گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۱ فصل ۱و۲

Contents

I Artificial Intelligence

1 Introduction

2 Intelligent Agents

II Problem-solving

3 Solving Problems by Searching

4 Beyond Classical Search

5 Adversarial Search

6 Constraint Satisfaction Problems

III Knowledge, reasoning, and planning

7 Logical Agents

8 First-Order Logic

9 Inference in First-Order Logic

10 Classical Planning

11 Planning and Acting in the Real World

12 Knowledge Representation

IV Uncertain knowledge and reasoning

13 Quantifying Uncertainty

14 Probabilistic Reasoning

15 Probabilistic Reasoning over Time

16 Making Simple Decisions

17 Making Complex Decisions

V Learning

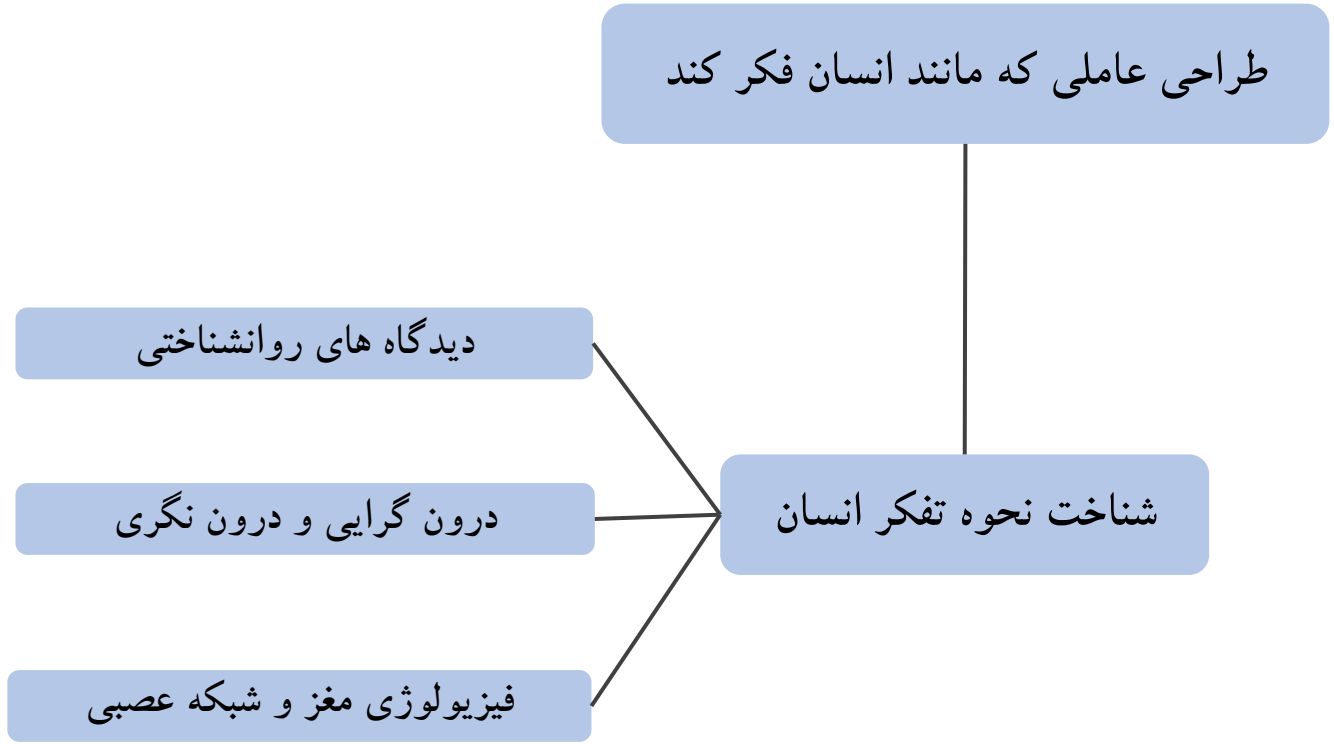
18 Learning from Examples

19 Knowledge in Learning

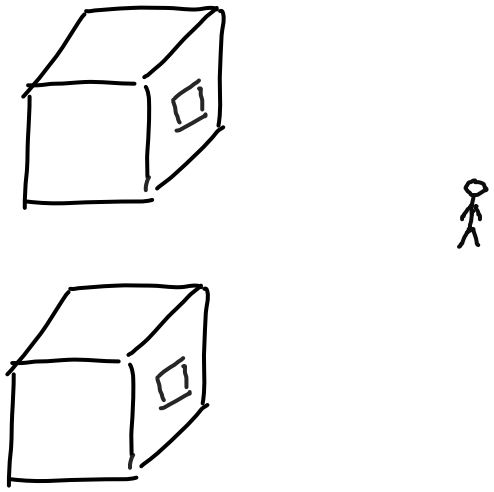
AI?

Humanly	Rationally	
مطالعه نحوه تفکر انسان، روانشناختی،درون گرایی، فیزیولوژی	یافتن الگوریتم های استنتاج منطقی نمایش دانش به زبان منطق	Thinking
تست تورینگ	مساله تعریف رفتار منطقی، معیار کارایی	Acting

Thinking humanly: The **cognitive modeling** approach (مدل سازی شناختی)



Acting humanly: The Turing Test approach



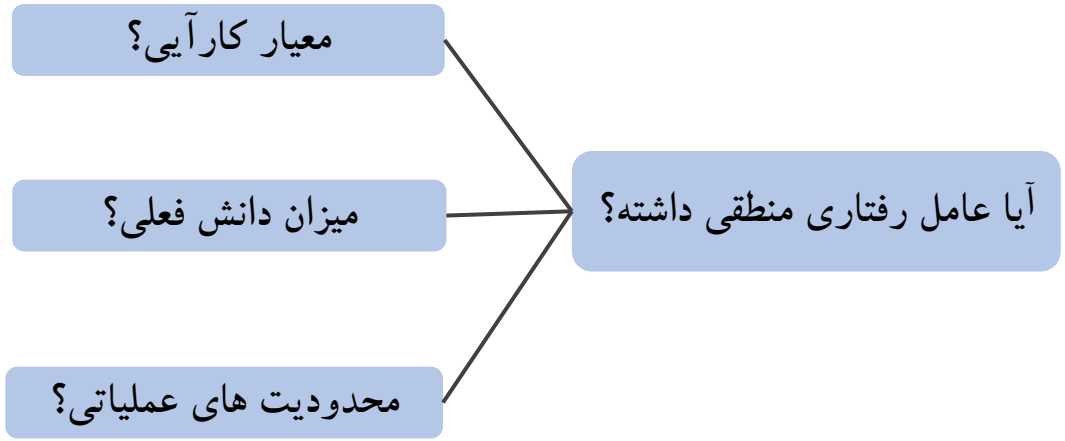
Turing Test

- natural language processing — پردازش زبان طبیعی
- automated reasoning — نمایش دانش
- knowledge representation — استدلال خودکار
- machine learning — یادگیری

Total Turing Test

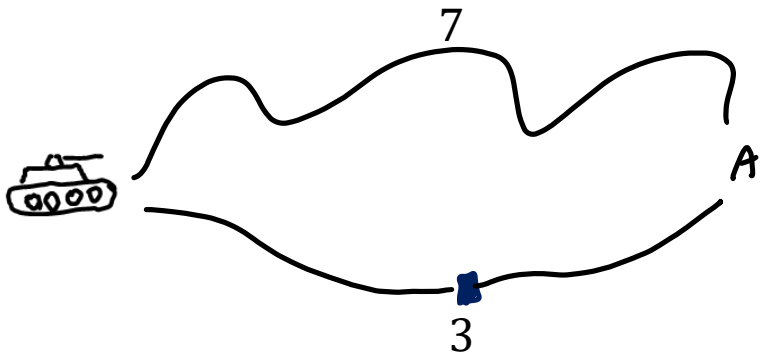
- computer vision
- robotics

Acting Rationally: performance measure



Rationality: maximizes expected performance.
Perfection: maximizes actual performance.

عامل خردمند: نتایج مورد انتظار را بهینه می‌کند.
عامل دانای کل: نتایج واقعی را که در عمل رخ می‌دهند بهینه می‌کند.



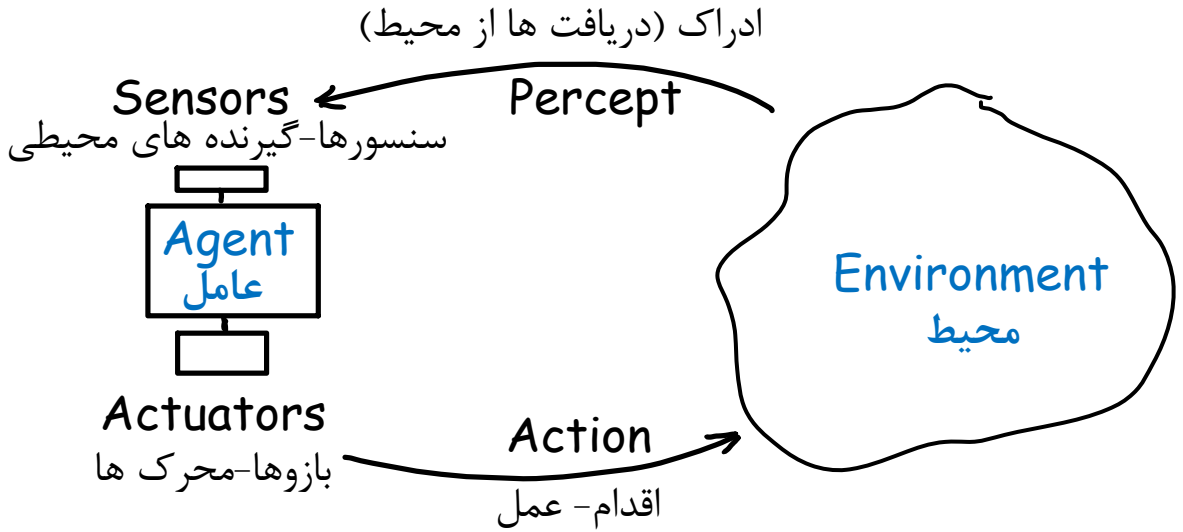
omniscience is impossible in reality.

ساخت موجود هوشمند دانای کل غیر ممکن است.

Thinking Rationally: logic language and inference algorithms

در فصل هفتم و فصل های بعد از آن به عوامل منطقی می‌پردازیم. این عوامل هوشمند، یک پایگاه دانش دارند و مجهز به الگوریتم های استنتاج برای تولید دانش و استخراج اطلاعات مفید با استفاده از دانش موجود در پایگاه دانش هستند.

ارتباط عامل و محیط:



چهارتایی محیط وظیفه:

Task environment: **PEAS** (**P**erformance, **E**nvironment, **A**ctuators, **S**ensors)

سنسورها، بازوها، محیط، کارآیی

مثال: محیط وظیفه مربوط به تاکسی هوشمند

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

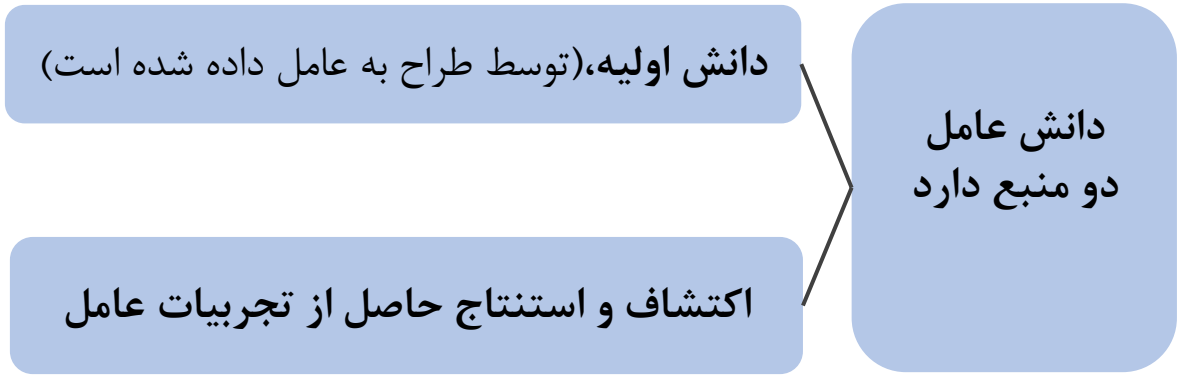
مثال های دیگر: محیط وظیفه مربوط به چند عامل

نوع کارگزار	مقیاس کارآیی	محیط	اقدام‌گرها	حس‌گرها
سامانه تشخیص پزشکی	بیمار سالم، کمینه کردن هزینه‌ها، مسائل حقوقی	بیمار، بیمارستان، کارکنان	نمایش پرسش‌ها، آزمایش‌ها، تشخیص‌ها، تجویزها و ارجاع‌ها	واردکردن علائم بیماری، یافته‌ها و پاسخ‌های بیمار از طریق صفحه‌کلید
سامانه تحلیل تصاویر ماهواره‌ای	دسته‌بندی صحیح تصویر	ارتباط رو به پایین ماهواره	نمایش دسته‌بندی تصویر	آرایه‌هایی از پیکسل‌های رنگی
روبات قطعه‌بردار	درصد قطعاتی که در سطل درست قرارگرفته‌اند	نوار نقاله همراه با قطعات، سطل‌ها	بازو و دست مفصل‌دار	دوربین، حس‌گرهای زاویه مفصل
پایش‌گر پالایشگاه	بیشینه کردن خلوص، بازده واقعیت	پالایشگاه، کاربران	شیرها، پمپ‌ها، گرم‌کن‌ها، صفحه‌های نمایش	حس‌گرهای دما، فشار و شیمیایی
معلم انگلیسی محاوره‌ای	بیشینه کردن نمره امتحان دانش‌آموز	مجموعه دانش‌آموزان، مؤسسه امتحان گیرنده	نمایش تمرین‌ها، پیشنهادها و پالایش‌ها	ورودی صفحه‌کلید

software agents

این دو نمونه عامل هوشمند نرم‌افزاری هستند.
گاه عامل هوشمند فقط یک برنامه است.

دانش عامل(Knowledge)



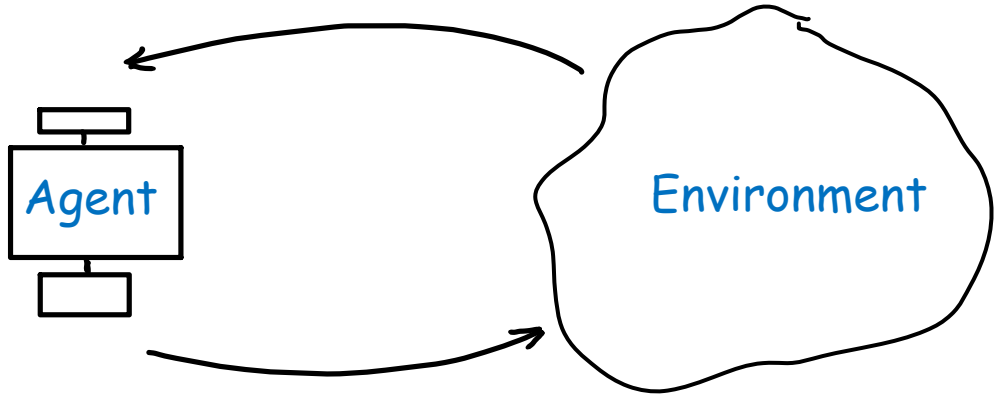
خود مختاری:

اگر عامل هوشمند، هیچ دانش اولیه ای نداشته باشد یا دانش اولیه آن حداقلی باشد و تمام دانش خود از محیط را با تجربه کردن،

و اکتشاف و استنتاج به دست آورد آن را خودمختار (autonomous) می‌گوییم.

از سوی دیگر عاملی که توان یادگیری و اکتشاف نداشته باشد و فقط بر مبنای دانش اولیه اش تصمیم گیری کند، فاقد خودمختاری است.

انواع محیط- انواع برنامه عامل



انواع محیط

Discrete vs. continuous	گسسته – پیوسته
Single agent vs. multi-agent	تک عاملی – چندعاملی
Episodic vs. sequential	اپیزودیک – ترتیبی
Fully observable vs. partially observable	کاملا مشاهده پذیر – جزئاً مشاهده پذیر
Deterministic vs. nondeterministic (stochastic)	قطعی – غیرقطعی
Static vs. dynamic	ایستا – پویا
Known vs. unknown	شناخته شده – ناشناخته

Discrete **گسسته:** اندازه محیط وتعداد اقدامات متناهی یا در تناظر با اعداد صحیح است. (مانند گراف ها، ماتریس ها و جداول اعداد صحیح، صفحه شطرنج)

Continuous **پیوسته:** اندازه و ابعاد محیط یا نوع اقدامات در تناظر با اعداد حقیقی است مانند زمان و مکان در دنیای واقعی.

Single agent **تک عاملی:** مانند موشی که در مارپیچ به دنبال راه خروج است.

multi-agent **چندعاملی:** مانند بازی های دو یا چند نفره، تاکسی هوشمند در محیطی که رانندگان دیگر هم در حال حرکتند.

ارتباط عوامل در محیط چندعاملی:

- **همکاری** (بازیکنان هم تیمی).
- **رقابت** (دو رقیب بازی شطرنج).
- **ترکیبی همکاری و رقابت** (رانندگان در یک مسابقه برای برنده شدن رقابت می‌کنند برای حفظ امنیت همکاری می‌کنند).

Episodic

رویدادی(اپیزودیک): در این نوع محیط، اقدام عامل در هر اپیزود (رویداد) هیچ تاثیری از رویداد قبلی نمی‌پذیرد و تاثیری بر رویداد بعدی نیز ندارد. رویدادها مستقل هستند. برای مثال رباتی که قطعات را روی یک تسمه نقاله، یکی یکی بررسی می‌کند تا قطعات خراب را جدا کند. بررسی هر قطعه، یک ماموریت مستقل است.

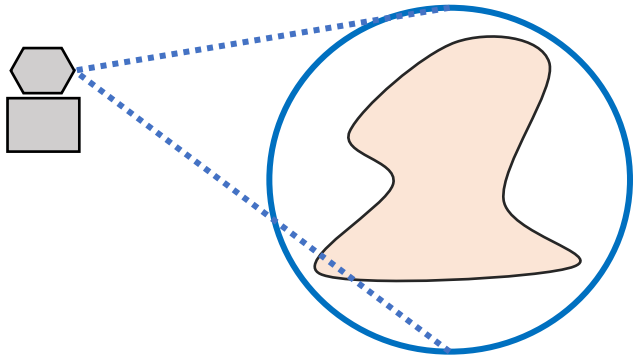
Sequential

ترتیبی(دنباله‌ای): در این محیط اقدام عامل در هر مرحله، روی وضعیت محیط و اقدام عامل در مراحل بعد اثرگذار است. مثلا در شطرنج انجام هر حرکت روی وضعیت مهره ها و حرکت بعدی اثر دارد.در تاکسی هوشمند انتخاب جهت حرکت در این چهار راه روی انتخاب های راننده در ادامه مسیر اثرگذار است.

Fully observable vs. partially observable

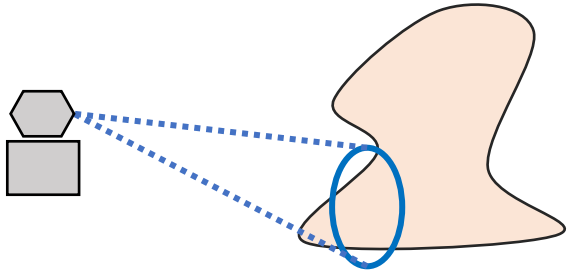
مشاهده پذیری :
اینکه محیط را کلاً مشاهده‌پذیر در نظر بگیریم یا جزئاً مشاهده‌پذیر ، وابسته به سنسورهایی است که عامل به آنها مجهز است .پس ویژگی ذاتی آن محیط نیست.یک محیط ، ممکن است برای یک عامل مشاهده پذیر و برای عامل دیگر جزئاً مشاهده‌پذیر باشد.

Fully observable



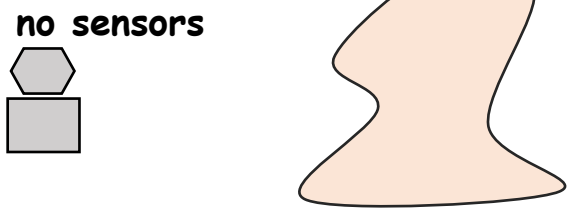
کلاً مشاهده‌پذیر

Partially observable



جزئاً مشاهده‌پذیر

Unobservable



مشاهده ناپذیر

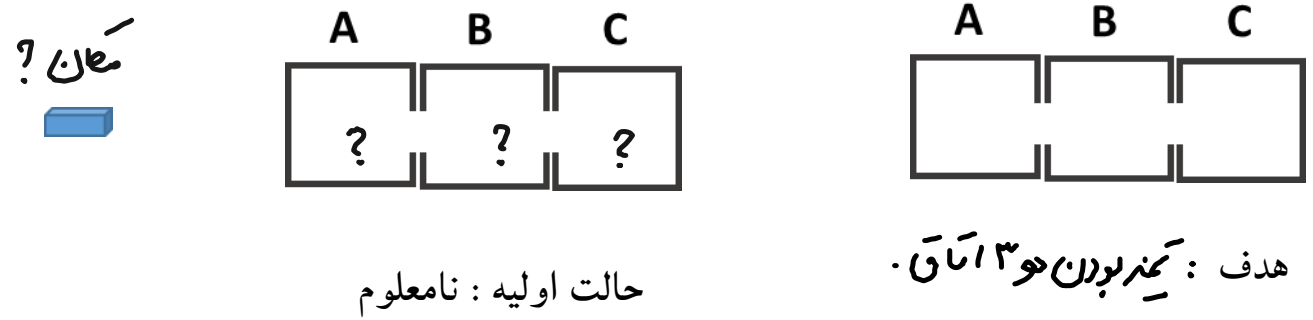
Unobservable

no sensors

حتی در محیط غیرقابل مشاهده هم می‌توان مسئله را حل کرد. کافیست به جای حالت، از مجموعه حالات باور استفاده کنیم.

مثال: جارو برقی هوشمند بدون سنسور: نمی‌داند در کدام اتاق است. نمی‌داند آن جا کثیف است یا تمیز.

اما باز هم می‌تواند یک دنباله اقدامات برای رسیدن به هدف بیابد.



مکان ؟

حالت اولیه : نامعلوم

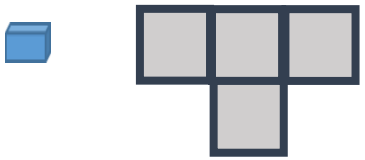
هدف : تمیز بودن هر ۳ اتاق.

دنباله جواب:

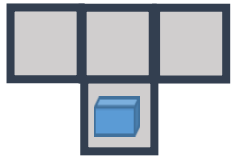
[LEFT,LEFT,SUCK,RIGHT,SUCK,RIGHT,SUCK]

مثال ۲ : یک عامل متحرک می‌خواهد از هر جایی که هست به اتاق هدف برسد.

no sensors



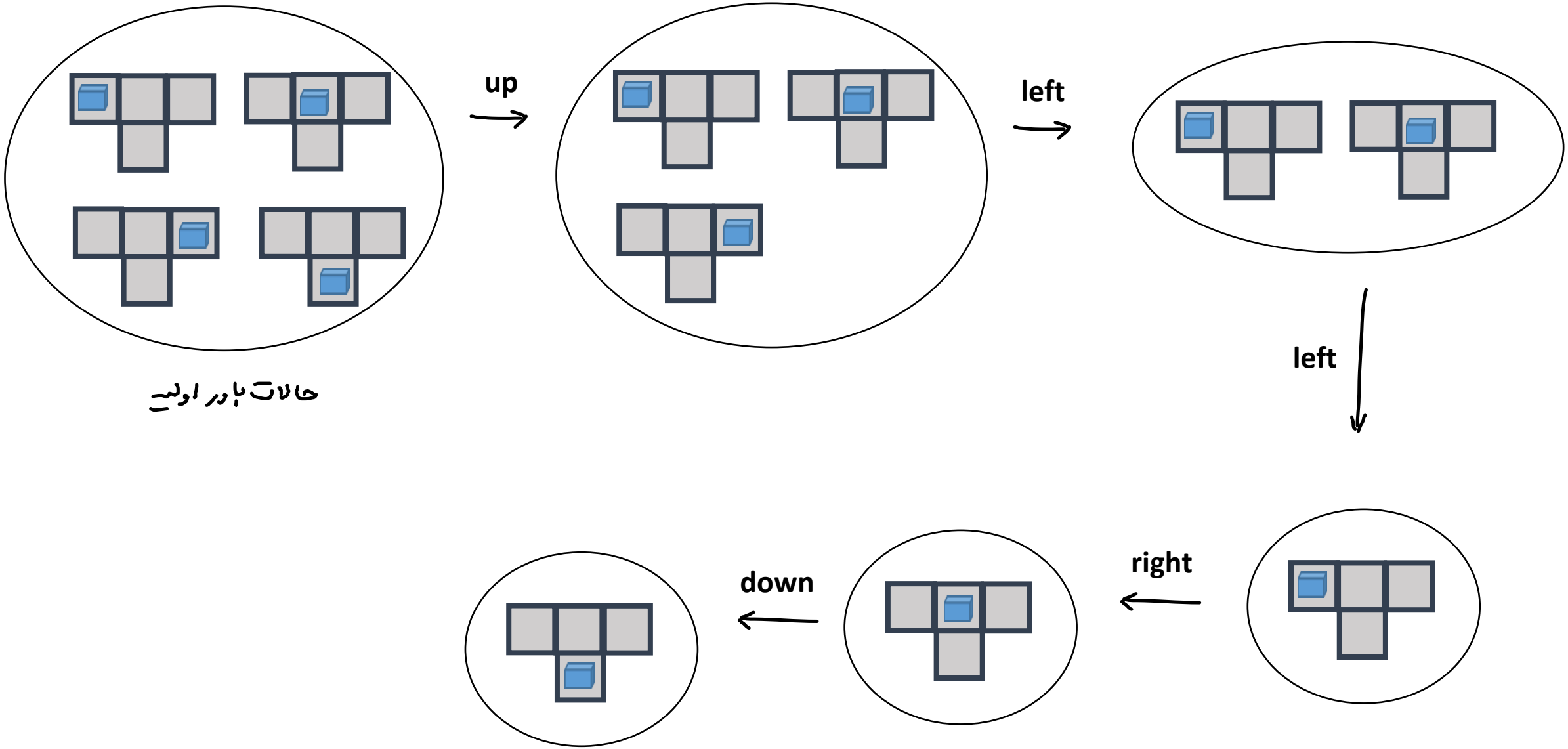
مکان اولیه : نامعلوم



حالت هدف

[up, left, left, right, down] دنباله جواب:

در اسلاید بعدی نحوه‌ی استفاده از فضای حالت با ورتکس داده شده.


up
left
left
right
down
حالات باور اولیه

Static vs. dynamic

Dynamic پویا: بدون دخالت و اقدام عامل، محیط درحال تغییر است.

Static ایستا: تا هنگامی که عامل اقدامی انجام ندهد، محیط تغییری نمی‌کند.

Semi-dynamic نیمه پویا: تا هنگامی که عامل اقدام انجام ندهد محیط تغییری نمی‌کند اما تابع سودمندی تغییر می‌کند
مثل شطرنج با ساعت. [زمان برای عامل می‌گذرد]

Deterministic vs. nondeterministic (stochastic)

Deterministic:

محیط قطعی : با دانستن وضعیت فعلی و اقدام انتخاب شده ، میتوانیم وضعیت بعدی را با قطعیت تشخیص دهیم.

current state + action executed ⇒ next state

این نتیجه گیری ها در حالت کلی درست نیستند اما در اغلب محیط ها درست هستند:

Dynamic ⇒ (معمولاً) Nondeterministic

Partially observable ⇒ (معمولاً) Nondeterministic

عدم اطمینان از وضعیت یعنی از دو چیز ممکن است به وجود آمده باشد:

Uncertainty ⇒ Nondeterministic actions (اقدامات با نتایج غیرقطعی) ∨ Partially observable (جزئی مشاهده‌پذیر بودن محیط)

پس:

Deterministic ∧ Fully observable ⇒ Certainty

Strategic environment

we ignore uncertainty that arises purely from the actions of other agents in a multi-agent environment

در محیط چند عامله : اگر تغییرات محیط را که حاصل اقدامات سایر عوامل هوشمند است لحاظ نکنیم (فرض کنیم سایر عوامل بی حرکت شده اند) آنگاه بینیم که محیط قطعی است ، به آن محیط استراتژیک گوییم.

در محیط استراتژیک : به شرط آن که سایر عوامل دخالت نکنند ، ما نتیجه ی قطعی اقدام خود را می دانیم.

برخی از محیط‌های غیرقطعی از نوع (تصادفی) هستند. یعنی درست است که نتیجه اقدام را نمی‌دانیم اما احتمال رخ دادن نتایج مختلف را می‌دانیم. مثلاً در پرتاب تاس، با فرایند تصادفی روبرو هستیم.

stochastic ~ **probabilities**

تصادفی — دانستن احتمالات

Known vs. unknown

کی وابسته به عامل یا طراح آن

Known: The agent knows the rules and model of the environment.

Unknown: The agent does not know the rules and model of the environment.

← محیط مشاهده پذیر و غیرمشاهده پذیر ربطی به محیط شناخته شده و ناشناخته ندارد.

در بازی ورق شما قوانین بازی را می دانید (مدل را در اختیار دارید) پس محیط شناخته شده است هرچند که مشاهده پذیر کامل نیست چون دست حریف را نمی بینید.

وقتی یک کودک برای اولین بار با بازی مار روبرو می شود قوانین بازی و حتی نوع حرکات را نمی داند (ناشناخته) اما تمام اشیاء در صفحه بازی را می بیند. (مشاهده پذیر).



گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

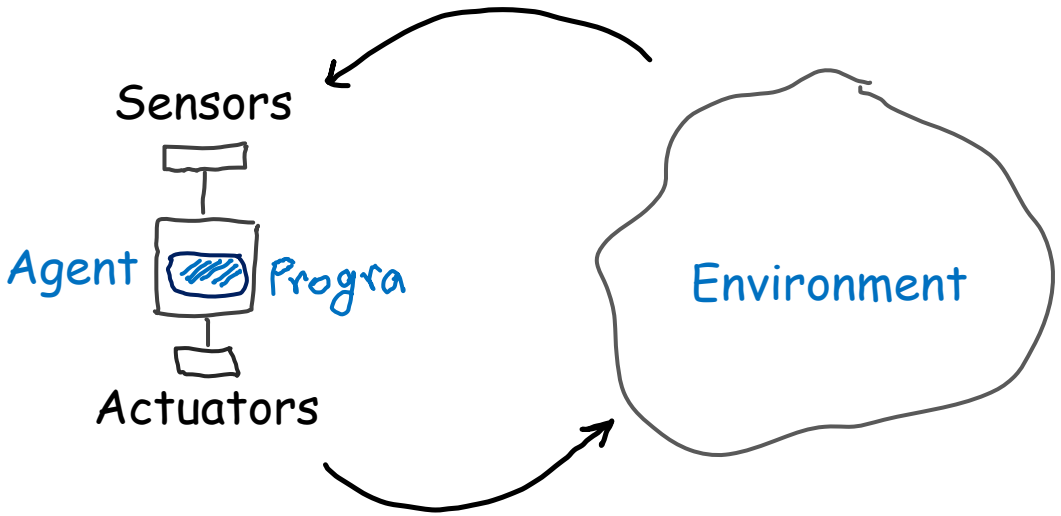
هوش مصنوعی

ابوالفضل گیلک
گروه بابان

بخش دوم جلسه ۱

عامل هوشمند = معماری + برنامه‌عامل

درس ما درمورد انواع برنامه‌های عامل است.



چند اصطلاح:

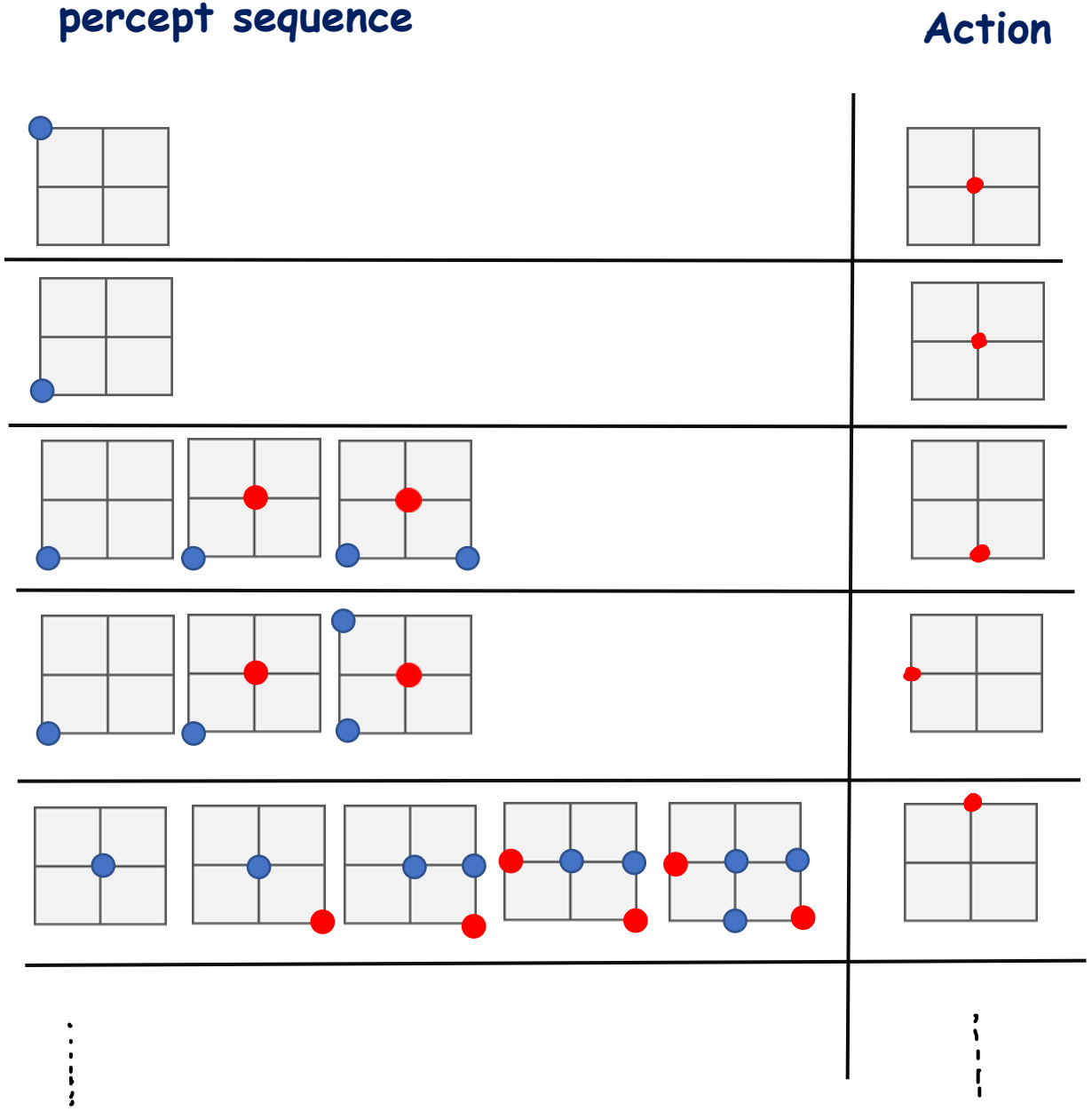
percept sequence دنباله ادراکات:

دنباله‌ای از ادراکات متوالی عامل از محیط.

agent program برنامه عامل: یک کد است که دنباله ادراکات را گرفته و اقدام مناسب را برمی‌گرداند.

agent function تابع عامل: یک مفهوم ریاضی (ذهنی) که مانند یک تابع هر دنباله ادراکات را به یک اقدام تصویر می‌کند.

نمایش چند سطر از تابع عامل بازی دوز :



مثلاً یک تابع عامل برای بازی دوز که به ازای هر دنباله از ادراکات، یک اقدام به عامل می‌دهد.
در اغلب مسائل، ساختن چنین جدولی در عمل ممکن نیست چون تعداد دنباله ادراکات بی‌نهایت می‌شود. پس تابع عامل یک مفهوم ذهنی و ریاضی است.

دنباله ادراکات $\xrightarrow{f}$ اقدام

اما برنامه عامل یک کد است که با روش‌های جبر، یا استفاده از منطق و ابزارهای دیگر نوشته می‌شود و می‌تواند همان کار تابع عامل را انجام دهد. با داشتن دنباله ادراکات، عمل بعدی را به عنوان خروجی به عامل بدهد.

تابع عامل :

percept sequence	
20	turn off
24,22,27,26	turn on
25,22,23	turn off
19,21	turn off
24,22,28,26,26,25,26	turn on
20,22,22,24,25,25,24,25,27	turn on

برنامه عامل : (تنظیم گر کولر بادی ۲۴)

```
get (heat);
if (heat>24) {turn on }
else{turn off}
```

انواع برنامه عامل:

Simple reflex agents;	واکنشی ساده
Model-based reflex agents;	واکنشی مدل‌گرا
Goal-based agents;	هدف‌گرا
Utility-based agents	مبتنی بر سودمندی

واکنش ساده:

برف پاک کن هوشمند، تعویض هوادردهای ۲۴ درجه، ...

چشم الکترونیکی برای شیر آب، درب پارکینگ، ....

واکنش مدل گرا (دارای حالت داخلی)

تشخیص دهنده‌ی چراغ چشمک زن، محاسبه گر سرعت، ....

هدف گرای مدل گرا: (مبتنی بر هدف)

مسیریاب، تاکسی هوشمند با هدف کاهش زمان مسافت،

هدف گرا، مدل گرا، (مبتنی بر سودمندی):

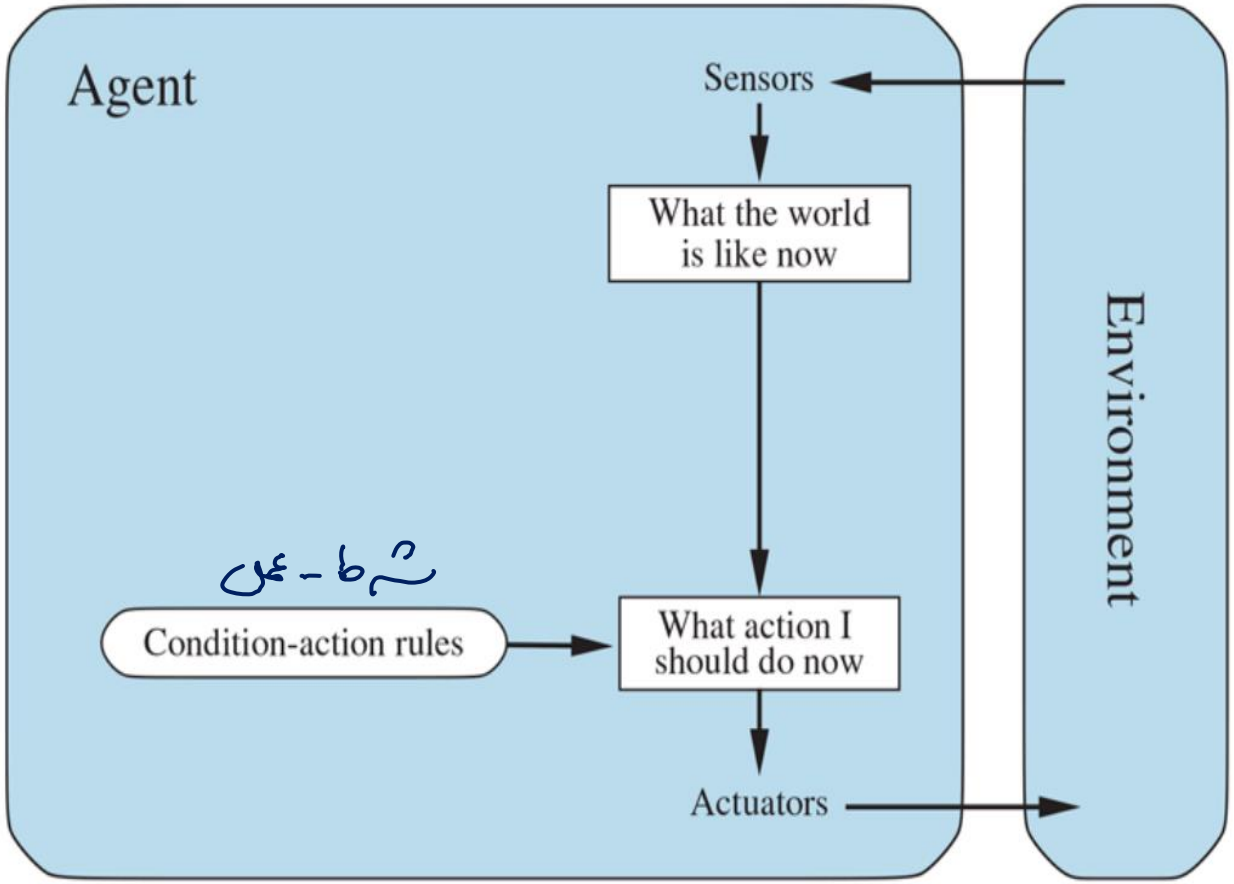
تاکسی هوشمند با اهداف: سرعت، امنیت، کاهش مصرف

عامل ساده

```
state ← INTERPRET-INPUT(percept)
rule ← RULE-MATCH(state, rules)
action ← rule.ACTION
return action
```

دقت کنید که باید بتوانند با سنسورهایشان وضعیت محیط را تشخیص دهند پس لازم است محیط مشاهده پذیر باشد

Simple reflex agents



عامل واکنش ساده: در محیط کاملاً مشاهده پذیر می‌تواند عقلانی عمل کند. (شرط لازم)

واکنش ساده: حالت داخلی ندارد. پس مثلاً چشمک زدن چراغ را نمی‌تواند تشخیص دهد.

واکنش ساده: براساس قوانین شرط-عمل اقدام را انتخاب می‌کند:

IF .... then ....

واکنشی، مدل‌گرا

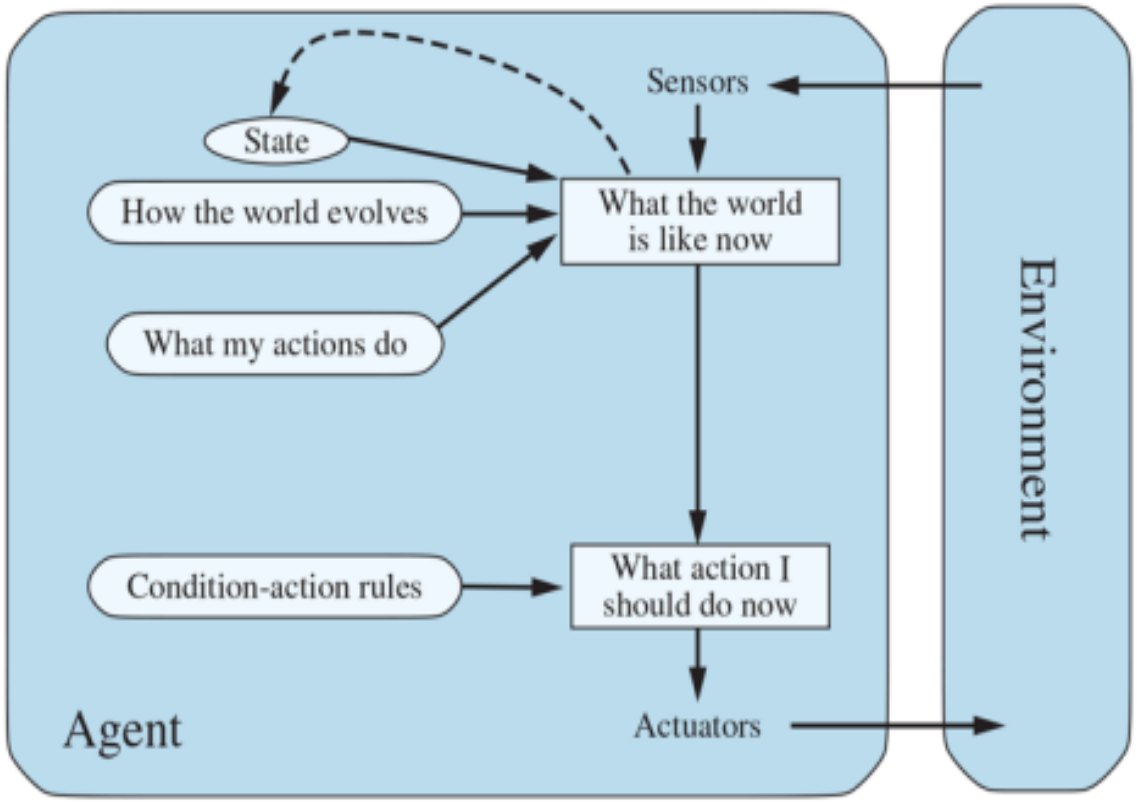
$state \leftarrow$ UPDATE-STATE($state, action, percept, model$)
$rule \leftarrow$ RULE-MATCH($state, rules$)
$action \leftarrow rule.$ACTION
return $action$

حالت داخلی دارد. (ذخیره‌ی وضعیت قبلی محیط)

مُدل را دارد (می‌داند این محیط چگونه تغییر می‌کند)

از شرط-عمل استفاده می‌کند.

Model-based reflex agents



نتیجه: به روزرسانی وضعیت محیط:

$state \leftarrow \text{UPDATE-STATE}(state, action, percept, model)$

- $state$: حالت داخلی
- $action$: اقدام انجام شده
- $percept$: دریافت سنسورها از محیط
- $model$: مدل این دنیا.

درخواست چهارم راسل، مدل به دو بخش تفکیک شده. [نحوه محاسبه سنسورها و چگونگی تغییر آن]

- مدل تغییرات
- مدل سنسورها

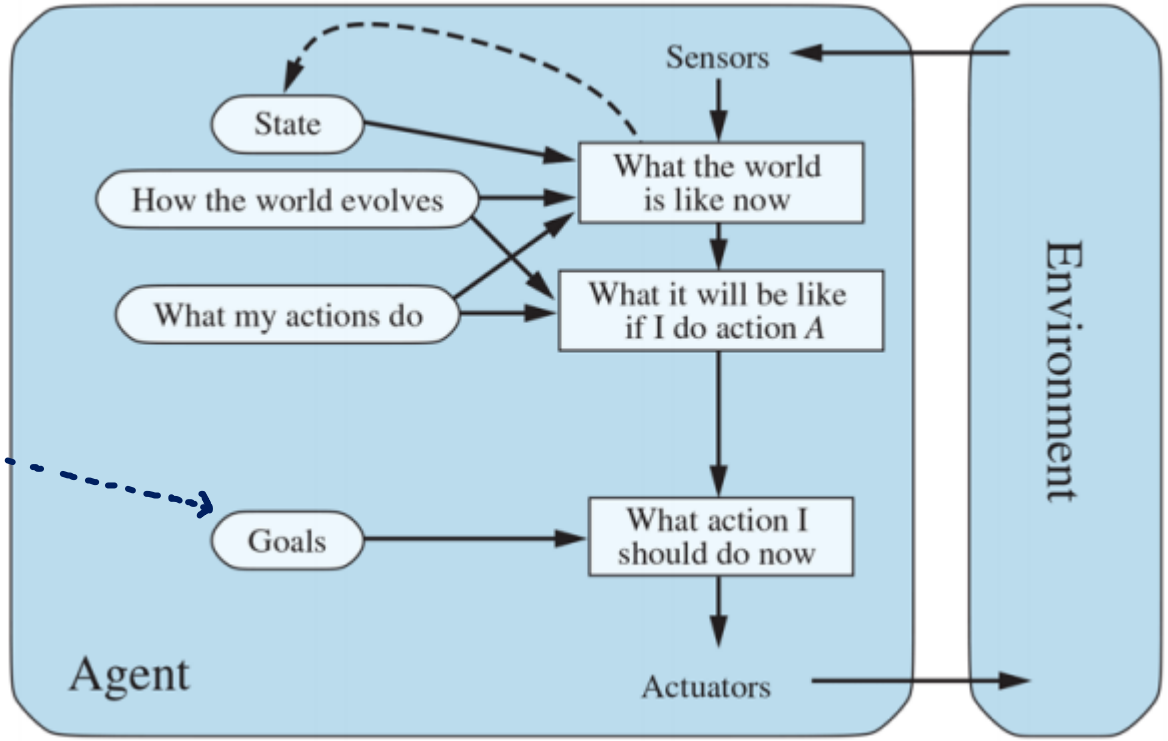
$state \leftarrow \text{UPDATE-STATE}(state, action, percept, transition_model, sensor_model)$

عامل هدف‌گرا- مدل‌گرا

Model + Goal-based agents

مدل محیط را دارد.
حالت داخلی دارد.

به جای قوانین شرط عمل
براساس رسیدن به هدف
اقدام مناسب تعیین می‌شود.



هدف‌گرا : علاوه بر ادراک فعلی ، و حالت داخلی (حافظه) یک هدف هم دارد.

پس دیگر استفاده از (شرط-عمل) جواب نمی‌دهد بلکه رسیدن به

هدف، ملاک تصمیم است.

نتیجه عامل هدف‌گرا: اگر هدف را تغییر دهیم ، احتیاجی نیست که برنامه را تغییر دهیم.

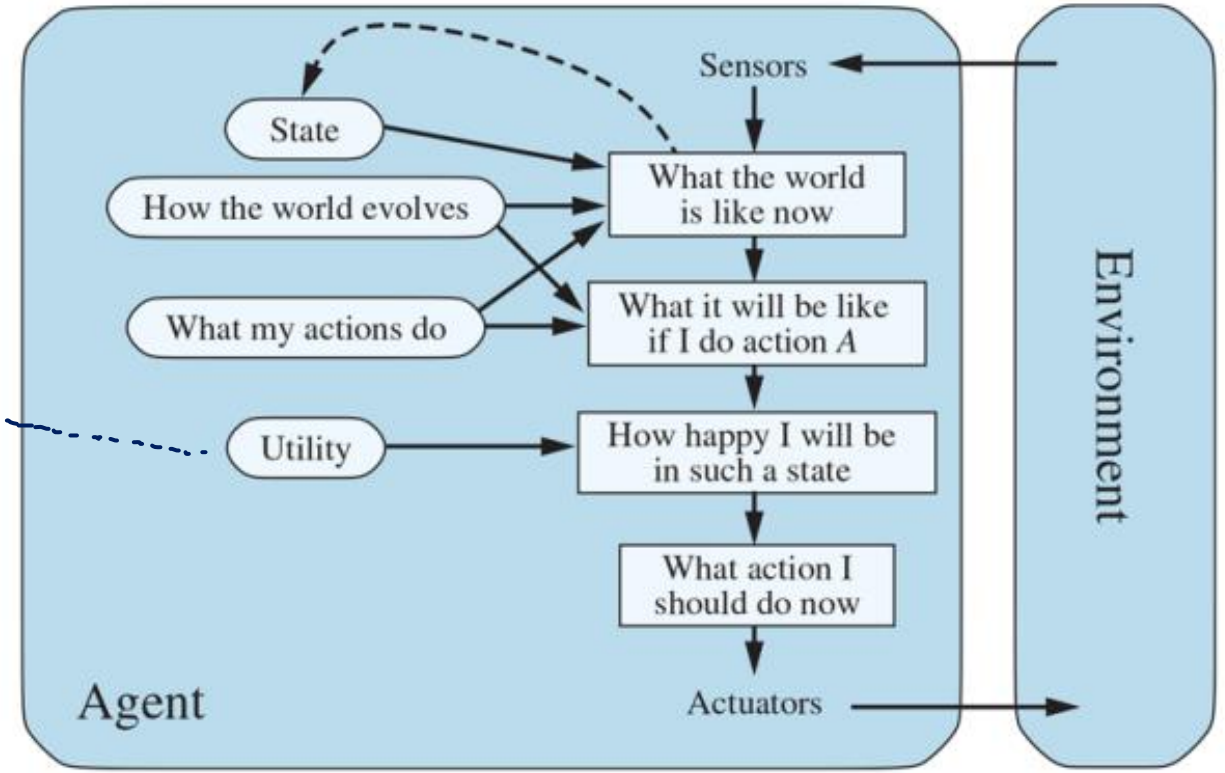
عامل مدل‌گرای هدف‌گرای مبتنی بر (سودمندی)

به این دلایل نیاز به تابع سودمندی داریم:

۱- اهداف متضاد (مانند سرعت و امنیت حرکت)

۲- عدم امکان تحقق همه اهداف

Model + Utility-based agents



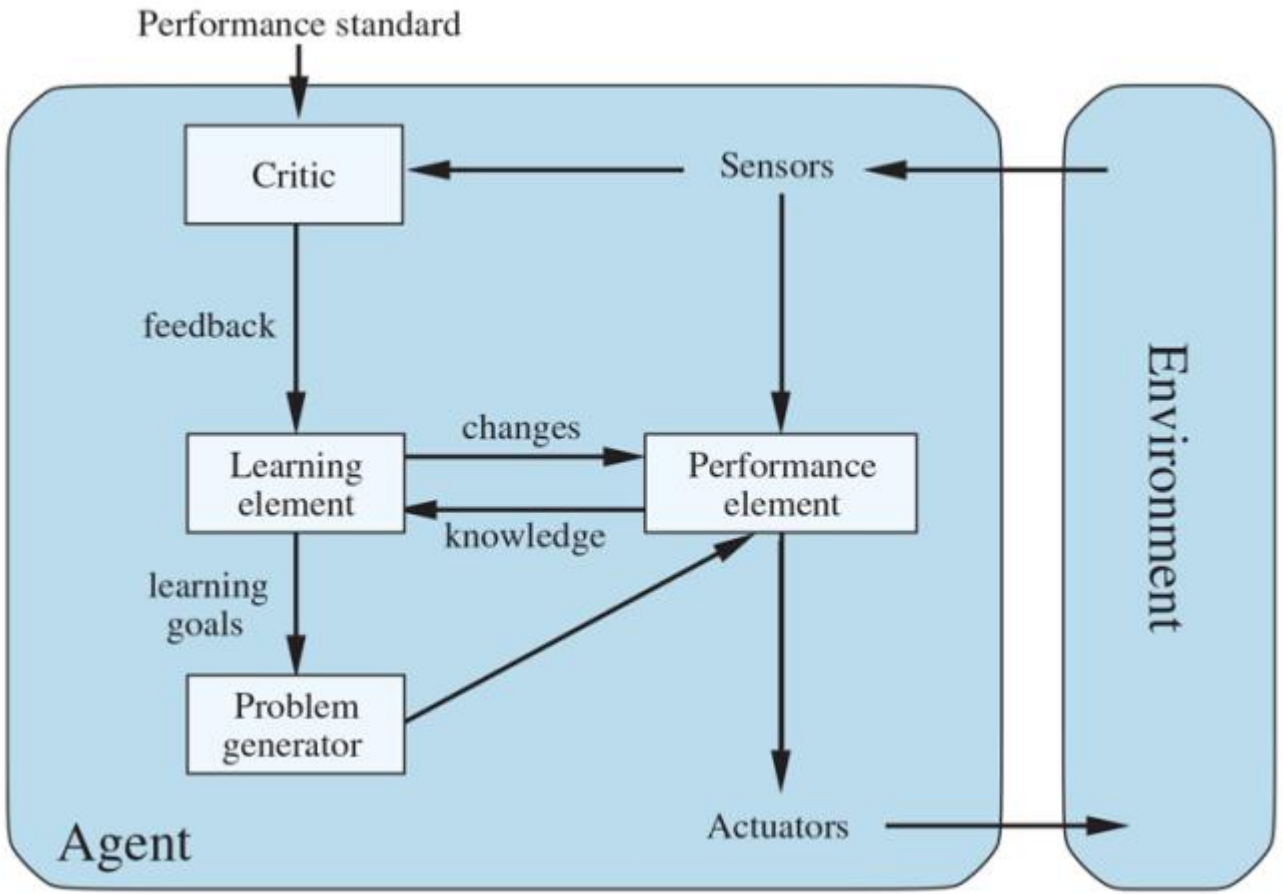
براساس پاسخ به بیشترین سودمندی اقدام مناسب تعیین می‌شود.

عوامل یادگیرنده: Learning agents

برنامه‌ی عامل، تا اینجا خودمختاری و یادگیری نداشتند.

عامل با توان یادگیری نیازمند چهار بخش مهم است:

A general learning agents



A general learning agents

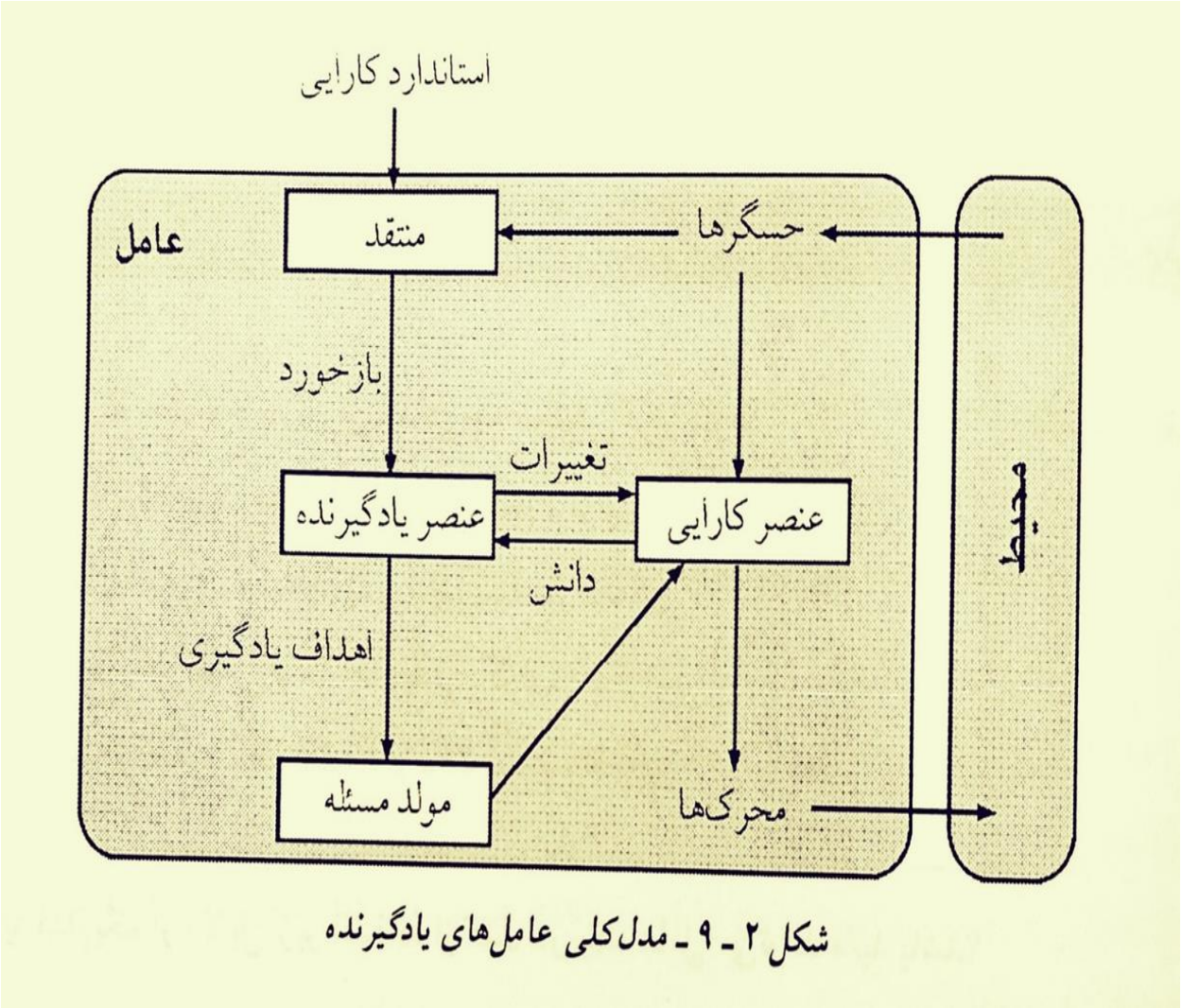
استاندارد کارایی: توسط طراح عامل تعیین می‌شود.

واحد منتقد: براساس استاندارد کارایی به اقدامات نمره مثبت یا منفی می‌دهد.

عنصر یادگیرنده: این امتیازات را در آینده لحاظ می‌کند.

مولد مسأله: همان تفکر کنجکاوی دانان را دارد.

مسأله جدید تولید می‌کند تا با حل آن تجربه و دانش عامل بیشتر شود.



شکل ۲ ـ ۹ ـ مدل کلی عامل‌های یادگیرنده

۲۷ـ روبات برف‌پاک‌کن در خودرو جزء کدام دسته از عامل‌های هوشمند است؟

(مهندسی کامپیوتر ـ دولتی ۹۶)

۱) واکنشی با حافظهٔ داخلی

۲) واکنشی ساده

۳) هدف‌گرا

۴) سودمندگرا

۸۷- اگر هدف رساندن یک مسافر از مبدأ به مقصد توسط یک عامل هوشمند باشد و معیار کارایی امنیت، زمان و مسیر خلوت باشد. چه نوع عاملی مناسب است؟

۱) هدف‌گرا
۲) مدل‌گرا
۳) سودمند
۴) واکنشی ساده

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۲ فصل ۳

حل مسئله با جستجو (جستجو در گراف فضای حالات یا درخت حالات با ورد)

برخی از مسائل را می‌توان به این صورت پیاده‌سازی کرد:

- state از محیط ← یک گره از گراف فضای حالات
- اقدام action ← یک یال
- آزمون هدف: goal test
  - گاه به صورت صریح (مجموعه‌ای از گره‌ها)
  - گاه به صورت ضمنی (مثلاً توصیف حالت ملات در شطرنج)

خروجی یک جستجو : دنباله‌ی اقدامات جواب.

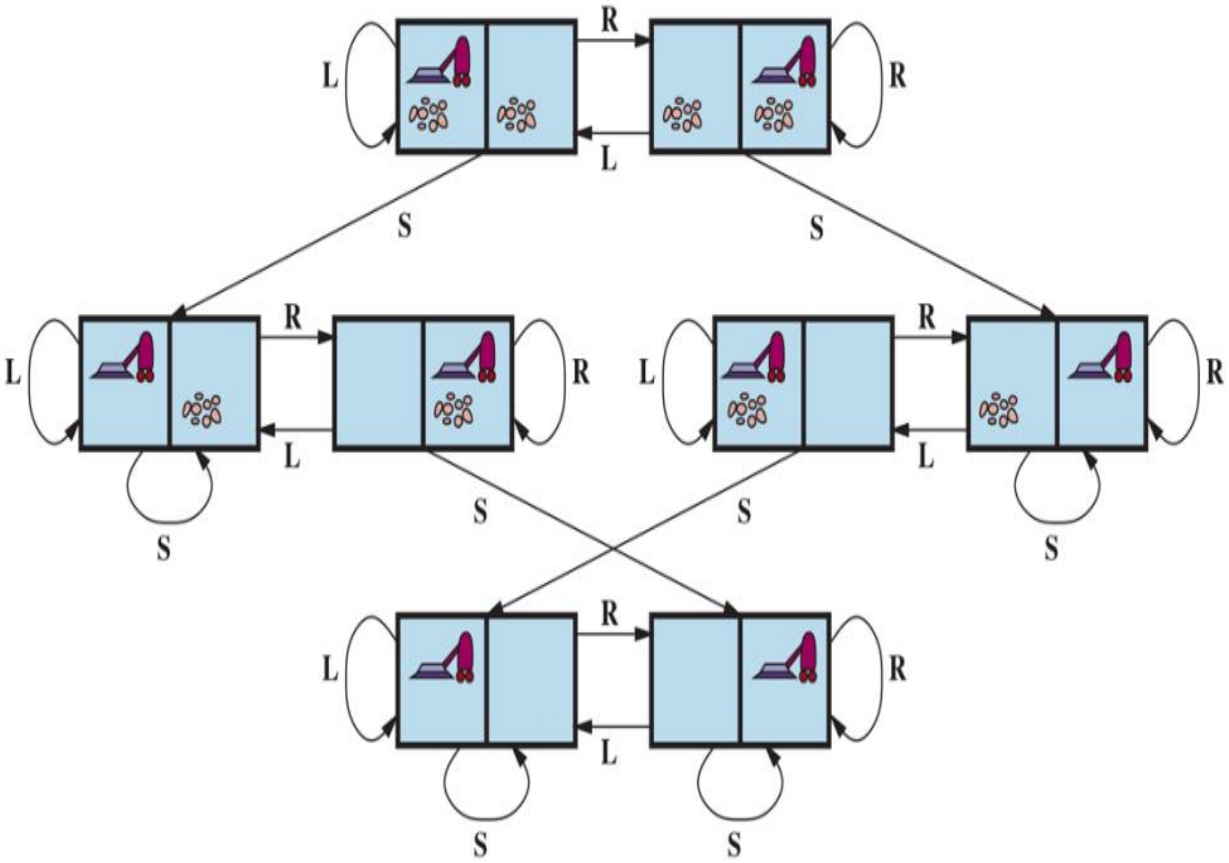
$[\ action_1\ ,\ action_2\ ,\ action_3\ ,\ \ldots\ ,\ action_d\ ]$

که در واقع یک مسیر است از حالت اولیه به حالت هدف

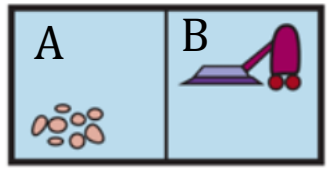
برخی الگوریتم‌ها می‌توانند دنباله‌ی اقدامات بهینه را پیدا کنند.

گراف فضای حالت مساله جاروبرقی هوشمند:

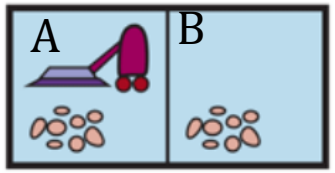
نمایش هر حالت به صورت یک سه تایی مرتب است:
دو مولفه اول تمیز یا کثیف بودن اتاق ها را نشان می‌دهد
و سومی مکان جاروبرقی است.



The state-space graph for the two-cell vacuum world. There are 8 states and three actions for each state: L = *Left*, R = *Right*, S = *Suck*.



(Dirty, Clean, B)

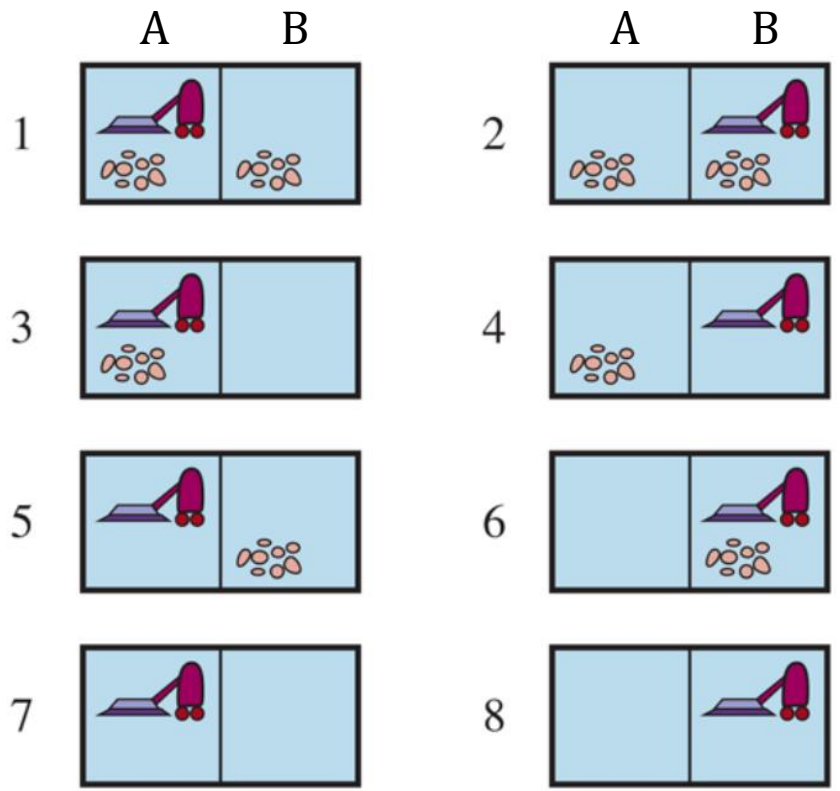


(Dirty, Dirty, A)

در این مساله هزینه هر اقدام واحد ۱ است.

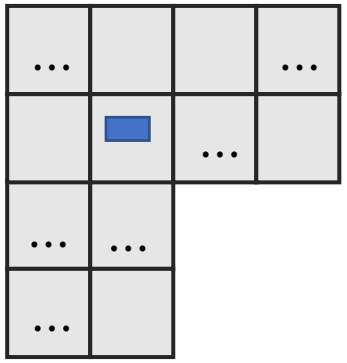
اندازه فضای حالات مساله جاروبرقی با دو اتاق: $2 \times 2 \times 2 = 8$

شماره گزاری حالات:



اندازه فضای حالات مساله جاروبرقی با n اتاق:

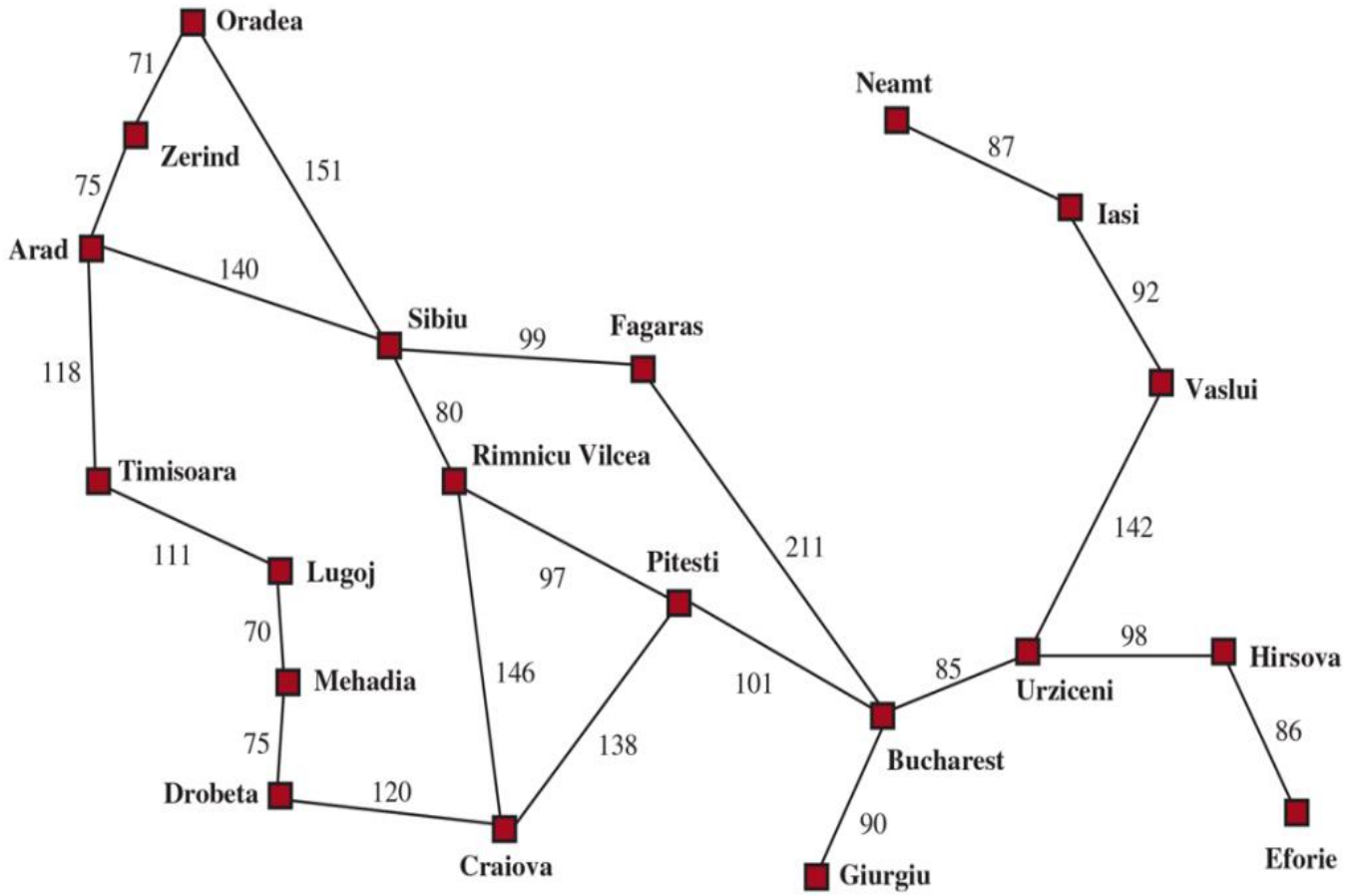
هراتاق دو حالت دارد (تمیز یا کثیف) محل عامل n حالت دارد.



تعداد حالات $= 2^n \times n$

مساله: یافتن کوتاهترین مسیر از Arad به Bucharest

هزینه‌ی اقدامات: طول جاده‌ی انتخاب شده



A simplified road map of part of Romania, with road distances in miles.

معمای ۸ :(8 – Puzzle)

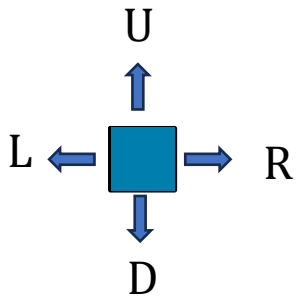
۹ خانه داریم. یکی خالی است و سایرین با مربع های متحرک ۱و۲و۳و....تا ۸ پر شده اند. پس تعداد کل جایگشت ها !۹ است.

به عبارتی به لحاظ ریاضی دقیقا !۹ چیدمان مختلف داریم. البته با شروع از حالت اولیه، فقط نیمی از این چیدمان ها قابل تولید هستند.

پس گراف فضای حالات با معلوم بودن حالت اولیه، $\frac{۹!}{۲}$ گره دارد..

چهار اقدام (جابجایی مربع خالی با یکی از همسایگانش).

هزینه هر اقدام یک واحد است.

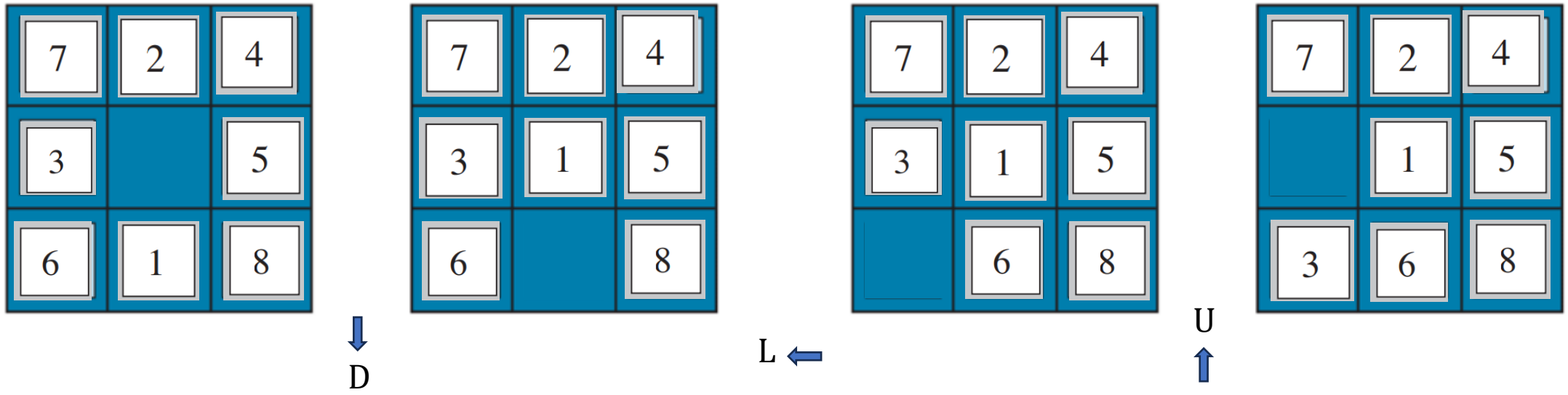


7	2	4
5		6
8	3	1

حالت اولیه

	1	2
3	4	5
6	7	8

حالت هدف


7 2 4
3 5
6 1 8
D
7 2 4
3 1 5
6 8
L
7 2 4
3 1 5
6 8
U
7 2 4
1 5
3 6 8

اندازه فضای حالات مساله $\left((n^2-1)\text{Puzzle}\right)$:

در جدول $n \times n$ با یک مربع خالی، تعداد کل حالات قابل تولید با شروع از حالت اولیه $\frac{n!}{۲}$ است.

پس گراف فضای حالات $\frac{n!}{۲}$ گره دارد.

مساله ۸ وزیر:

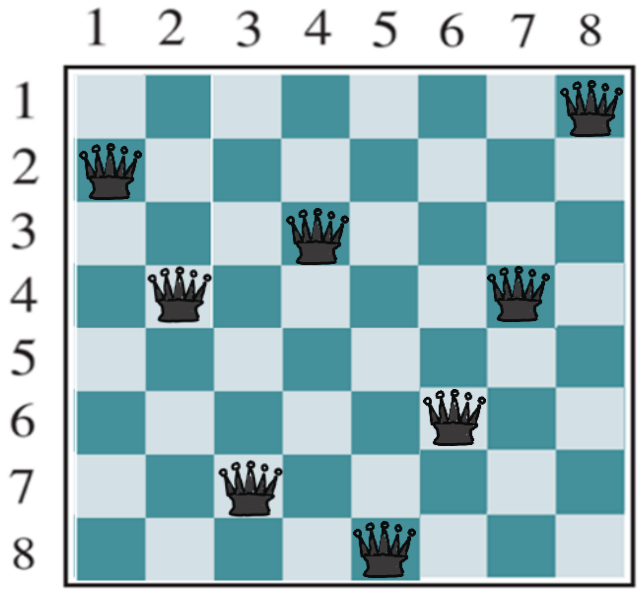
یافتن چیدمانی از ۸ وزیر در صفحه شطرنج که هیچکدام دیگری را تهدید نکند.
این مساله دو نوع پیاده سازی دارد.

پیاده سازی ناکارآمد:

دراین روش در هر حالت تعداد k وزیر در صفحه جانمایی شده اند$(k = 0,1, \dots ,8)$.
در هر حرکت، یک وزیر اضافه می‌شود که ممکن است در هر جای خالی قرار داده شود.
البته شرط عدم برخورد بررسی می شود.
حالت اولیه صفحه خالی است و سپس در هر حرکت یک وزیر به صفحه اضافه می‌کنیم.

پیاده سازی کارآمد(افزایشی)

در هر حالت برای تعدادk ستون سمت چپ در هر کدام یک وزیر جانمایی شده است،
به گونه ای که این k وزیر یکدیگر را تهدید نمی‌کنند. $(k = 0,1, \dots ,8)$.
ابتدا از یک صفحه خالی آغاز می‌شود.
در هر اقدام به اولین ستون خالی یک وزیر اضافه می‌شود طوری که قبلی ها را تهدید نکند.



نحوه نمایش حالت فوق:

[2 , 4 , 7 , 3 , 8 , 6 , 4 , 1]

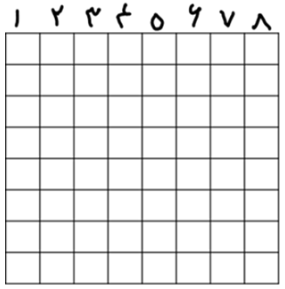
دیباگ‌سازی ناکارآمد (عادی)

اگر هر نوع چینش از $0 \le K \le 8$ وزیر را یک حالت فرض کنیم.

state: K وزیر در صفحه $0 \le K \le n$

initial: $K=0$

Action: قراردادن یک وزیر دیگر



$N = 1,8 \times 10^{14}$

$b = 64$

(حداکثر) فاکتور انشعاب: $b = 64$

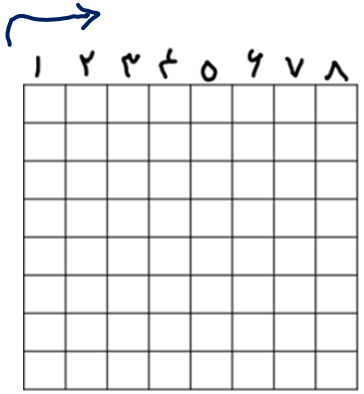
$$N = 1 + 64 + 64 \times 63 + \cdots + 64 \times 63 \times \cdots \times 57$$

- 1: خانه‌ی خالی
- 64: یک وزیر
- 64×63: دو وزیر
- $64 \times 63 \times \cdots \times 57$: ۸ وزیر

پیاده سازی افرایشی (کارآمد)

در این پیاده سازی ستون ها از چپ به راست پر می شوند.

اگر k وزیر داریم باید در k ستون سمت چپ قرار گرفته باشند.



state: k وزیر بازگذار در k ستون اول $0 \le k \le n$

initial: $k=0$

Action: قرار دادن یک وزیر در ستون $k+1$ که بازگذار باشد.

$N = 2^{0} \times 7^{5}$

$b = 8$

For example, we can reach 5 from 4 as follows:

$$\left\lfloor \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{(4!)!}}}}} \right\rfloor = 5.$$

یک مثال متفاوت:

کسی دنبال اقدامات جواب را مشخص می‌کند

حالات: اعداد صحیح نامنفی (یا اعداد حقیقی نامنفی)

حالت اولیه: ۴

هدف: ۵

اقدامات: جذر گرفتن، حد پایین گرفتن، فاکتوریل گرفتن

A simple problem-solving agent.

function SIMPLE-PROBLEM-SOLVING-AGENT(*percept*) **returns** an action
  persistent: *seq*, an action sequence, initially empty
    state, some description of the current world state
    goal, a goal, initially null
    problem, a problem formulation

برنامه عامل حل مسئله با جستجو دو بخش اصلی دارد [جستجو / اجرا

search.

```
state ← UPDATE-STATE(state, percept)
if seq is empty then
    goal ← FORMULATE-GOAL(state)
    problem ← FORMULATE-PROBLEM(state, goal)
    seq ← SEARCH(problem)

if seq = failure then return a null action
```

فاز جستجوی جواب:

بحث اصلی ما در این فصل، مربوط به الگوریتم‌های فاز جستجو است.

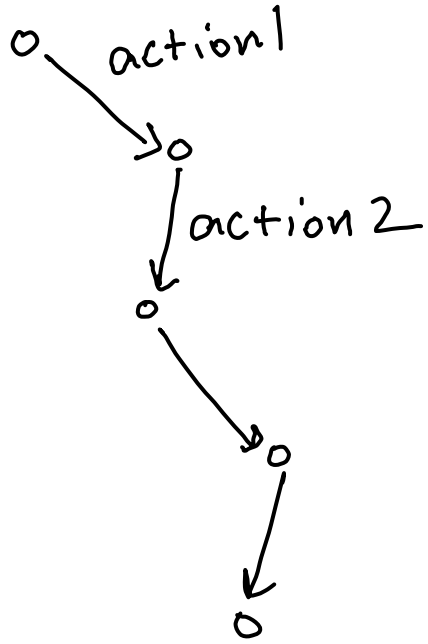
خروجی: یک دنباله از اقدامات:

seq = [action1, action2, ----]

execution phase.

$action \leftarrow \text{FIRST}(seq)$
$seq \leftarrow \text{REST}(seq)$
return $action$

فاز اجرا :
دنباله اقدامات به دست آمده را اجرا می‌کند



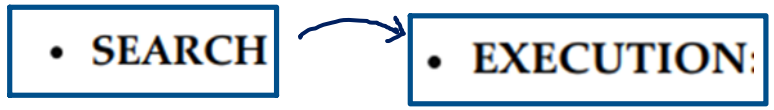
جستجو و اجرا به دو شکل می‌توانند استفاده شوند:

- سیکل باز : Open-loop
- سیکل بسته : Closed-loop

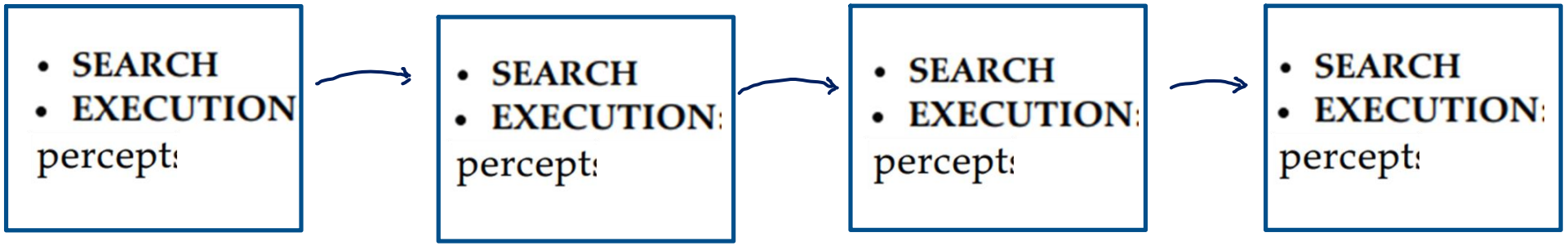
در محیط‌های مشاهده‌پذیر قطعی و ایستا، عامل می‌تواند اینبار دنباله اقدامات بهینه را جستجو کند سپس شروع به اجرای آن اقدامات کند. (سیکل باز)

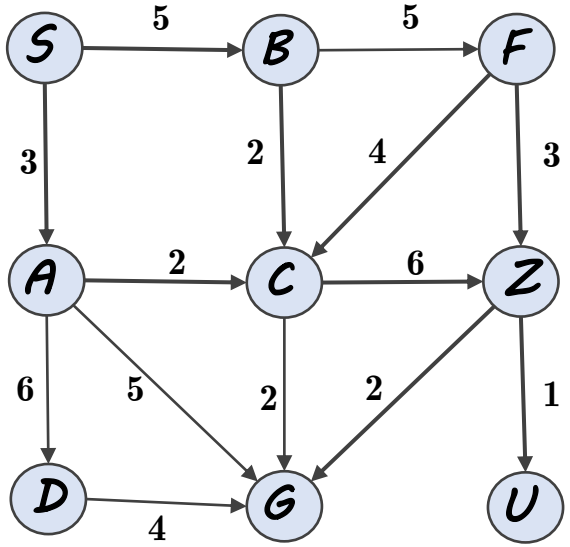
اما اگر محیط پیچیده‌تر، پویا، غیرقطعی، ... باشد باید پس از اجرای هر اقدام دوباره درک از محیط را گرفته و جستجو برای اقدام بعدی را تکرار کنیم. (سیکل بسته)

Open-loop ابتدا دنباله‌ی اقدامات بهینه جستجو می‌شود. سپس آن دنباله اجرا می‌شود

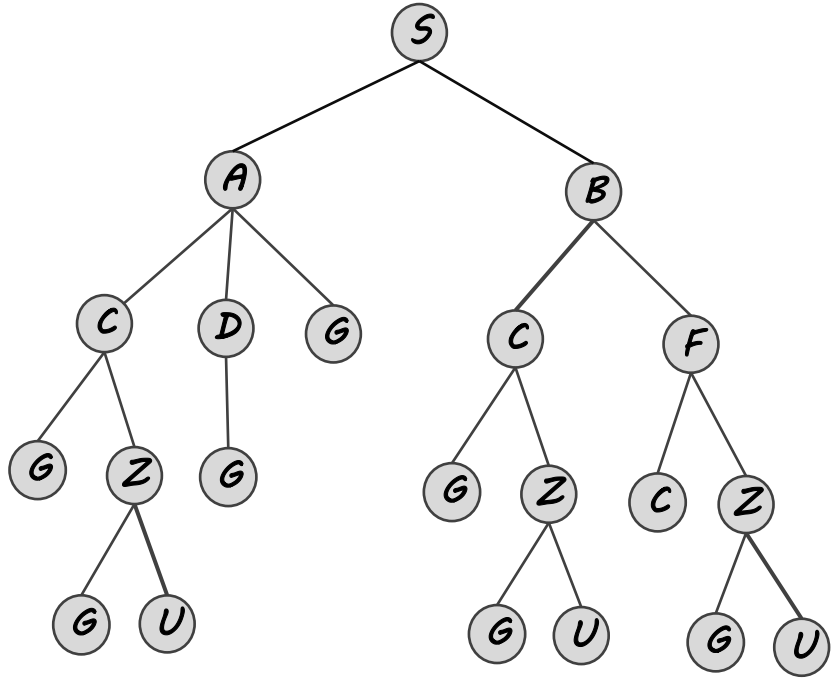


Closed-loop گام به گام، جستجوی اقدام بهینه، اجرای آن، ادراک محیط، دوباره حرکت از نو





گراف فضای حالات

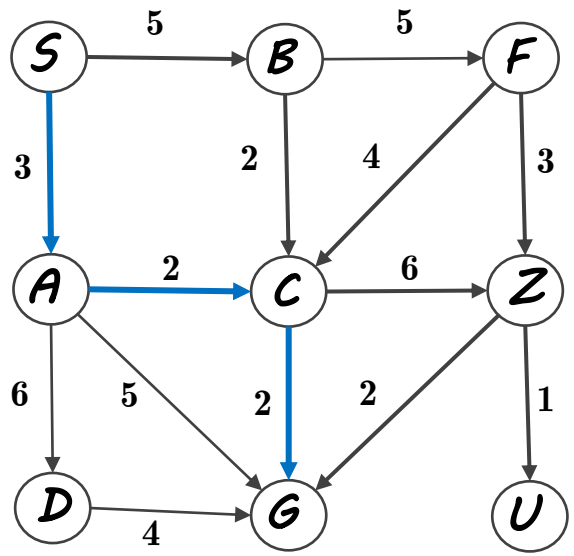


درخت جستجو

هر حالت از گراف فضای حالات ممکن است چندین گره در درخت جستجو داشته باشد. مثلاً گره C دوبار در درخت ظاهر شده چون مسیرهای S-B-C و S-A-C به آن وجود دارد.

اگر یال‌های گراف مثلاً یال S→A جهت‌دار نبود گره S ممکن بود دوباره هم تولید شود.

اولیه



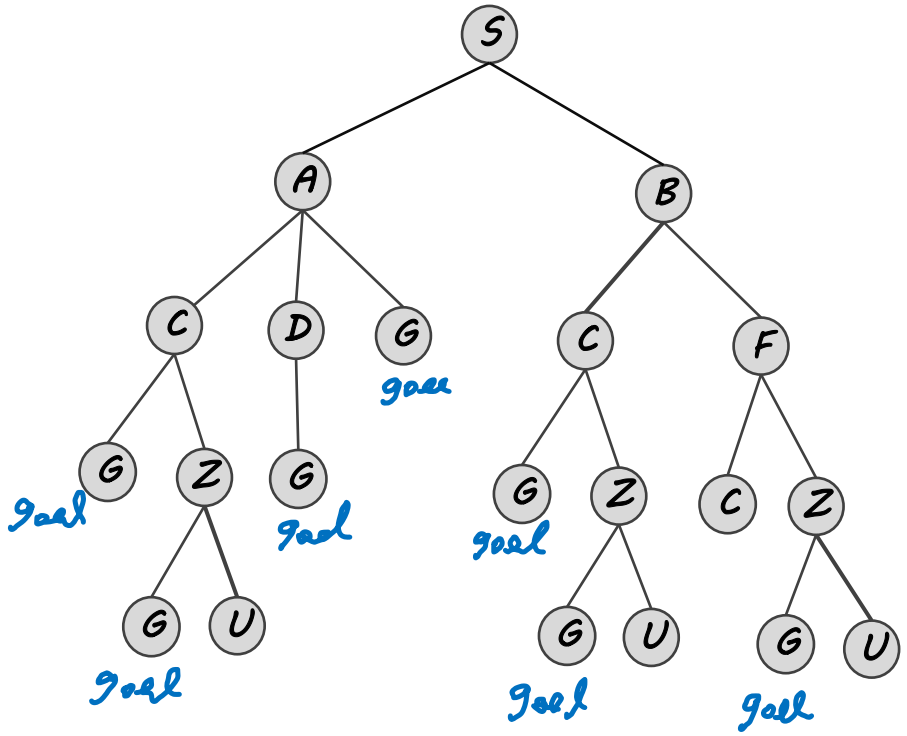
هرف

دو مسیر تا هدف، یک دنباله اقدامات است.

هزینه‌ی مسیر بهینه c^* نامیده می‌شود.

$$c^* = ۳+۲+۲ = ۷$$

چند اصطلاح:

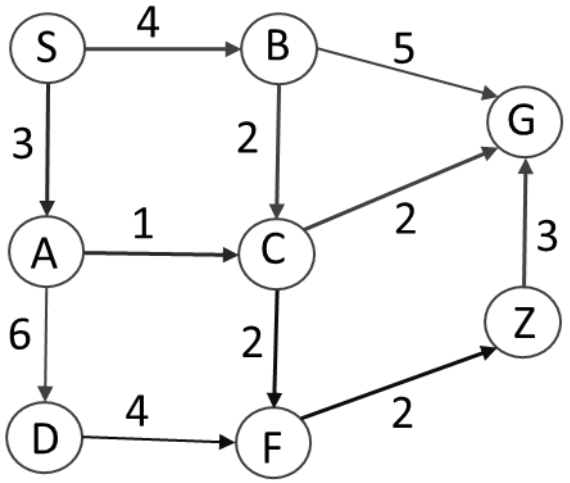


b : فاکتور انشعاب؛ بیشترین تعداد فرزندان گره‌ای در شکل مقابل $b=3$ است.

d : عمق هدف؛ عمقِ نزدیک‌ترین هدف به لحاظ تعداد یال‌ها در مثال مقابل $d=2$ است.

h یا m : ارتفاع درخت جستجو، (حداکثر عمق در درخت) در شکل مقابل $m=4$ است.

Breadth-first search



مثالی از یک گراف فضای حالات:

در این مثال: S وضعیت اولیه است. initial state

G هدف است. goal state

در این مثال: اقدامات برگشت‌پذیر هستند (یال‌ها جهت دارند)

هزینه اقدامات کنار آن‌ها نوشته شده

در این مثال: مسیر بهینه SACG است با هزینه‌ی $C^* = 6$.

BFS search

BFS:

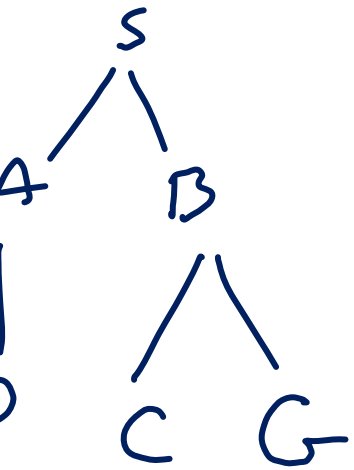
از صف استفاده می‌کند (FIFO)

آزمون هدف هنگام تولید گره انجام می‌شود.

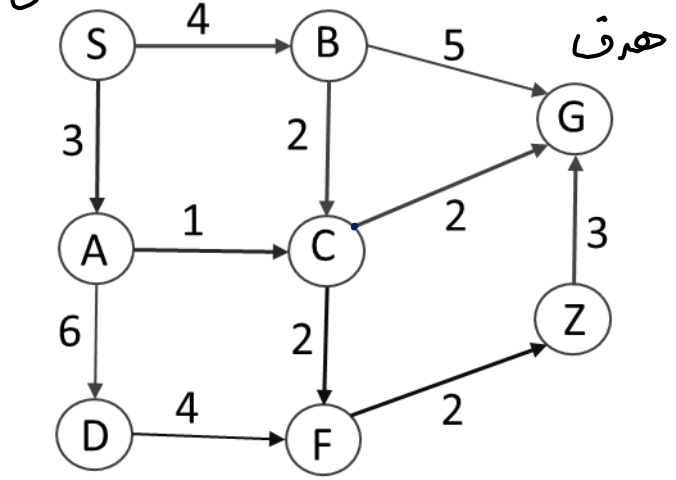
اگر هدف نباشد، گره رابطی می‌دهد و فرزندانش

تولید می‌شوند.

درخت BFS



اولی



هدف

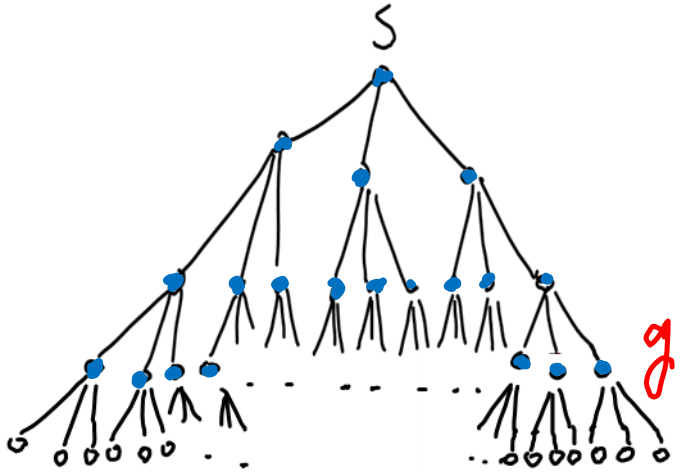
گسترش S ترتیب بطبق دادن گره‌ها:

S, A, B

به محض تولید G کار تمام می‌شود.

مسیر یافت شده S–B–G (لزوماً بهینه نیست BFS)

مرتبه‌ی زمانی و مرتبه‌ی فضایی BFS:

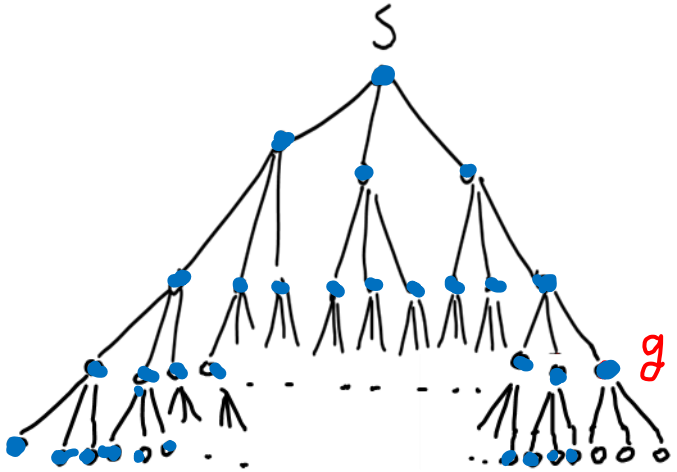


آزمون هدف
هنگام تولید گره

$$1+b+b^2+\cdots+b^d = \frac{b^{d+1}-1}{b-1} = O(b^d)$$

$(b>1)$

مرتبه‌ی زمانی و مرتبه‌ی فضایی BFS:

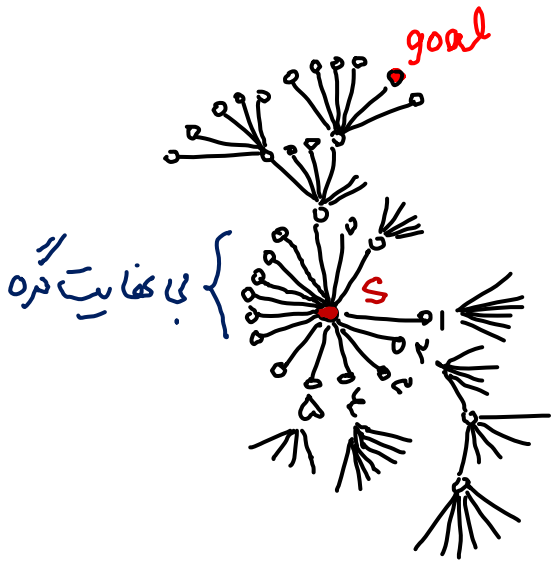


اگر آزمون هدف
هنگام ملاقات گره (خروج از صف) انجام شود:

$$1+b+b^2+\cdots+b^d+b^{d+1}-b = O(b^{d+1})$$

تولید گره‌ها تا عمق $d+1$ به جز فرزندان هدف

آیا BFS، کامل است؟ آیا BFS بهینه است؟



اگر $b = \infty$ باشد، BFS کامل نیست. یعنی لزوماً مسیر رسیدن به هدف را پیدا نمی‌کند.

اما اگر $b < \infty$ باشد، BFS کامل است.

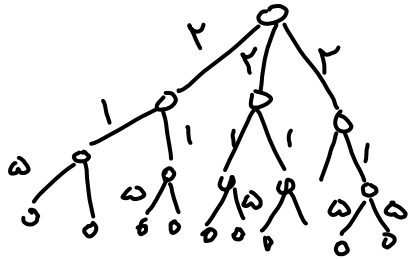
BFS در حالت کلی بهینه نیست.

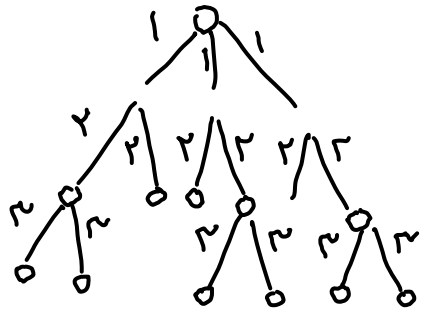
در حالات خاص بهینه است: اگر هزینه رسیدن به گره‌های با عمق یکسان، برابر باشد و هزینه‌ی رسیدن به گره‌های با عمق بیشتر، بیشتر شود، BFS (جستجوی اول سطح) بهینه است.



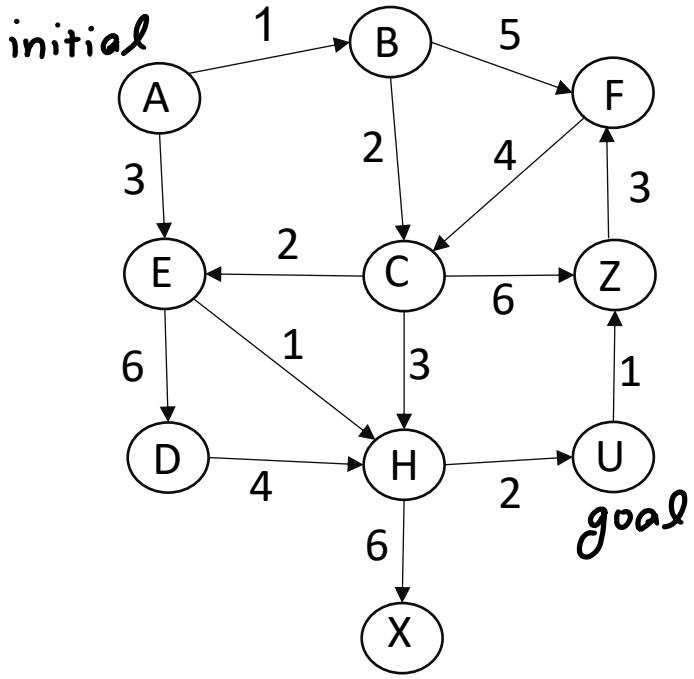
مثلاً وقتی هزینه‌ها همه مثبت و یکسان است.

یا در شکل‌های زیر:





DFS search



جستجوی عمقی DFS:

از پشته استفاده می‌کند (LIFO)

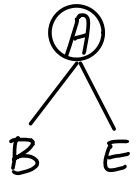
به هزینه اقدامات توجهی ندارد پس لزوماً بهینه نیست.

آزمون هدف: هنگام تولید

A

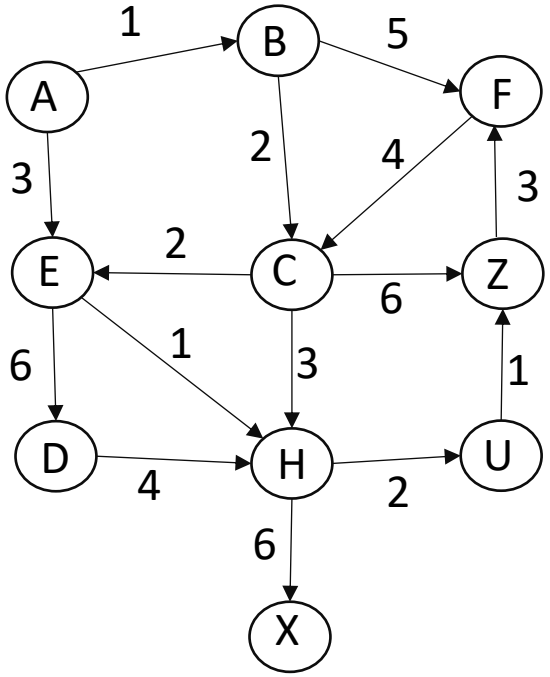


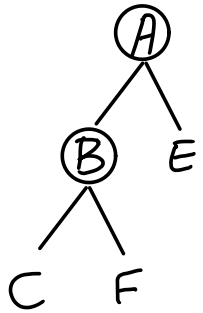
A خارج می‌شود (ملاقات) و بسط داده می‌شود (فرزندان تولید می‌شوند):



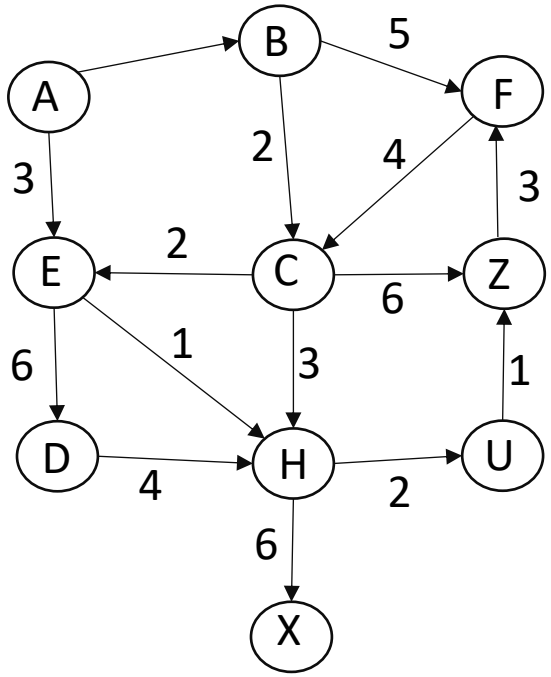


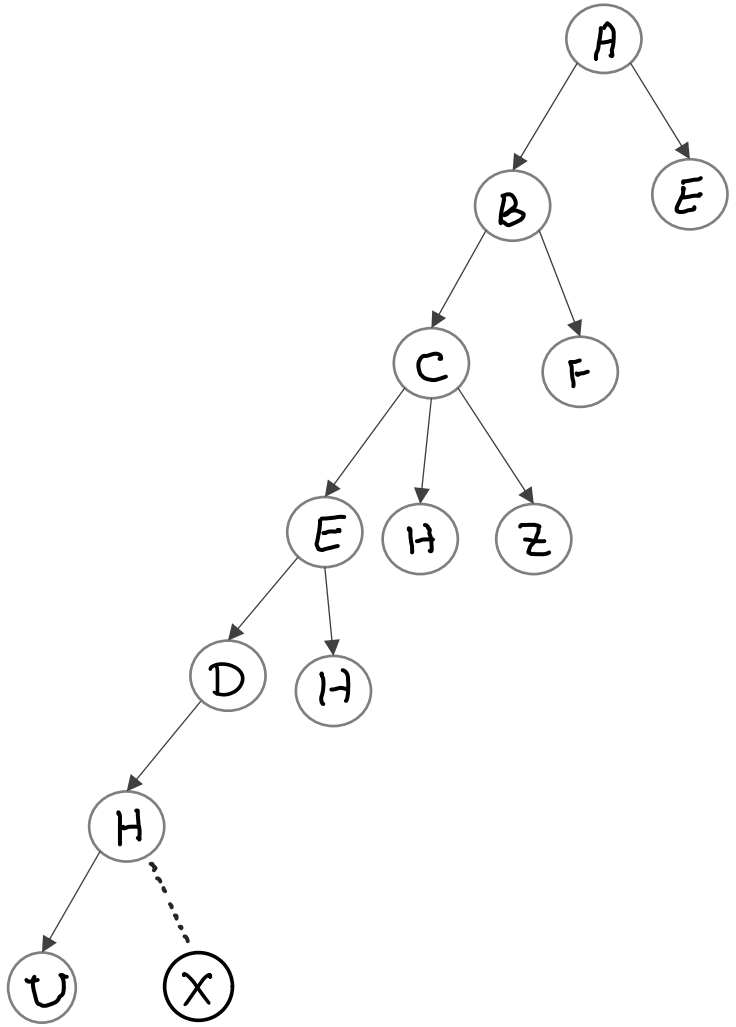
B خارج می‌شود (ملاقات) و بسط داده می‌شود (فرزندان تولید می‌شوند):


A
B
F
E
C
Z
D
H
U
X
1
5
3
2
4
3
2
6
6
1
3
1
4
2
6


A
B
E
C
F

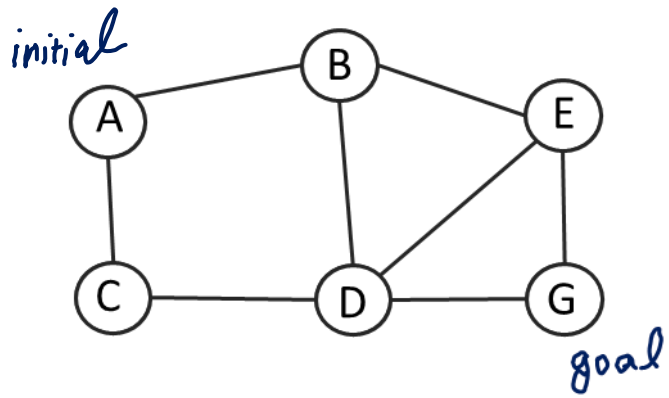

C
F
E





درخت بیانگر مختصر
توالیِ گره هدف
جستجو تمام است.

گرفتار شدن در حلقه بینهایت در جستجوی درختی:

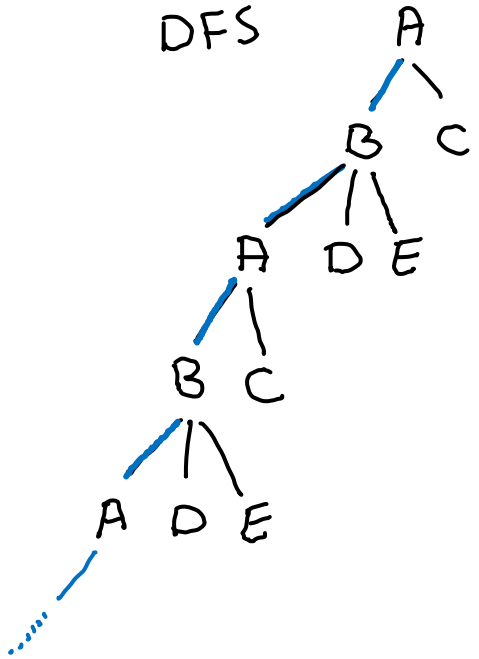


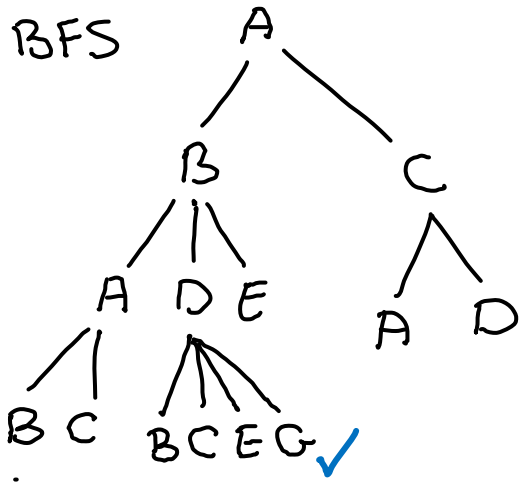
در این مثال اقدامات مکرری نیز وجود دارند که غیر هدف دارند.

جستجوی درختی از تولید گره‌های تکراری جلوگیری نمی‌کند.

می‌بینیم که جستجوی درختی DFS در loop قرار گرفته و کامل نیست

اما جستجوی درختی BFS در loop گرفتار نمی‌شود.



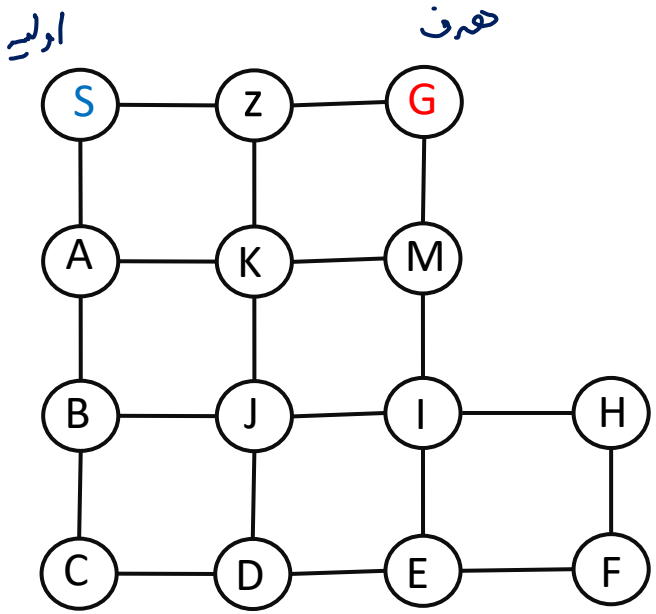


مرتبه زمانی DFS:

حتی اگر در حین loop قرار نگیرد، ممکن است برای رسیدن به هدف تمام گره‌های موجود را تولید کند (مثلاً جستجوی گرافی DFS)

پس اگر درخت جستجو دارای عمق m و فاکتور انشعاب b باشد،

مرتبه زمانی $O(b^m)$ است.



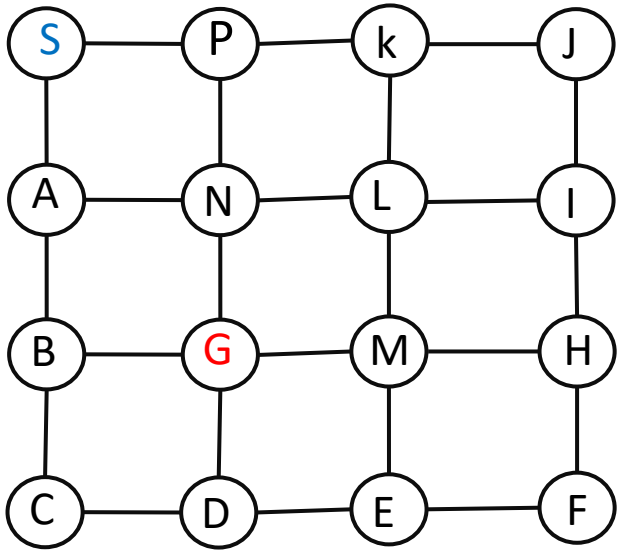
برای مثال در شکل بالا اگر DFS استفاده کنیم و ترتیب الفبایی رعایت جستجوی DFS به این صورت می‌رسد:

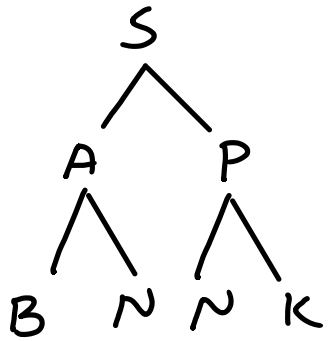
S A B C D E F H I J K M G

Depht -Limited search

برای جلوگیری از قرارگرفتن در loop جستجوی عمقی را با محدودیت عمق l اجرا می‌کنیم

در شکل مقابل حاصل DLS با محدودیت $l=2$ این است (جستجوی گذرنی):





این جستجو به هدف دست نیافت چون هدف در عمق

$d=3$ قرار دارد. و $l<d$ است.

Depth-limited search

جستجوی عمق با محدودیت عمق :

در حلقه‌ی بی‌نهایت تکرار نمی‌افتد.

$l \geq d$ ← کامل غیربهینه.

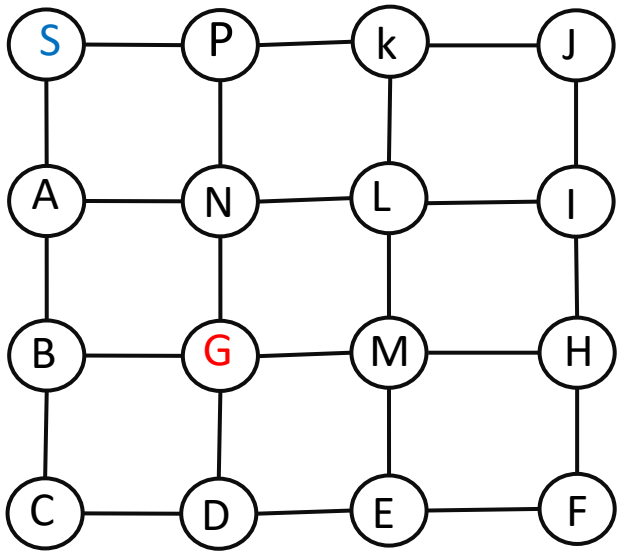
$l < d$ ← غیرکامل

$l = d$ و هزینه‌ها یکسان $c > 0$ ← بهینه و کامل

IDS:Iterative deepening depth first search

از آنجایی که نمی‌دانیم هدف در چه عمقی قرار دارد بهتر است DLS را با محدودیت عمق

$l=0, 1, 2, \ldots$ آنقدر تکرار کنیم تا $l=d$ شود.

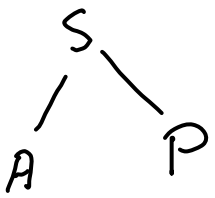


اجرای IDS در این مثال:

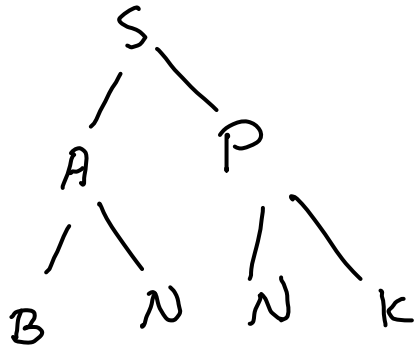
$\ell = 0$

S

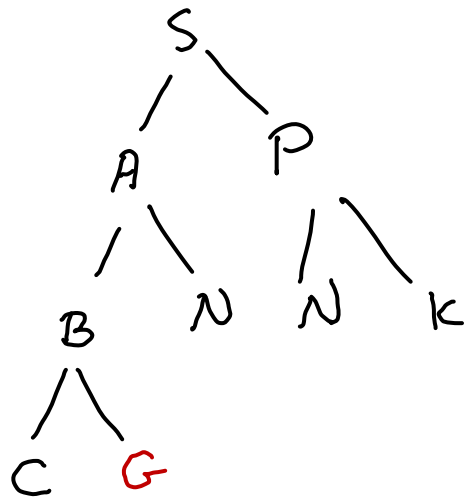
$\ell = 1$



$\ell = 2$

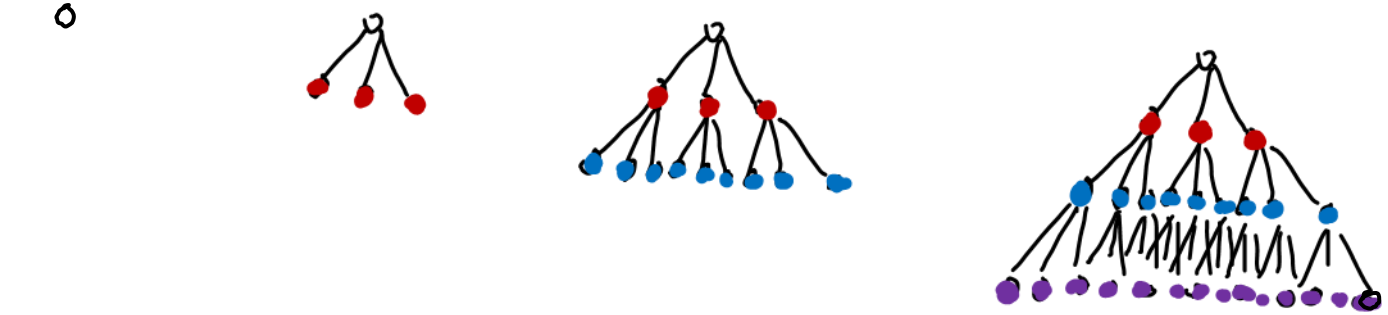


$\ell = 3$



حداکثر تعداد گره‌های تولید شده در IDS : (مرتبه زمانی)

$l=0$



در عمق یک ، b گره که $d+1$ بار تولید می‌شوند

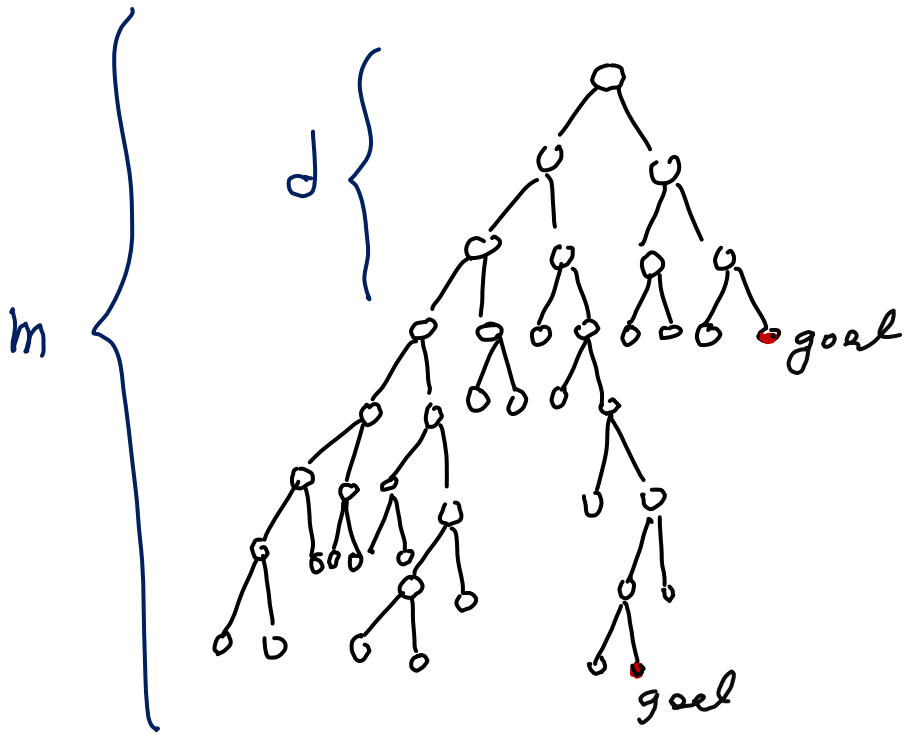
در عمق دو : b^2 گره که d بار تولید می‌شوند،

⋮

در عمق d : b^d گره که 1 بار تولید می‌شوند.

$$N(\text{IDS}) = (d)b + (d-1)b^2 + \cdots + (1)b^d = O(b^d)$$

مقایسه روش‌های سطحی و عمق:



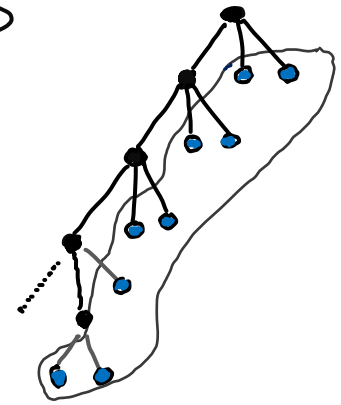
مرتبه زمانی:

$O(b^d)$: BFS

$O(b^m)$: DFS

مرتبه فضای (فضای موردنیاز حافظه برای لیست باز) نقطه قوت جستجوی عمق خطی بودن فضای موردنیاز است.

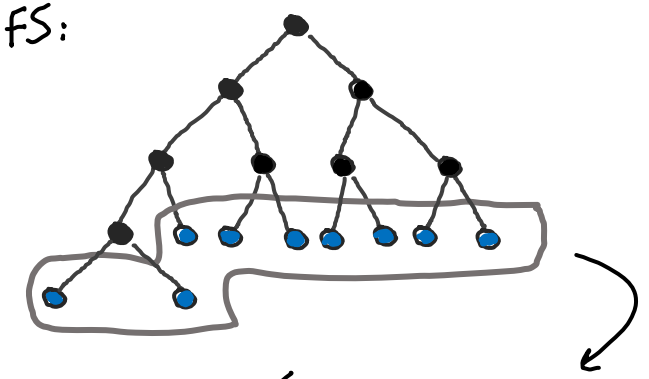
DFS



تعداد لیست باز (درون پشته)
حداکثر m گره که هرکدام b فرزند دارند:

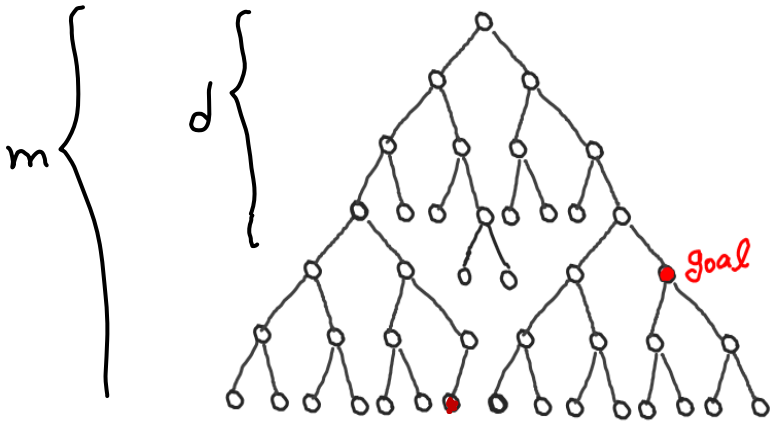
$O(bm)$

BFS:



لیست باز (صف) حداکثر به اندازه‌ی $O(b^d)$.

جستجوی کورکورانه



	زمان:	فضا:
BFS	$O(b^d)$	$O(b^d)$
DFS	$O(b^m)$	$O(bm)$
DLS	$O(b^l)$	$O(bl)$
IDS	$O(b^d)$	$O(bd)$

*توجه: از نظر مجانبی:

فرقی بین $O(b^l)$ و $O(b^{l+1})$ نیست.

Bidirectional search

جستجوی دوسویه

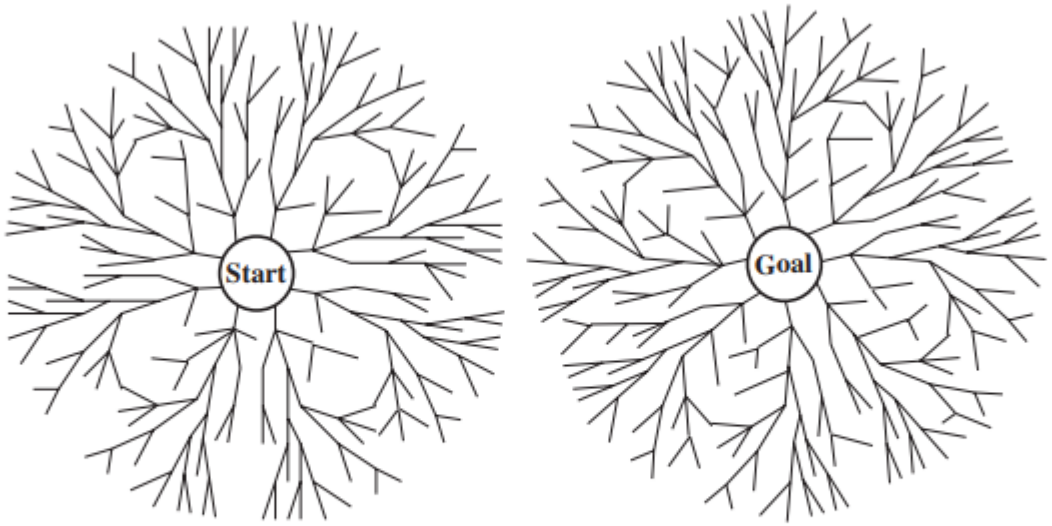


Figure 3.20 A schematic view of a bidirectional search that is about to succeed when a branch from the start node meets a branch from the goal node.

اگر از هر دو طرف به صورت BFS اجرا شود و $b<\infty$ باشد کامل است. (هزینه‌ها مثبت)

و زمان اجرای آن نیز

$$\text{زمان} = O(b^{\frac{d}{2}} + b^{\frac{d}{2}}) = O(2b^{\frac{d}{2}}) = O(b^{\frac{d}{2}})$$

جستجوی اول بهترین: (Best First)

جستجوی BFS و DFS، براساس عمق کمتر یا عمق بیشتر اولویت برای ملاقات گره‌ی (خروج از لیست باز) داشتند.

اما جستجوهای زیر که همگی از خانواده‌ی (اول بهترین) هستند براساس $f(n)$ کمتر، اولویت دهی می‌کنند.

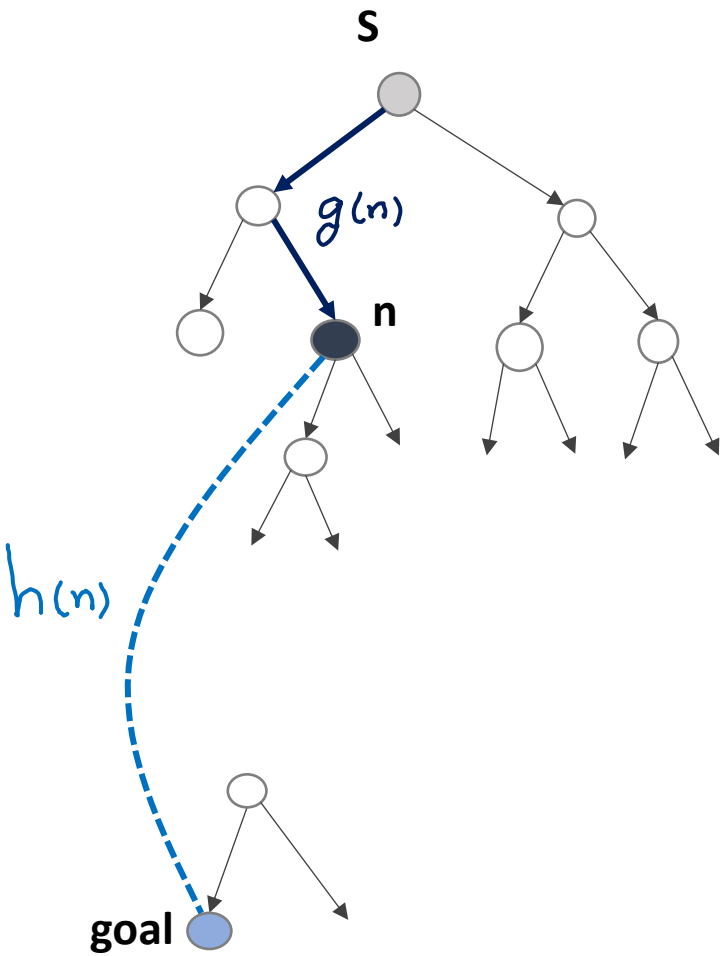
البته هرکدام $f(n)$ را به شیوه‌ی خاص خود محاسبه می‌کنند:

Greedy best first search

Uniform cost search

A*

در این الگوریتم‌ها اولویت ملاقات با گره‌ای است که کمترین $f(n)$ را دارد. البته تفاوت آنها در نحوه‌ی محاسبه $f(n)$ است.



$g(n)$ = هزینه مسیر از s تا n

تابع هیوریستیک (اکتشاف):

$h(n)$ = تخمین هزینه از n تا هدف

در جستجوی حریصانه GBFS : $f(n) = h(n)$

در جستجوی هزینه یکنواخت UCS : $f(n) = g(n)$

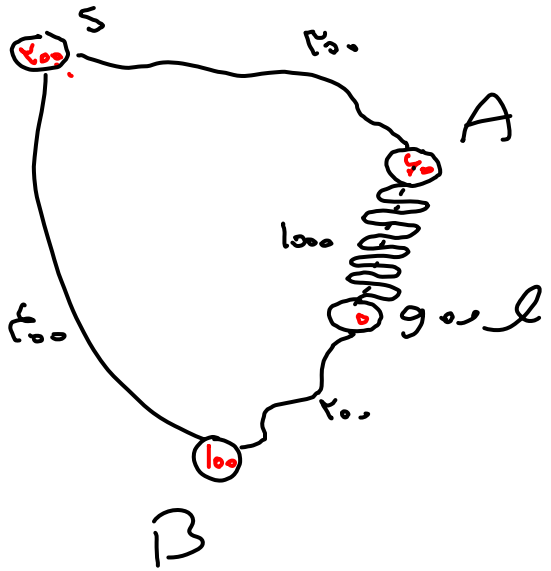
در جستجوی A^* : $f(n) = g(n) + h(n)$

جستجوی اول بهترین حریصانه :

GBfS

کامل نیست (ممکن است در حلقه قرار بگیرد)

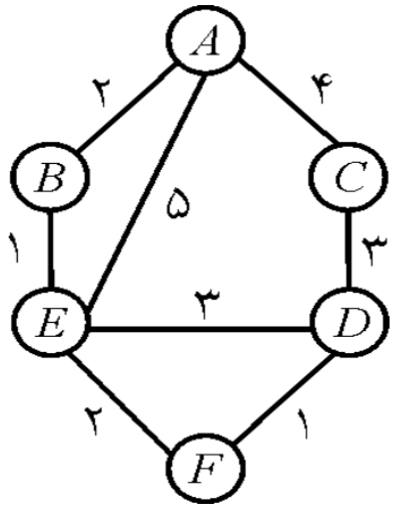
بهینه نیست (زیرا به هزینه‌ی $g(n)$ توجهی ندارد)



جست‌وجوهای آگاهانه آنهایی هستند که از یک تابع تخمین $h(n)$ استفاده می‌کنند.

آگاهانه	ناآگاهانه
GBFS	BFS
A*	DFS
IDA*	DLS
SMA*	IDS
	UCS

۲۲ـ گراف زیر را در نظر بگیرید. گره A وضعیت شروع و گره F وضعیت هدف را نشان می‌دهند. با استفاده از روش جست و جوی *Uniform Cost Search* ترتیب ملاقات گره‌ها به صورت کدام یک از موارد زیر است؟ عدد کنار هر یال هزینهٔ عبور از آن یال است. (مهندسی IT ـ دولتی ۹۵)



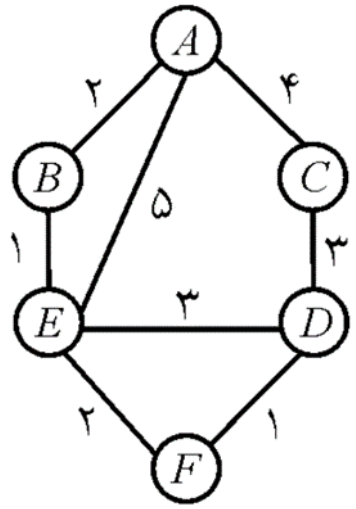
۱) A, B, E, C, F

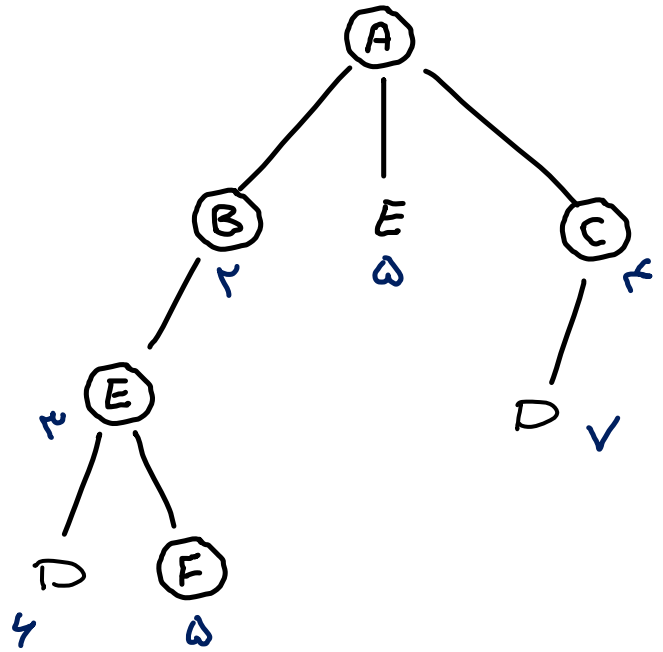
۲) A, B, E, D, F

۳) A, B, E, D, C, F

۴) A, B, E, C, D, F

UCS: $f(n) = g(n)$





ترتیب گسترش گره‌ها:

A, B, E, C, F

البته گره هدف F گسترش داده نمی‌شود فقط ملاقات می‌شود.

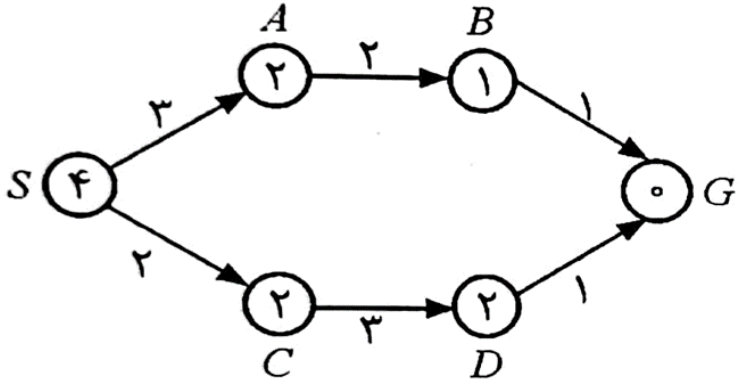
در جلسه بعد خواهیم دید که:

UCS
- کامل است (اگر $b < \infty$)
- بهینه است. (اگر هزینه اقدامات از یک عدد مثبت بزرگتر می‌باشد).

در UCS:

وقتی n برای بط انتخاب می‌شود، مسیر بهینه تا n به دست آمده است.

۷۵ـ در شکل زیر هزینه کنش‌ها روی یال‌ها و مقدار تابع ابتکاری روی گره‌ها نوشته شده است. اگر S گره شروع و G گره هدف باشد، کدام مورد درست است؟ (در شرایط یکسان برای دو گره از ترتیب الفبایی استفاده شود) (مهندسی کامپیوتر ـ دولتی ۹۷)



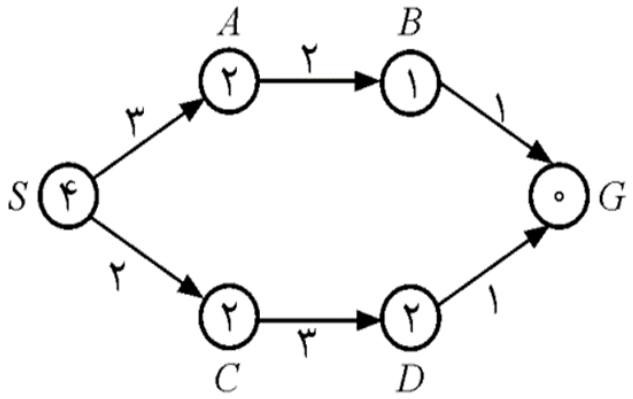
۱) تابع ابتکاری استفاده شده قابل قبول (*admissible*) است.

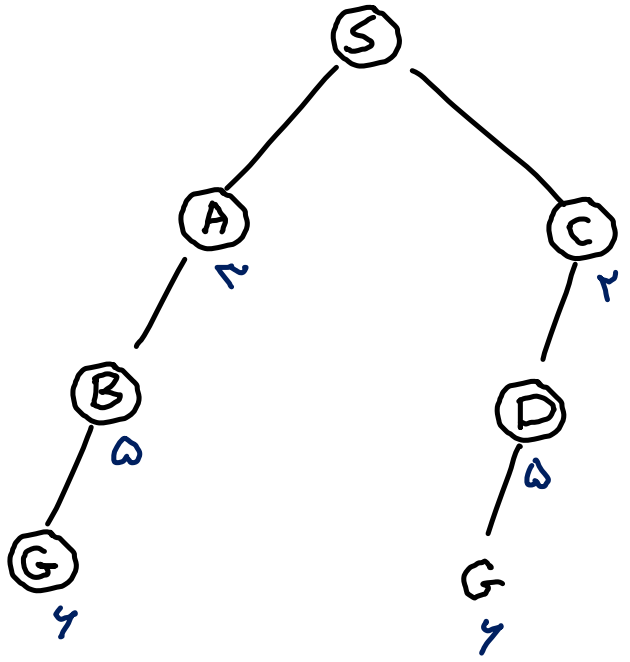
۲) ترتیب گسترش گره‌ها در الگوریتم A^*، از چپ به راست S,C,A,B,G است.

۳) ترتیب تولید گره‌ها در الگوریتم A^*، از چپ به راست S,A,C,B,D,G است.

۴) ترتیب تولید گره‌ها در الگوریتم UCS، از چپ به راست S,A,C,B,D,G است.

UCS: $f(n) = g(n)$



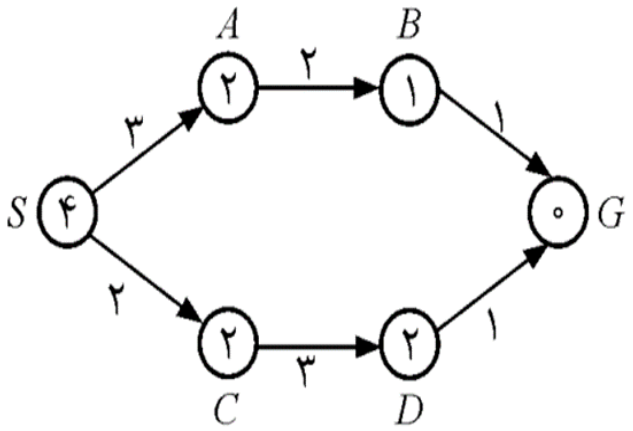


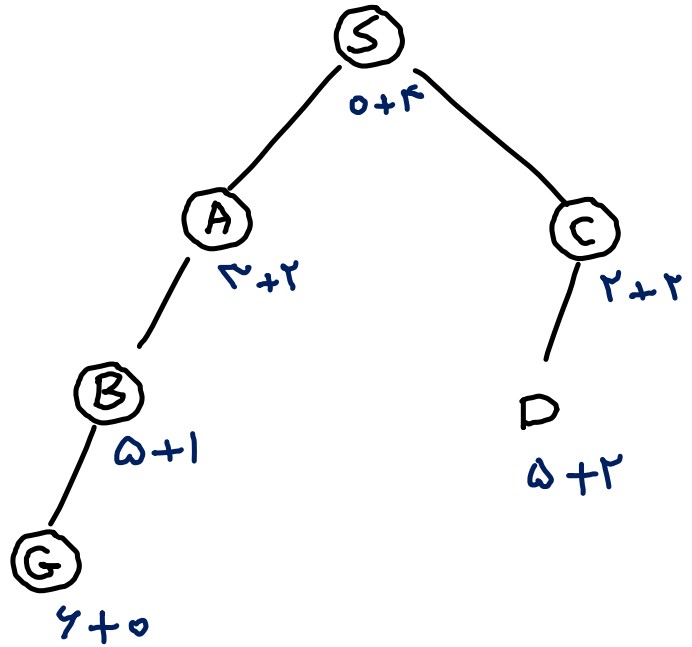
ترتیب گسترش گره‌ها:

S, C, A, B, D, G

البته گره هدف G گسترش داده نمی‌شود فقط ملاقات می‌شود.

A^* $f(n) = g(n) + h(n)$





ترتیب گسترش گره‌ها در A^*:

S , C , A , B , G

البته گره هدف G گسترش داده نمی‌شود فقط ملاقات می‌شود.

در حالت کلی می‌بینیم که در A^* اگر $f(n) > C^*$ باشد گره n بسط داده نمی‌شود.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

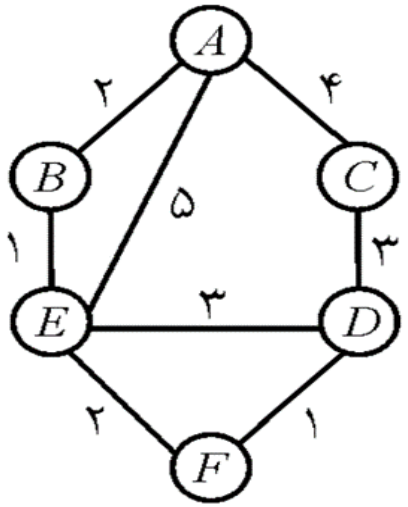
هوش مصنوعی

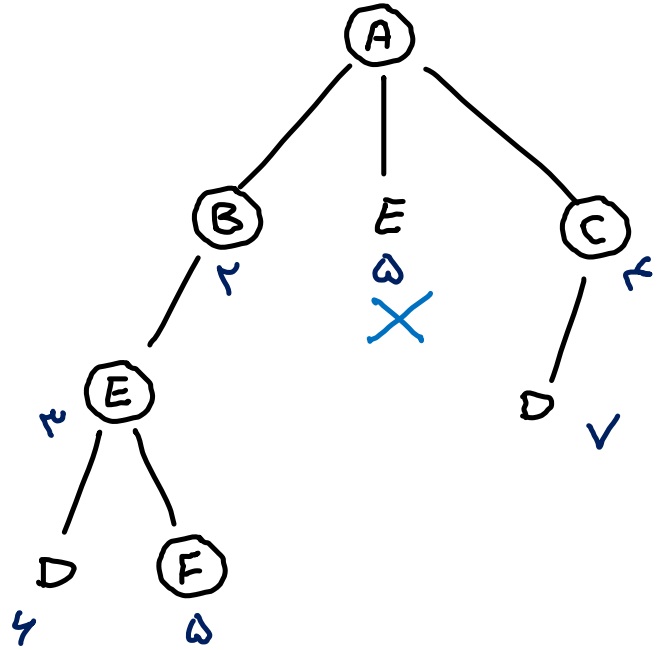
ابوالفضل گیلک
گروه بابان

جلسه ۳ فصل ۳

UCS $f(n) = g(n)$

یادآوری UCS





ترتیب گسترش گره‌ها:

A, B, E, C, F

گره هدف F گسترش داده نمی‌شود

فقط ملاقات می‌شود.

UCS

شرط کافی کامل بودن

$b < \infty$ و $\min(\text{cost}) \geq \varepsilon > 0$

شرط کافی بهینه بودن

بهینه است.

ویژگی خاص

هر گره‌ای که ملاقات می‌شود، مسیر بهینه تا آن گره پیدا شده است.

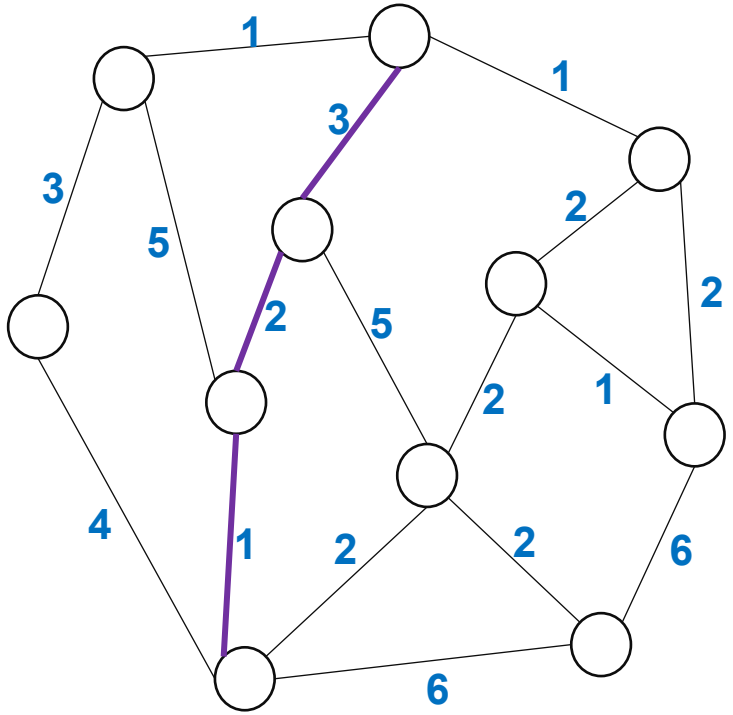
مرتبه زمانی :

مرتبه فضایی :

$O(b^{1+\lfloor \frac{C^*}{\varepsilon} \rfloor})$

زیرا حداکثر عمق هدف در این روش $d = \lfloor \frac{C^*}{\varepsilon} \rfloor$ است.

استفاده از این گراف فضای حالت برای تحلیل روش A^*:



مسیر بهینه از A تا هدف:

$c^* = ۶$

تحلیل روش A^* :

شرط کامل بودن A^* مانند UCS است اما در مورد بهینه بودن و زمان اجرای آن،

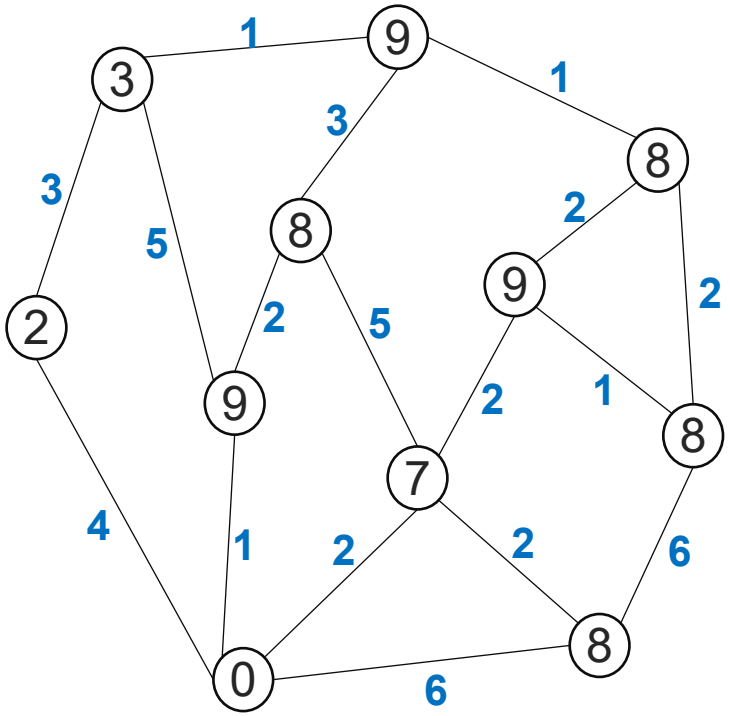
این کار وابسته به تابع هیوریستیک استفاده شده است.

در مثال زیر برای یک گراف، از ۳ تابع هیوریستیک استفاده کرده‌ایم.

h_1 : در برخی گره‌ها $h_1(n) > h^*(n)$ است. (بیشتر از هزینه‌ی بهینه است)

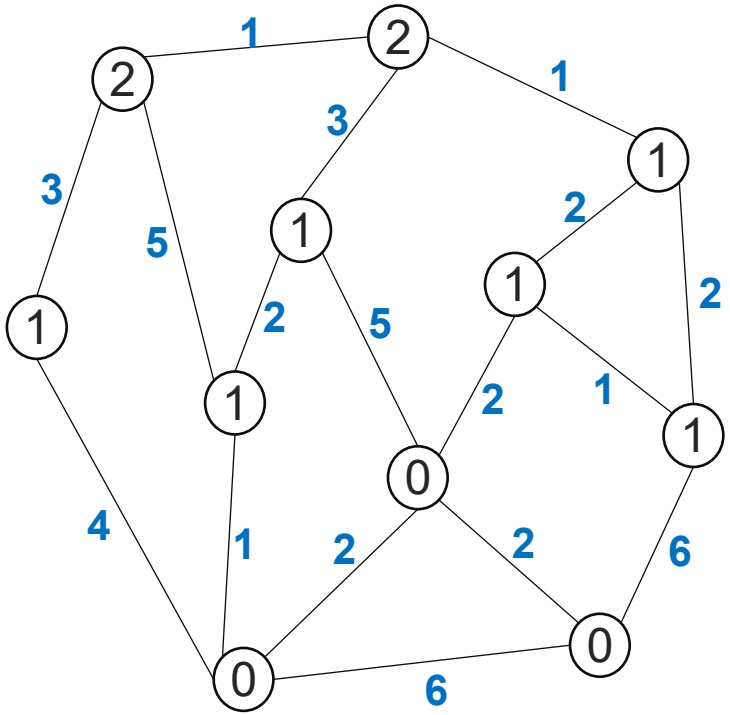
h_2 : همیشه $h_2(n) \leq h^*(n)$ اما بسیار کمتر است. (خیلی نادقیق است)

h_3 : همیشه $h_3(n) \leq h^*(n)$ و تقریباً نزدیک به h^* است.

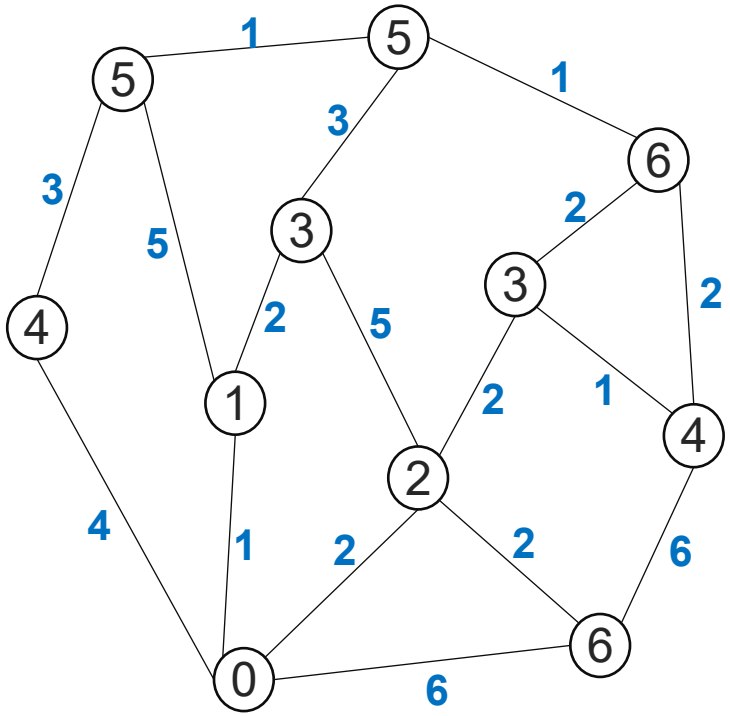


$h_1 > h^*$

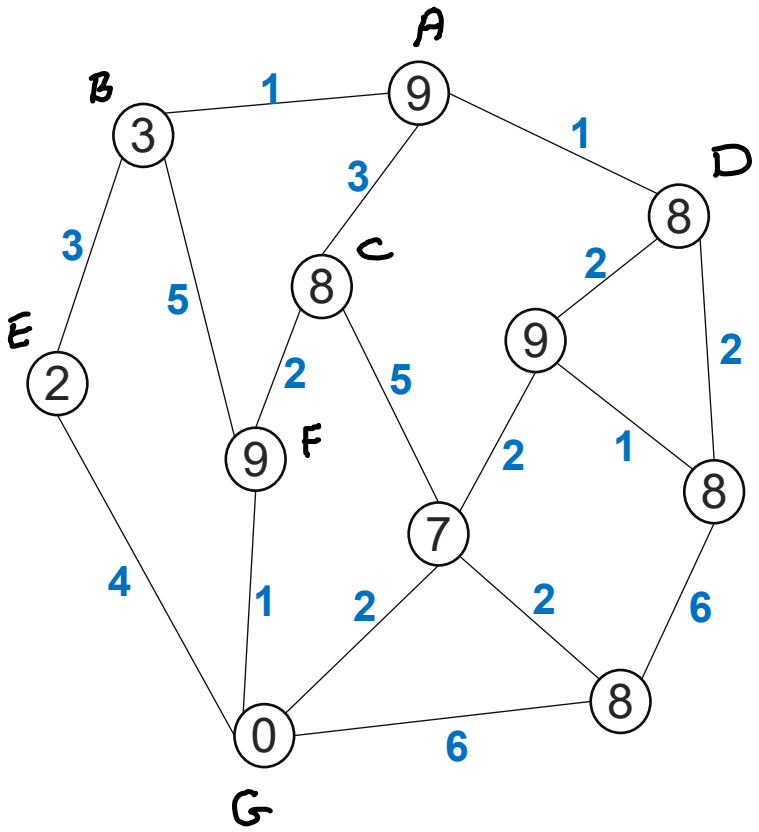
در برخی گره‌ها



$h_2 \ll h^*$

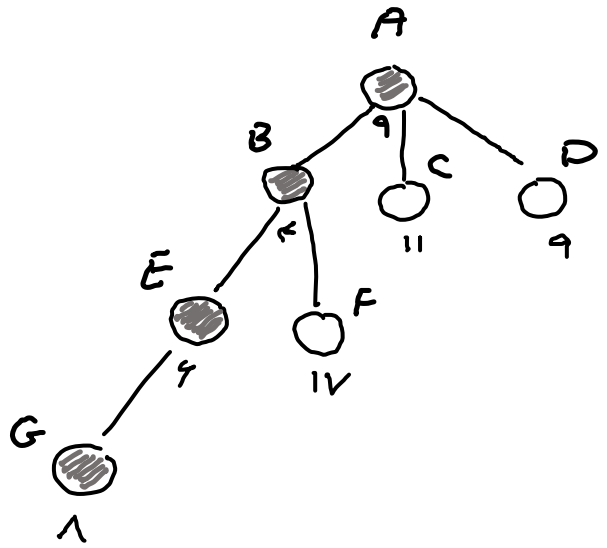


$h_3 \leq h^*$

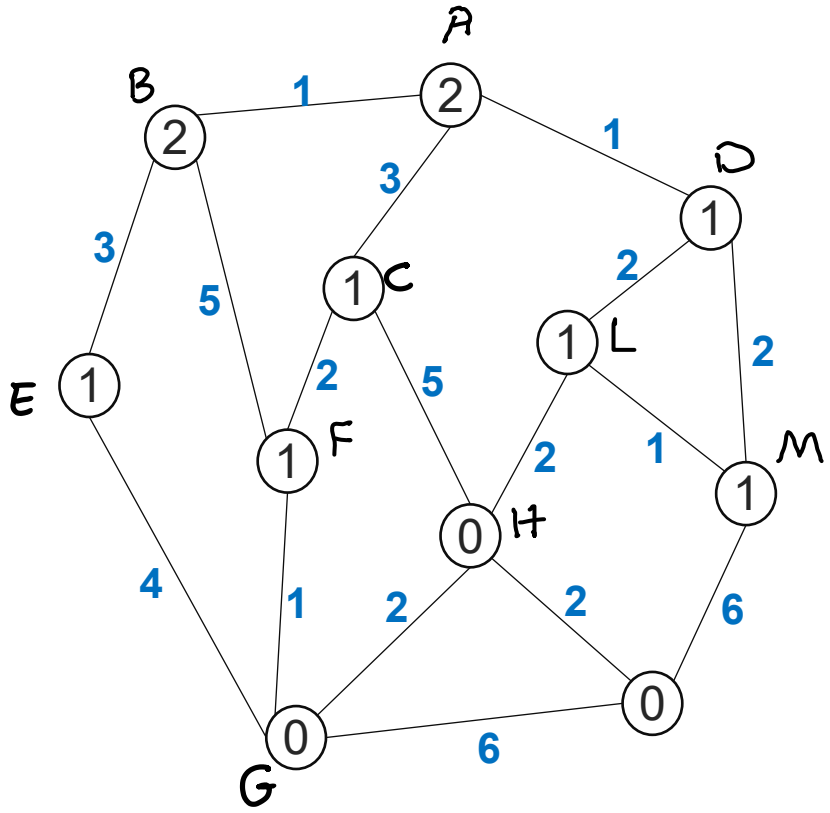


h_1

A^* با استفاده از h_1:

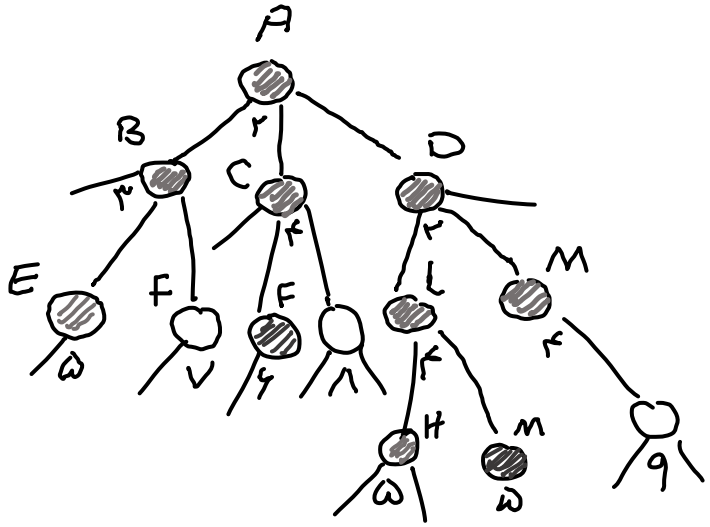


وقتی در برخی گره‌ها $h > h^*$ باشد، جستجوی A^* مسیری غیربهینه را ممکن است پیدا کند.



$h_2 \ll h^*$

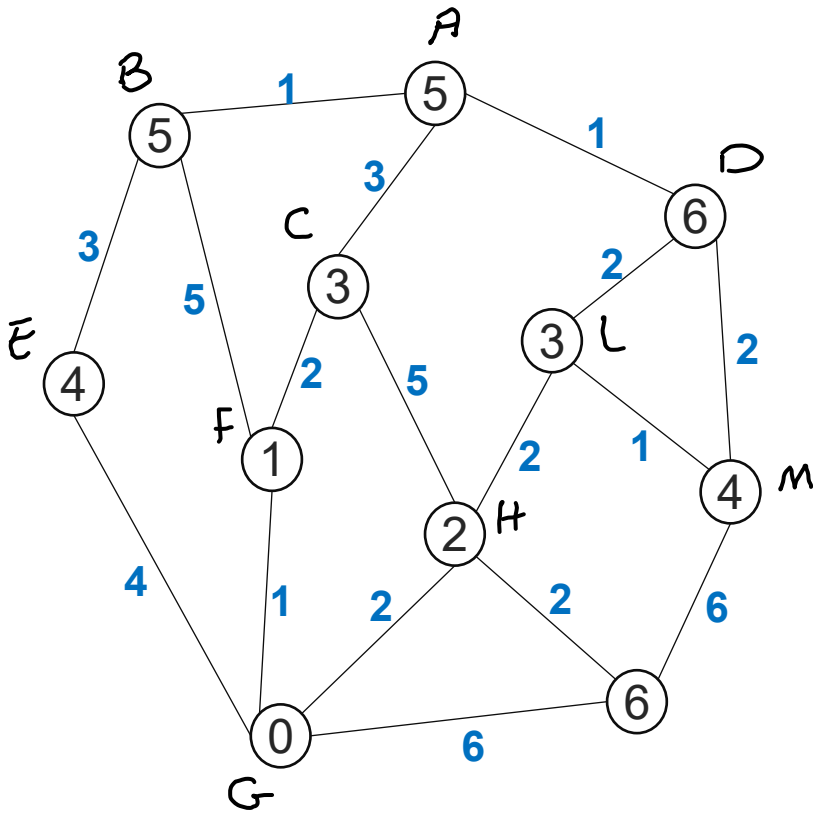
A^* با استفاده از h_2:

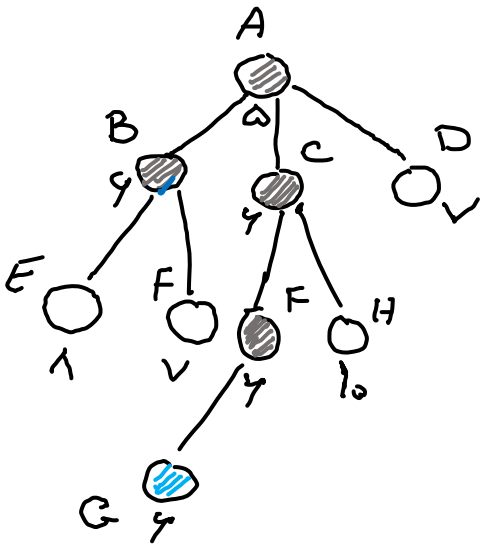


(درخت را کامل نکرده‌ایم)

وقتی $h < h^*$ باشد اما خیلی کمتر از h^* را نشان بدهد، جستجوی A^* درنهایت مسیر بهینه را خواهد یافت (جستجوی درختی) اما این کار با بطی دادن تعداد زیادی از گره‌ها تمام می‌شود. در واقع بیشتر شبیه UCS می‌شود.

اجرای A^* با استفاده از h_3 : (جستجوی گراف A^*)





(ما جستجوی گراف را انجام
داده ایم شما می توانید
جستجوی درختی را هم امتحان کنید)

$h_3 \leq h^*$ h_3 هم از h^* کمتر مساوی است اما به آن نزدیکتر است.

وقتی $h \leq h^*$ اما نزدیک به h^* باشد، جستجوی A^* با بسط دادن تعداد کمی از گره ها مسیر بهینه را پیدا می کند.

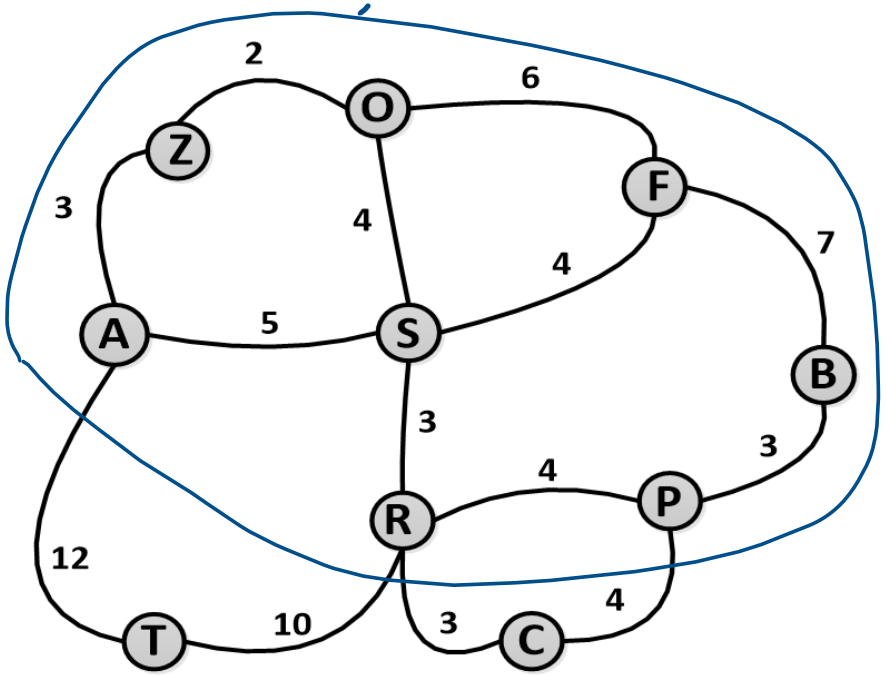
اگر به مثال آخر دقت کنید متوجه می‌شوید که A^* گره‌ای را که در آن $f(n) > c^*$ است، بط نزده.

$f(n) < c^*$ حتماً بط داده می‌شود

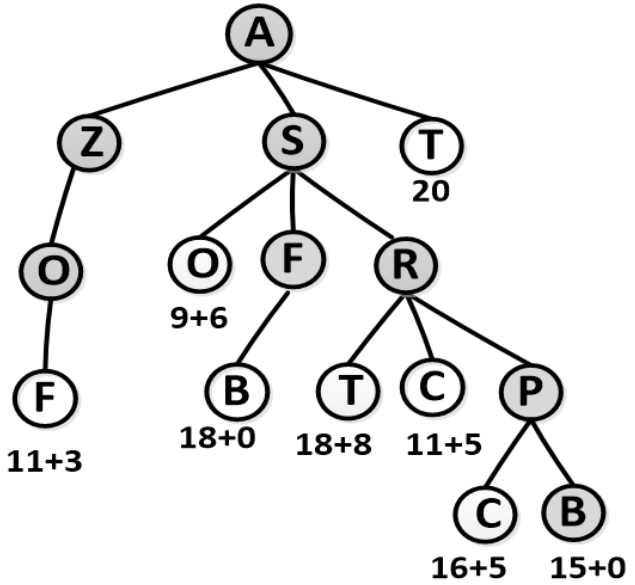
$f(n) = c^*$ ممکن است بط داده شود.

$f(n) > c^*$ بط داده نمی‌شود. (البته ممکن است تولید شود)

$f(n) \leq C^*$ کانتور



n	h(n)
A	7
Z	8
T	8
O	6
S	4
R	5
F	3
P	2
C	5
B	0



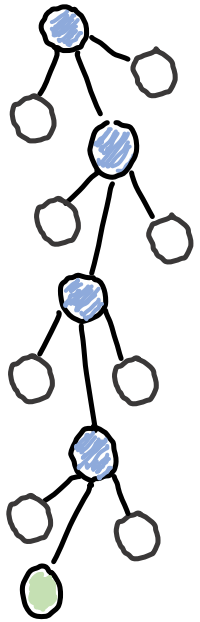
$C^* = 15$

در حالت خاص اگر h قابل قبول و به قدر کافی نزدیک به h^* باشد:

$$h(n) \leq h^*(n)$$

$$|h^*(n) - h(n)| \leq O(\log h^*(n))$$

مرتبه فضایی A^* و تعداد گره‌های تولید شده $O(bd)$ می‌شود.



هیوریستیک قابل قبول :

اگر برای هر گره n داشته باشیم $h(n) \leq h^*(n)$ گوییم h قابل قبول است.

شرط کامل بودن A^* مانند UCS است: $b<\infty$ و $\min(\text{cost}(\text{actions}))=\varepsilon>0$.

A^*: شرط کافی بهینه بودن A^* چنین است:

h ق.ق ← جستجوی درختی A^* بهینه است.

h سازگار ← جستجوی گرافی A^* بهینه است.

سازگار بودن یعنی چه؟ (در ادامه خواهیم گفت)

مثال هایی از توابع هیوریستیک قابل قبول:

در مساله مسیریابی نقشه:

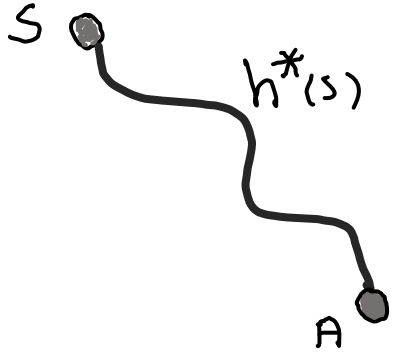
$h_1 \leq h^*$: فاصله مستقیم قابل قبول است

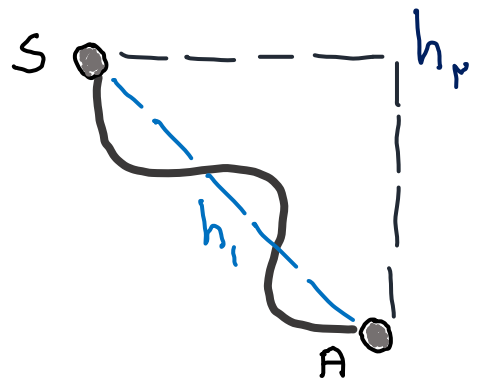
اما فاصله منهتن لزوما قابل قبول نیست.

$h^*(s)$ = کمترین فاصله n تا هدف روی نقشه

$h_1(s)$ = فاصله‌ی مستقیم (اقلیدسی)

$h_2(s)$ = فاصله‌ی شهری (منهتن)

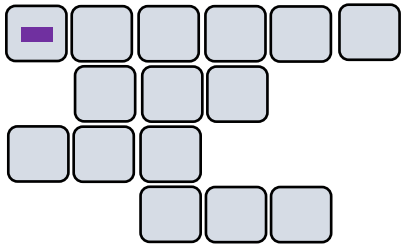




درستی‌ی جاروبرقی هوشمند:

$h^*(n)$ = تعداد کمترین اقدامات (کوتاهترین مسیر) برای رسیدن به هدف

$h(n)$ = تعداد آتای‌های کثیف



$h(n) \leq h^*(n)$ قابل قبول است.

درمسله‌ی معمای ۸ : 8-puzzle

$h^*(n)$: کمترین تعداد حرکات مورد نیاز برای رسیدن به هدف



حالت اولیه



حالت n

$h_1(n) = 3$ تعداد کاشی‌هایی که در جای خود نیستند misplaced

$h_2(n) = 1 + 3 + 2$ مجموع فواصل منهتن کاشی‌ها از جای هدف خود sum – Manhattan

$h_3(n) = 4$ تعداد حرکات لازم اگر بتوانیم هر کاشی را برداشته و در جای خالی قرار دهیم. Gaschnig

چند نکته که در کلاس گفته شد:

① روش ساخت هیوریستیک قابل قبول، ساده‌سازی مسأله است ← اجازه اقدامات جدید
← کاهش محدودیت‌ها

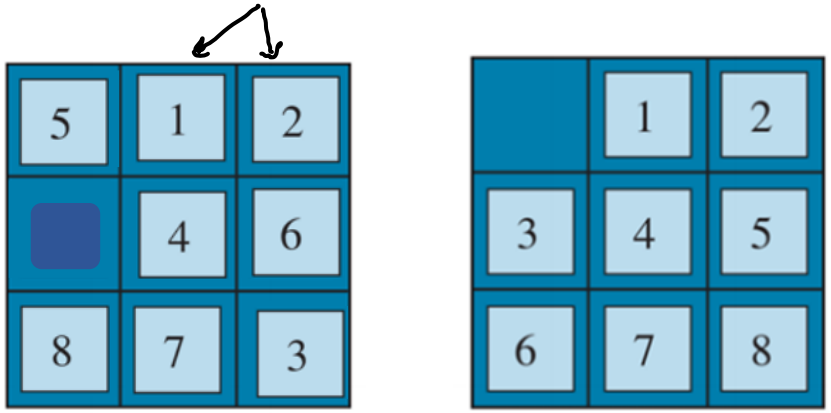
جواب مسأله‌ی ساده شده، یک هیوریستیک قابل قبول برای مسأله اصلی است.

② در مثال‌های 8:

$$h_1 \leq h_2 \leq h^*$$
$$h_1 \leq h_3 \leq h^*$$

مثال [شکل درست رسم شده است]

در کلاس، شکل با محاسبات همخوانی نداشت.



(۵ و ۶ و ۸ و ۳)

$$h_1(n) = 4$$

$$h_2(n) = 3 + 3 + 2 + 3$$

$$h_3(n) = 4$$



n



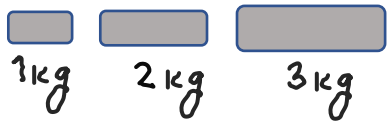
goal

این مثال نشان می دهد که h_2 از h_1 بهتر است.

تعداد نابجاها $h_1(n) = 2$

مجموع خواص $h_2(n) = 1 + 1$

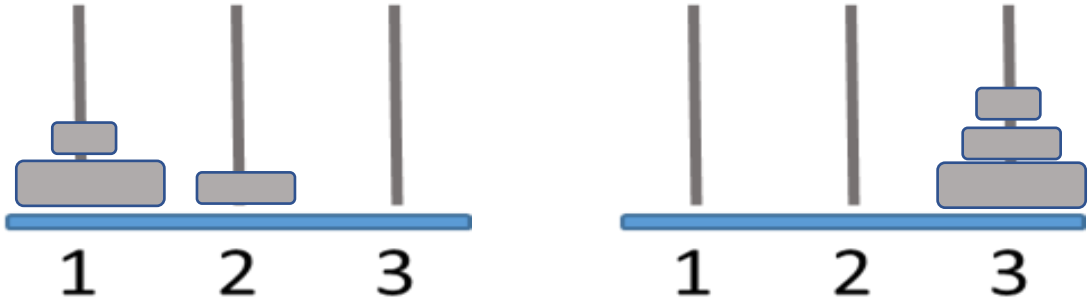
گاشنیگ $h_3(n) = 3$



یک قبل: هزینه انتقال وزنه‌ی با وزن w:

w = هزینه انتقال ساده

۳w = هزینه انتقال دوبل



یک هیوریستیک قابل قبول:

مجموع وزن روی میله‌های اول و دوم را S_1 و S_2 بنامیم.

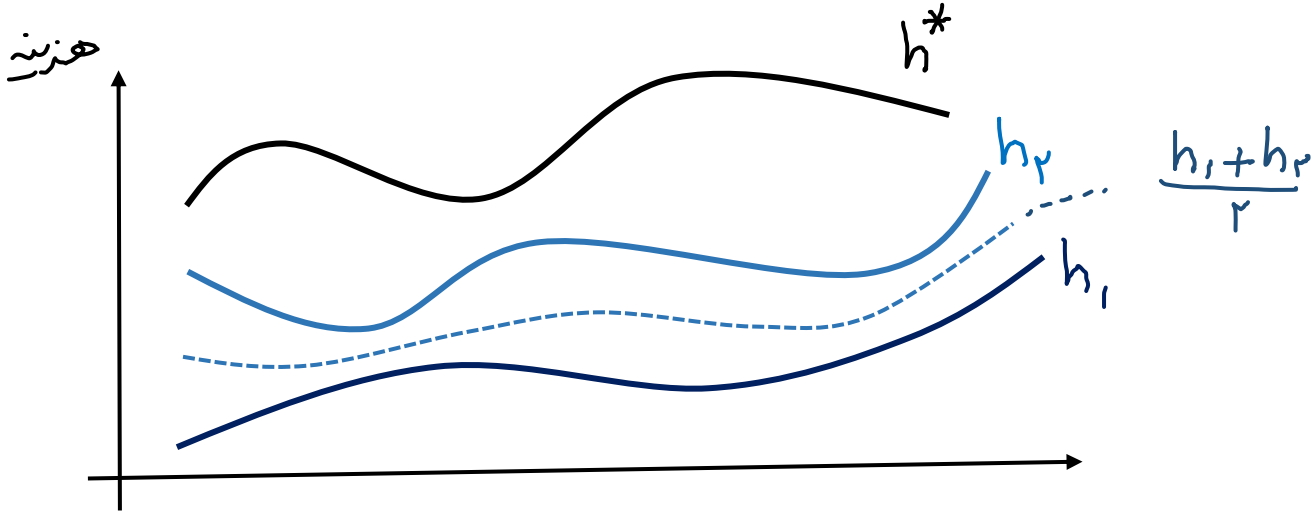
برای انتقال آنها به میله هدف؛ اگر از حرکات ساده استفاده کنیم که کم‌هزینه‌ترین است،

حداقل $2S_1 + S_2$ هزینه لازم است پس:

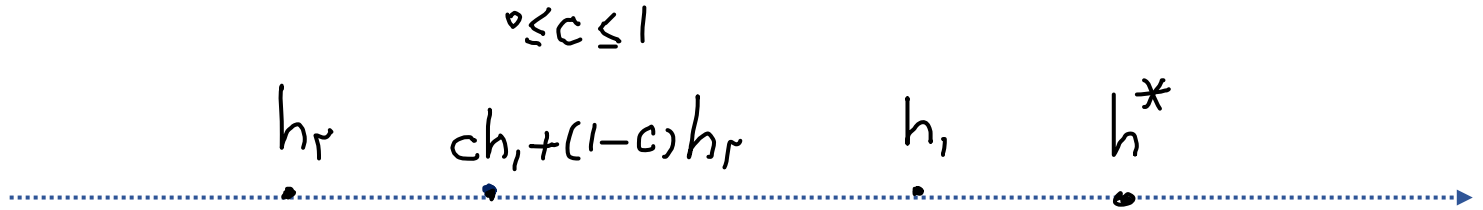
$$h(n) = 2S_1 + S_2 \leq h^*(n)$$

توجه کنید: در اصل h_1 و h_2 و h^* توابع هستند ولی وقتی می‌گوییم $h_1 \le h_2 \le h^*$ یعنی نمودار آنها چنین است.

$\frac{h_1+h_2}{2}$ نمودارش در وسط h_1 و h_2 قرار دارد.



اما از نظر مقادیر عددی مقدار را فاصله گرفته n : مثلاً اگر $h_2 \le h_1 \le h^*$ باشد،



به ازای $c=\frac{1}{2}$ ، معدل (میانگین حسابی) به دست می‌آید

که در وسط h_1 و h_2 قرار دارد.

ترکیب‌های خطی محدب از h_1 و h_2 عبارتند از مقادیر: $ch_1 + (1-c)h_2$

که c یک عدد بین ۰ و ۱ است.

برای $c=0$ به h_2 می‌رسیم.

برای $c=1$ به h_1 می‌رسیم.



اگر h_1 و h_2 قابل قبول باشند

این‌ها هم قابل قبولند:

ترکیب خطی محدب

$$\min(h_1, h_2) \leq c_1 h_1 + c_2 h_2 \leq \max(h_1, h_2) \leq h^*$$

$$c_1 + c_2 = 1$$

$$0 \leq c_1, c_2 \leq 1$$

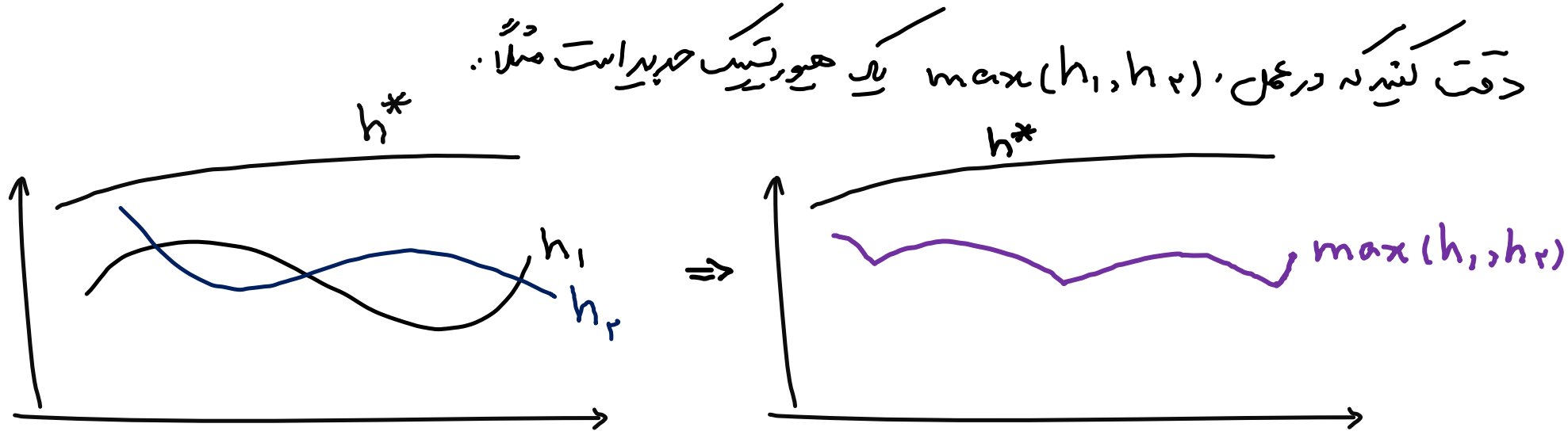
$$\min(h_1, h_2) \leq \sqrt{h_1 h_2} \leq \frac{h_1 + h_2}{2} \leq \max(h_1, h_2) \leq h^*$$

میانگین هندسی — میانگین حسابی

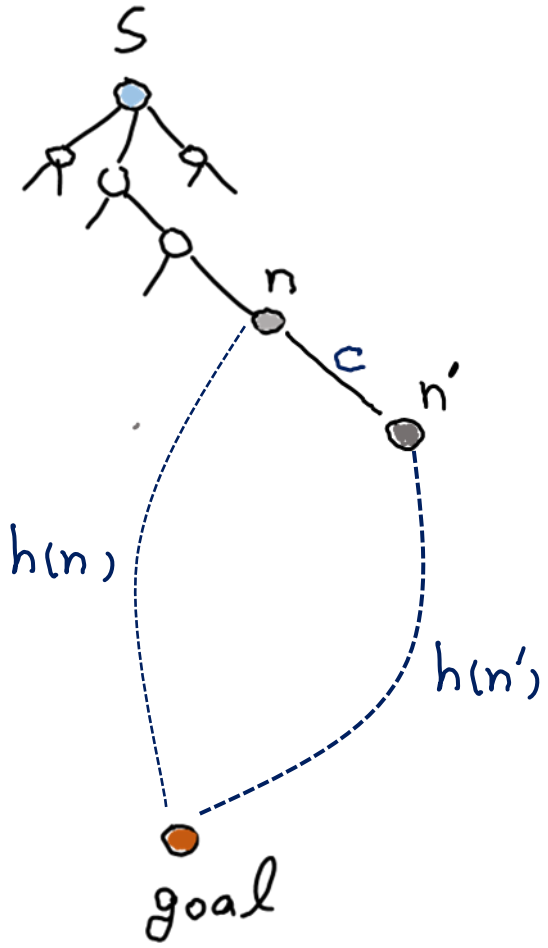
تکمیلی:

* در بین چند تابع هیوریستیک قابل قبول، هرکدام که بزرگتر باشد به h^* نزدیکتر و کارآمدتر است پس اگر $h_1, h_2, \dots, h_m$ قابل قبول باشند، $\max(h_1, h_2, \dots, h_m)$ بهترین هیوریستیک قابل قبول است.

دقت کنید که در عمل، $\max(h_1, h_2)$ یک هیوریستیک جدید است مثلاً:



سازگار بودن تابع هیوریستیک: (یکنوا بودن)



اگر در هر گره n و فرزندان n' داشته باشیم

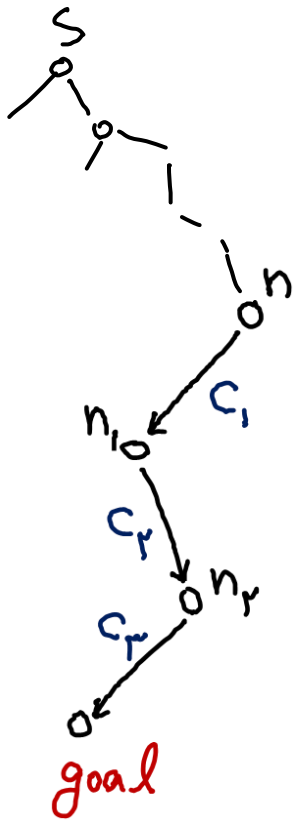
$$h(n) \leq c + h(n')$$

(نامساوی مثلث) . می‌گوییم h سازگار است.

قضیه: اگر $h(n)$ سازگار باشد آنگاه قابل قبول است.

اثبات:

مسیر بهینه از n تا هدف را در نظر بگیرید.



$$
\begin{aligned}
h(n) &\leq c_1 + h(n_1) \\
&\leq c_1 + c_2 + h(n_2) \\
&\leq c_1 + c_2 + c_3 + h(g) \\
&= c_1 + c_2 + c_3 + 0 \\
&= h^*(n)
\end{aligned}
$$

پس اگر h سازگار باشد حداقل دو نوع روش پیاده‌سازی A^* بهینه است.

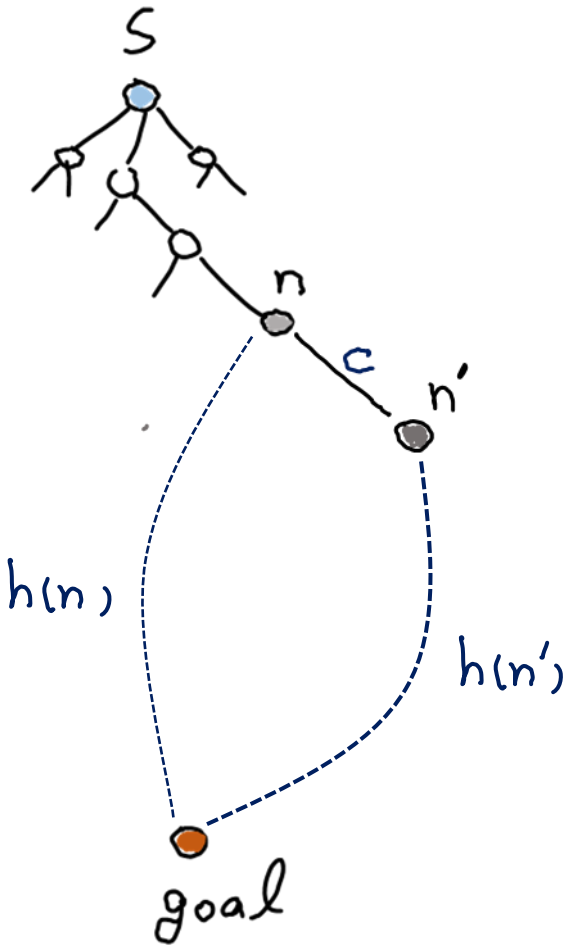
$h \leq h^*$ و h سازگار ← A^* کران بهینه است.

↑

$h \leq h^*$ و h ← A^* درختی بهینه است

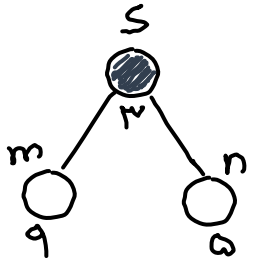
قضیه:

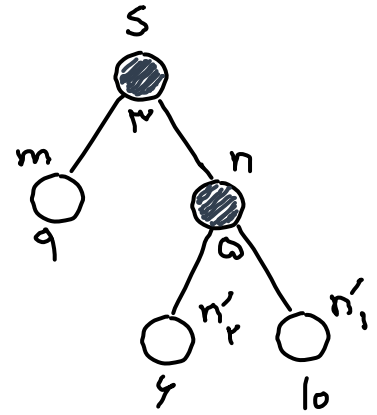
h سازگار باشد $\Rightarrow$ تابع f در هر مسیر صعودی است $(f(n') \geq f(n))$



$$
\begin{aligned}
f(n') &= g(n') + h(n') \\
&= g(n) + c + h(n') \\
&\geq g(n) + h(n) = f(n)
\end{aligned}
$$

واطبی سقه ارهزینه گره های موجود در صف و گره‌ای که در مرحله بعد به صف وارد می شوند.





openlist: m, n

openlist: n'_1, m, n'_2

گره ی که خارج شد: کمترین f(n) را داشت.

فرزندانش n'_1 و n'_2 حتماً هزینه‌ی بیشتر مبدا را دارند اما نسبت به سایر گره‌ها
ممکن است هزینه کمتر یا بیشتری داشته باشند.

۱۲۲- اگر h_1 و h_2 توابعی سازگار (consistent) باشند، کدام عبارت نادرست است؟

۱) $\alpha h_1 + (1-\alpha)h_2$ درحالتی که $0 \leq \alpha \leq 1$ سازگار است.

۲) $\frac{\min(h_1, h_2) + \max(h_1, h_2)}{2}$ سازگار است.

۳) $\max(h_1, h_2)$ سازگار است.

۴) αh_1 سازگار است.

همه نکاتی که برای قابل قبول بودن گفتیم برای سازگارها هم درست است.

ممکن است α نزدیک باشد. گزینه ۴ نادرست است.

۹۲- فرض کنید می‌خواهیم در یک مسئله جستجو از روش A^* استفاده کنیم. در این مسئله دو تابع مکاشفه قابل‌قبول h_1 و h_2 تعریف شده‌اند. کدام یک از روش‌های زیر حالت بهینه هدف را پیدا می‌کند؟

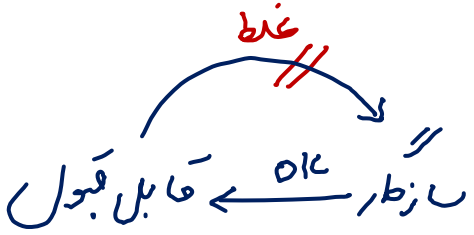
۱) جستجوی مبتنی بر گراف A^* با تابع مکاشفه $h = h_1 + h_2$

۲) جستجوی مبتنی بر درخت A^* با تابع مکاشفه $h = h_1 * h_2$

۳) جستجوی مبتنی بر گراف A^* با تابع مکاشفه $h = \min(h_1, h_2)$

۴) جستجوی مبتنی بر درخت A^* با تابع مکاشفه $h = (2h_1 + h_2)/3$

مهندسی آی تی ۱۴۰۰



h_1 و h_2 قابل قبولند پس درمورد سازگار بودن آنها نمی‌توان نظری داد.

h_1 و h_2 قابل قبولند پس $\frac{2}{3}h_1 + \frac{1}{3}h_2$ قابل قبول است و قابل قبول بودن، برای بهینه بودن جستجوی درختی کافی است.

h_1 و h_2 قابل قبولند پس $\min(h_1, h_2)$ قابل قبول است پس برای جستجوی درختی، شرط بهینه بودن ندارد اما نه برای جستجوی گراف.

رابطه‌ی بین میزان دقت هیوریستیک قابل قبول با مرتبه زمانی A^*

$h<h^*$ $\qquad \varepsilon = \left\| \frac{h^* - h}{h^*} \right\| \longrightarrow O(b^{\varepsilon d})$ $\qquad$ زمان A^*:

فاکتور انشعاب موثر $= O(b^{\varepsilon})$

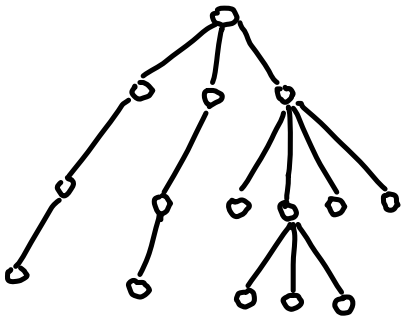
مثال عددی (اگر ۳ گره داشته باشیم) [معنای نُرم]

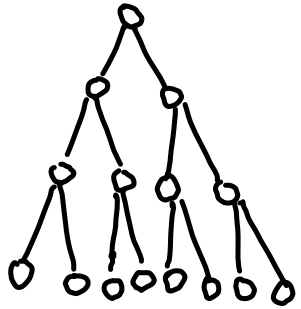
گره	h^*	h	$\lvert h-h^* \rvert$	$\frac{\lvert h-h^* \rvert}{h^*}$
A	۵	۲	۳	$\frac{۳}{۵}$
B	۶	۴	۲	$\frac{۲}{۶}$
C	۰	۰	۰	۰

$\Rightarrow^{max}$

$$\varepsilon_1 = \left\| \frac{h-h^*}{h^*} \right\| = \frac{۳}{۵}$$

فاکتور انشعاب موثر:





$$N = 1 + b + \cdots + b^{d} = \frac{b^{d+1} - 1}{b - 1}$$

$$N = \frac{b^{*d+1} - 1}{b^{*} - 1}$$

با دانستن N و d

می‌توان b^* را یافت.

مثلاً اگر در یک جستجو $N=25$ گره تولید و هدف در عمق $d=3$ یافت شود،

فاکتور انشعاب را حساب کنیم:

$$N=\frac{b^{*d+1}-1}{b^*-1}$$

$$25=\frac{b^{*4}-1}{b^*-1} \quad \rightarrow \quad 2<b^*<3$$

کاهش مرتبه زمانی A^* ← از طریق انتخاب $h(n)$ مناسب

کاهش مرتبه فضایی A^* ← روش هایی که A^* را با محدودیت حافظه اجرا کند

مثلاً به صورت تحتی، تکراری، بازگشتی، ...

(موضوع جلسه بعدی)

اجرای A^* با محدودیت حافظه:

Memory-bounded heuristic search

IDA*

RBFS

SMA*

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۴ فصل ۳ و ۴

مشکل اصلی A* نمایی بودن مرتبه فضایی (حافظه) است. با این روش‌ها این اشکال رفع می‌شود:

Memory-bounded heuristic search

		فضا
IDA*	عمق-نخست با محدودیت هزینه	O(bd)
RBFS	پیاده‌سازی بازگشتی	
SMA*	با حافظه O(c)	O(c)

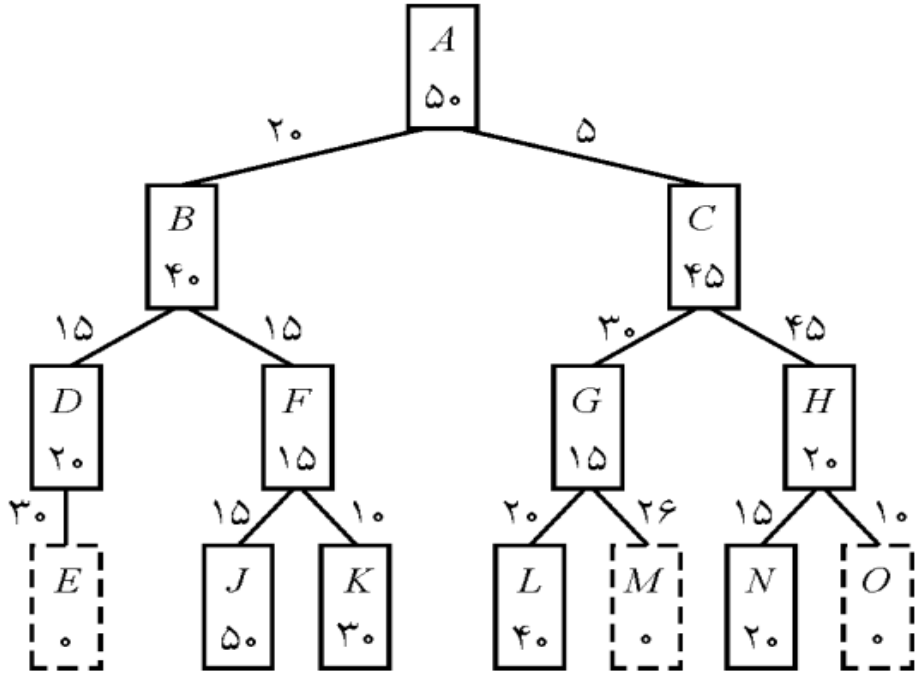
۶۹ـ درخت جستجوی زیر داده شده است. گره A، وضعیت اولیه می‌باشد. وضعیت‌های جواب نیز با مربع‌های نقطه‌چین نشان داده شده‌اند. اعداد روی یال‌ها هزینه استفاده از آن مسیر (یال) را نشان می‌دهد. اعداد داخل هر گره نیز تخمین فاصله تا هدف را مشخص می‌کند. اگر برای جستجو از روش IDA^* استفاده شود، میزان حد آستانه که جهت انتخاب گره‌ها، هنگام ورود به صف در نظر گرفته می‌شود پس از گرفتن مقدار اولیه، چند بار تغییر می‌کند؟ (مهندسی کامپیوتر ـ دولتی ۹۶)

۱) ۰

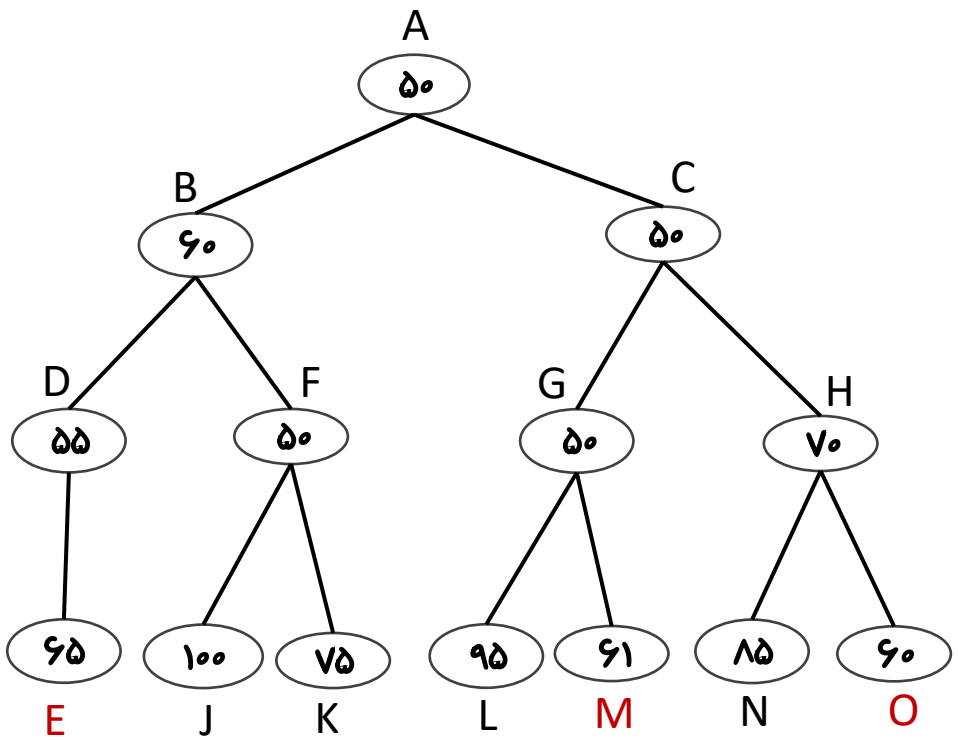
۲) ۱

۳) ۲ ✓

۴) ۳



دوبار مقدار f_cut تغییر می‌کند.



برای راحتی بیشتر،
مقادیر $f(n)$ همه گره‌ها را حساب
کرده‌ایم. گره‌های قرمز، هدف
هستند.

IDA* با شروع از گره A :

$$f_cut = f(A)$$

اجرای A^* بدون بسط دادن گره‌هایی که هزینه آنها بیشتر از f_cut است .

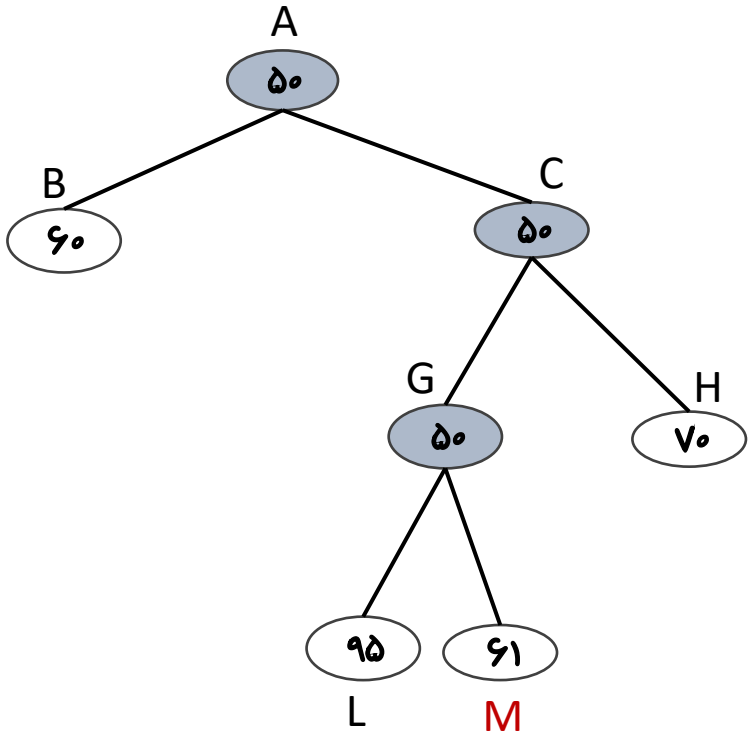
اگر هدف ملاقات نشد:

$$f_cut = \min(f(n) \mid n \in open_list)$$

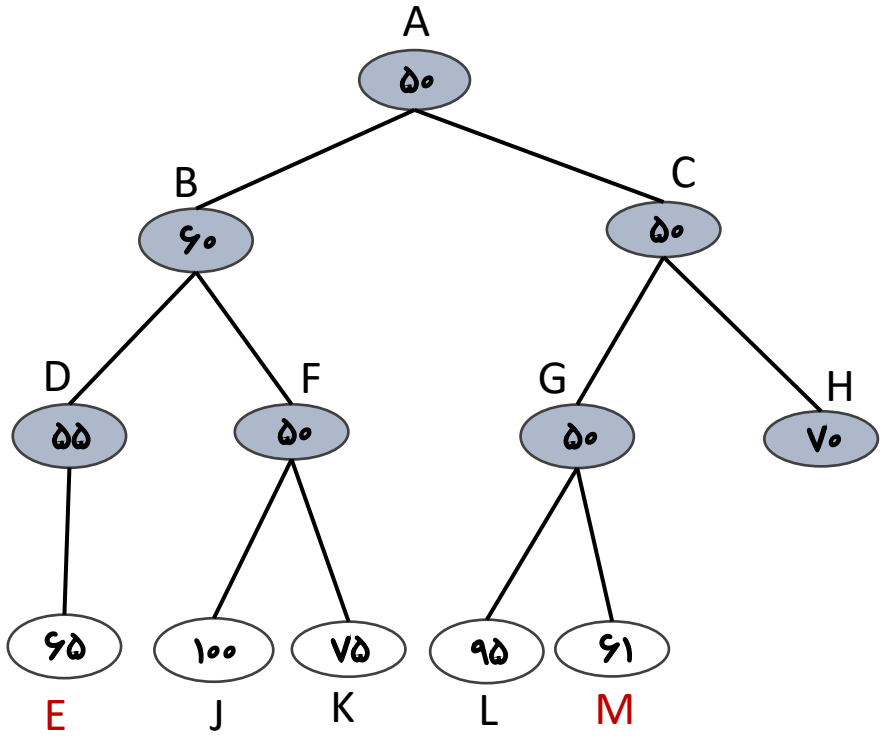
تکرار دوباره .

مرتبه فضای خطی است (جست‌وجوی عمقی است) $O(bd)$

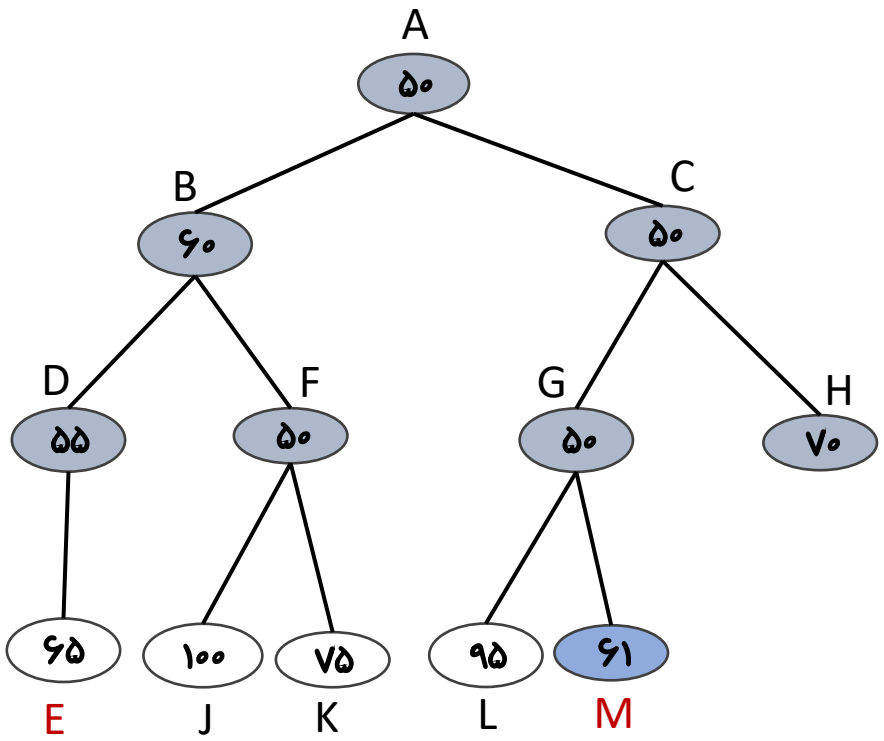
ع، ~ f_cut=۵۰



f_cut=۶۰



$f_cut = 61$



گره حرف M
ملاقات می‌شود.

R.B.F.S **Recursive best-first search** بازگشت A*

در ابتدا : $f_limit = \infty$

اگر n برای ملاقات انتخاب شده باشد:

قبل از بسط دادن باید f_limit آن محاسبه و ذخیره شود.

[کمترین مقدار f در بین گره‌هایی به جز n که در openlist هستند]

$f_limit =$ کمترین مقدار f در بین همگره‌های رقیب

یعنی در بین همه‌ی برادران و دیده عموها

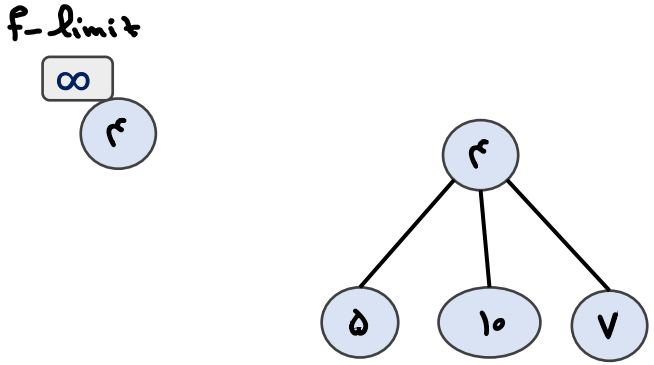
پس از بطل دادن گره ما:

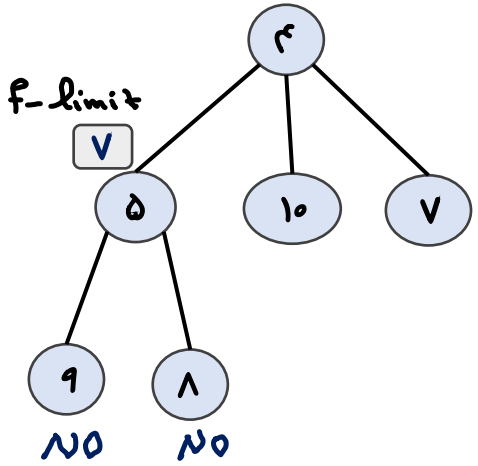
از بین فرزندان، کم هزینه ترین انتخاب می‌شود البته با این شرط که محدودیت f_lim را نقض نکند.

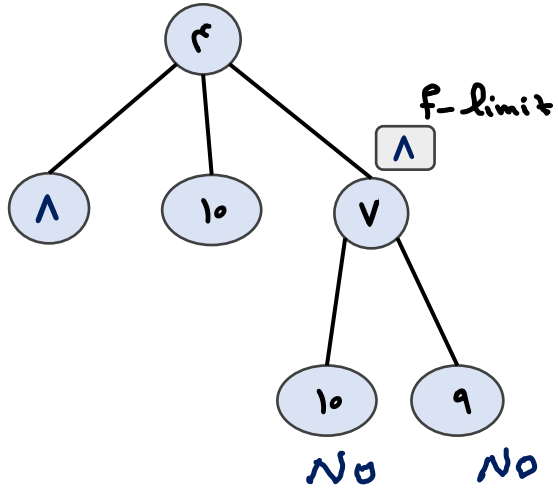
اگر همه فرزندان هزینه‌ای بیش از f_lim دارند باید بازگشت از ما انجام شود.

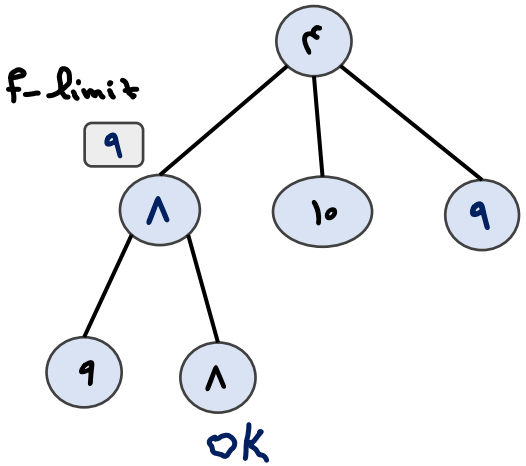
در بازگشت از ما:

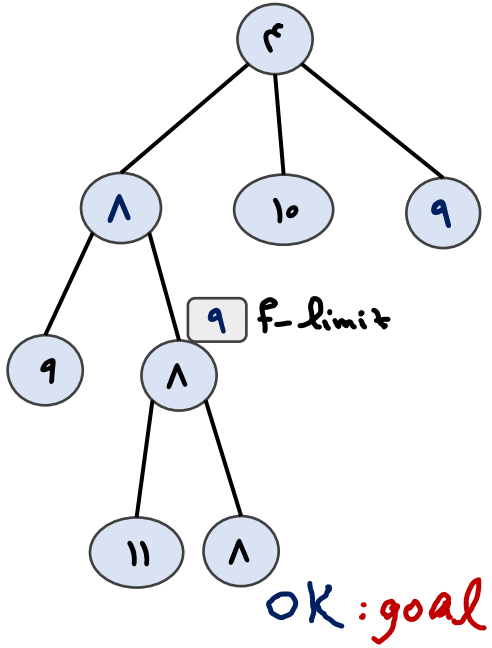
[به روزرسانی f(u): (min هزینه فرزندان) انجام می‌شود.]


f-limit
∞
۴
۴
۵
۱۰
۷


f-limit
۷
۴
۵
۱۰
۷
۹
۸
NO
NO


۴
f-limit
۸
۸
۱۰
۷
۱۰
۹
NO
NO


f-limit
۹
۴
۸
۱۰
۹
۹
۸
OK


۴
۸
۱۰
۹
۹
f-limit
۹
۸
۱۱
۸
OK : goal

RBFS

مرتبه فضایی $O(bd)$

مرتبه زمانی بستگی به هیوریستیک دارد. اما حداکثر همان $O(b^{\varepsilon d})$ است.

شرط بهینه بودن: مانند A^* .

در مقایسه با IDA^* : معمولاً تعداد گره‌های کمتری بسط می‌دهد اما برای هر گره تعداد محاسبات بیشتری باید انجام شود. در کل معمولاً سریع‌تر است.

MA* (memory-bounded A*) and **SMA*** (simplified MA*)

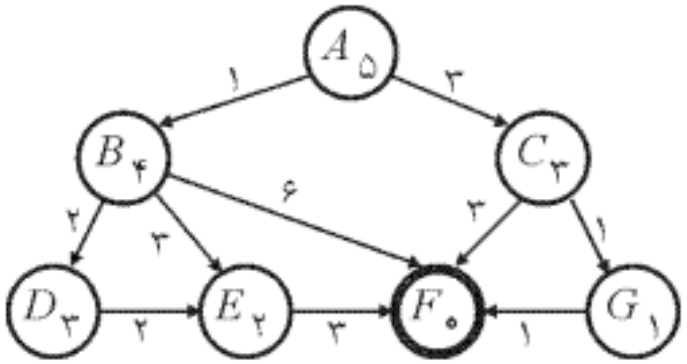
SMA* با محدودیت m گره در حافظه: مرتبه فضای مورد نیاز: O(c)

① اگر با مسیری شامل m گره (عمق m-1) نتوان به هدف رسید، کامل است.

② در بین مسیرهای تا عمق (m-1) مسیری را که کم هزینه‌تر باشد می‌یابد

③ می‌تواند همان A* را اما روی مسیرهای با عمق حداکثر m-1 اجرا کند.

۱۱ـ حاصل جستجوی SMA^* با حداکثر ۳ خانه حافظه بر روی گراف مقابل چیست؟ (A گره شروع و F گره هدف است. اعداد روی یال‌ها هزینه مسیر و اعداد داخل گره‌ها هزینه تخمینی گره تا هدف است. ترتیب ملاقات فرزندان به ترتیب حروف الفباست.) (مکاترونیک و مهندسی IT ـ دولتی ۸۸)

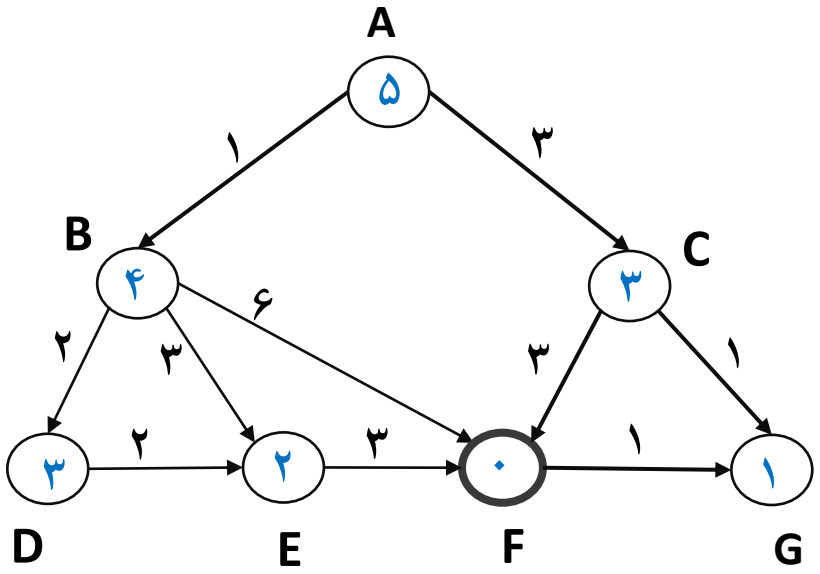


۱) ACF

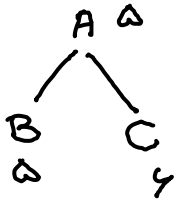
۲) ABF

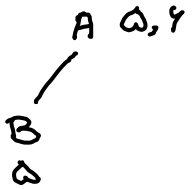
۳) $ACGF$

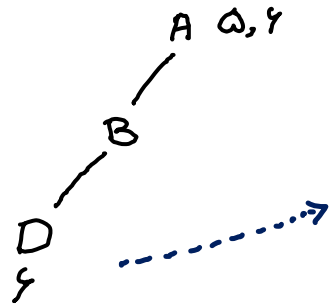
۴) SMA^* پاسخی برای این مسئله پیدا نمی‌کند.

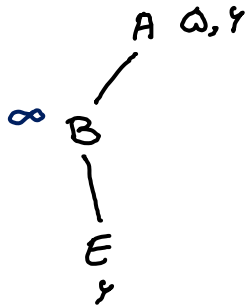


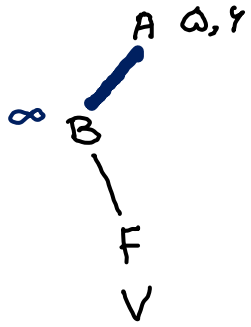
با محدودیت $m=۳$ گره:

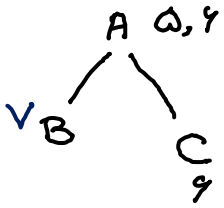


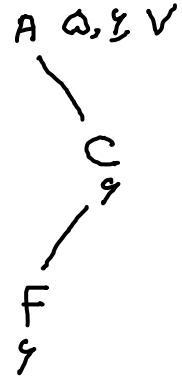










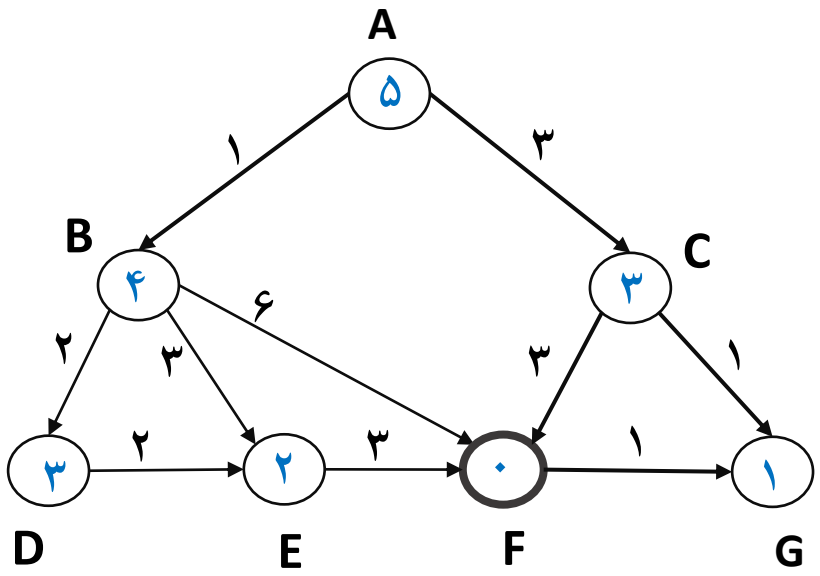


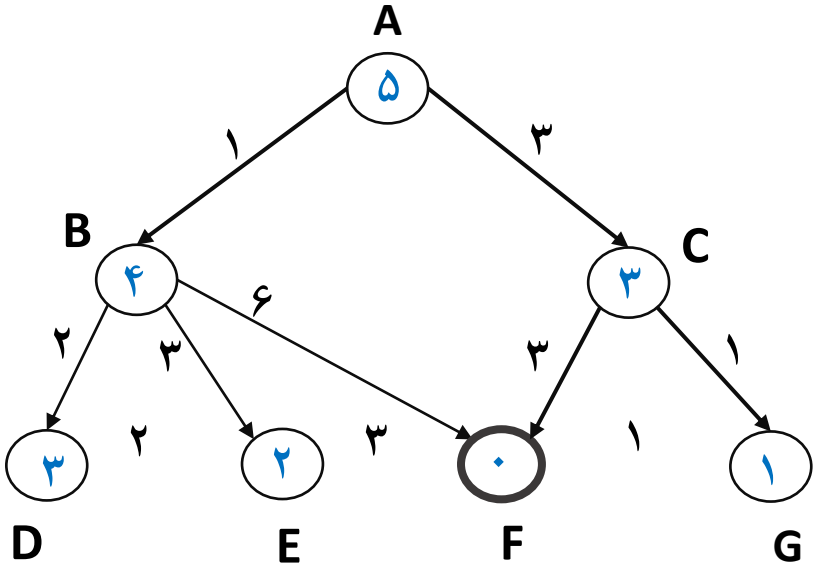
D تک برگ و غیرهدف است.

مقدار $f(D)$ در پدر ∞ ذخیره می‌شود

برای یافتن خروجی SMA^* :

می‌توانیم مسیرهای با طول بیشتر از $m-1$ (دارای بیش از m گره) را حذف کنیم سپس همان A^* را اجرا کنیم.





اجرای A^*

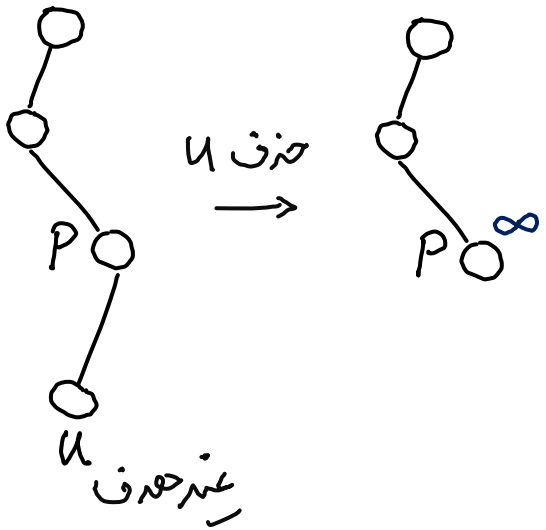
A^* مسیر بهینه تا هدف را می‌یابد چون این مسیر در دسترس است.

خلاصه‌ی SMA^* :

جستجوی A^* را اجرا کن، اما هرگاه فضای ذخیره برای تولید گره جدید وجود ندارد:
گره با بیشترین $f(n)$ را حذف اما مقدار آن را در پدر ذخیره کن.

* اولویت حذف در بین چند گره یکسان: گره با عمق کمتر (قدیمی‌تر)
* اولویت بسط در بین چند گره یکسان: گره با عمق بیشتر (جدیدتر)

* اگر یک برگ داریم و فضای حافظه پر است و آن برگ باید حذف شود و آن برگ، هدف نیست: مقدار ∞ در پدر ذخیره شود.



گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

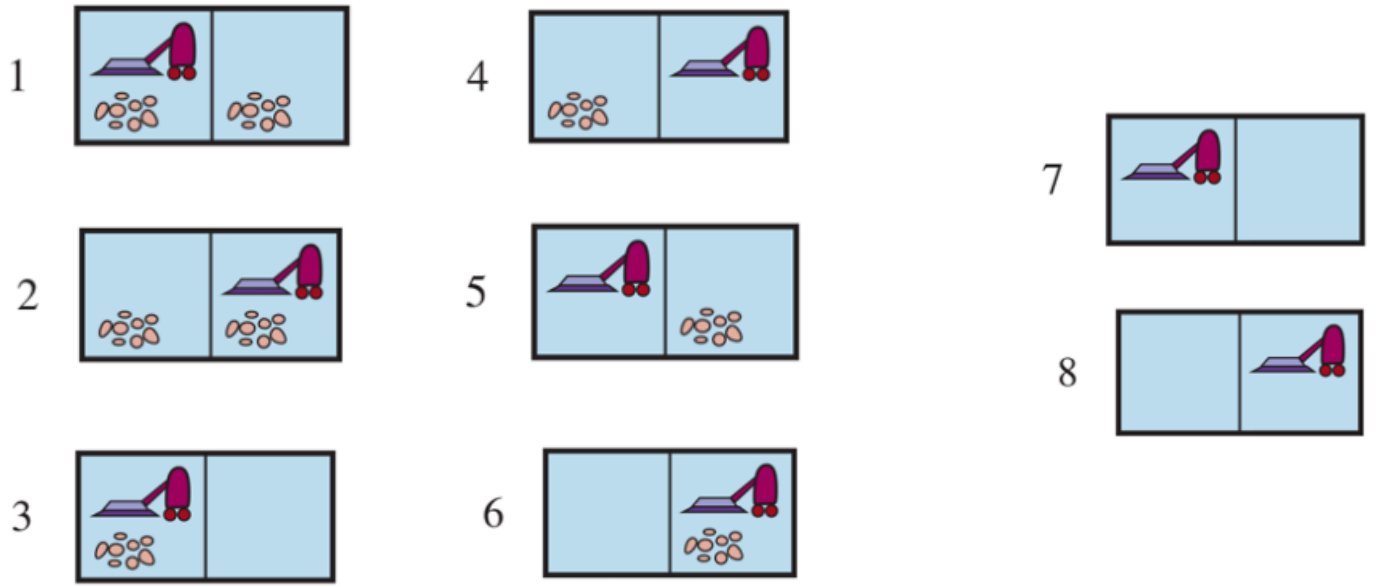
هوش مصنوعی

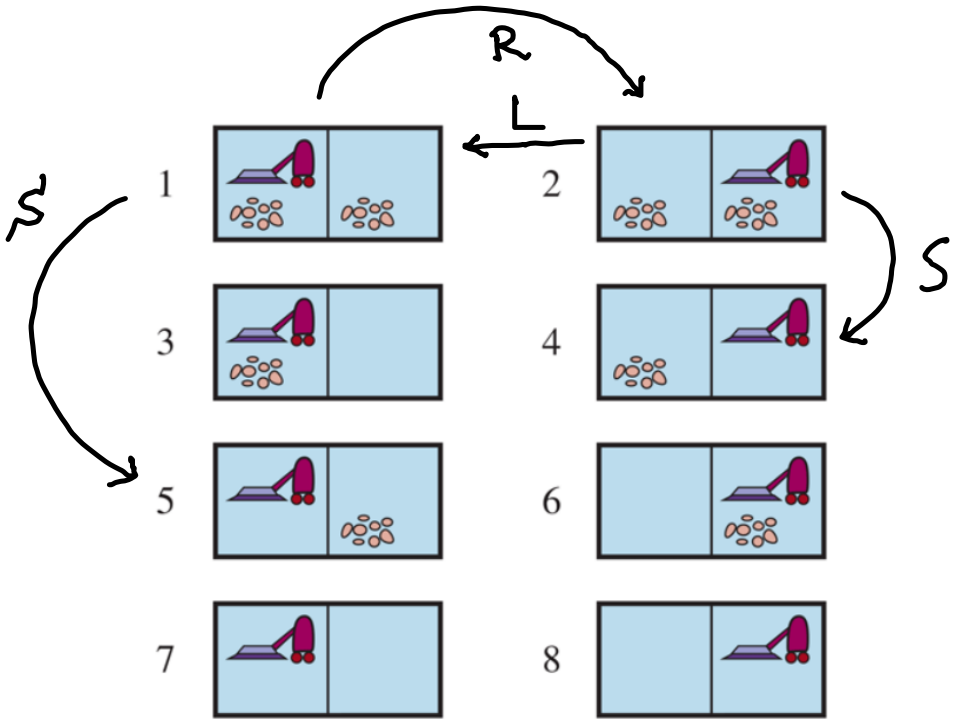
ابوالفضل گیلک
گروه بابان

بخش دوم جلسه ۴

عامل حل مساله با جستجو در محیط های پیچیده ی که

- غیر حتمی
- مشاهده پذیر جزئی
- مشاهده ناپذیر





جارو برقی هوشمند در مثال ساده آن

دارای ۸ حالت است.

اقدامات ممکن، راست، چپ

و سنسورهای مکان و تشخیص تمیز بودن اتاق

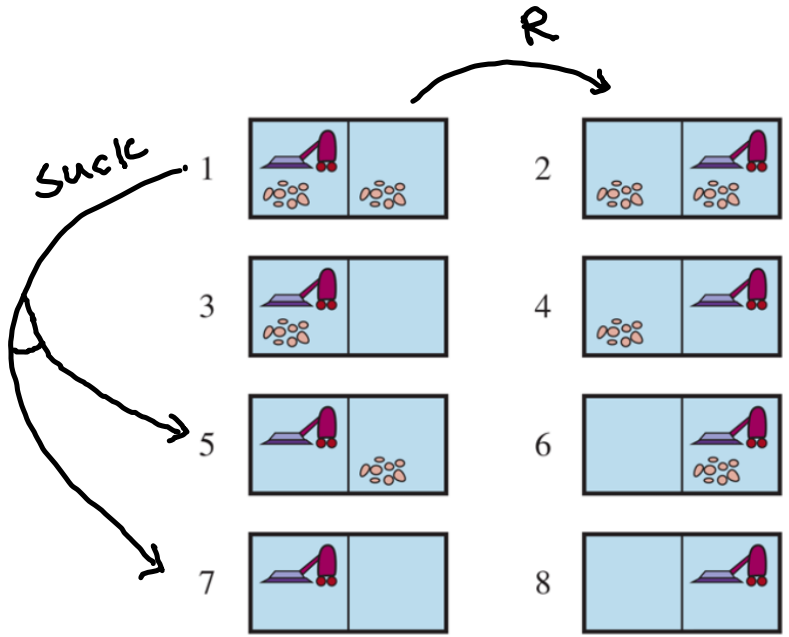
را دارد.

حالا وزن های دیگر را برای یکم:

erratic vacuum world

جاروبرقی غیرقطعی

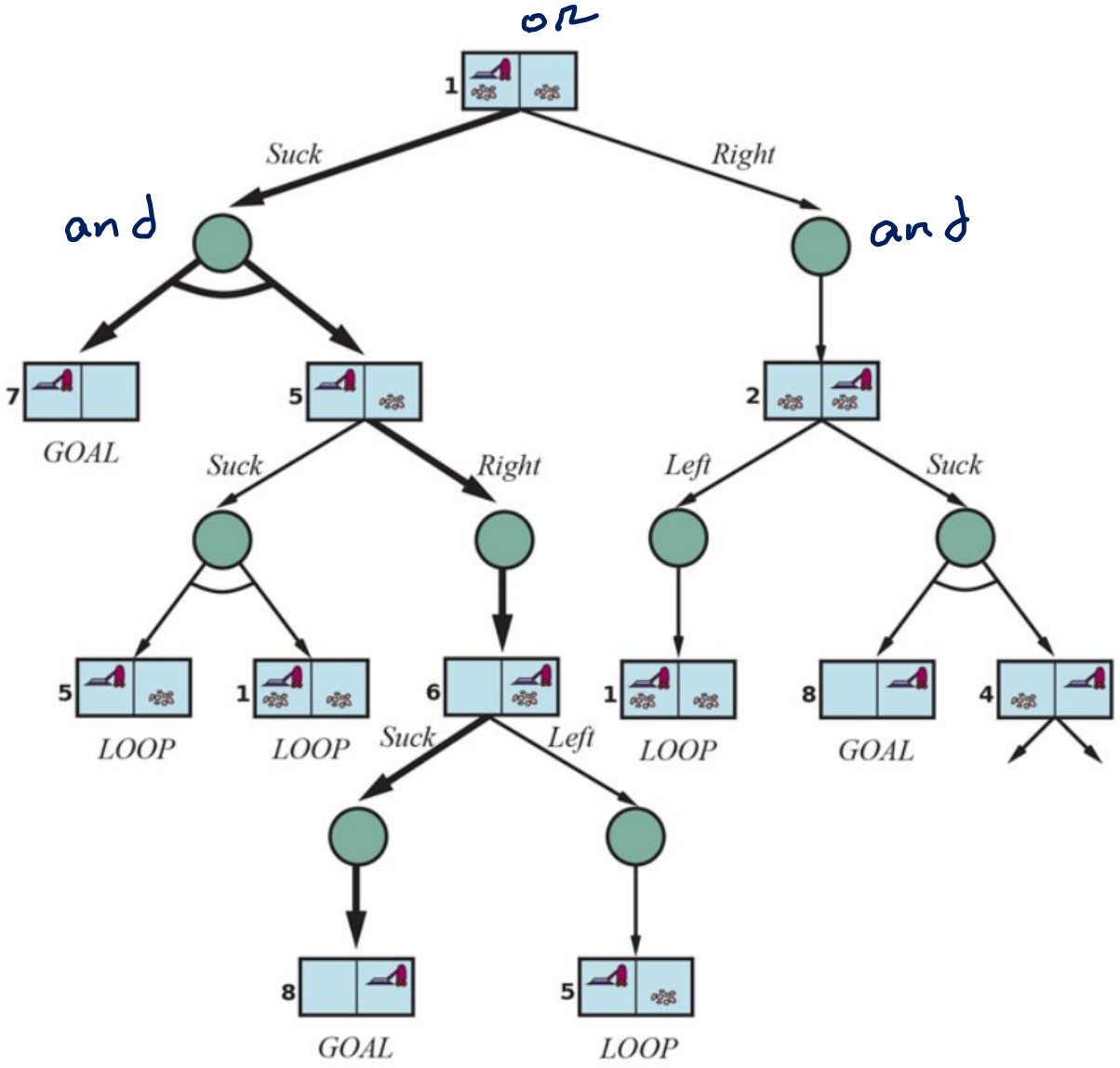
هنگامی که در اتاق کثیف اقدام به مکش می‌کنیم گاه همین اتاق و گاه هر دو اتـاق تمیـز می‌شوند. هنگامی که در اتاق تمیز اقدام به مکش می‌کنیم گاهی اوقـات گـرد و غبـار از اتاق کثیف مجاور به این اتاق نیز کشیده می‌شود.



در این موارد (اقدام با نتیجه غیرقطعی) از درخت جستجوی AND — OR استفاده می کنیم.

گره‌ای AND : نشان دهنده نتایج ممکن یک اقدام

AND–OR search trees



گره‌های OR : در آن حالت چه اقداماتی قابل اجراست.

گره‌های and : نتایج ممکن هر اقدام.

برخی برگ‌ها هدف هستند.

به پدر یا جد برمی‌گردند.

برخی برگ‌ها، هدف هستند.

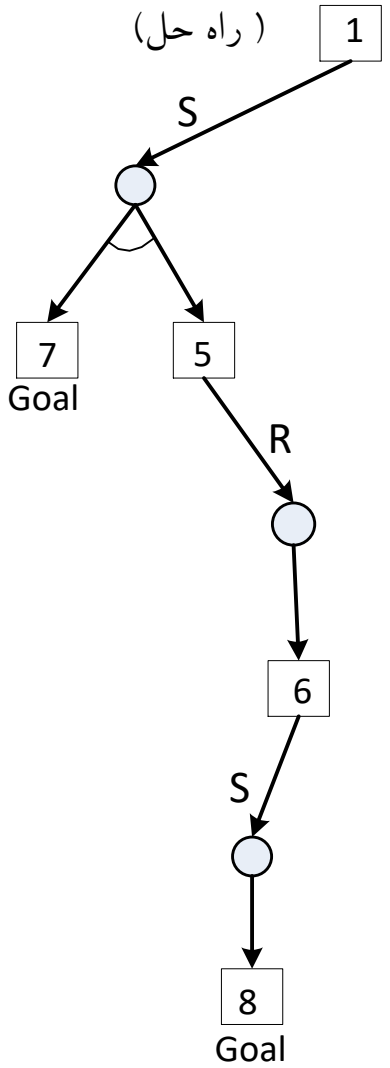
مسیری که همه برگ‌های غیر هدف آن، هدف باشند، جواب است.

دنباله جواب :

[Suck, if state=5 then [Right, Suck] else []]

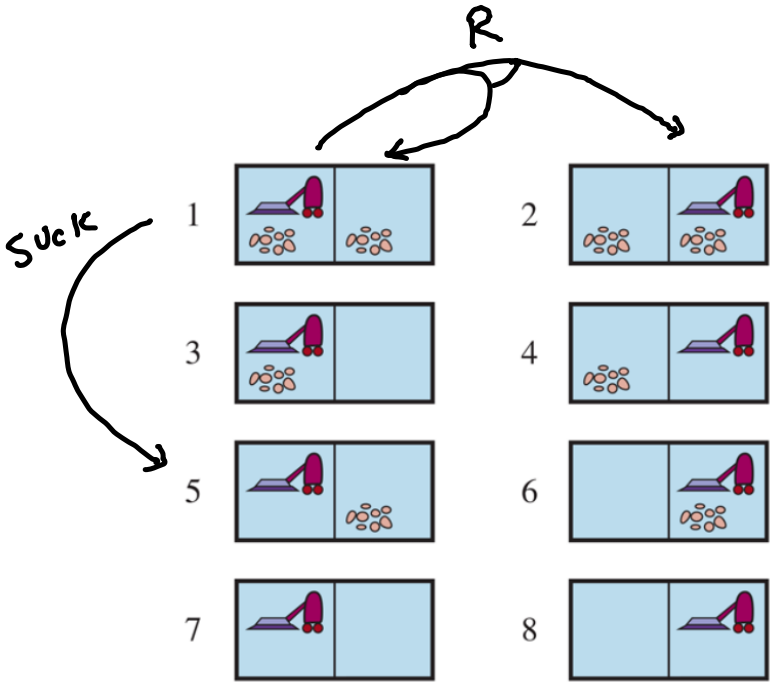
دنباله جواب (شرطی)

دنباله جواب : (حداکثر ۳ گام دارد)

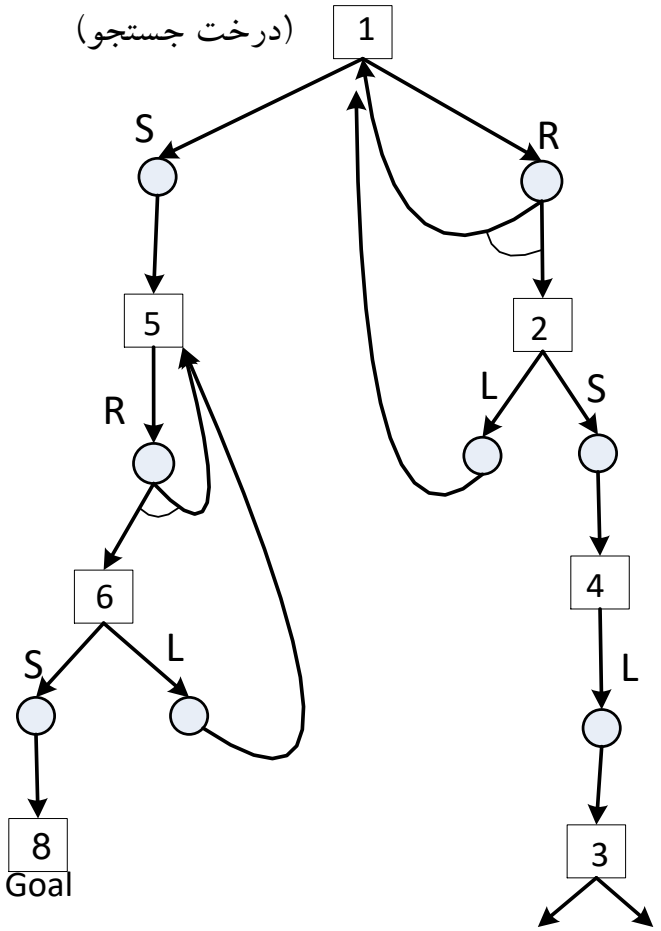


نمونه دیگری از جاروبرقی غیرقطعی

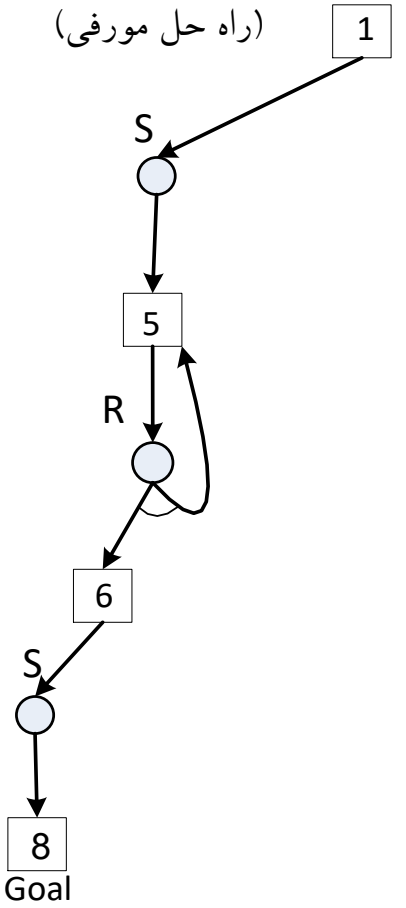
به دلیل نقص در سیستم حرکتی، هنگامی که اقدام حرکت به راست اجرا میشود گاهی اوقات هیچ جابجایی رخ نمی‌دهد.



درخت AND–OR را رسم می‌کنیم. در این نمونه‌ها، دنبالهٔ جواب از نوع دوری، (صورتی) به دست می‌آید.



جواب



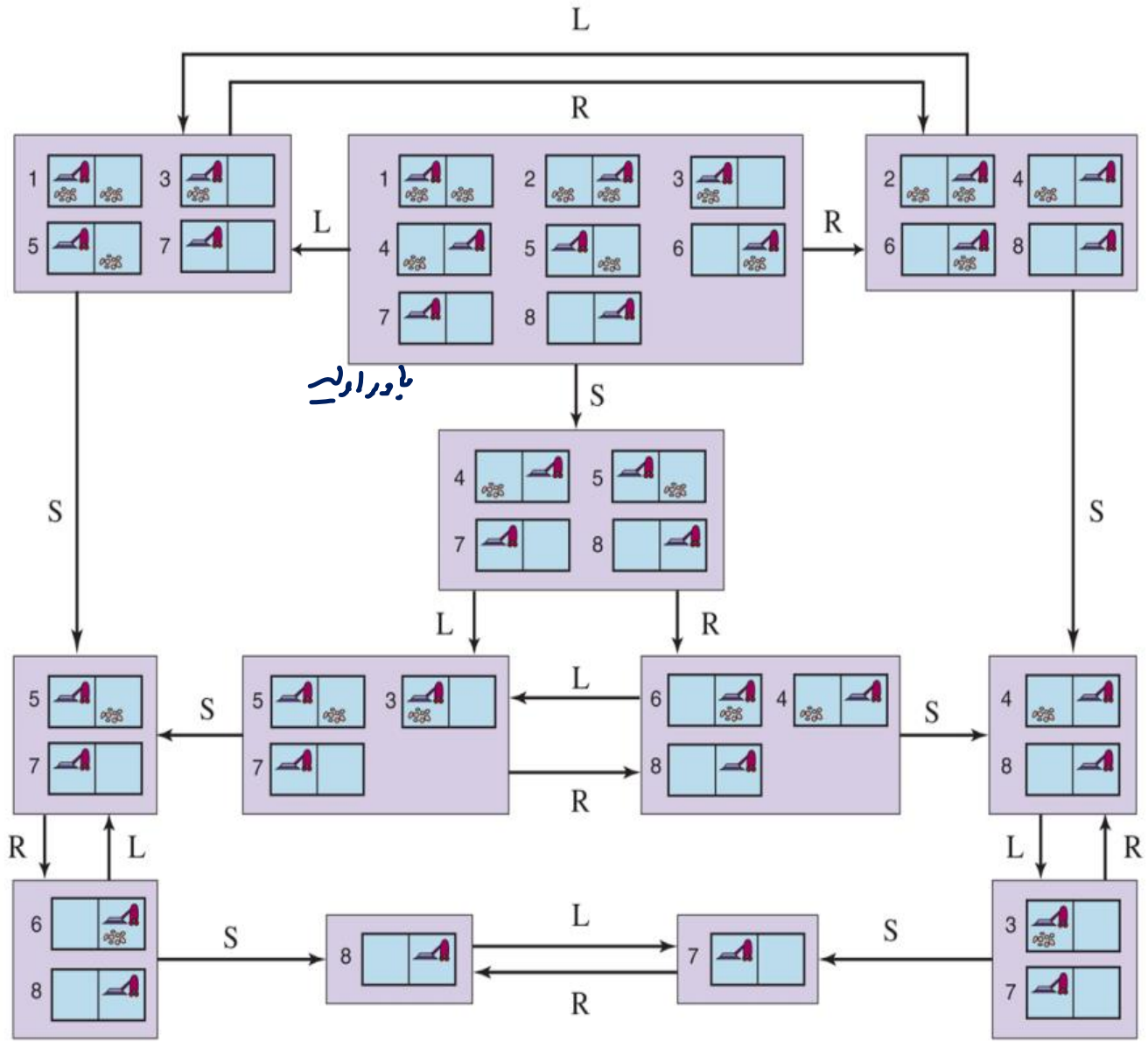
دنباله جواب مورفی: (دوری)

[Suck,{ while state=5 do Right }, Suck]

Try, try again

نمی‌توان حداکثر تعداد گام‌ها را تعیین کرد.

Cyclic solution

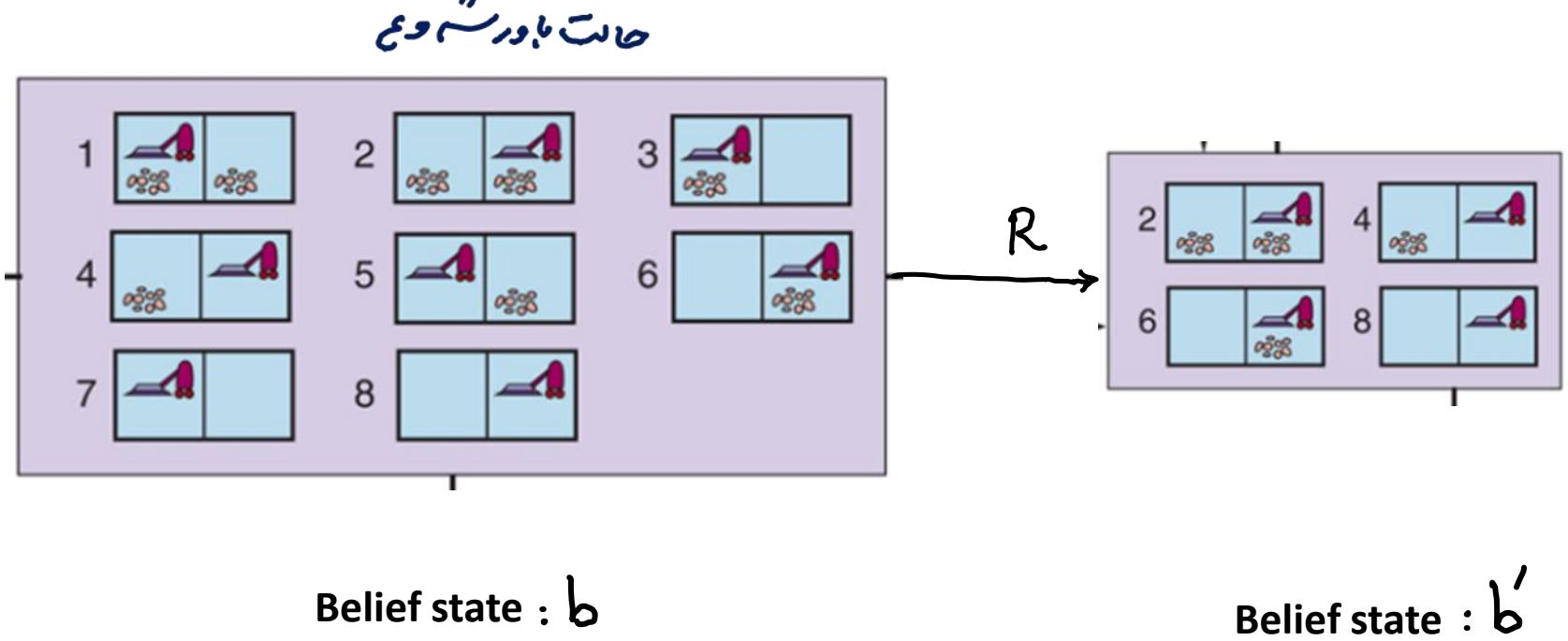


در محیط مشاهده ناپذیر یا جزئاً مشاهده پذیر به جای جستجو در فضای حالات باید از جستجو در فضای حالات باور استفاده کنیم.

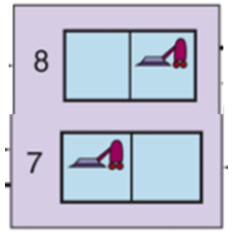
هر زیرمجموعه‌ای از مجموعه‌ی تمام حالات، یک حالت باور است (چون ۸ خانه داریم 2^8 حالت باور داریم)

حالت باور اولیه دارای تمام ۸ حالت ممکن است.

مثلاً با اقدام حرکت به راست از حالت باور اولیه‌ی b به حالت باور b' خواهیم رسید.



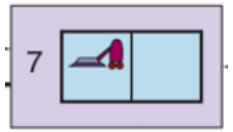
GOAL TEST : حالت یا وضعیت کیف است goal است اگر و تنها اگر هر کیف s∈b باشد goal باشد.



goal



goal



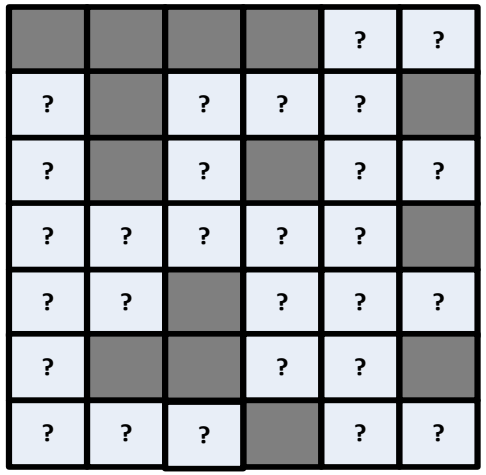
goal

عامل در محیط غیر قطعی و مشاهده پذیر نسبی:

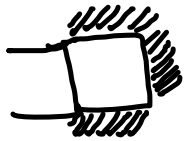
اقدام move : تصادفاً به یکی از 4 همسایه غیرمسدود می رود.

سنسورها: فقط همسایه های مسدود را تشخیص می دهد اما محل عامل در نقشه را نمی داند.

هدف فعلی: عامل می خواهد محل خود در نقشه را تشخیص دهد.



حالت باور اولیه b_0 : همه خانه‌های غیر مسدود.

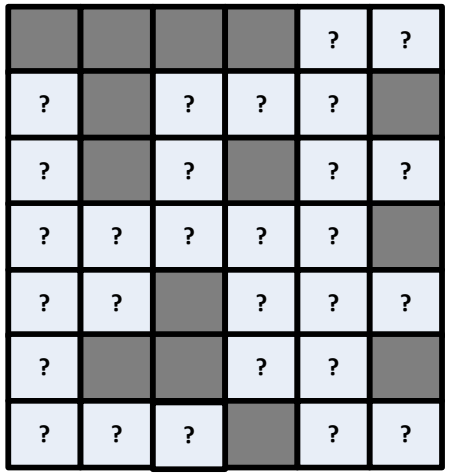


گرفتن ادراک از محیط : N, S, E یعنی شمال جنوب شرق ، مسدود است .

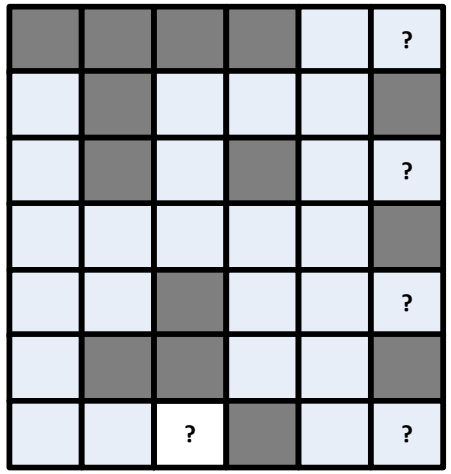
حالا با توجه این ادراک ، باور اولیه را update می‌کنیم : b_1

یک اقدام move انجام می‌دهیم. قاعدتاً باید به b_{act} رسیده باشیم.

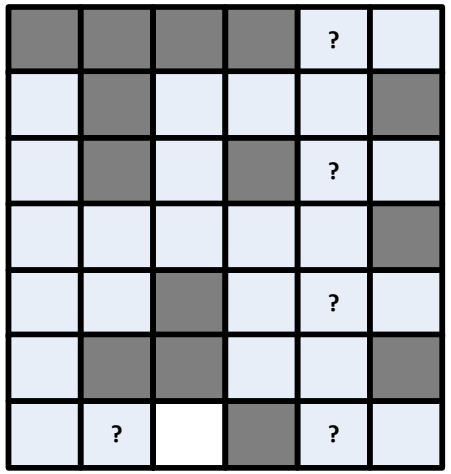
(این چرخه را تکرار می‌کنیم)



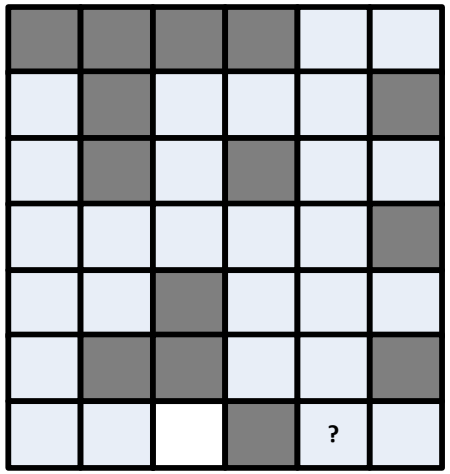
b_0



b_1

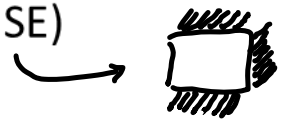


b_{action}



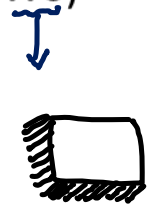
b_2

$b_1 = \text{Update}(b_0, \text{NSE})$



$b_{action} = \text{Predict}(b_1, \text{Move})$

$b_2 = \text{Update}(\text{Predict}(\text{Update}(b_0, \text{NSE}), \text{Move}), \text{WS})$

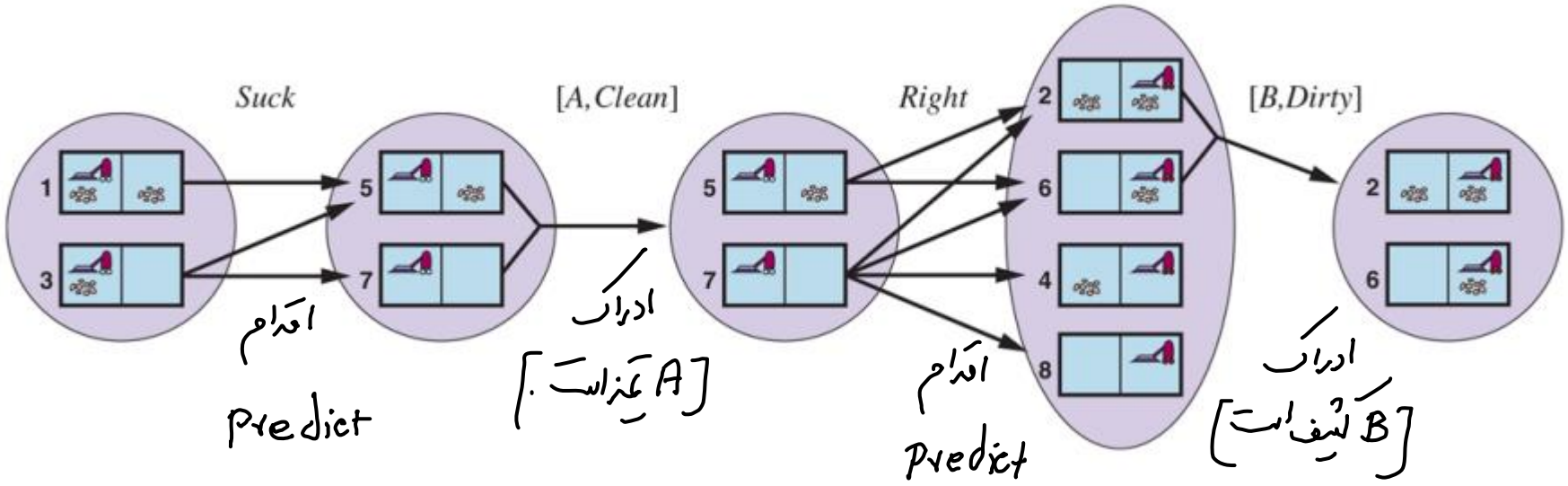


فیلترکردن و به‌روزرسانی

توجه: وقتی اقدامات غیرقطعی بوده و چند برایند ممکن داشته باشند، هر گام Predict باعث بزرگتر شدن حالت باور و هر بار Update باعث فیلتر شدن آن می‌شود. البته برخی مواقع ادراکات نمی‌توانند کمک چندانی بکنند مثلا اگر در یک تونل افقی طولانی که تمام خانه های مسیر از شمال و جنوب مسدودند قرار داشته باشیم ادراک NS کمکی به تعیین موقعیت نمی‌کند

صحنه‌ای که جستجو در محیط پویا (اتاق‌ها ممکن است در مرحله بعدی کثیف شده باشند) و مشاهده‌پذیر جزئی (فقط سنسور تشخیص وجود آشغال در اتاق فعلی)

البته با شبیه‌سازی از بالا و اولیه‌ی پاسخ می‌دهد:



درباره‌ی A^*:

$m \geq 1$

$h \leq m h^*$ $\Rightarrow$ هزینه‌ی مسیر پیدا شده توسط A^* کمتر مساوی m برابر هزینه‌ی مسیر بهینه است.

$p^* \leq m c^*$

هزینه مسیر بهینه ← ← هزینه مسیر یافت شده

الگوریتم جستجوی مسیر اکتشافی :

$$f(n) = (2 - w)g(n) + w \cdot h(n)$$

- اگر $w = 0$ باشد این الگوریتم به الگوریتمی معادل با جستجوی هزینه یکنواخت تبدیل می‌شود.
- اگر $w = 1$ باشد این الگوریتم به جستجوی A^* تبدیل می‌شود.
- اگر $w = 2$ باشد این الگوریتم به جستجوی حریصانه تبدیل می‌شود.

برای $h(n)$ قابل قبول، یک روش بهینه است. $\Longrightarrow$ $0 \leq w \leq 1$

مانند روش حریصانه، بهینه نیست $\Longrightarrow$ $1 < w \leq 2$

علت این امر آن است که اگر $h(n) \leq h^*(n)$ باشد برای $0 \leq w \leq 1$ داریم:

$$f(n) = (2-w)\left(g(n) + \underbrace{\frac{w}{2-w} h(n)}_{\downarrow} \right)$$

$$f(n) = (2-w)\left(g(n) + h'(n) \right)$$

$0 \leq w \leq 1 \longrightarrow \frac{w}{2-w} \leq 1 \Longrightarrow h'(n) \leq h(n) \leq h^*(n)$

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

بخش سوم جلسه ۴

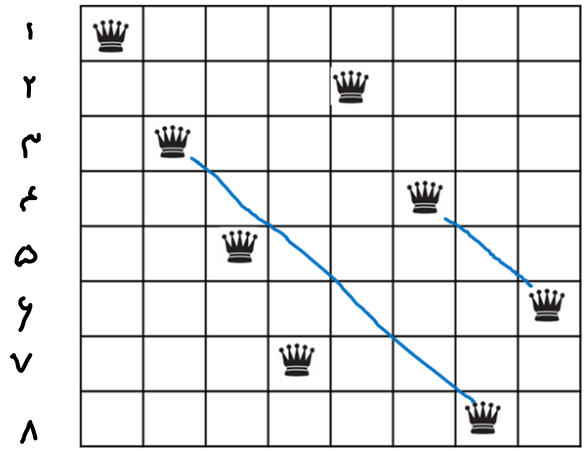
Local search

جستجوی محلی local search

در برخی مسائل یافتن مسیر بهینه هم نیت فقط می خواهیم از بین همه حالات، حالت بهینه را بیابیم.

- هر حالت یک مقدار سودمندی (یا یک مقدار هزینه) نسبت می دهیم و به دنبال حالت با کمترین هزینه یا بیشترین سودمندی هستیم.

مثلاً تعداد جفت وزیرهایی که یکدیگر را تهدید می کنند یک تابع هزینه برای ۸-وزیر است.



State: [۱, ۳, ۵, ۷, ۲, ۴, ۸, ۶]

۲ = مقدار هزینه

Hill-climbing search **greedy local search**

تپه نوردی بهترین انتخاب :

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    current ← problem.INITIAL
    while true do
        neighbor ← a highest-valued successor state of current
        if VALUE(neighbor) ≤ VALUE(current) then return current
        current ← neighbor
```

در تمام این جستجو فقط گره فعلی (current) و همسایگان آن و مقدار سودمندی آنها در هر مرحله مورد نیاز است پس فضای مورد نیاز : $O(c)$ است .

کپه نوردی بهترین انتخاب (تیزترین طلوع) [یک روش حریصانه محلی است]

اکنون ما هستیم.

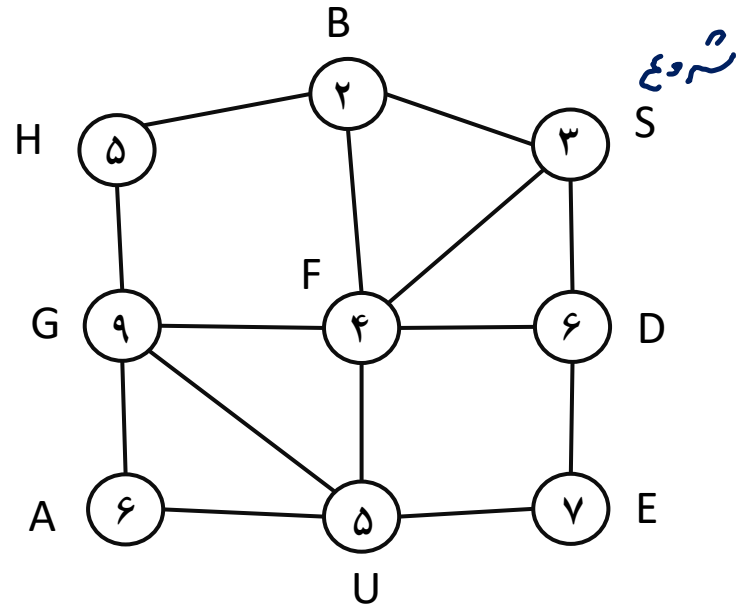
هم سالدهای ما تولید می‌شوند. بهترین آنها گسترش می‌یابد.

اگر آن همسایه بهتر از ما باشد، تغییر وضعیت می‌دهیم.

درغیراین صورت در ما متوقف می‌شویم.

نقطه ضعف: احتمال توقف در بهینه‌ی موضعی و نرسیدن به بهترین حالت.

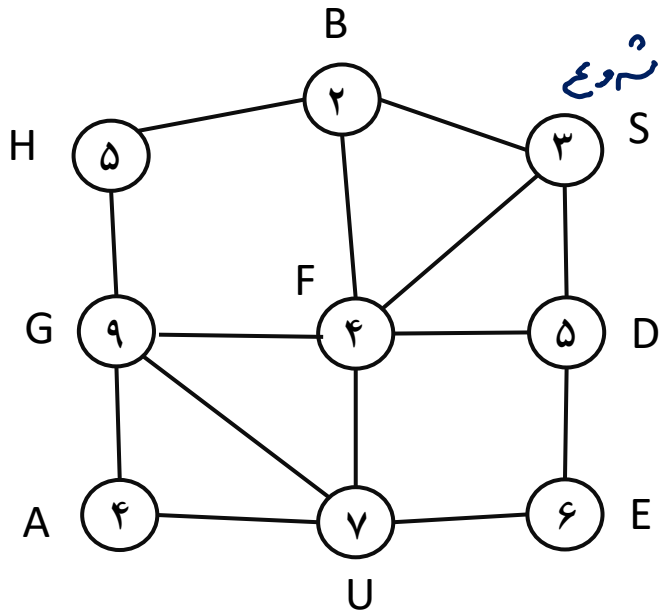
مثال: توقف در یک بهینه‌ی محلی (موضعی)



حالات:

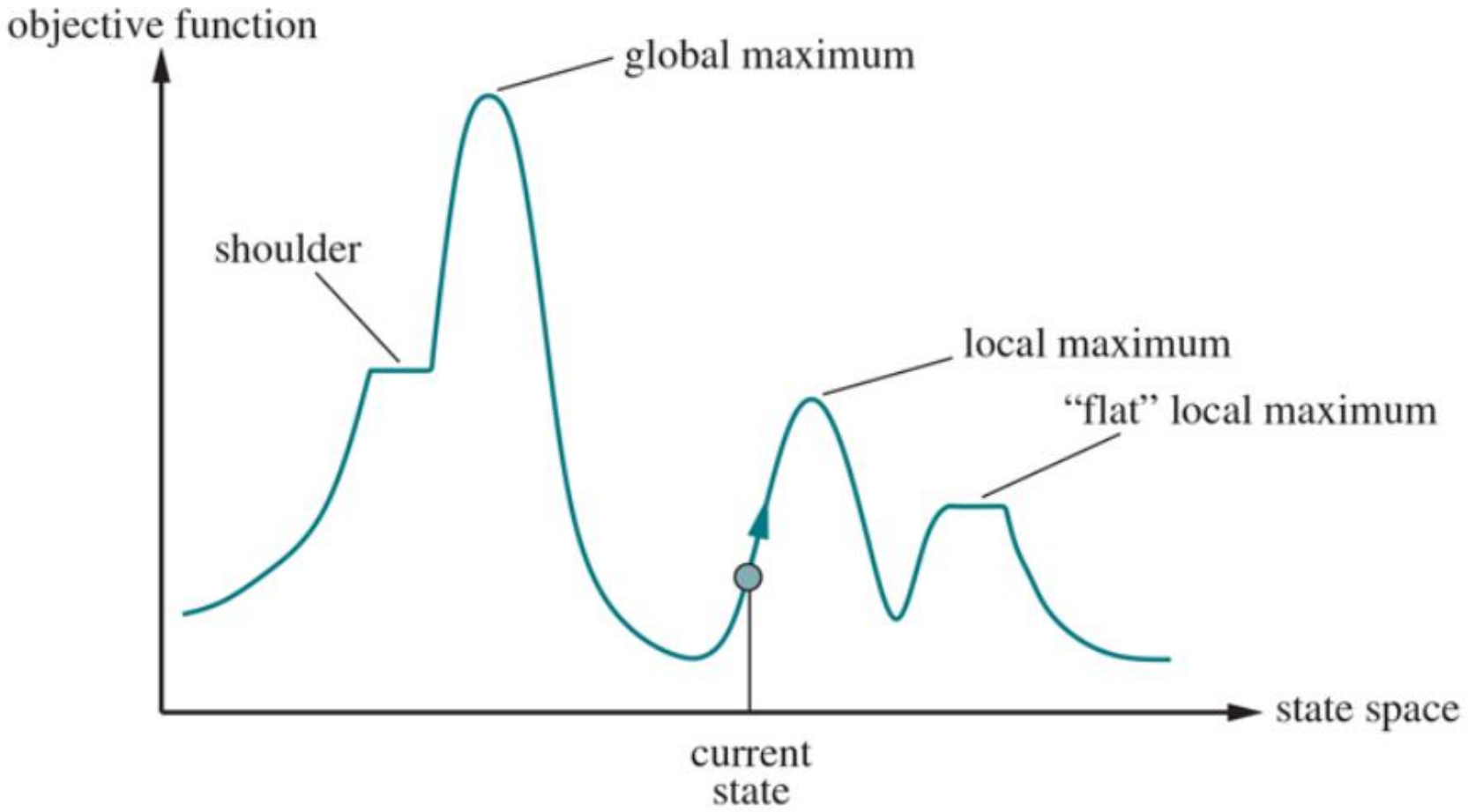
S, D, E

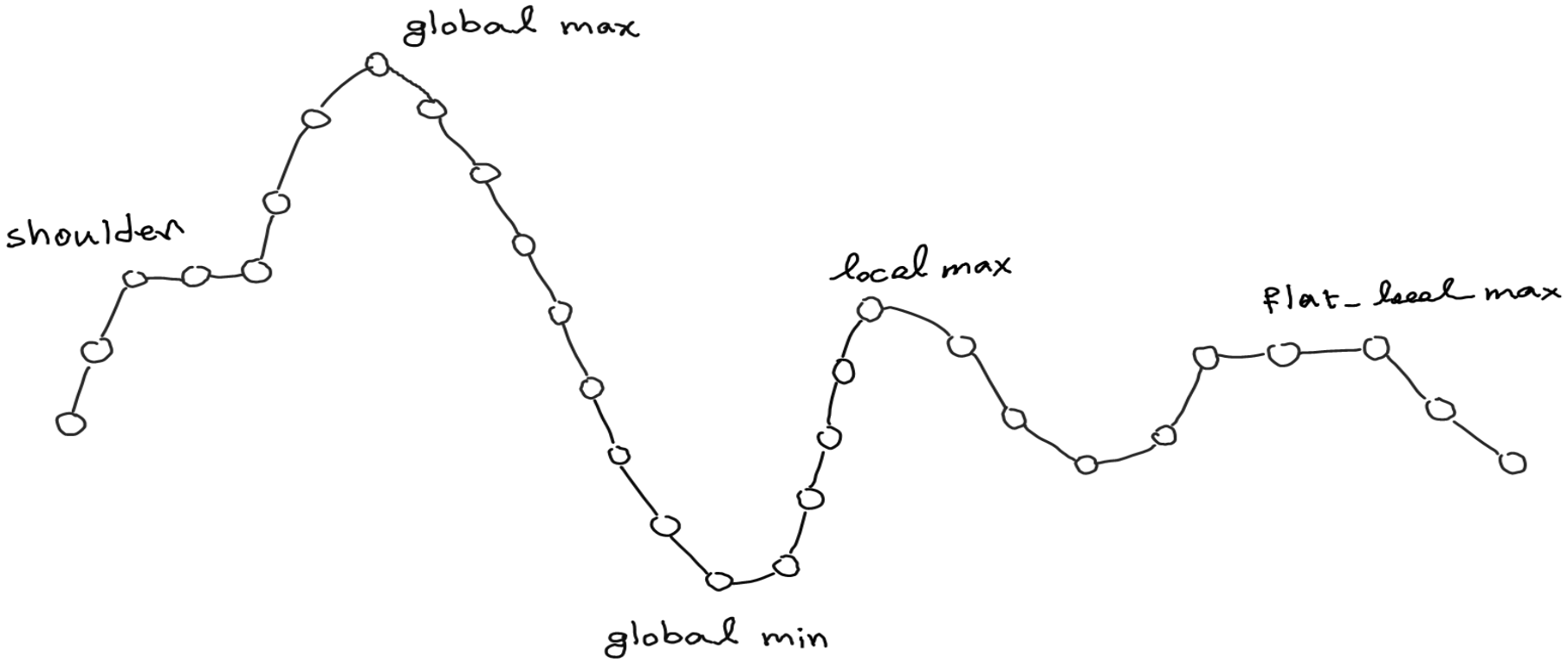
مثال: موفقیت در یافتن بهینه مطلق



حالات:

S, D, E, U, G


objective function
global maximum
shoulder
local maximum
"flat" local maximum
state space
current
state


global max
shoulder
local max
flat_local max
global min

نسخه‌های مختلف تکه‌نوردی:

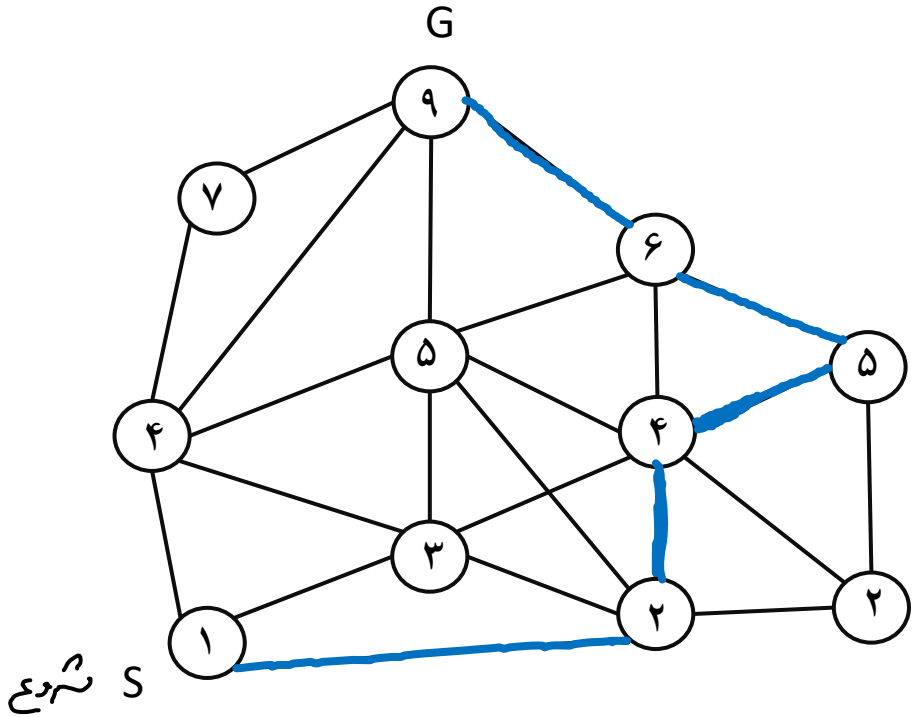
- تکه‌نوردی معمولی
- تکه‌نوردی بهترین انتخاب
- تکه‌نوردی تیزترین صعود

← تولید همگان، یافتن بهترین آنها، مقایسه آنها با هم، تغییر وضعیت به شرط سودمندی بیشتر.

- تکه‌نوردی تصادفی
- تکه‌نوردی اولین انتخاب
- اول بهترین

← تولید همگان با ترکیب تصادفی، به محض یافتن اولین همسایه‌ای که سودمندی بیشتری از گره فعلی دارد، تغییر وضعیت می‌دهیم.

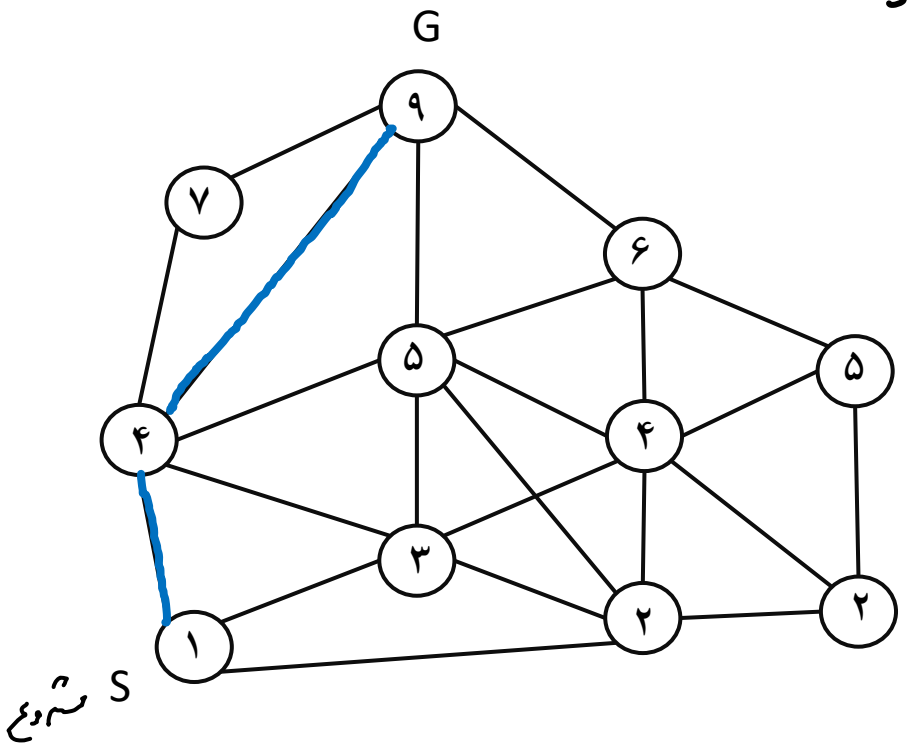
تپه‌نوردی اول بهترین



مدت زمان کمتر ، اما تعداد گام‌های بیشتر.

تپه‌نوردی بهترین انتخاب

تپه‌نوردی تندترین صعود



مدت زمان بیشتر، اما تعداد گام‌های کمتر

اولین انتخاب

۲۸ـ کدام‌یک از موارد زیر در مورد مقایسه دو روش جستجوی تپه‌نوردی ساده و تپه‌نوردی از تندترین شیب صحیح است؟ **(مهندسی IT ـ دولتی ۸۳)**

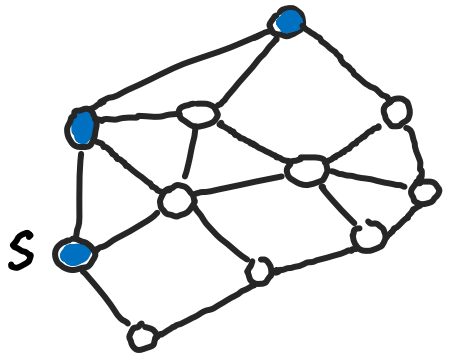
۱) تپه نوردی ساده کمتر در ماکزیمم محلی قرار می‌گیرد.

۲) تپه نوری ساده با سرعت بیشتری حرکت می‌کند اما مسیر طولانی‌تری را می‌یابد.

۳) تپه نوردی از تندترین شیب با سرعت بیشتری حرکت می‌کند اما حافظه بیشتری نیز مصرف می‌کند.

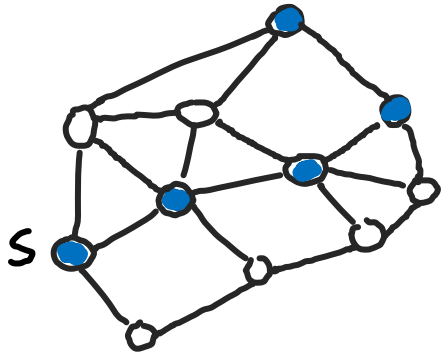
۴) تپه نوردی از تندترین شیب پاسخ بهینه را می‌یابد، در حالی که تپه نوردی ساده اینطور نیست.

Best choice



تعداد گام‌های کمتر اما
مدت زمان اجرای بیشتر

First choice

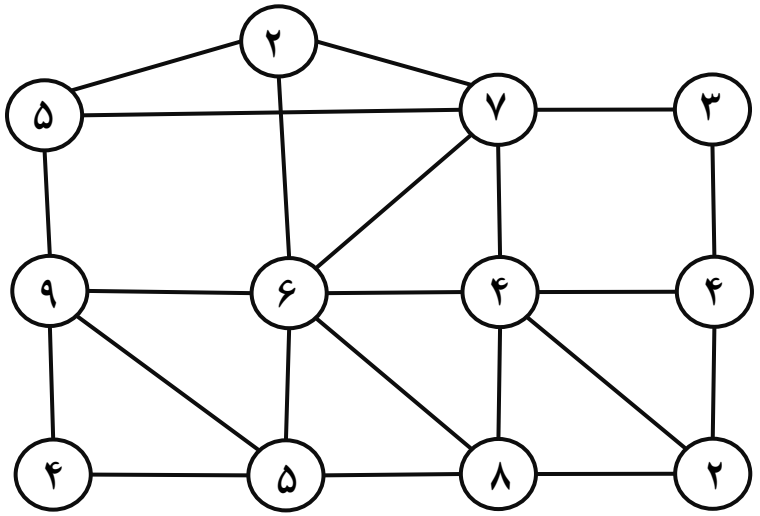


تعداد گام‌های بیشتر اما
مدت زمان اجرای کمتر

رفع مشکل توقف و گیر افتادن در بهینه موضعی

- تکرار از نقاط شروع تصادفی — random restart
- جستجوی پرتو محلی (به‌جای یک گره از یک جمعیت اولیه) — local beam

random restart



چند بار اجرای یک نوردی با شروع محبه دلخواه.

اگر احتمال موفقیت در هر تکرار p باشه

به طور متوسط باید $\frac{1}{p}$ بار چنین راتکرار

کرد تا برای اولین بار به موفقیت (جواب مطلق)

برسیم.

احتمال موفقیت در هر تکرار $= p$

هر بار از یک حالت تصادفی شروع می‌کنیم

تکرارها مستقل هستند.

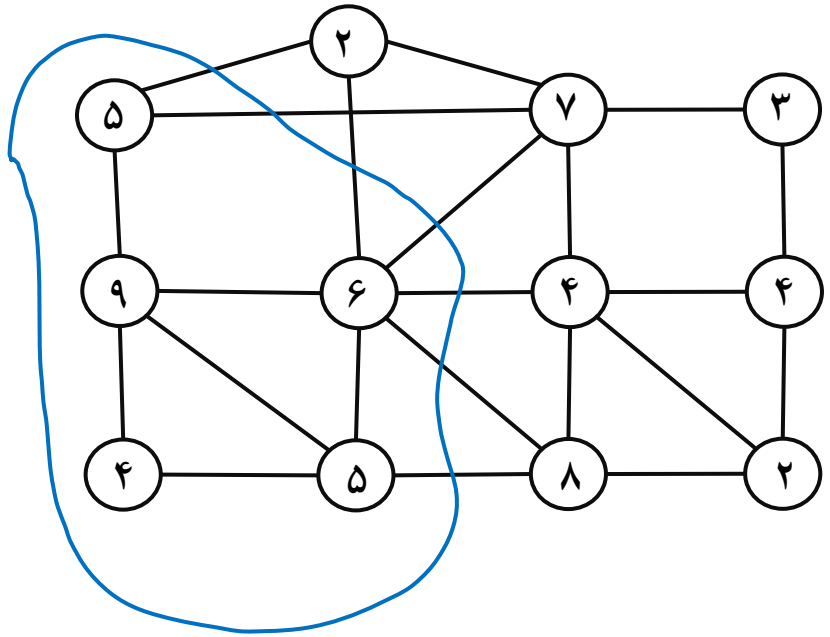
تعداد دفعات $k = \frac{1}{p}$

$k-1$ بار شکست.

1 بار موفقیت.

با احتمال نزدیک به 1، پس از k بار تکرار: جمله‌ی مطلق یافت می‌شود.

$$k = \frac{1}{p}$$



در این شکل ۱۲ گره داریم که با شروع از ۵ تا
می‌توان به حالت بعدی رسید.

$$p = \frac{5}{12}$$

$$\frac{1}{p} = \frac{12}{5} \qquad \left\lceil \frac{12}{5} \right\rceil = 3$$

دس: اگر ۳ بار تیم نوردی را تکرار کنیم با احتمال نزدیک به 1، بعدی مطلق را خواهیم یافت.

۷۸- برای حل یک مسئلهٔ جستجو، از روش‌های محلی تپه‌نوردی استفاده کرده‌ایم. فرض کنید احتمال موفقیت در جستجویی که از یک حالت تصادفی شروع می‌شود، برابر با ۲۵ درصد است. زمانی که جستجو موفقیت‌آمیز باشد، به‌صورت متوسط نیاز به طی کردن ۷ گام دارد و در صورتی که به یک کمینه محلی غیربهینه همگرا شود، به صورت متوسط ۹ گام طی می‌شود. به منظور حصول اطمینان از به جواب رسیدن روش، در صورت همگرایی به کمینه محلی غیر بهینه، از حالت تصادفی اولیه دیگری جستجو را آغاز می‌کنیم. به‌صورت متوسط چند گام برای رسیدن به پاسخ بهینه سراسری باید طی شود؟

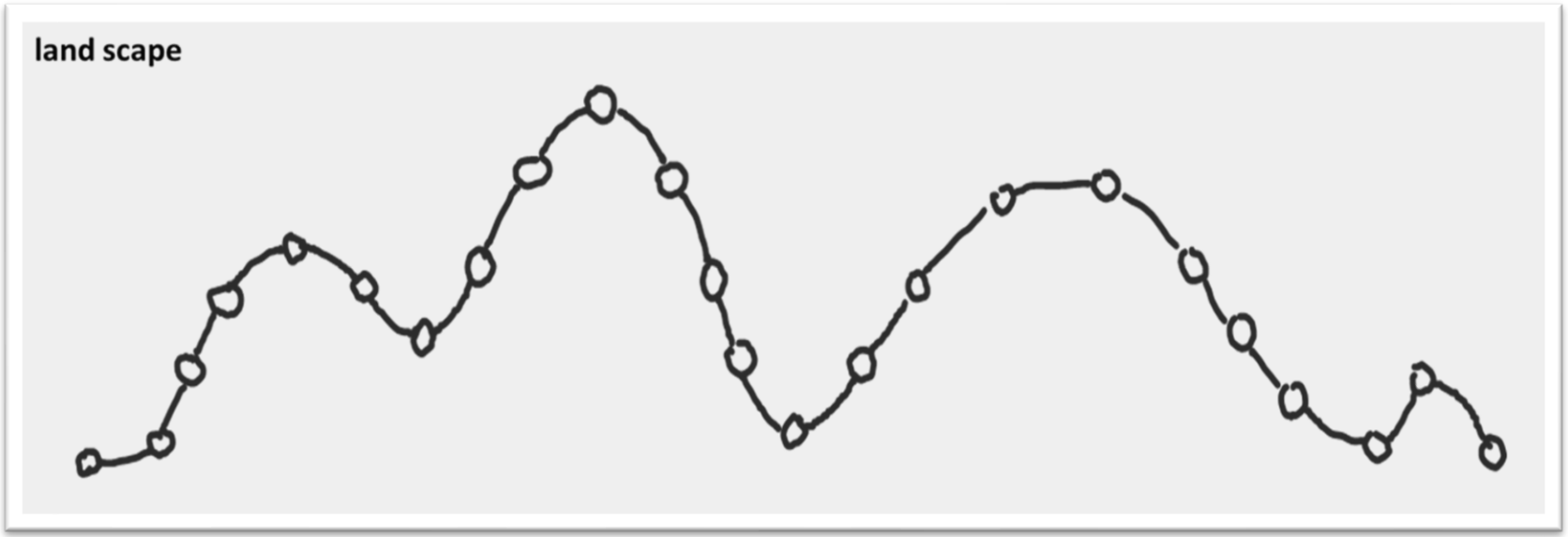
۱) ۲۷
۲) ۲۸
۳) ۳۴
۴) ۴۳

$p = 0.25$ $\quad \frac{1}{p} = 4 \quad \Rightarrow$ ۳ جبری ناموفق 1 موفق

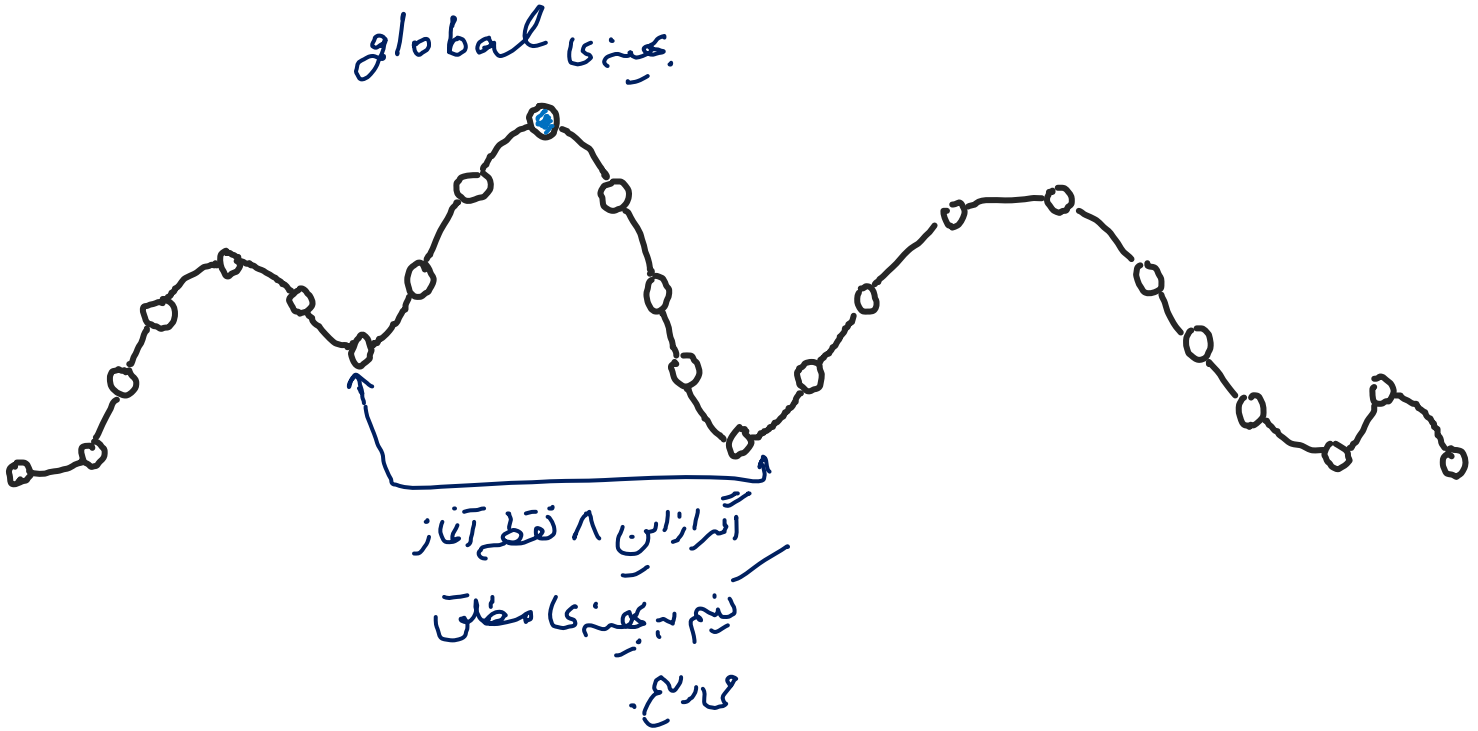
$\Rightarrow \quad 3 \times 9 + 1 \times 7 = 34$

مثال

در فضای حالات زیر ارتفاع همان سودمندی است. چند بار مستقل تپه نوردی را انجام دهیم تا با احتمال نزدیک به یک حالت بهینه به دست آمده باشیم؟



پاسخ:

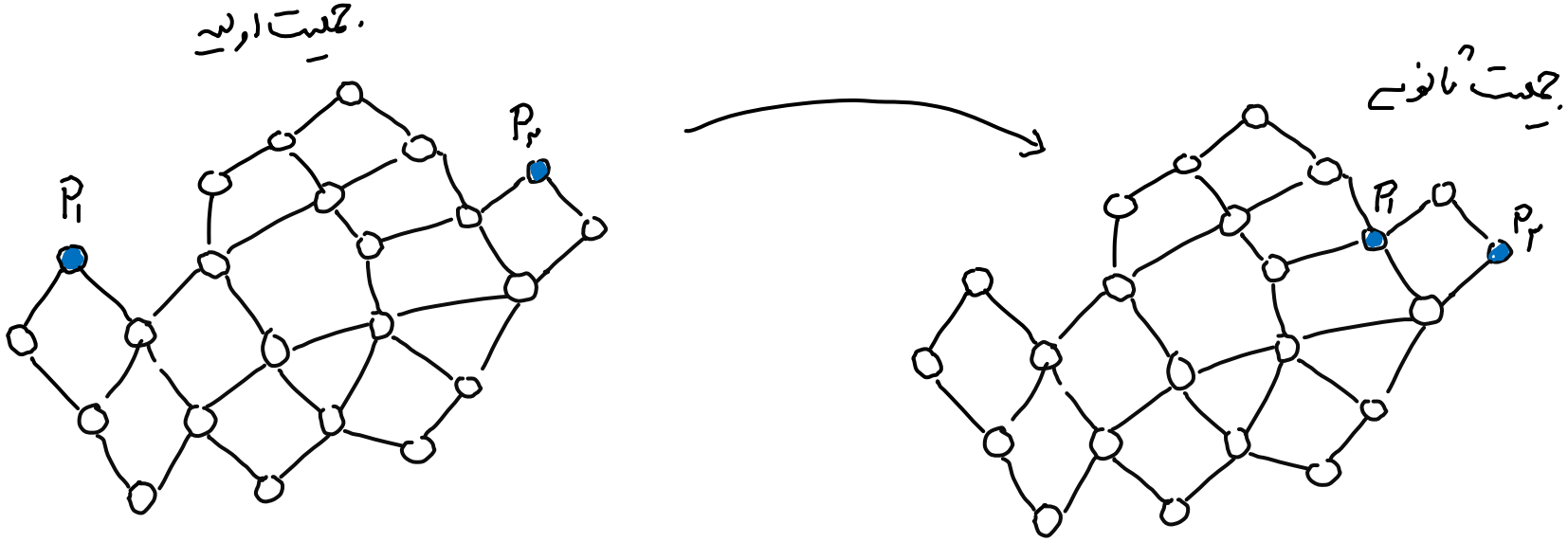


$$p = \frac{۸}{۲۴} \quad \Longrightarrow \quad k = \frac{۱}{p} = ۳$$

Local beam search

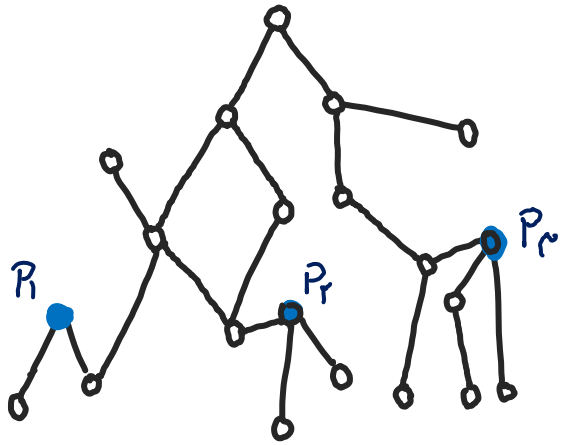
جستجوی پرتوی محلی:

در روش (local beam) اگر $\{P_1, \dots, P_k\}$ اعضای جمعیت اولیه باشد، در مرحله بعدی حتماً از بین همسایگان این نقاط، K عضو بهتر را انتخاب می‌کنیم. حالا ممکن است همه‌ی آن K عضو جدید، بهتر از همه‌ی اعضای جمعیت اولیه باشند. برای مثال به این تصویر توجه کنید:

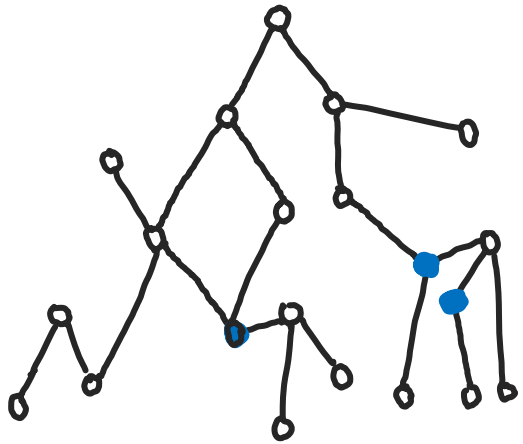


البته درصورتی به اقدامات برگشت‌پذیر باشند، با گذشت زمان ($t \to \infty$) سودمندی جهت بهینه خواهد شد.

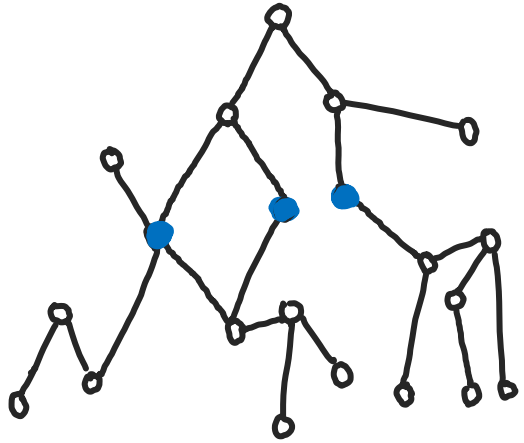
مثال ($k=3$)



نل اولیه $= \{P_1, P_2, P_3\}$



نل دوم

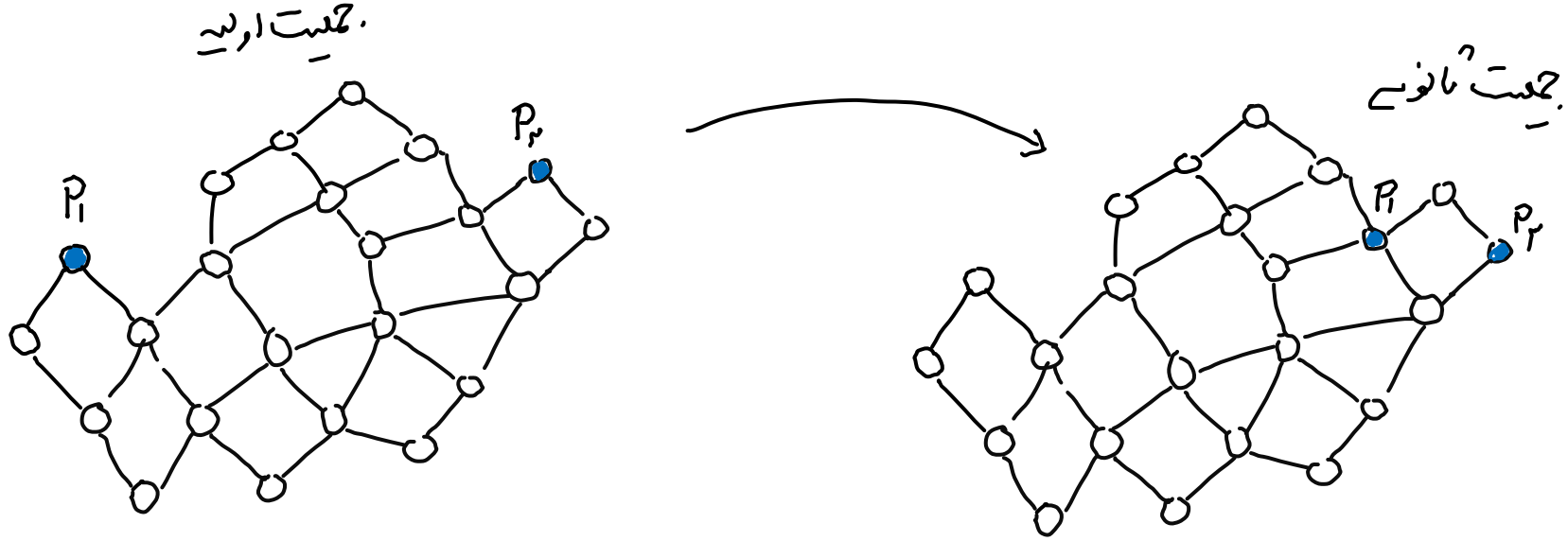


نل سوم

Local beam search

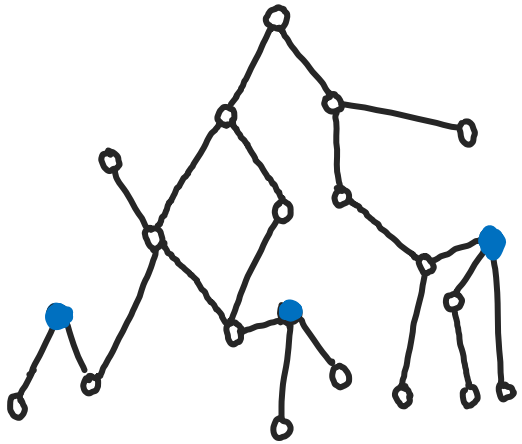
جستجوی پرتوی محلی:

در روش (local beam) اگر $\{P_1, \dots P_k\}$ اعضای جمعیت اولیه باشد، در مرحله بعدی حتماً از بین همسایگان این نقاط، K عضو بهتر را انتخاب می‌کنیم. حالا ممکن است همه‌ی آن K عضو جدید، بهتر از همه‌ی اعضای جمعیت اولیه باشند. برای مثال به این تصویر توجه کنید:

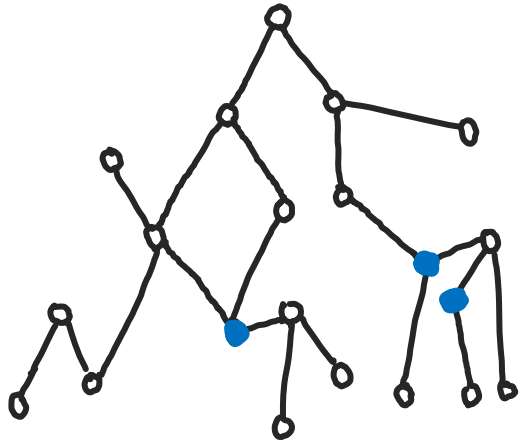


البته درصورتی به اقدامات برگشت پذیر باشند، با گذشت زمان ($t \to \infty$) بهبودی حاصل خواهد شد.

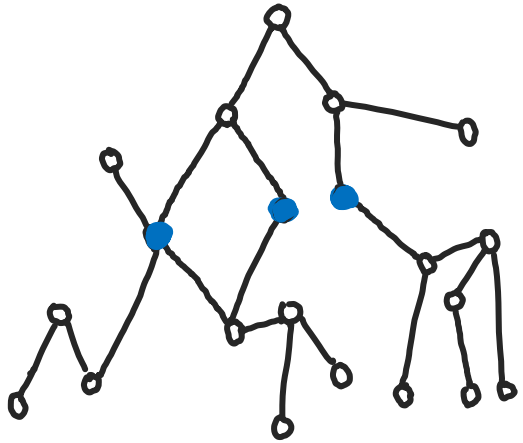
مثال ($k=3$)



نسل اول $\{P_1, P_2, P_4\}$



نسل دوم



نسل سوم

۱۲۴- درخصوص روش‌های جستجوی محلی (local search) کدام جمله درست است؟

۱) در روش local beam search ممکن نیست همهٔ اعضای جمعیت در یک زمان t' (از نظر تابع هدف) بدتر از همهٔ اعضای جمعیت دریکی از زمان‌های قبلی $t < t'$ باشند.

✓ ۲) در روش local beam search چنانچه کنش‌ها برگشت‌پذیر باشند، اعضای جمعیت در طول زمان بهبود پیدا می‌کنند (یا حداقل تنزل پیدا نمی‌کنند.)

۳) با نقطه شروع برابر ممکن نیست hill-climbing به نتیجه بهتری نسبت به (simulated Annealing) برسد.

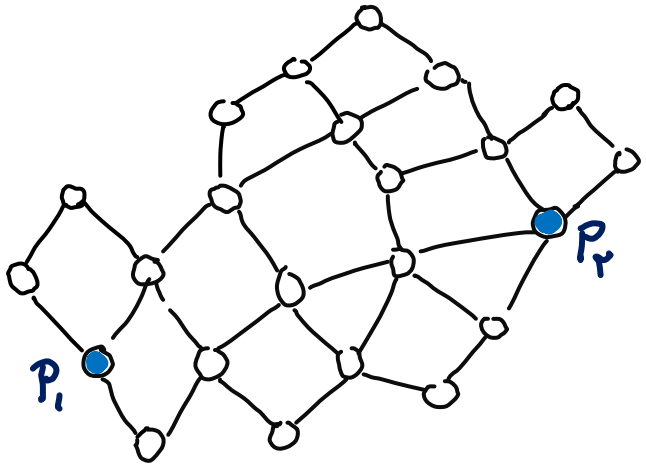
۴) در الگوریتم hill-climbing ممکن است در انتها به نقطه‌ای بدتر از نقطه شروع برسیم.

درست (جواب تست)

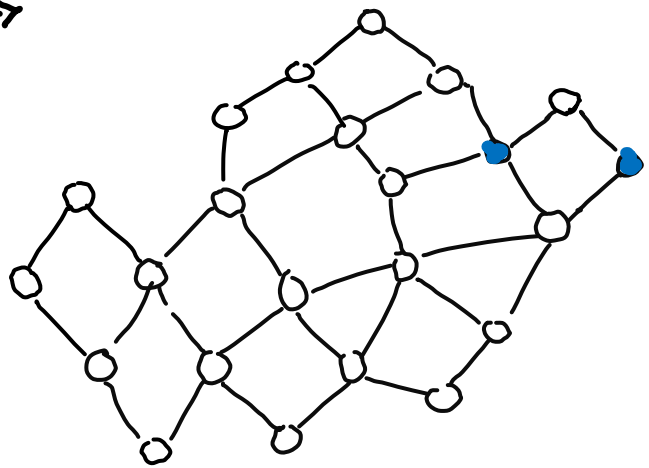
تپه‌نوردی ممکن است به جهت مطلق نرسد اما هرگز به بدتر از نقطه شروع نمی‌شود.

توجه: جستجوی پرتو محلی با جمعیت اولیه k عضوی، با k جستجوی تپه نوردی که به صورت موازی و مستقل از هم اجرا می‌شوند تفاوت دارد. به این مثال به ازای $k=2$ توجه کنید.

جبری درجه‌ی ۲ عضوی:

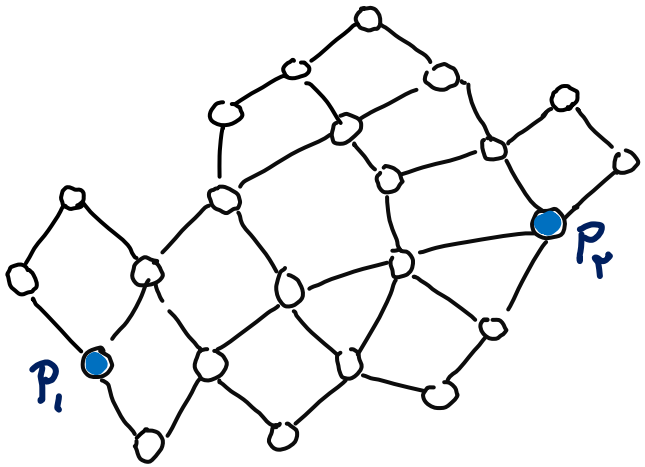


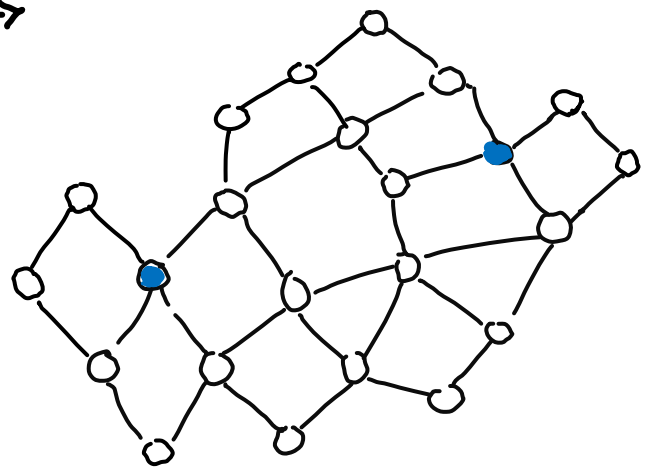
جبری درجه‌ی



اجرای موازی و مستقل ۲=k نیه نوردی:

دو جبری نیه نوردی مستقل





گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

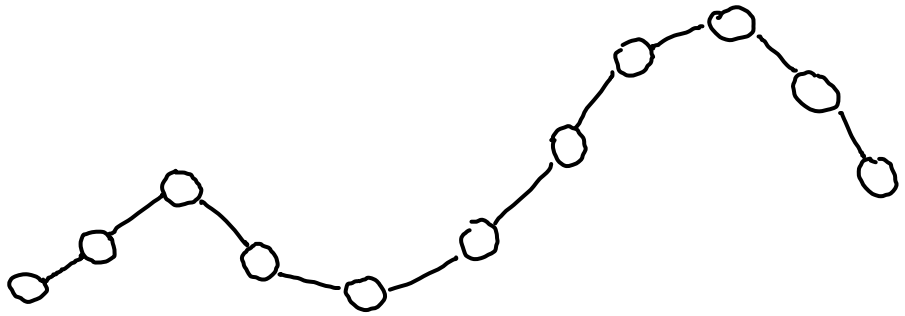
Artificial Intelligence

A Modern Approach

هوش مصنوعی

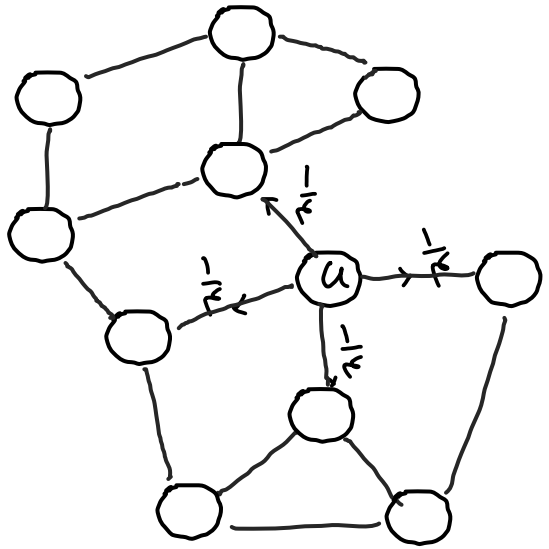
ابوالفضل گیلک
گروه بابان

جلسه ۵ فصل ۴



مشکل بزرگ تیم نوردی، متوقف شدن در بهینه ی موضعی است. برای رفع این مشکل می‌توانیم گاهی اوقات به تصادف (مانند قدم زدن تصادفی) اجازه دهیم از وضعیت بهتر به وضعیت بدتر برود.

قدم زدن تصادفی:



در قدم زدن تصادفی (random) وقتی u دارای m مابعد است احتمال تغییر وضعیت به هر کدام $\frac{1}{m}$ است.

شبیه‌سازی ذوب :

یک متغیر سرعت زمان داریم t

تابع دما (تابع احتمال) وابسته به زمان است $T = T(t)$

در شروع کار دما زیاد است $T(1) \simeq \infty$ $(t=0$ یا $t=1)$

با گذشت زمان دما به صفر میل کند $T(t) \simeq 0$ $t \to \infty$

شرط توقف و خاتمه جستجو: $(T=0)$

توضیح گام به گام شبیه‌سازی ذوب (SA):

۱) در C هستیم. سودمندی V_C است

۲) با احتمال یکنواخت (random) یکی از همسایگان تولید می‌شود: (next)

۳) اگر $V_{next} > V_C$ آنگاه حتماً تغییر وضعیت دهیم: $C \leftarrow next$

۴) اگر $V_{next} \leq V_C$ آنگاه با احتمال $p = e^{-\frac{|V_{next} - V_C|}{T}}$ تغییر وضعیت دهیم.

با احتمال p تغییر وضعیت می‌دهیم: $C \leftarrow next$ و به مرحله‌ی بعدی می‌رویم $t = t+1$.

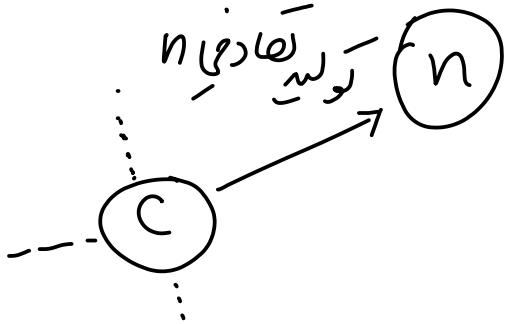
با احتمال $1-p$ در همان وضعیت C می‌مانیم. و به مرحله‌ی بعدی می‌رویم $t = t+1$.

این گام‌ها را آنقدر ادامه می‌دهیم که دما به قدر کافی کم شود: $T = 0$.

شبیه سازی ذوب فلزات :

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
    current ← problem.INITIAL
    for t = 1 to ∞ do
        T ← schedule(t)
        if T = 0 then return current
        next ← a randomly selected successor of current
        ΔE ← VALUE(current) – VALUE(next)
        if ΔE > 0 then current ← next
        else current ← next only with probability e^(−ΔE/T)
```

در واقع $e^{-\frac{|\Delta E|}{T}}$ درست است .



تولید تصادفی n

$v_n > v_c \Longrightarrow$ حتماً: $C \leftarrow n$

$v_n \leq v_c \Longrightarrow$ با احتمال $p = e^{-\frac{|\Delta E|}{T}}$ تغییر وضعیت.

چند تابع مناسب برای $T(t)$ به این صورت است:

$$T(t) = 100\, e^{-t} \qquad t = 1, 2, 3, \ldots$$

$$T(t) = \frac{1000}{1+t} \qquad t = 0, 1, 2, \ldots$$

[در لحظات اولیه $T(t)$ زیاد باشد و وقتی $t \to \infty$ دما به صفر میل کند]

سوال ۲۱:

در یک پیاده سازی از جستجوی محلی simulated annealing پارامتر زمان در بازه $0 \leq t < \infty$ تعریف شده است.
برای آنکه این پیاده سازی به احتمال نزدیک به یک بهینه شود کدام تابع دما مناسب است؟

۱- $T(t) = 10t\, e^{\frac{1}{t+0.1}}$

۲- $T(t) = (10 + t)e^{-\frac{1}{t+0.1}}$

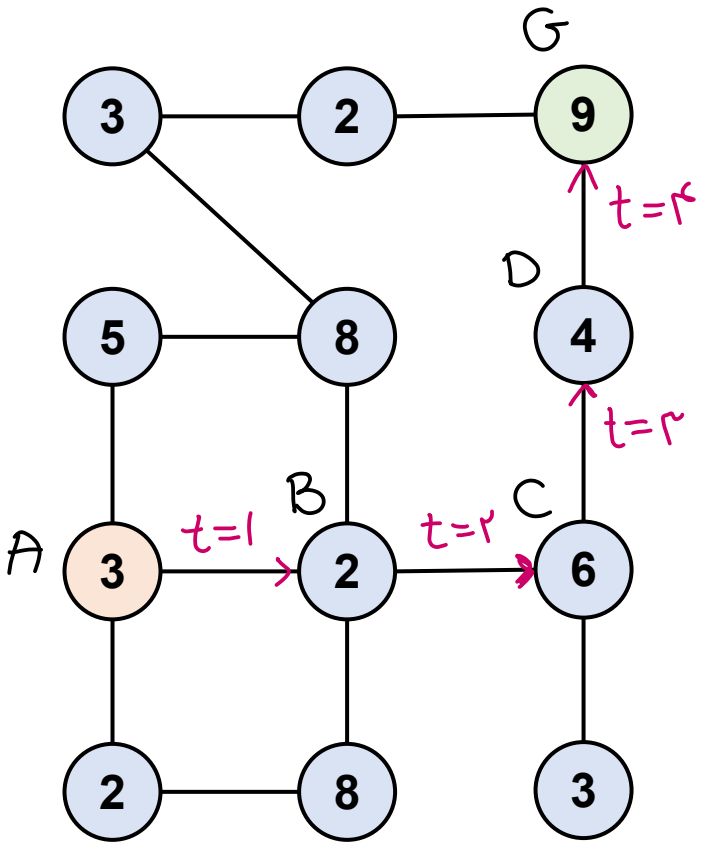
۳- $T(t) = 10te^{-t}$

۴- $T(t) = (10 + t)\, e^{-t}$

در شروع دمای کوره زیاد باشد تا شانس اینکه قدم زدن تصادفی مهیا باشد. پس در $t = 0$ باید $T(0)$ بزرگ باشد تا حد امکان.
با گذشت زمان کوره سرد شود ($t \to \infty \quad T(\infty) = 0$). گزینه (۴) صحیح است.

تمرین: در شبیه سازی ذوب ، با شروع از A ، احتمال آن که در ۴ گام به G برسیم چقدر است؟

$T(t)= 2e^{-t}$ t=1,2,3,..



تنها مسیری که با ۴ گام به G می‌رسد $ABCDG$ است.

احتمال تولید B: $\frac{1}{3}$ احتمال تغییر وضعیت: $p=e^{-\frac{1}{2e^{-1}}}$ است.

احتمال تولید C: $\frac{1}{4}$ احتمال تغییر وضعیت: $p=e^{-\frac{2}{2e^{-3}}}$

احتمال تولید D: $\frac{1}{3}$ احتمال تغییر وضعیت: $p=1$.

احتمال تولید G: $\frac{1}{2}$ احتمال تغییر وضعیت: $p=1$.

$$\text{جواب} = (1)(\frac{1}{2})(e^{-\frac{2}{2e^{-3}}})(\frac{1}{3})(1)(\frac{1}{4})(e^{-\frac{1}{2e^{-1}}})(\frac{1}{3})$$

تحلیل شبیه‌سازی ذوب در ابتدا و انتها:

در شروع کار $T \approx \infty$ پس $p = e^{-\frac{\Delta E}{\infty}} = e^{0} = 1$. بنابراین مانند قدم زدن تصادفی است:

کولیس با احتمال یکسوم گفت انتها

اگر ارزش بالاتری نسبت به قبلی داشت: حتماً به آن حالت خواهیم رفت (با احتمال یک)

اگر ارزش پایین‌تری نسبت به قبلی داشت: حتماً به آن حالت خواهیم رفت. ($p=1$)

در پایان کار $T \simeq 0$ دس: $p = e^{-\frac{\Delta E}{0^+}} = e^{-\infty} = 0$ بنابراین مانند تکیه نوردی اولین انتخاب است.

کولیس با اتمام نگرفت آنطان

اگر ارزشی بالاتری نسبت به فعلی داشت؛ حتماً به آن حالت خواهیم رفت (با اتمام نگرفت)

اگر ارزشی پایین تری نسبت به فعلی داشت: قطعاً به آن حالت نخواهیم رفت. ($p=0$)

دمای کوره:

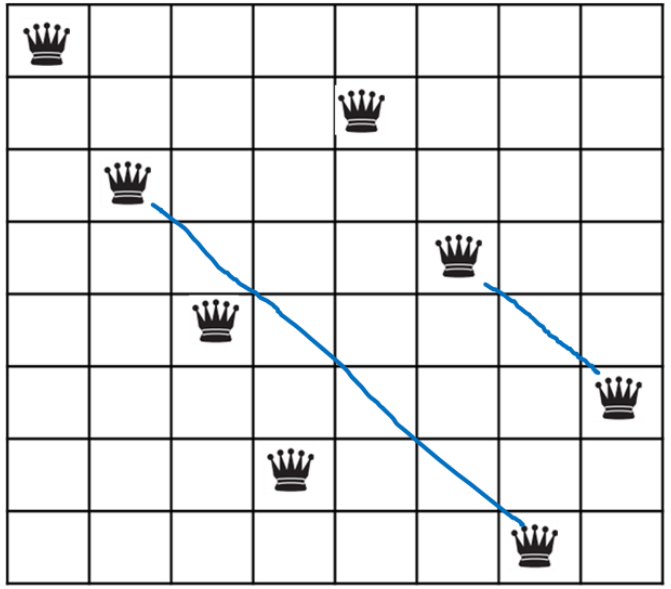
$T \approx \infty$ → قدم زدن تصادفی

$T \to 0$ → تپه نوردی اولین انتخاب (ساده)

می دانیم که هر حالت از محیط دارای یک تابع سودمندی (ارزندگی) fitness است.

مثلاً در ۸-وزیر کلاً $\binom{8}{2} = 56$ جفت وزیر داریم تعداد جفت‌های متقاطع را (هزینه) و تعداد جفت‌های غیر متقاطع را سودمندی، تلقی می‌کنیم.

u: [1 3 5 7 2 4 8 6]



$\binom{8}{2} = \frac{7 \times 8}{2} = 28$

$C(u) = 2$ تعداد برخوردها

ارزندگی (لودمنی)

$f(u) = 26$

هر عضو از فضا را یک کروموزوم و مولفه های آن را ژن ها می نامیم. مثلاً u: [1 3 5 7 2 4 8 6]

یک کروموزوم است دارای ۸ ژن.

شبه کد الگوریتم ژنتیک

```
function GENETIC-ALGORITHM(population, fitness) returns an individual
  repeat
    weights ← WEIGHTED-BY(population, fitness)
    population2 ← empty list
    for i = 1 to SIZE(population) do
      parent1, parent2 ← WEIGHTED-RANDOM-CHOICES(population, weights, 2)
      child ← REPRODUCE(parent1, parent2)
      if (small random probability) then child ← MUTATE(child)
      add child to population2
    population ← population2
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to fitness

function REPRODUCE(parent1, parent2) returns an individual
  n ← LENGTH(parent1)
  c ← random number from 1 to n
  return APPEND(SUBSTRING(parent1, 1, c), SUBSTRING(parent2, c + 1, n))
```

مراحل الگوریتم ژنتیک :

① انتخاب K کروموزوم برای نسل اولیه { تصادفی (انتخاب K کروموزوم) / تورنمنت (از K دسته، هرکدام یک برنده وارد نسل اول شود)

هر عضو از نسل اول دارای تابع برازش است: $f(u)$ ، احتمال انتخاب u به‌عنوان والد برابر است با:

$$p(u) = \frac{f(u)}{\sum f(x)} \times 100$$

② انتخاب والدین (K والد می خواهیم) هر عضو ممکن است چندین بار به‌عنوان والد انتخاب شود.

③ تقاطع ، برخورد، ترکیب، ادغام

④ امکان جهش تصادفی ژن ها

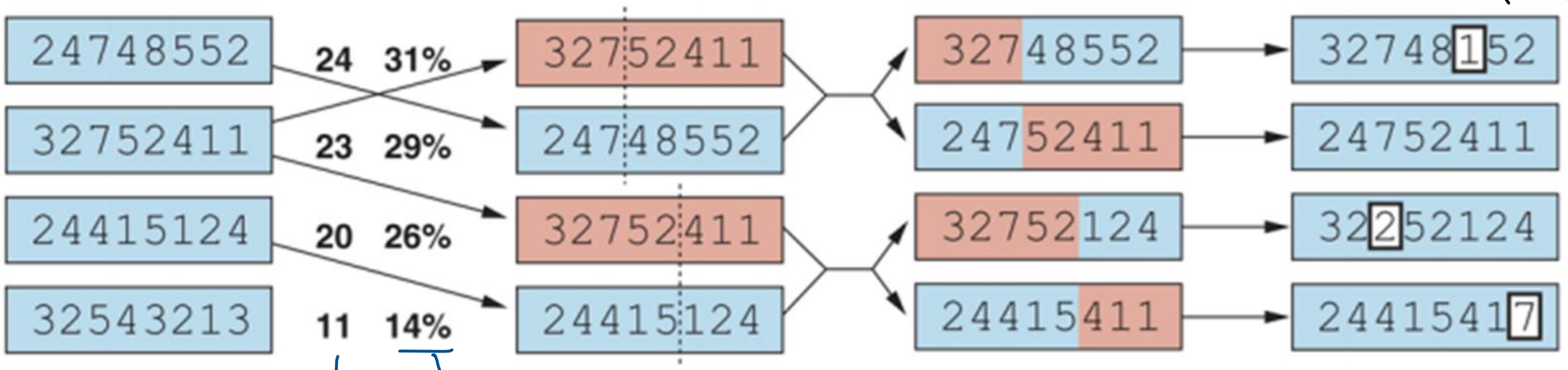
نسل اولی
Initial population

انتخاب
Selection

تقاطع
Crossover

جهش
Mutation

نسل دوم



Fitness function

$$P = \frac{11}{24+23+20+11} \times 100 = 14\ \%$$

اگر اندازه نسل اول k باشد، نسل‌های بعدی هم k تا هستند. هرچه k بزرگ‌تر باشد احتمال یافتن حالت بهینه افزایش می‌یابد.

دو نوع شرط توقف الگوریتم ژنتیک:
- تعداد نسل‌ها (رسیدن به نسل n اُم).
- رسیدن به مقدار مشخصی از f در نسل فعلی.

توجه: هر عضو از نسل فعلی با بارآوری f_i و احتمال انتخاب $p_i = \frac{f_i}{\sum f_j} \times 100$ می‌تواند چندین بار به‌عنوان والد انتخاب شود. مثلاً برای دومین عنصر:

$$p_2 = \frac{f_2}{f_1 + f_2 + \dots + f_k}$$

احتمال آن‌که دقیقاً ۲ بار به‌عنوان والد انتخاب شود:

$$\binom{k}{2} p_2^2 (1-p_2)^{k-2}$$

احتمال آن‌که هرگز به‌عنوان والد انتخاب نشود:

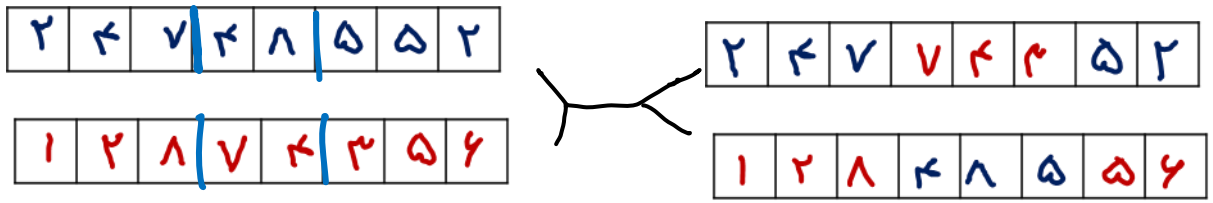
$$\binom{k}{0} p_2^0 (1-p_2)^k = (1-p_2)^k$$

روش های تقاطع (ادغام):

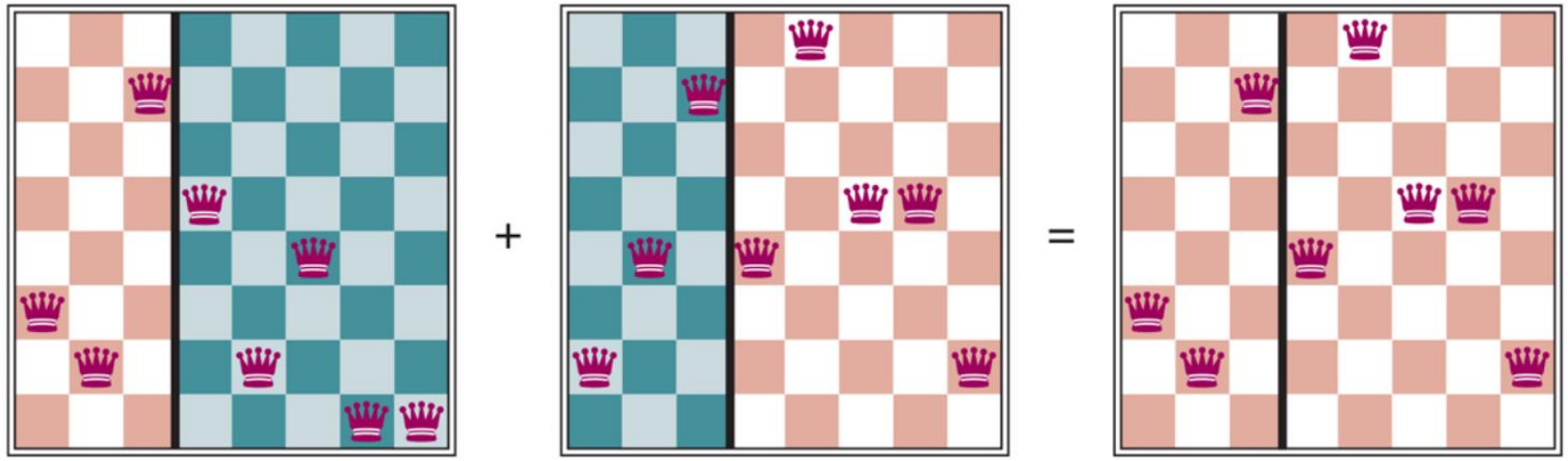
Crossover

- یک، دو یا چند نقطه‌ای
- یکنواخت (uniform)

two-point



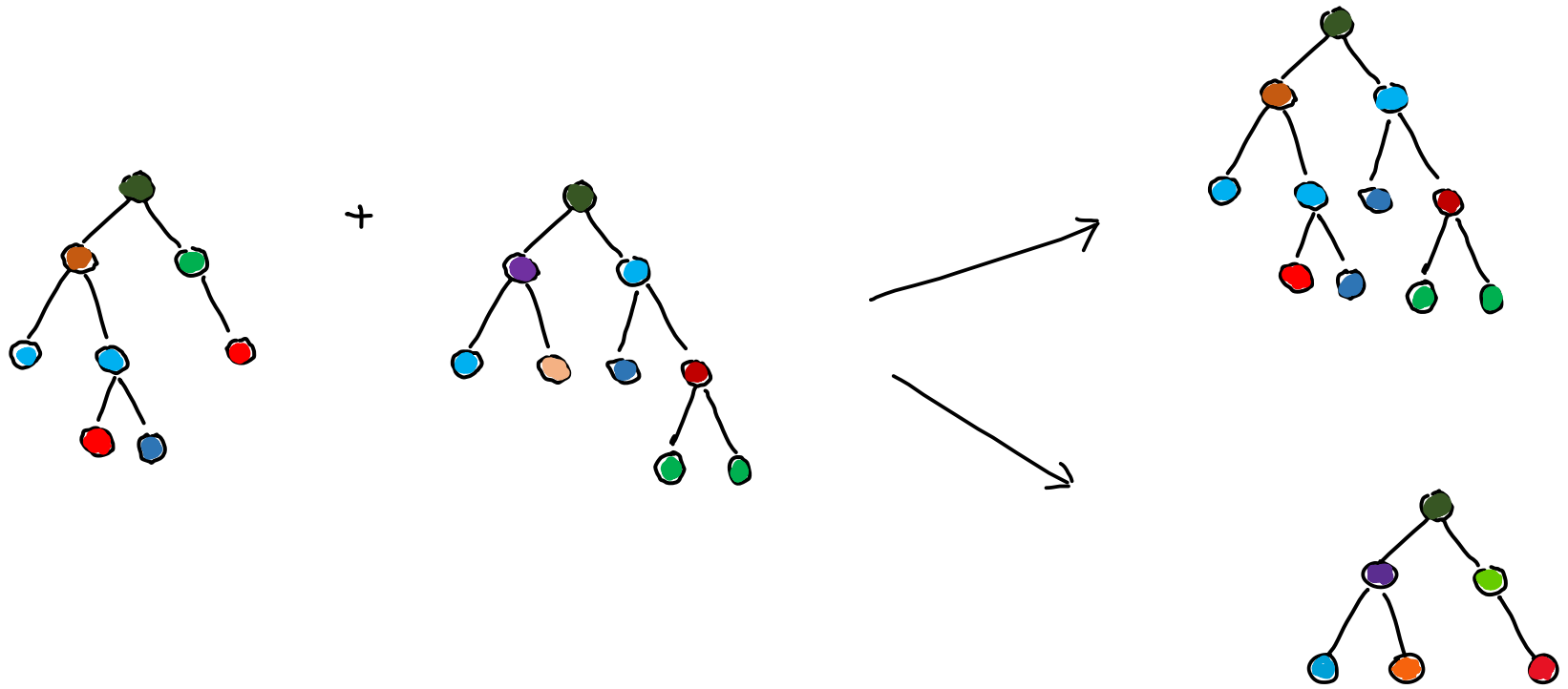
مثال از Crossover تک نقطه‌ای



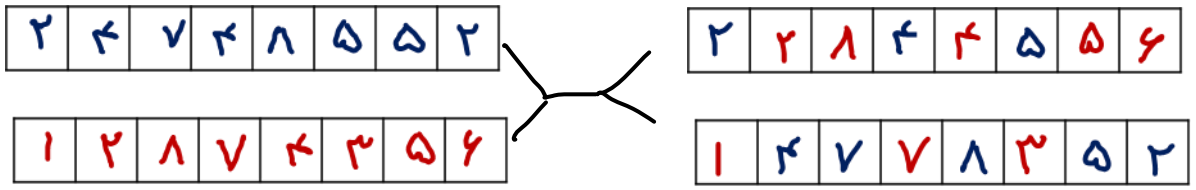
[6 7 2 | 4 7 5 8 8] + [7 5 2 | 5 1 4 4 7] = [6 7 2 5 1 4 4 7]

گاه نهایی درخت دودویی برای هوکرموزوم به کار می‌رود. با استفاده از تلفیق زیردرختان چیاورات، crossover انجام می‌شود.

مثلاً:



uniform



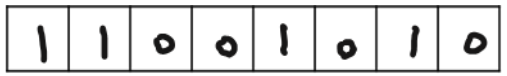
در روش یکنواخت هر ژن با احتمال $\frac{1}{2}$ از یکی از والدین گرفته می‌شود.

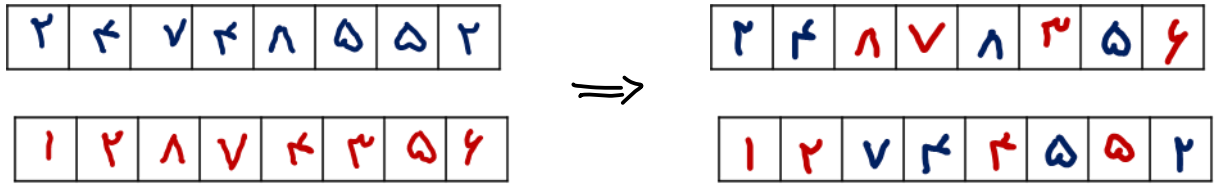
برای پیاده‌سازی این روش ابتدا دنباله‌ای bit mask تولید می‌شود. سپس در هر ژن:

1 ⟶ جابه‌جایی ژن‌ها

0 ⟶ عدم جابه‌جایی ژن‌ها

bit mask





بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

بخش دوم جلسه ۵

learning real-time A* (**LRTA***)

جستجوی محلی آگاهانه و دارای توان یادگیری برای خروج از مینیمم‌های یا مکان‌نماهای موضعی

برای هر همسایه مانند n':

$$cost(n') = c(n;n') + H[n']$$

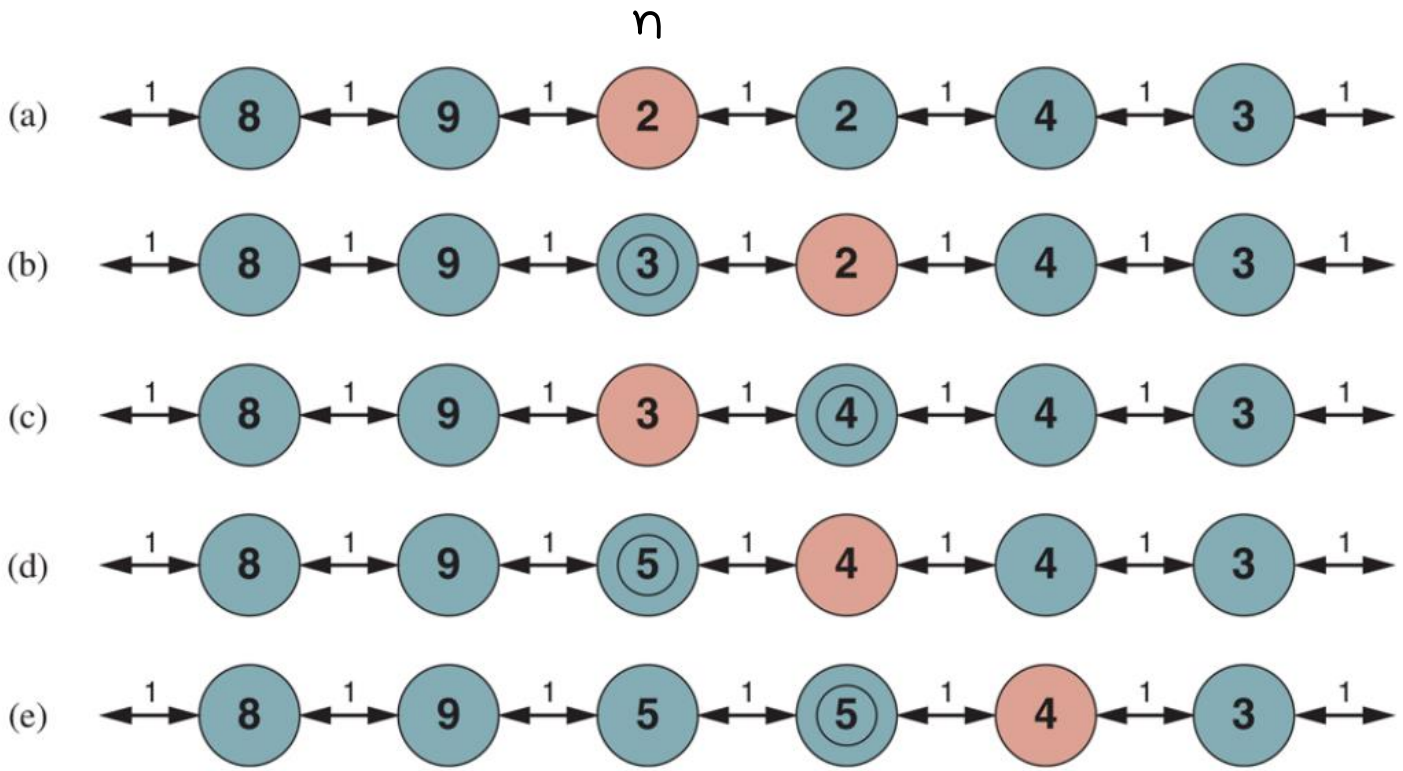
همسایه‌ی با کمترین هزینه را m بنامیم.

پیش از تغییر وضعیت از n به m مقدار $H[n]$

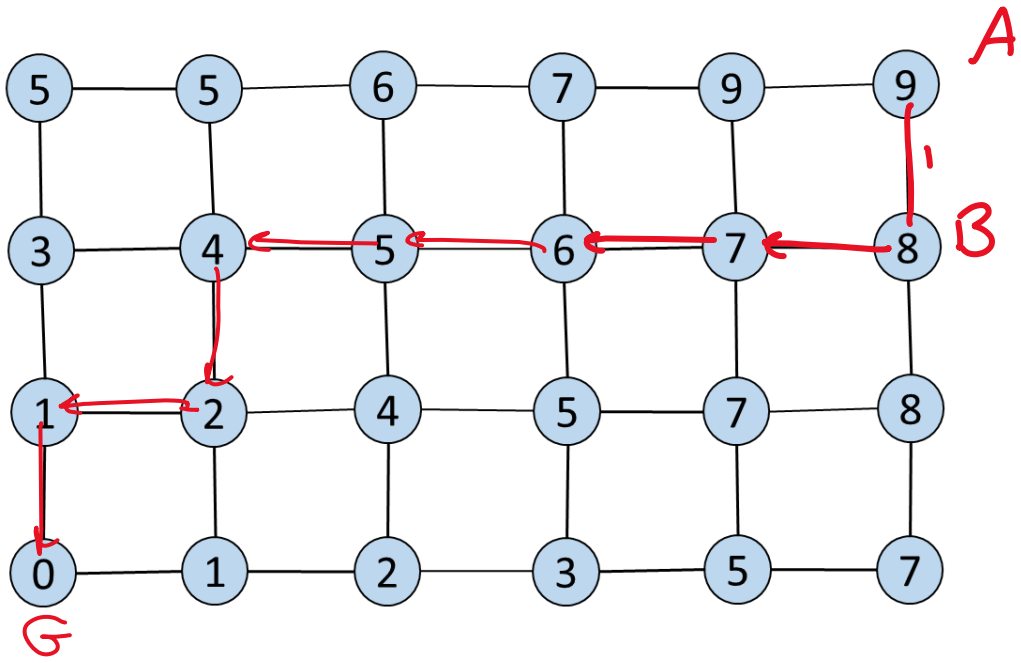
به روزرسانی می‌شود:

$$H[n] \leftarrow cost(m)$$

Online local search

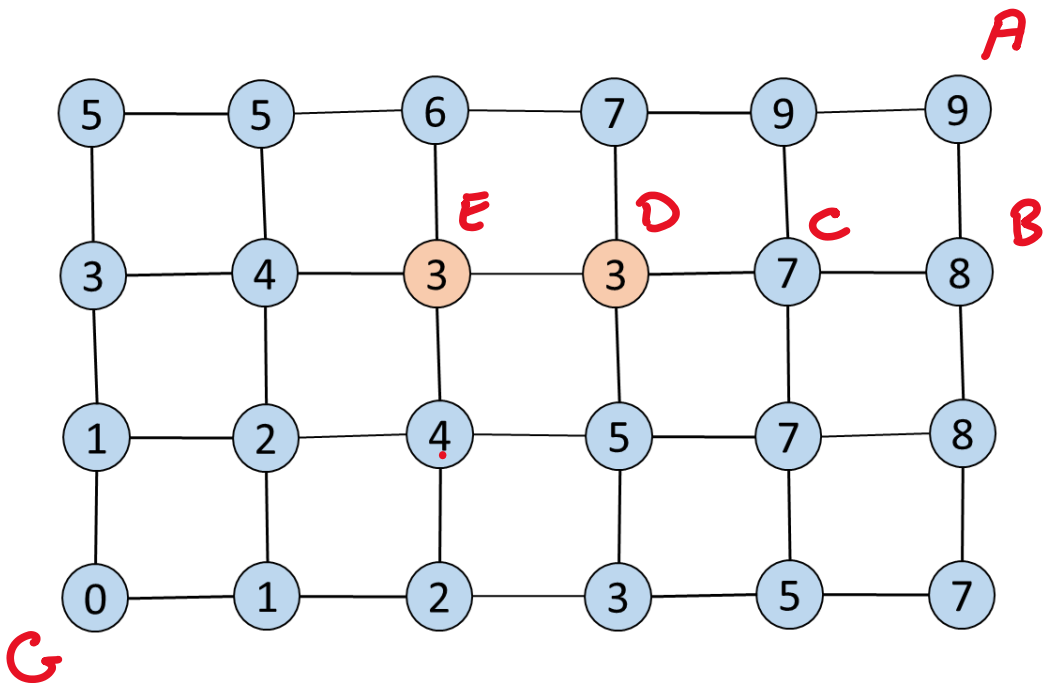


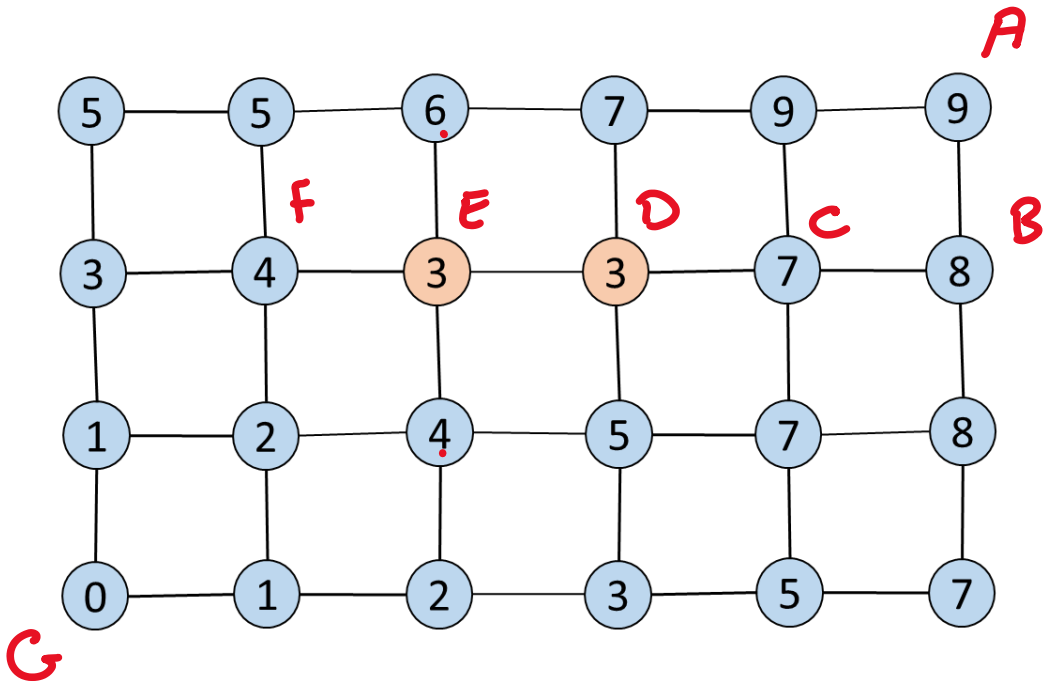
در این مثال مقادیر $H(n)$ تخمین فاصله تا هدف هستند و عامل از A بدون مشکل به G میرسد زیرا تخمین ها تا حدی دقیق هستند. هزینه اقدامات 1 است.



در این مثال در D و E یک بهینه موضعی داریم و بدون استفاده از یادگیری LRTA* عامل در آنجا متوقف می‌شود.

اما با استفاده از LRTA* از آن خارج می‌شود:





دی از خروج از A به B :

$H(A) = 1+8 = 9$

دی از خروج از B به C : $H(B) = 1+7 = 8$

دی از خروج از C به D : $H(C) = 1+3 = 4$

از D به E : $H(D) = 1+3 = 4$

از E به D : $H(E) = 1+4 = 5$

به این ترتیب $H(E)$ و $H(D)$ برابر می شوند

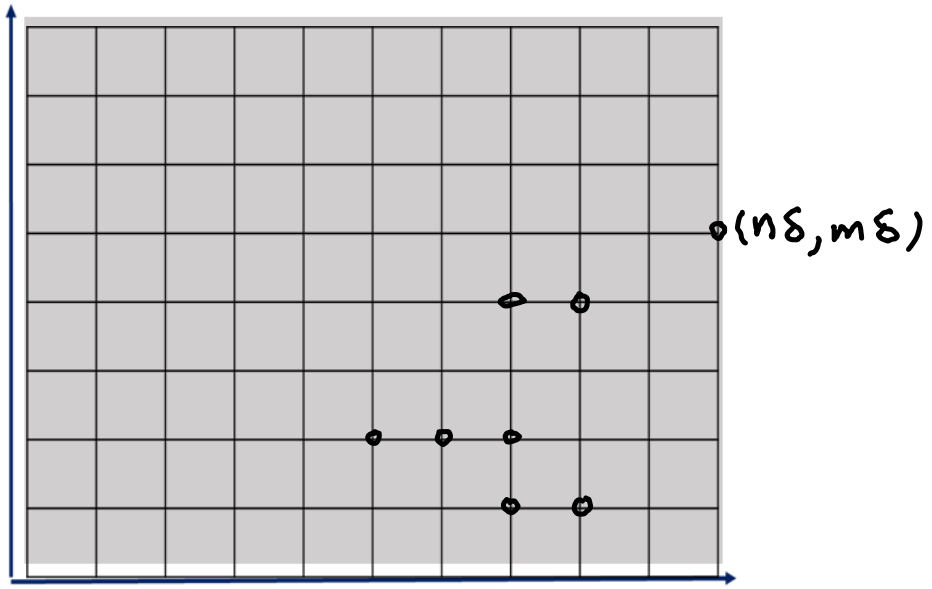
و در نهایت عامل از D و E

خارج شده به سمت G می رود.

جستجو در محیط‌های پیوسته

- گسسته سازی
- بردار گرادیان برای کمترین صعود
- روش جستجوی نیوتن رفسون

LOCAL SEARCH IN CONTINUOUS SPACES



در فضای دو بعدی پیوسته:

حالات: هر نقطه (x,y) یک حالت است.

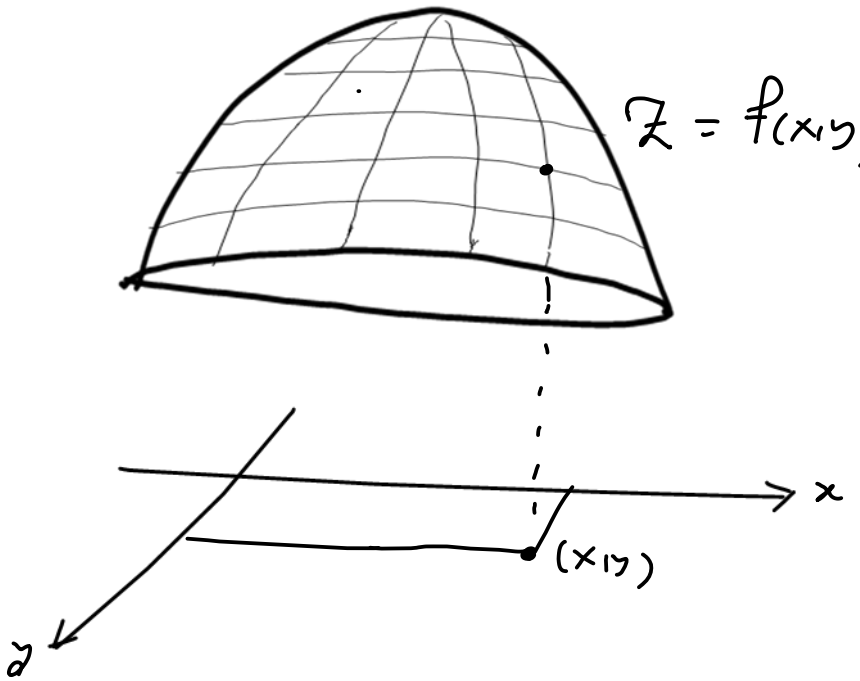
مثل: هر نقطه‌ی (x,y) دارای بی‌نهایت همسایه است.

راه حل کار کردن در فضای پیوسته: گسسته سازی با استفاده از شبکه $\delta >$

discretize گسسته سازی:

حالا حالات: $(n\delta, m\delta)$ هستند.

مثال: سودمندی در هر حالت (x,y) با تابع $z=f(x,y)$ داده شده است:



$$z = f(x,y) = 10 - x^2 - y^2 + 4x$$

از حالت (x,y) به کدام حالت برویم که بیشترین افزایش سودمندی رخ دهد؟

جواب: جهت بردار گرادیان

بردار گرادیان (جهت بیشترین صعود) (تندترین صعود)

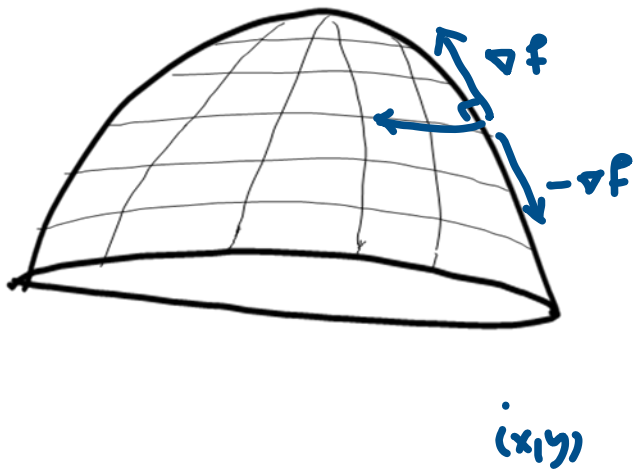
$f(x,y)$ تابع

$$\vec{\nabla} f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

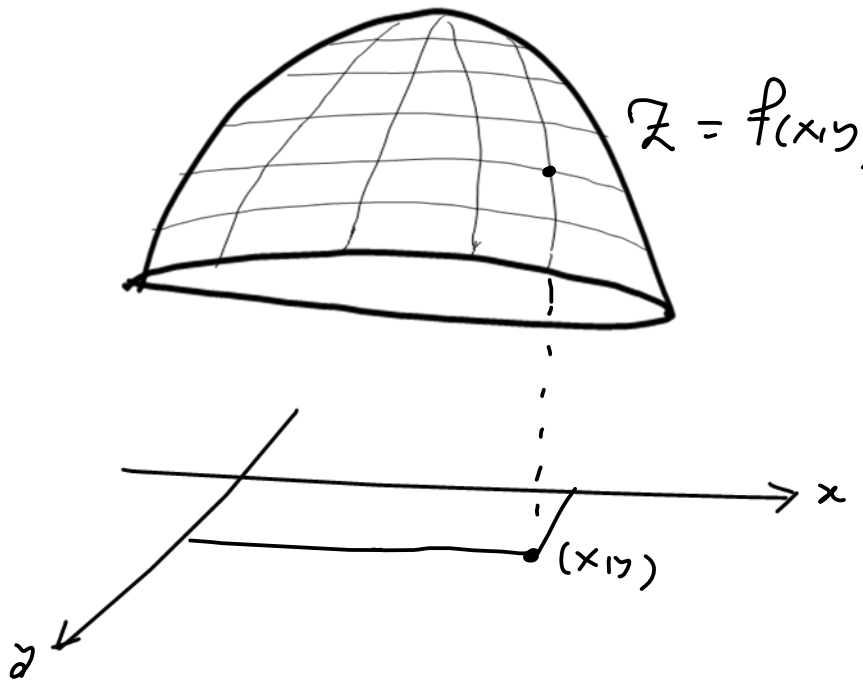
در جهت بردار گرادیان، تندترین صعود (شیب مثبت) را داریم.

در جهت $-\vec{\nabla} f$ تندترین نزول (شیب منفی) را داریم.

در جهت عمود بر گرادیان، مقدار f تغییری نمی‌کند.



مثال:

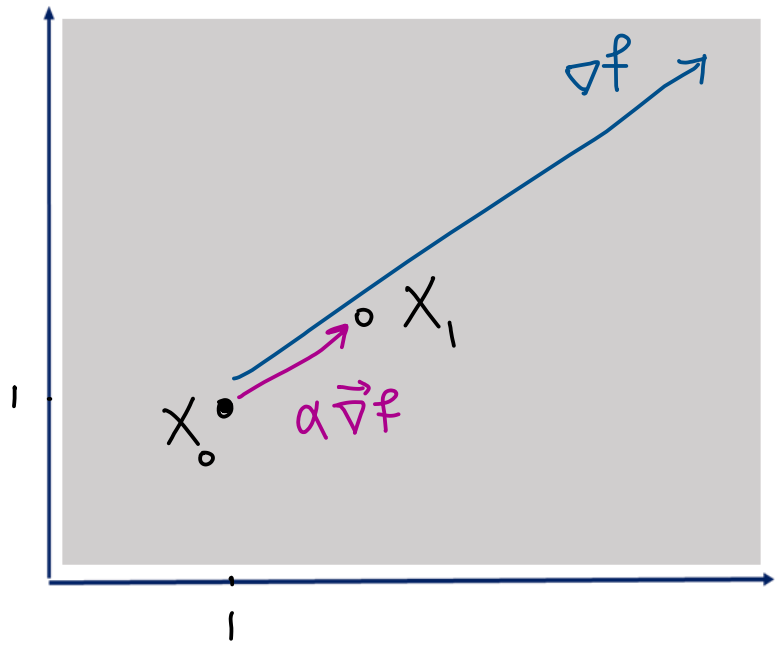


$$z = f(x,y) = 10 - x^2 - y^2 + 2x$$

در این مثال:

گرادیان $\vec{\nabla} f = \begin{bmatrix} -2x+2 \\ -2y \end{bmatrix}$

مثال: تابع $f(x,y) = x^2 + 2y^2 + 6x$ را در نظر بگیرید.



اگر در حالت $X_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ باشیم حالت بعدی چگونه به دست می‌آید.

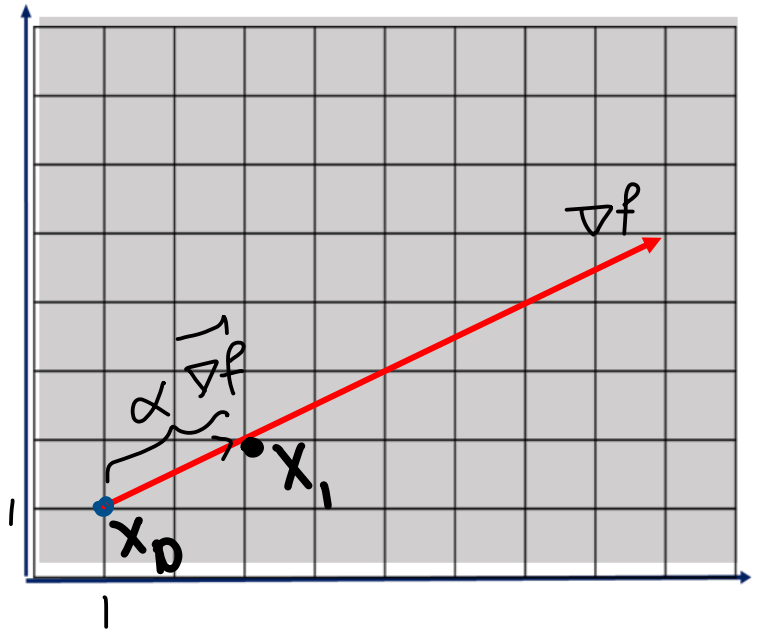
$$X_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad f(X_0) = 1 + 2 + 6 = 9$$

$$\vec{\nabla} f = \begin{bmatrix} 2x + 6 \\ 4y \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \end{bmatrix}$$

در جهت بردار $\begin{bmatrix} 8 \\ 4 \end{bmatrix}$ البته با اندازه‌ی قدم مناسب، حرکت می‌کنیم

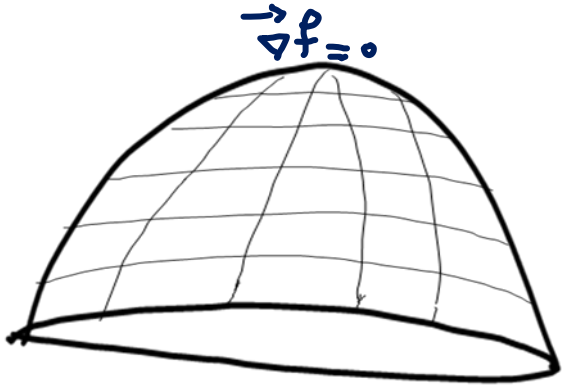
step size $\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x})$

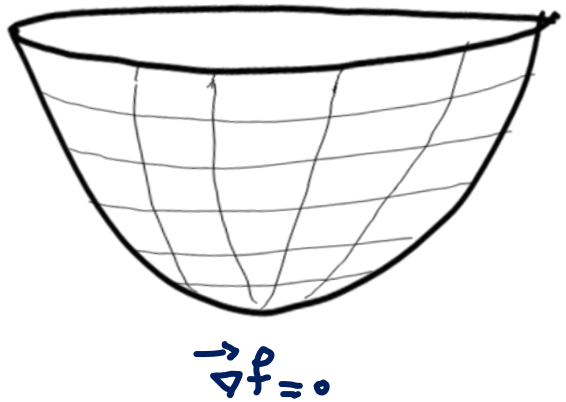
مثلاً با طول گام $\alpha = 0.2$ داریم:



$$X_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0.2 \begin{bmatrix} 8 \\ 4 \end{bmatrix} = \begin{bmatrix} 2.6 \\ 1.8 \end{bmatrix}$$

نیوتن رفسون : یافتن نقاط بحرانی . در این روش به دنبال یافتن نقطه ای هستیم که $\vec{\nabla} f = 0$ شود.





$\mathbf{H}_f(\mathbf{x})$ is the **Hessian** matrix of second derivatives, $\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}_f^{-1}(\mathbf{x})\nabla f(\mathbf{x})$

$$H_f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}$$

ماتریس هسین:

مثلاً برای توابع دو متغیره:

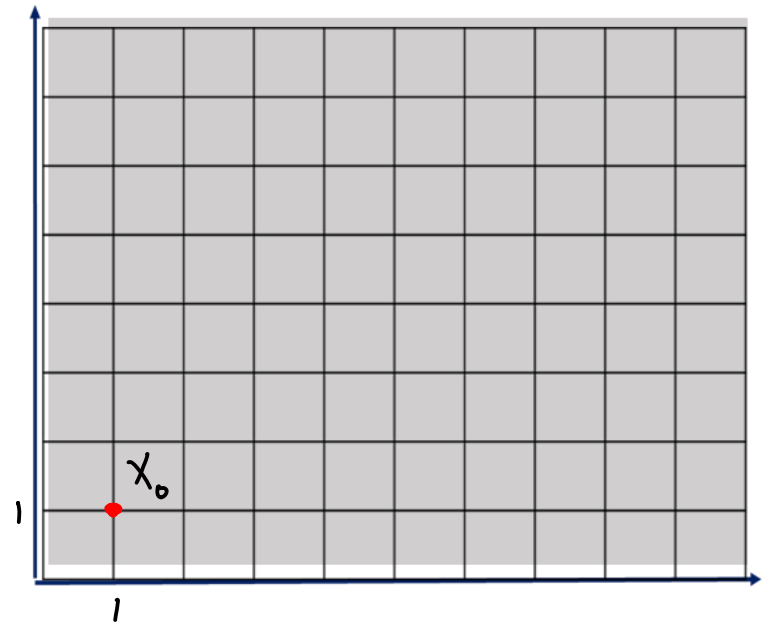
$$H = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix}$$

مثال روش جستجوی محلی با متد تبرید تابع سودمندی زیر:

$$f(x,y) = 2xy + x^2y$$

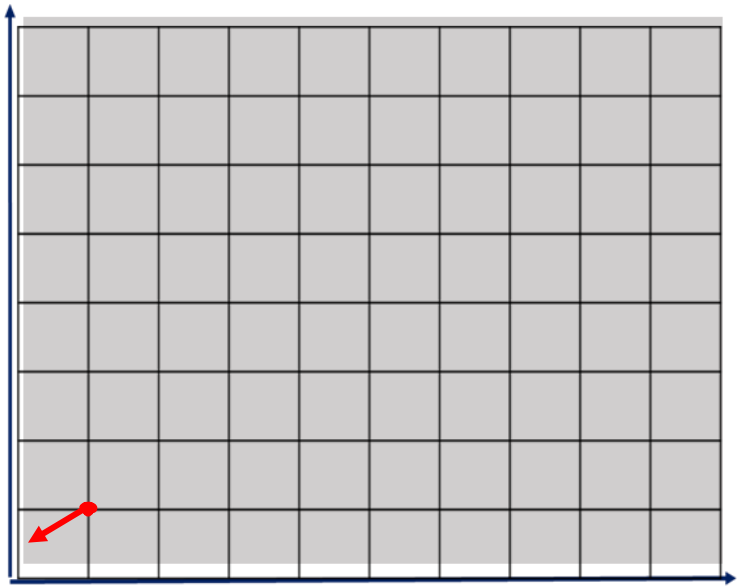
اگر از حالت اولیه‌ی $X_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

آغاز کنیم نقاط بعدی چگونه به دست می‌آیند؟



$$f(x,y) = 2xy + x^2 y$$

$$\nabla f = \begin{bmatrix} 2y + 2xy \\ 2x + x^2 \end{bmatrix} = \begin{matrix} f_x \\ f_y \end{matrix}$$



$$H = \begin{bmatrix} 2y & 2+2x \\ 2+2x & 0 \end{bmatrix}$$

$$\vec{\nabla} f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2y + 2xy \\ x^2 + 2x \end{bmatrix}$$

$$H_f = \begin{bmatrix} 2y & 2+2x \\ 2+2x & 0 \end{bmatrix} \qquad H_f^{-1} = \frac{1}{-(2+2x)^2} \begin{bmatrix} 0 & -2-2x \\ -2-2x & 2y \end{bmatrix}$$

$$\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}_f^{-1}(\mathbf{x}) \nabla f(\mathbf{x})$$

حالا هر نقطه X_i را در لیست راست تکرار هم نقطه اولیه‌ی در لیست چپ به دست می‌آوریم.

$$\begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} x \\ y \end{bmatrix} - \frac{1}{-(2+2x)^2} \begin{bmatrix} 0 & -2-2x \\ -2-2x & 2y \end{bmatrix} \begin{bmatrix} 2y + 2xy \\ x^2 + 2x \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} x \\ y \end{bmatrix} - \frac{1}{-(2+2x)^2} \begin{bmatrix} 0 & -2-2x \\ -2-2x & 2y \end{bmatrix} \begin{bmatrix} 2y+2xy \\ 2x+x^2 \end{bmatrix}$$

در ست راست $x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ را قرار دهیم تا قدم بعدی به دست آید:

$$X_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \frac{1}{16} \begin{bmatrix} 0 & -4 \\ -4 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \frac{1}{16} \begin{bmatrix} -12 \\ -10 \end{bmatrix}$$

$$\simeq \begin{bmatrix} 0.25 \\ 0.4 \end{bmatrix}$$

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

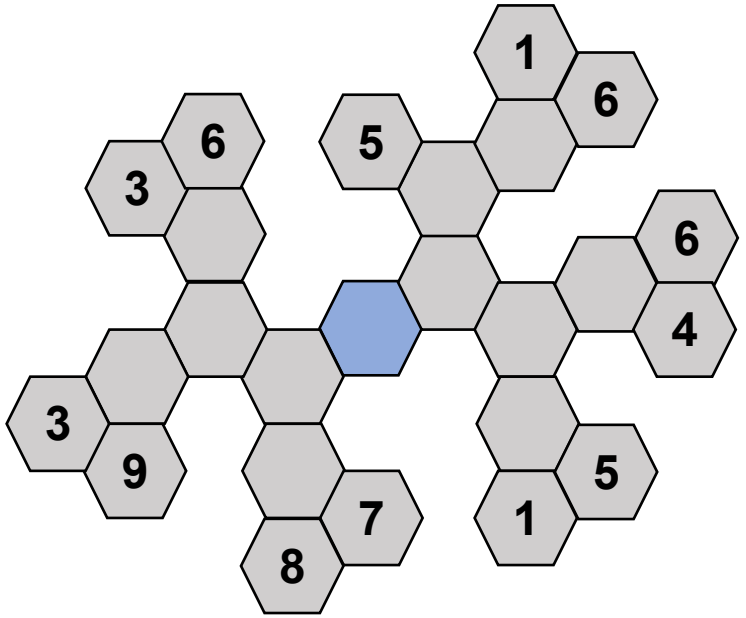
ابوالفضل گیلک
گروه بابان

جلسه ۶ فصل ۵

Adversarial Search and Games

محیط رقابتی:
- دونفره: عامل جست‌وجو کننده (MAX) در برابر رقیب (MIN)
- چندنفره:

محیط رقابتی:
- مجموع صفر: یعنی مجموع سودمندی‌ها ثابت باشد (مثلاً در شطرنج مجموع امتیازات 1 است)
- مجموع غیرثابت: مجموع سودمندی‌ها می‌تواند افزایش یا کاهش یابد.



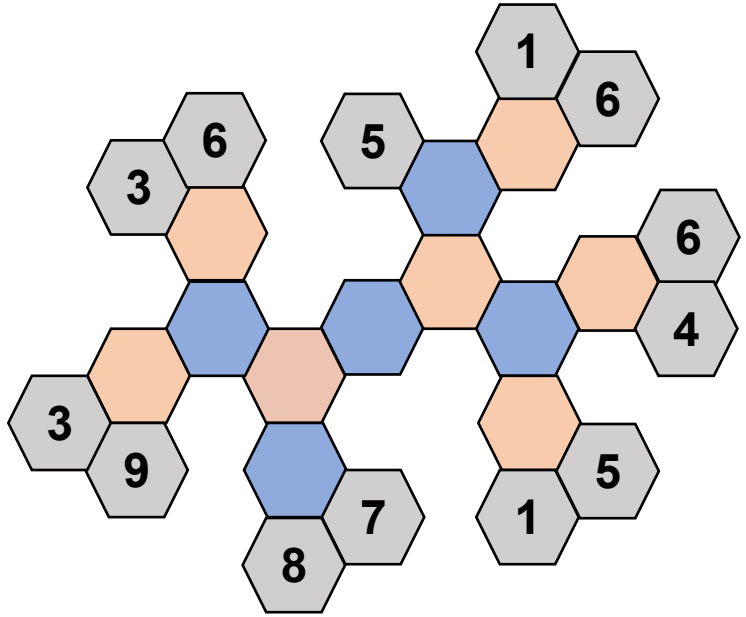
مثال عامل در محل آبی است و شروع کننده بازی است.

عامل و رقیب یکی در میان حرکت می کنند.

هر بار به یکی از خانه های مجاور که تاکنون کسی به آن وارد نشده

می رود. اعداد، امتیازهای عامل هستند.

در شکل زیر نوبت های عامل و رقیب رنگ آمیزی شده اند.

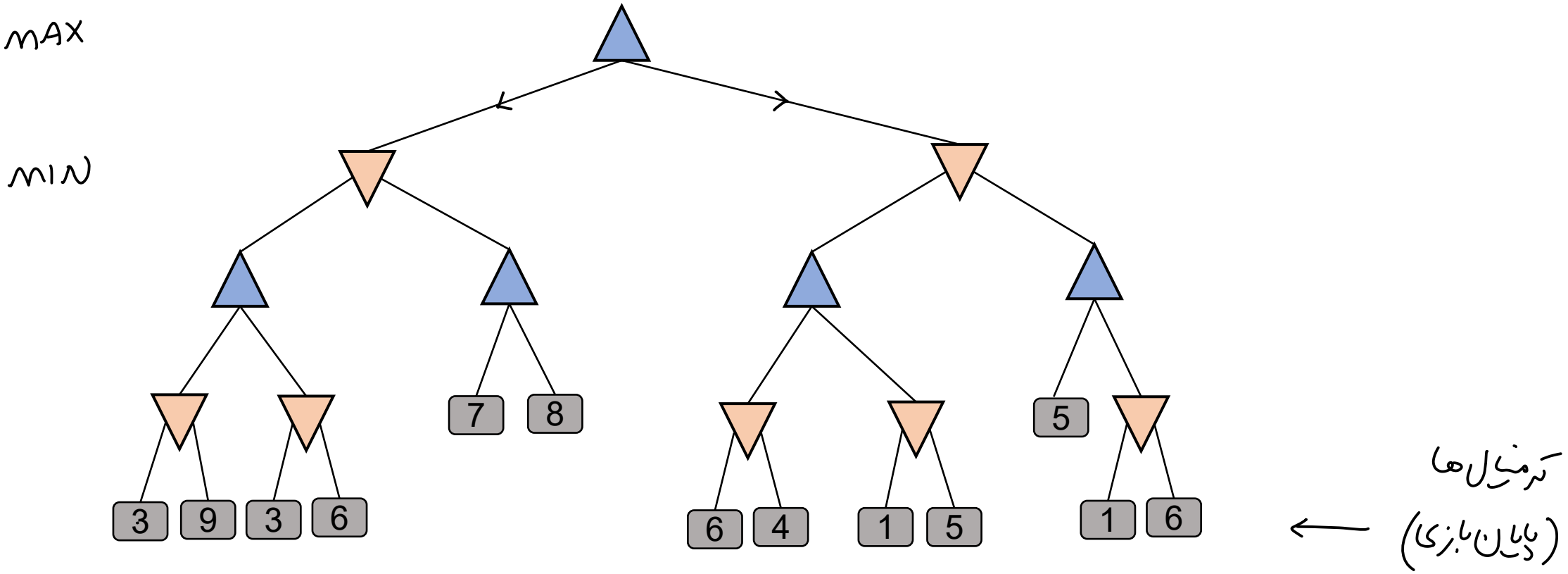


آبی ← نوبت عامل

زرد ← نوبت رقیب

درخت جستجوی مثال قبل:

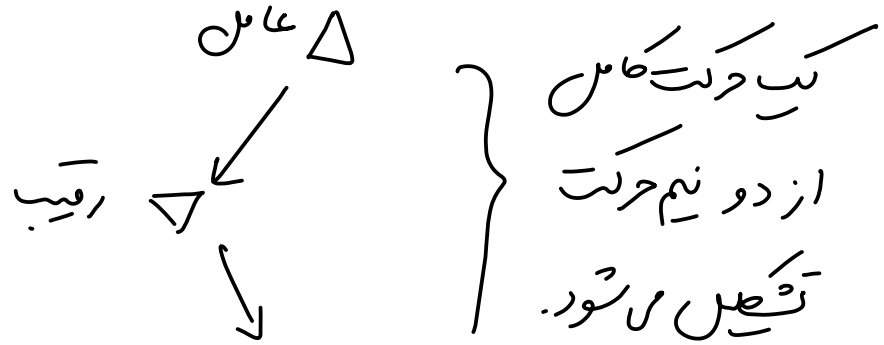
سوال: چگونه اقدام بهینه انتخاب شود؟



فرض کنیم که:

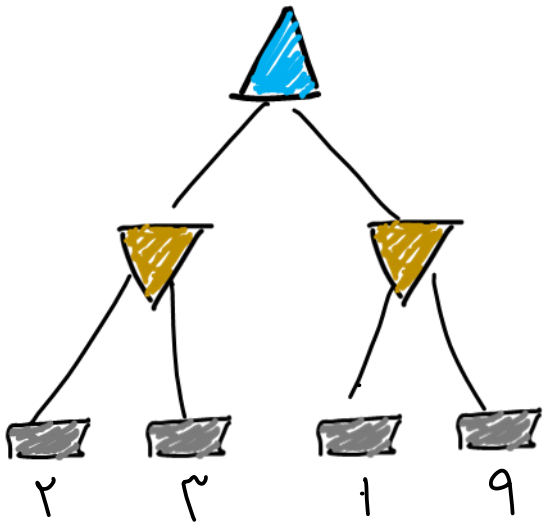
- بازیکن‌ها به نوبت عمل می‌کنند. بازی ۲ نفره است.
- MAX نشان‌دهنده‌ی عامل حریص‌مصنوعی است. MiN نشان‌دهنده‌ی بازیکن رقیب است.
- به هر عمل یا حرکتی که یک بازیکن انجام می‌دهد نیم‌حرکت گوییم.

یک حرکت کامل = نیم‌حرکت + نیم‌حرکت



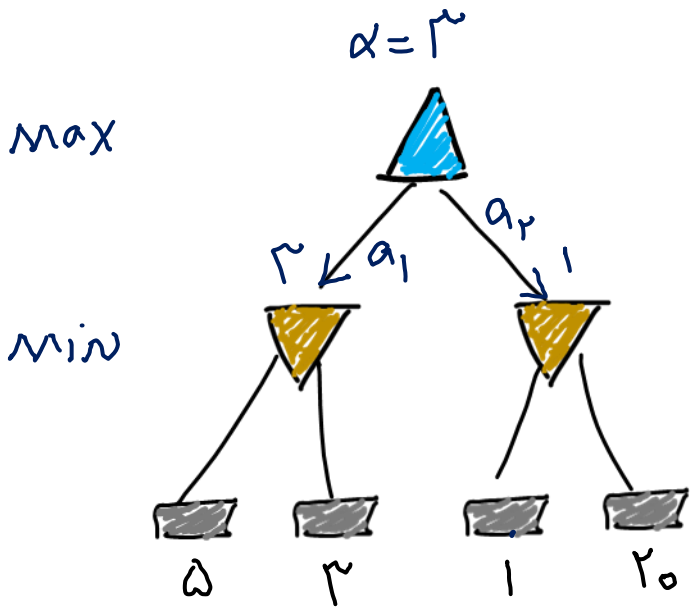
- به جای تابع goal_test این بار نیاز به تابع Terminal-test داریم که تشخیص می‌دهد در آن حالت بازی به پایان رسیده است. [پایانه]
- تابع سودمندی (امتیاز) (Utility) به هر حالت پایانه یک عدد را نسبت می‌دهد. دقت کنید دو عددی که می‌نویسیم فقط سودمندی عامل حوزه‌مند منظورمان است.
- بازی با مجموع ثابت است.
- محیط بازی: قطعی و کاملاً مشاهده پذیر است.

بازی با مجموع صفر (مجموع ثابت): کافیست سودمندی عامل در ترمینال‌ها نوشته شود.



(فقط سودمندی MAX در گره‌ها نوشته می‌شود.)

ایده‌ی جستجوی Minimax برای بازی با مجموع ثابت ۲ نفره:



گره MIN از بین فرزندانش مینیمم می‌گیرد. گره MAX از بین فرزندانش ماکزیمم می‌گیرد.

اجرا: DFS

برای گره MAX از $\alpha = -\infty$ شروع کنیم

برای گره MIN از $\beta = +\infty$ شروع کنیم.

مقدار ریشه: $\alpha = ۳$

سیاست بهینه: a_1

اجرای الگوریتم Minimax به صورت عمقی است:

α_u : سودمندی فعلی گره‌های Max

β_u : سودمندی فعلی گره‌های Min

$-\infty \leq \alpha$ ← یا از کمترین مقدار ممکن

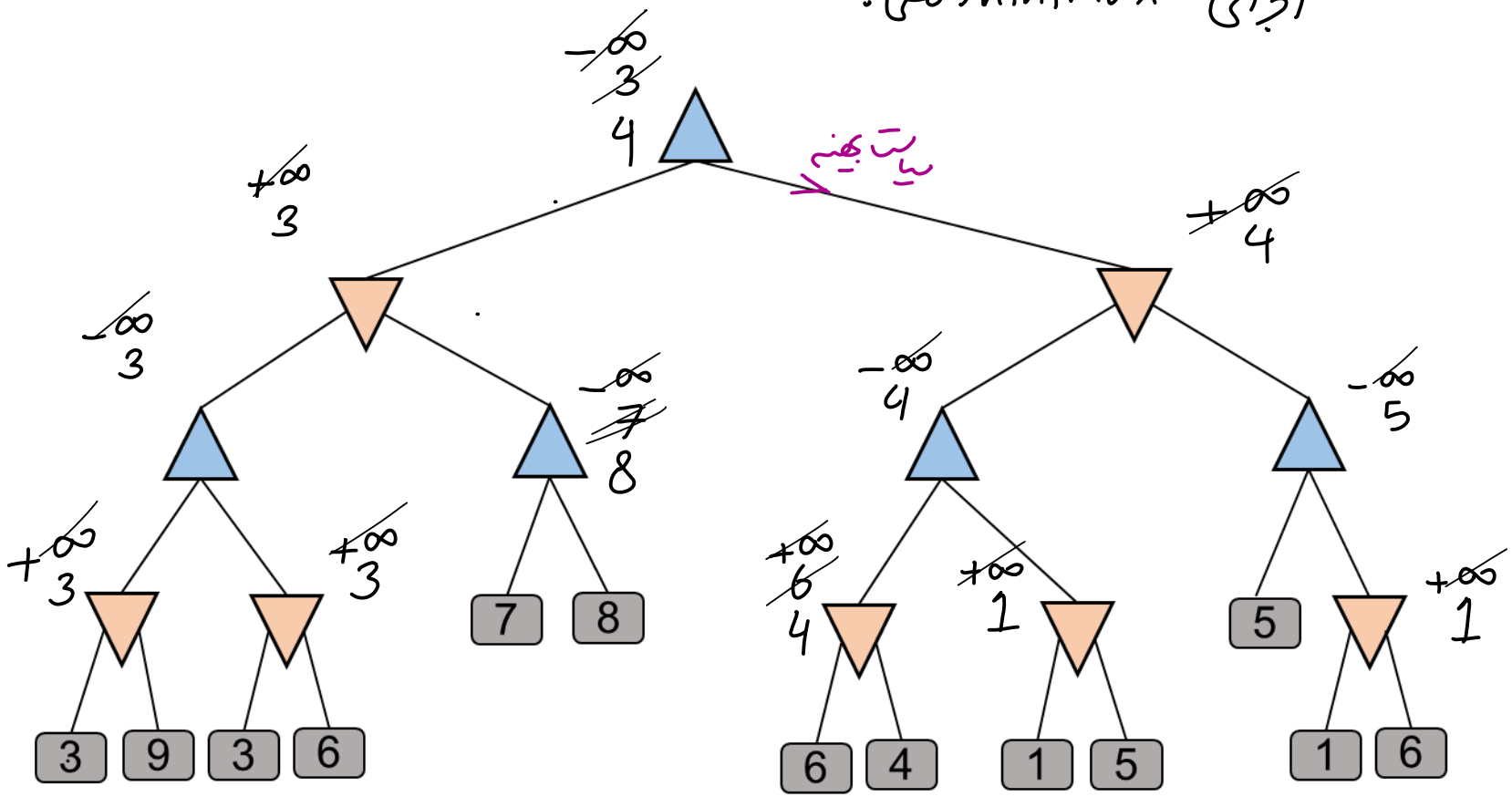
$\beta \leq +\infty$ ← یا بیشترین مقدار ممکن

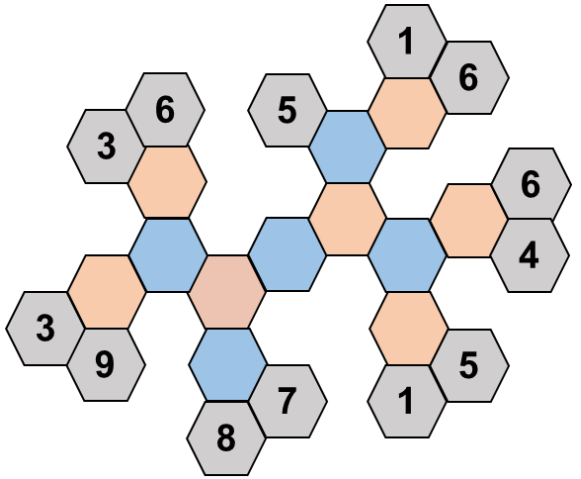
α_u : از کمترین مقدار ممکن آغاز می‌شود و در ادامه فقط صعودی به افزایش دارد.

β_u : از بیشترین مقدار ممکن آغاز می‌شود و در ادامه فقط صعودی به کاهش دارد.

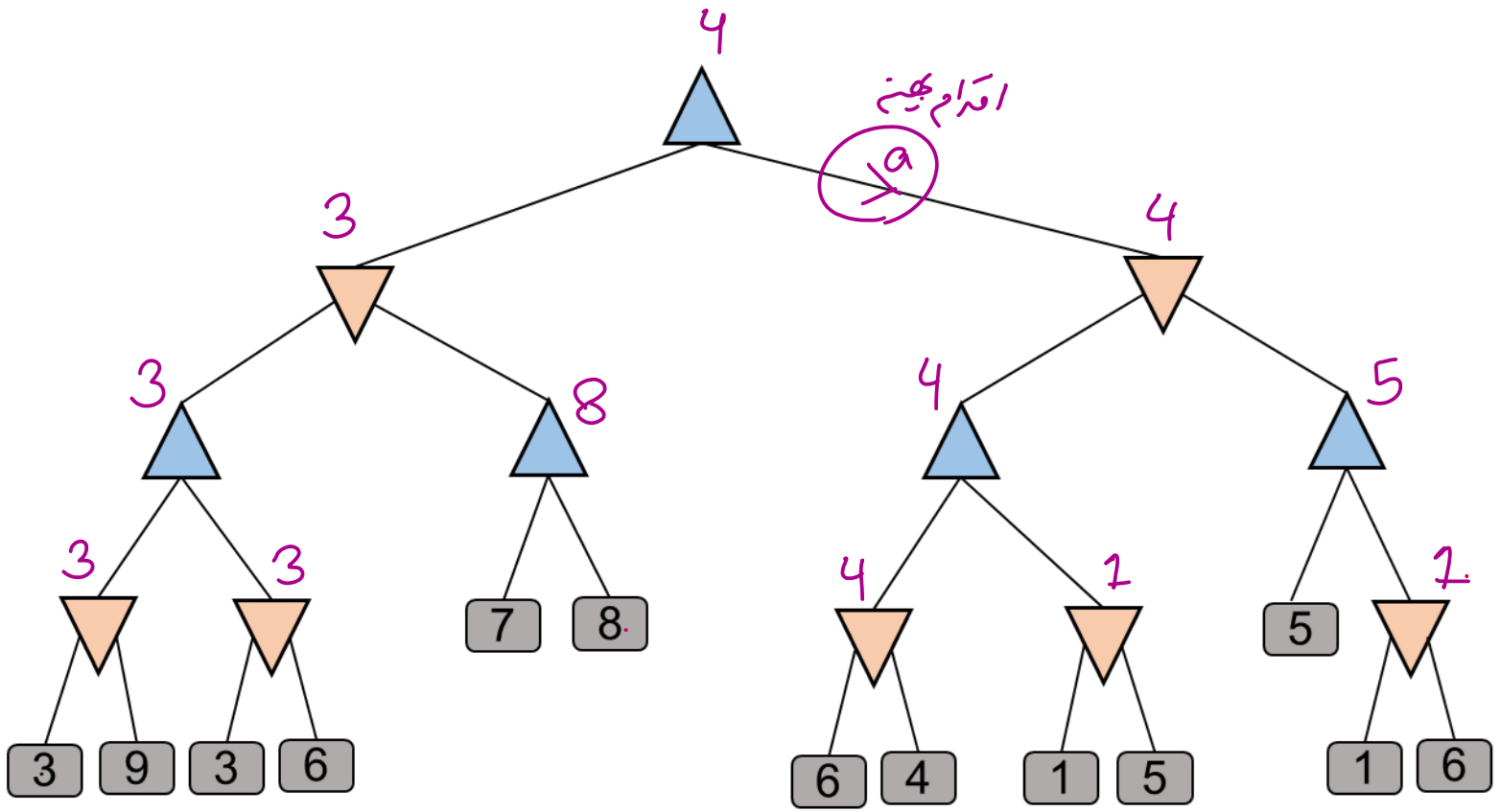
(*) [Minimax بهینه است.]

اجرای Minimax عمق:

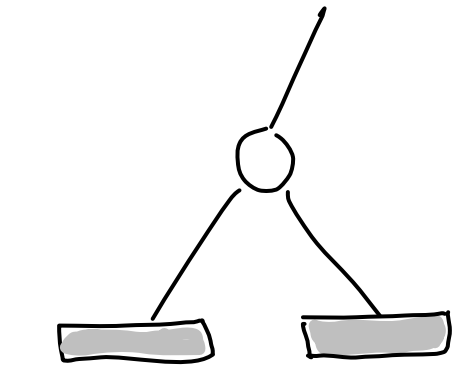




نتیجه نهایی :



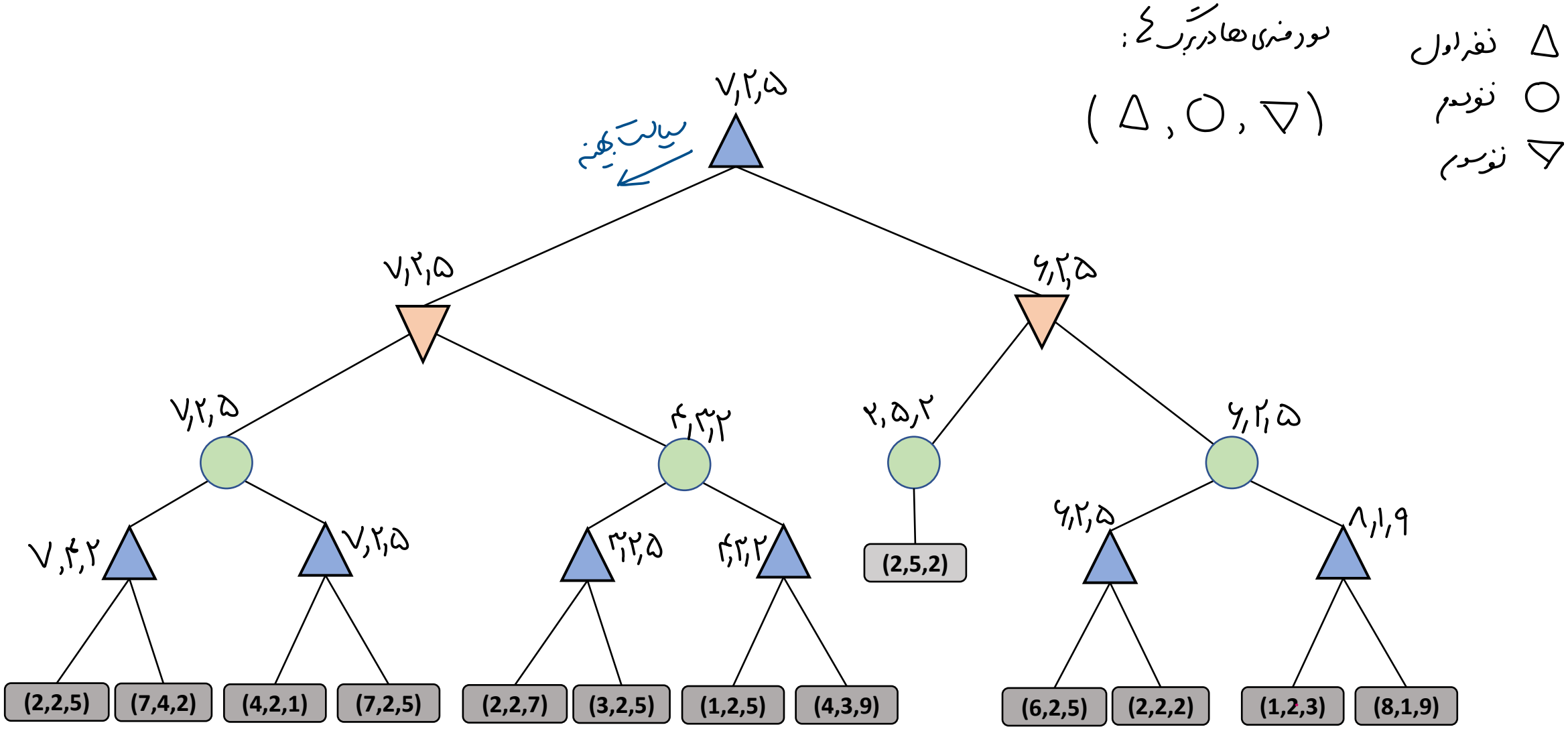
در بازی چنه‌نفره (۳ نفره) و در بازی با مجموع غیر ثابت :



(۲,۵,۷) (۰,۴,۱۰)

در تعادل (یا سایر گره‌ها) سودمندی‌های نفرات نوشته شود:

هر بازیکن، گره‌ای را که سودمندی خود را حداکثر کند انتخاب می‌کند.

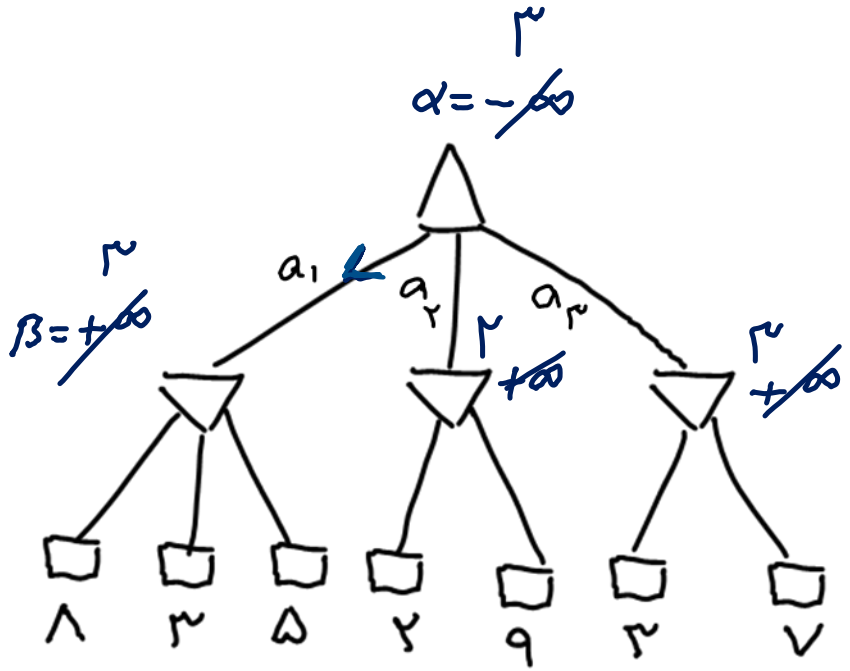

نفر اول
نفر دوم
نفر سوم
نود مرحله‌ی هادر برگ ها:
(△ , ○ , ▽)
سیاست بهینه
۷,۲,۵
۷,۲,۵
۶,۲,۵
۷,۲,۵
۴,۳,۲
۲,۵,۲
۶,۲,۵
۷,۴,۲
۷,۲,۵
۳,۲,۵
۴,۳,۲
(2,5,2)
۶,۲,۵
۸,۱,۹
(2,2,5)
(7,4,2)
(4,2,1)
(7,2,5)
(2,2,7)
(3,2,5)
(1,2,5)
(4,3,9)
(6,2,5)
(2,2,2)
(1,2,3)
(8,1,9)

تاثیر ترکیب ملاقات برگ‌ها بر Minimax :

الف) مقدار نهایی ریشه تغییر می‌کند

ب) سیاست بهینه (اقدام بهینه) ممکن است تغییر کند.

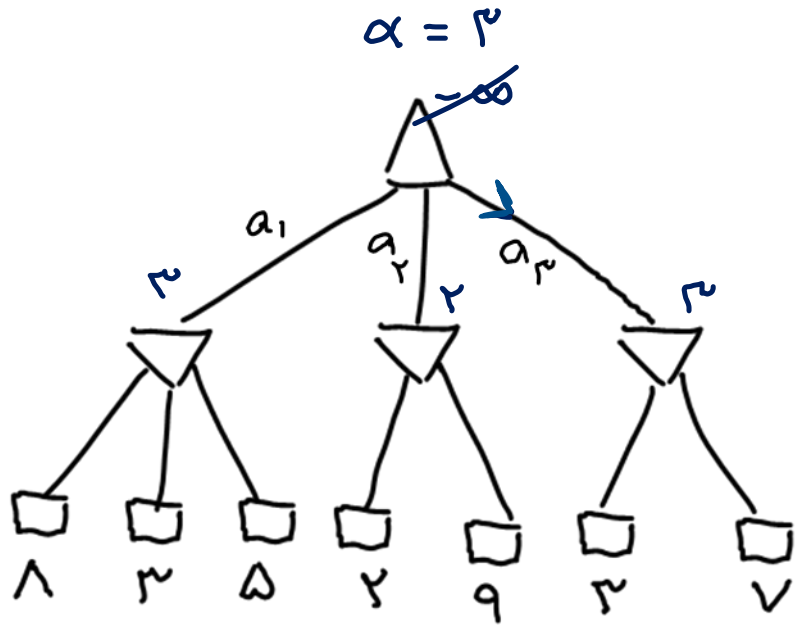
مثال: اگر ترتیب از چپ به راست باشد:



مقدار نهایی ریشه $\alpha=3$

سیاست بهینه a_1

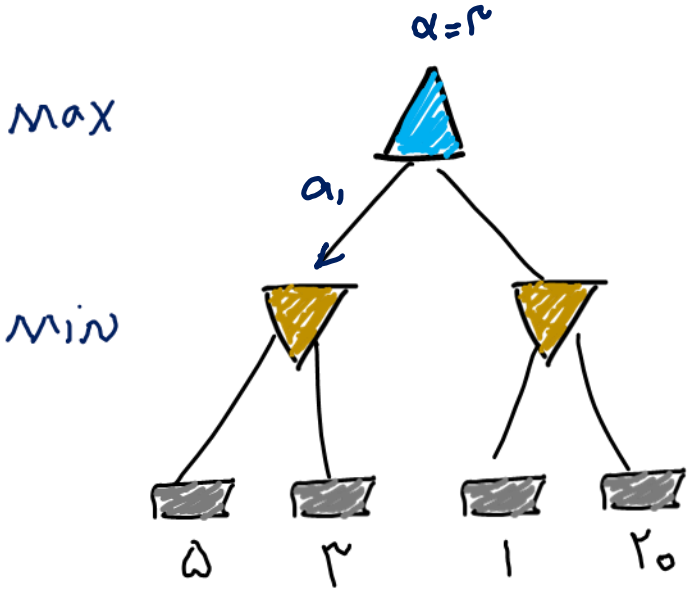
اگر ترتیب از راست به چپ باشد:



مقدار ریشه $\alpha = 3$

سیاست بهینه a_{13}

Minimax در مقابل حریف غیربهینه (تصادفی).



در این مثال:

جواب minimax: $\alpha = ۳$

اگر حریف بهینه بازی کند: ۳ = سودمندی MAX

اگر حریف غیربهینه باشد: ۵ = سودمندی MAX

مقدار سودمندی برای MAX در برابر حریف غیربهینه همیشه بزرگتر مساوی مقدار سودمندی به دست آمده توسط روش minimax است.

در واقع روش minimax با این فرض که حریف بهینه (به نفع خودش) بازی میکند محاسبه می‌شود. حالا اگر حریف غیربهینه و تصادفی بازی کند ممکن است MAX به سودمندی بیشتری از جواب minimax برسد اما کمتر نه.

خلاصه: همواره داریم $U' \geq U$

U: معماری سودمندی به دست آمده برای گره MAX ریشه، با روش Minimax (که حریف را بهینه فرض می‌کند)

U': معماری سودمندی به دست آمده برای گره MAX ریشه، در برابر حریفی که غیر بهینه یا تصادفی عمل می‌کند.

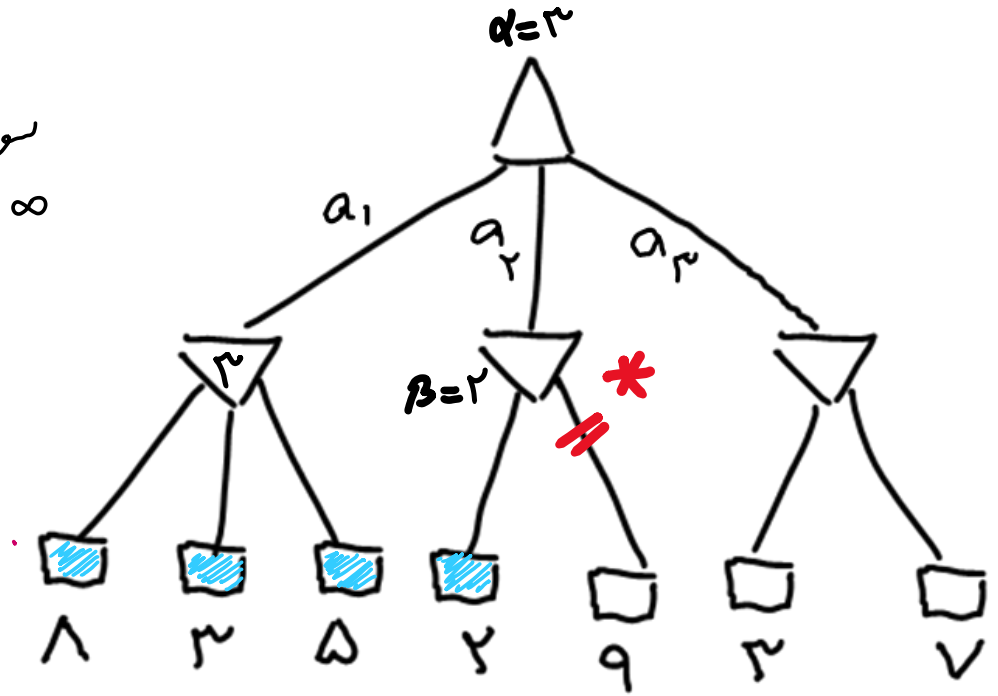
هرس $\alpha-\beta$: برای آن که سرعت اجرای Minimax را بیشتر کنیم می خواهیم گره هایی که

ملاقات آنها تاثیری در نتایج نهایی (مقدار نهایی ریشه) ندارد را هرس کنیم.

در واقع هرس $\alpha-\beta$ اندازه ای برای کاهش زمان اجرای Minimax

است و خودش یک الگوریتم جستجوی مستقل نیست.

توضیح (*) :



بعد از دیدن اولین فرزند در شاخه a_2 مقدار $\beta=2$ است

در واقع $(-\infty, 2]$ است.

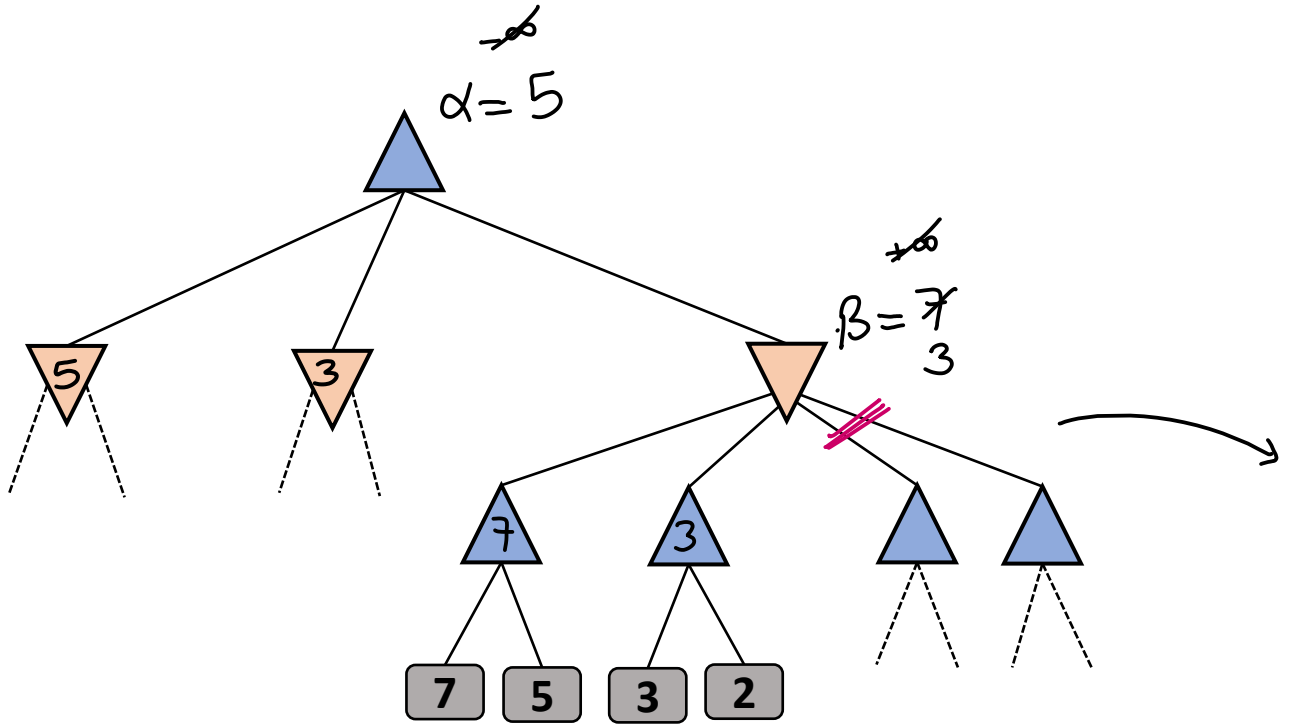
اما $\alpha=3$ است در واقع $[3, +\infty)$.

در این لحظه $\alpha \geq \beta$ است و

بازه‌ی $[3, +\infty)$ و $(-\infty, 2]$ تداخل ندارند پس ملاقات سایر فرزندان MIN لازم نیست.

بودجه‌ی بالادان
$-\infty < u < +\infty$

هرس α :



$\alpha = 5$
$\beta = 3$

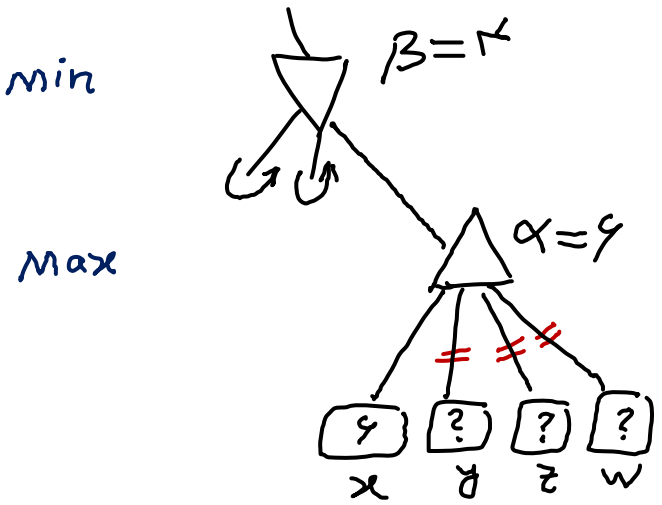
در این لحظه:

$\alpha \geq \beta$

شده پس

سایر فرزندان هرس می‌شوند.

هرس از نوع β:



در این لحظه $\alpha=۶$ داریم

$\beta \geq \alpha$ می‌شود. پس ملاقات

سایر فرزندان لازم نیست.

پس از ملاقات x

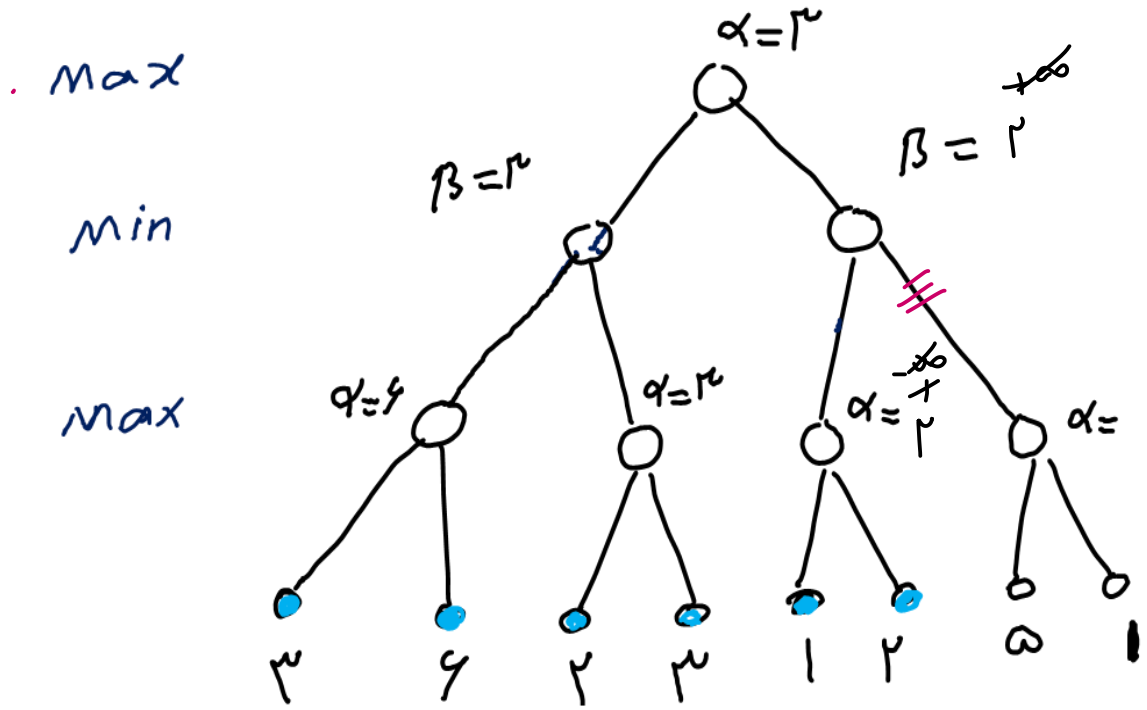
آنها هرس می‌شوند.

در واقع $[-\infty, ۴]$ برای min

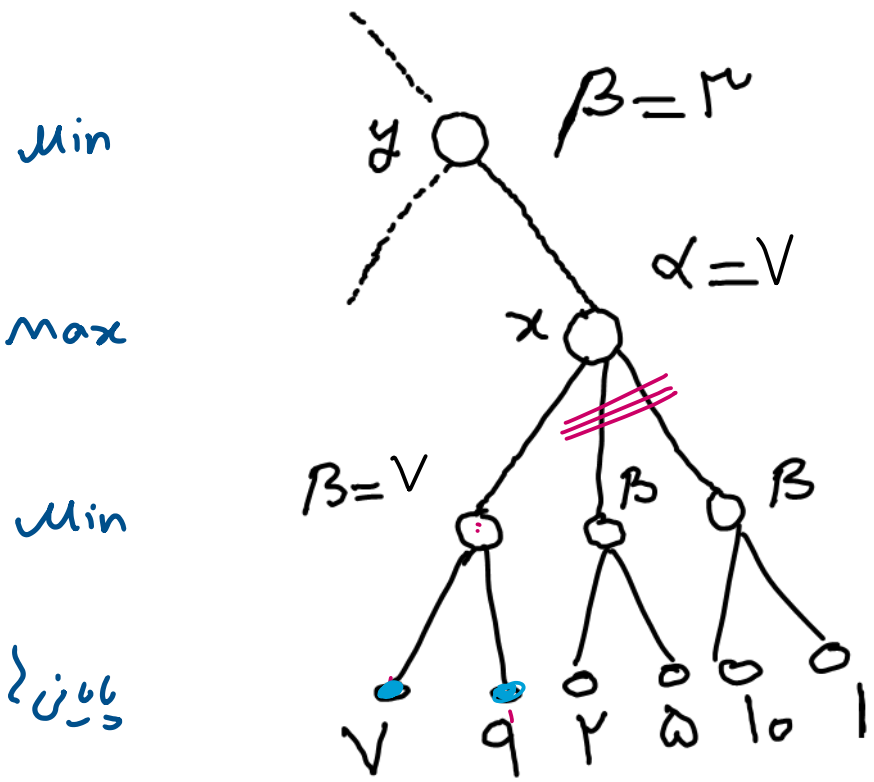
$[۶, \infty)$ برای max

قطعه‌ی مشترک ندارند پس هرس انجام می‌شود.

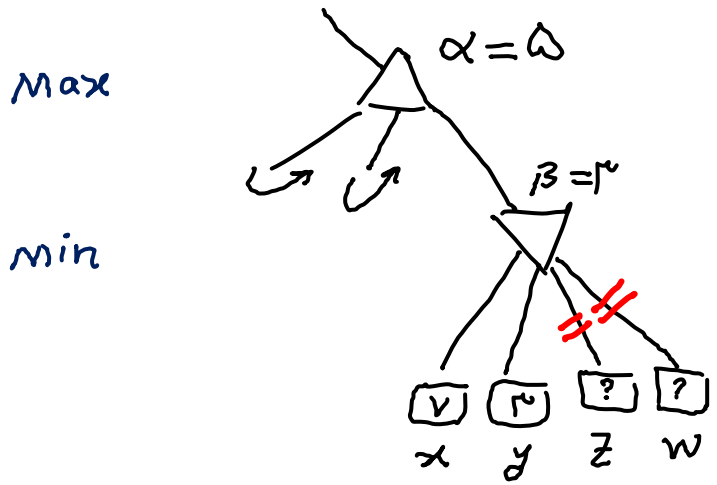
مثال از هرس نوع α :



یک مثال از هرس نوع β:



مثال هرس از نوع α :



پس از ملاقات x و y

داریم $\beta=3$. در این لحظه

$\beta \geq \alpha$ می‌شود. پس ملاقات

سایر فرزندان لازم نیست.

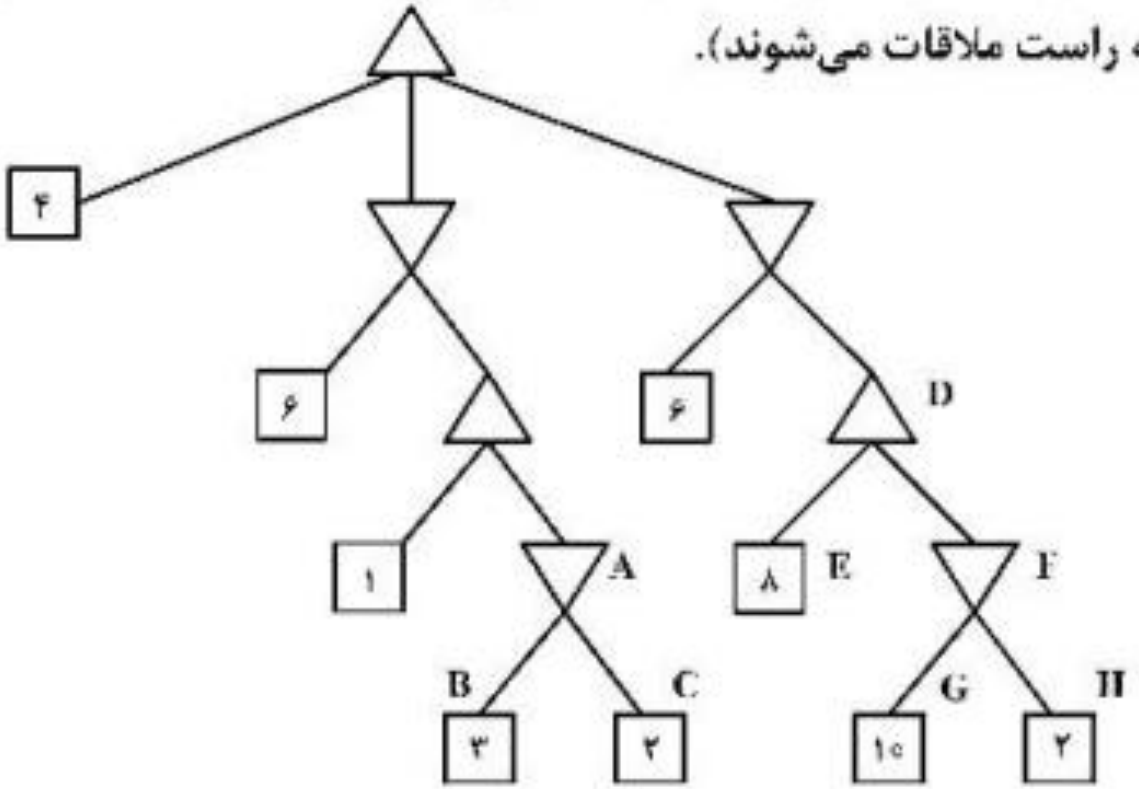
آنها هرس می‌شوند.

در واقع $[5,\infty)$ برای max

$[-\infty,3]$ برای min

قطعه‌ی مشترک ندارند پس هرس انجام می‌شود.

۱۲۰- در درخت **Minimax** زیر کدام گزینه مجموعهٔ کامل گره‌های هرس شده توسط روش هرس آلفا - بتا را نشان می‌دهد؟ (با فرض اینکه فرزندان رئوس از چپ به راست ملاقات می‌شوند).



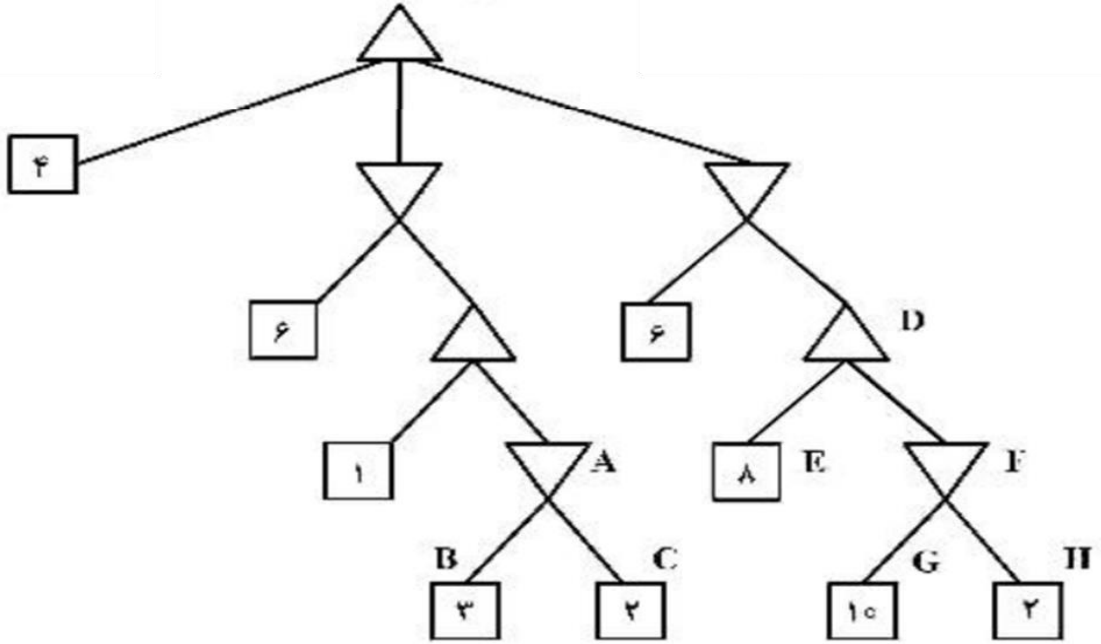
۱) $\{A,B,C\}$

۲) $\{F,G,H\}$

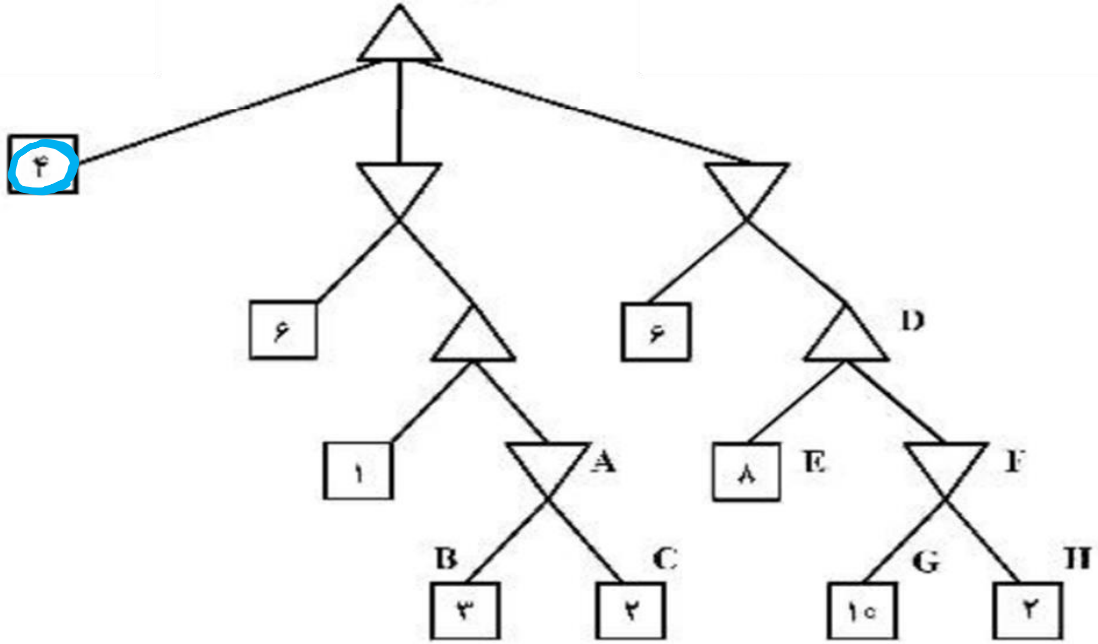
۳) $\{C,F,G,H\}$

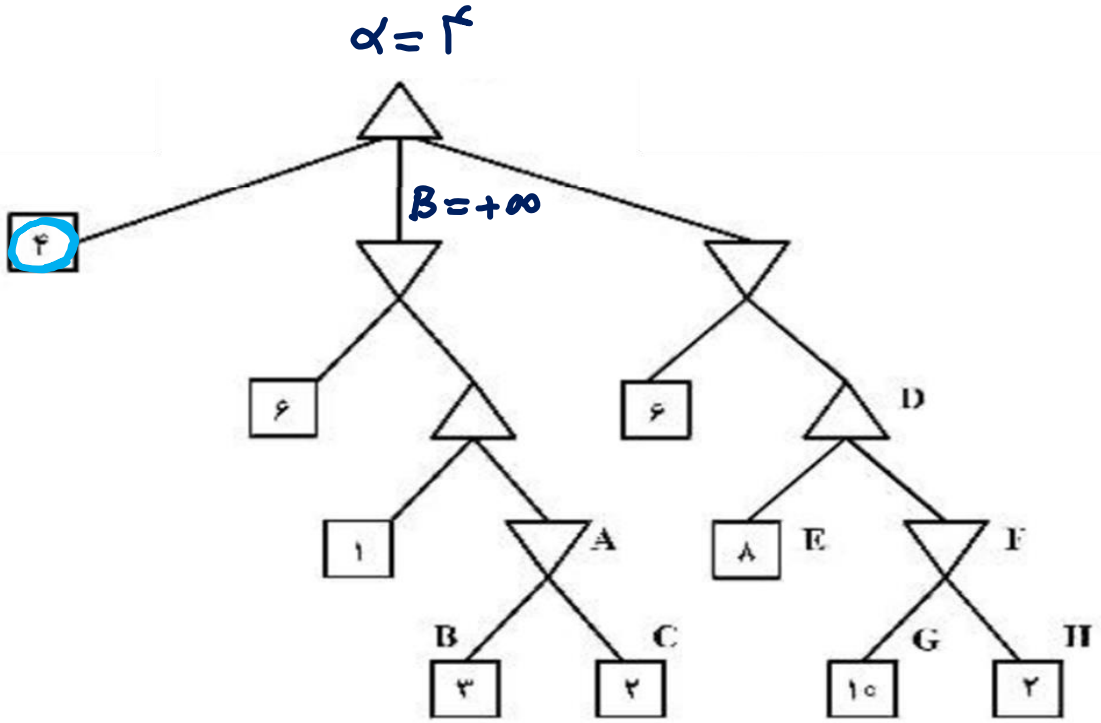
۴) $\{A,B,C,F,G,H\}$

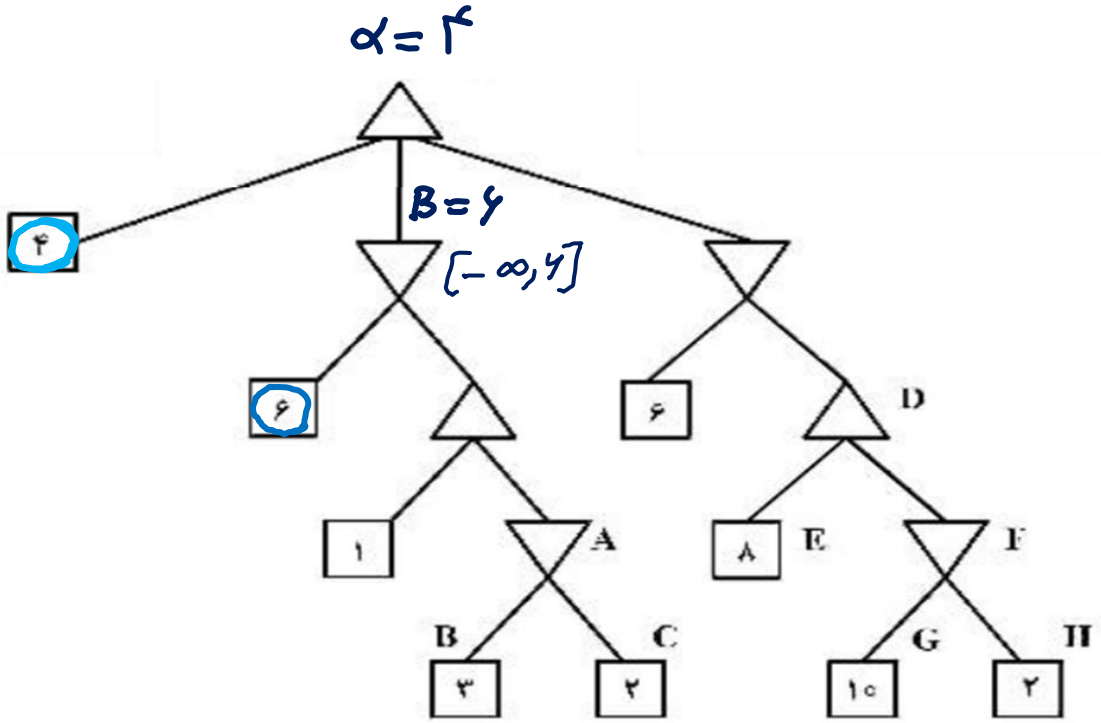
$\alpha = -\infty$ $[-\infty, +\infty]$

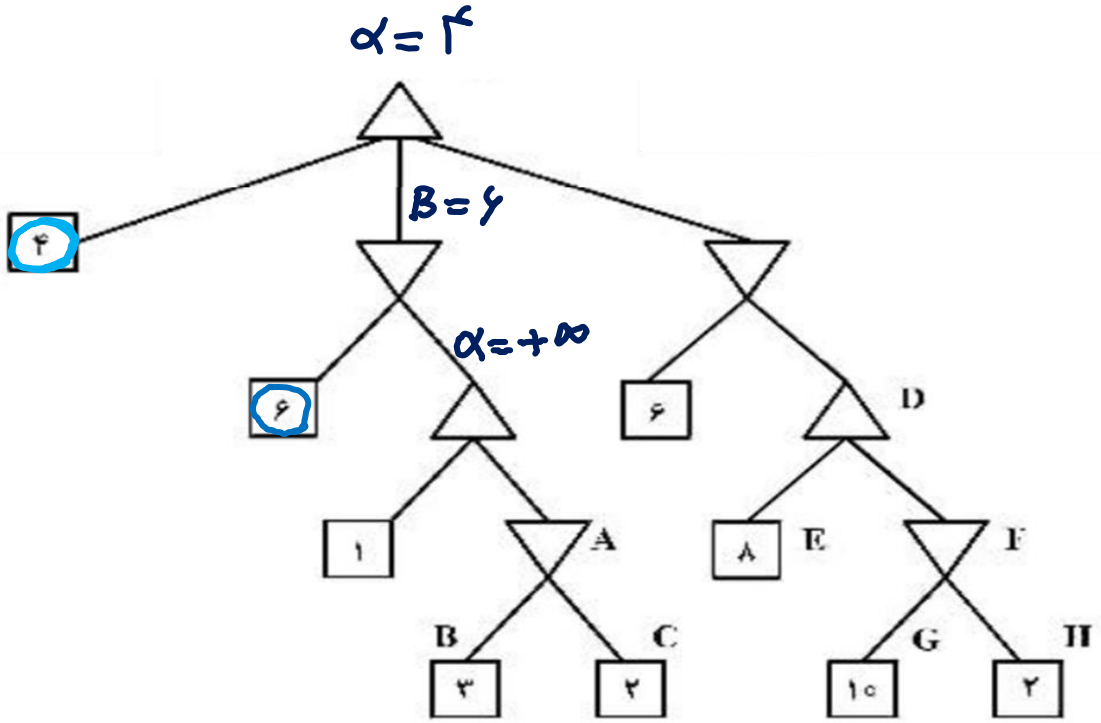


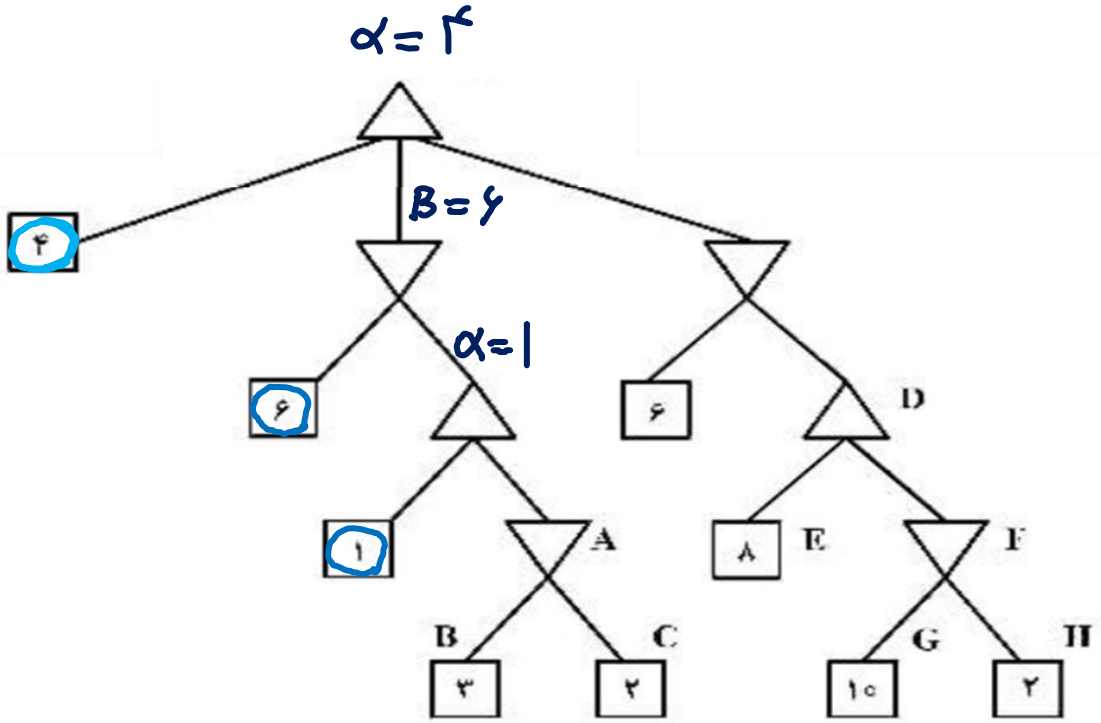
$\alpha = ۴$ $[۴, +\infty]$

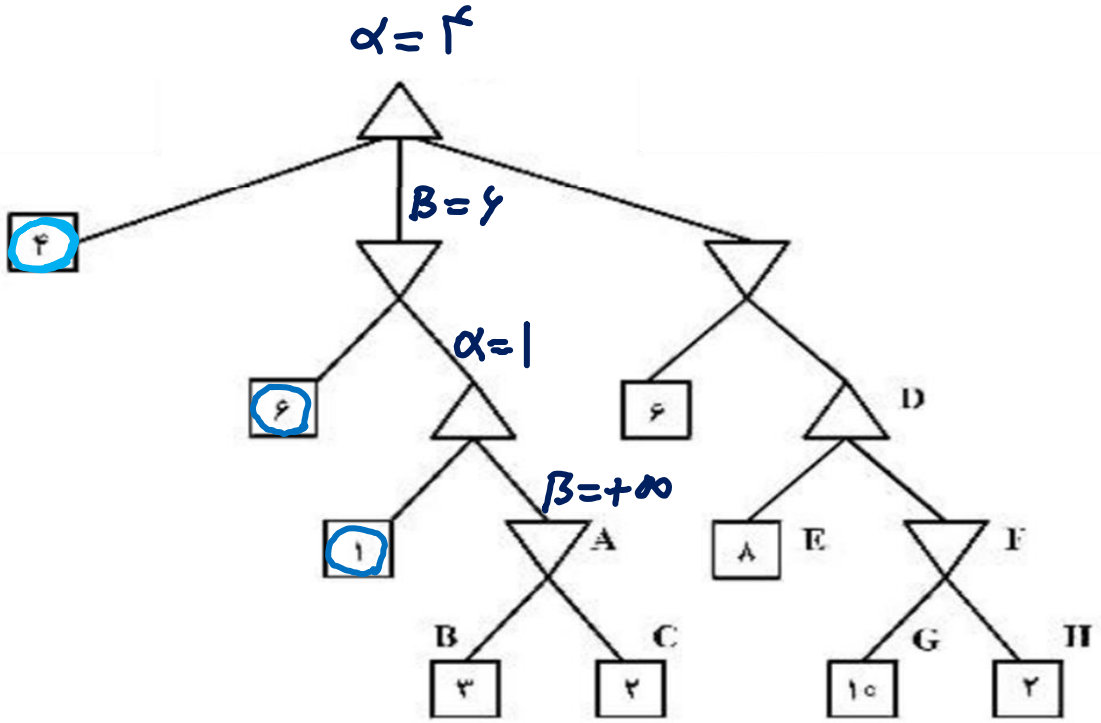


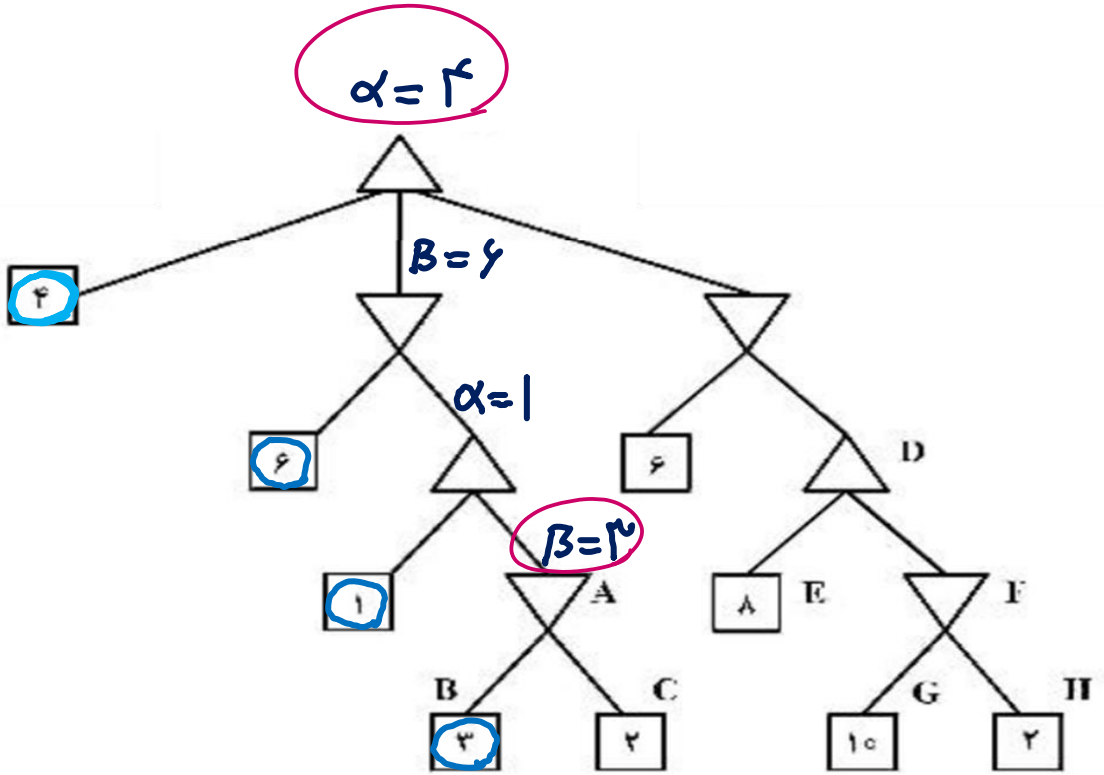

α = ۴
β = +∞
۴
۶
۶
D
۱
A
۸
E
F
B
C
G
H
۳
۲
۱۰
۲


α = ۴
B = ۶
[−∞, ۶]
۴
۶
۶
D
۱
A
۸
E
F
B
C
G
H
۳
۲
۱۰
۲


α=۴
B=۶
۴
α=+∞
۶
۶
D
۱
A
۸
E
F
B
C
G
H
۳
۲
۱۰
۲

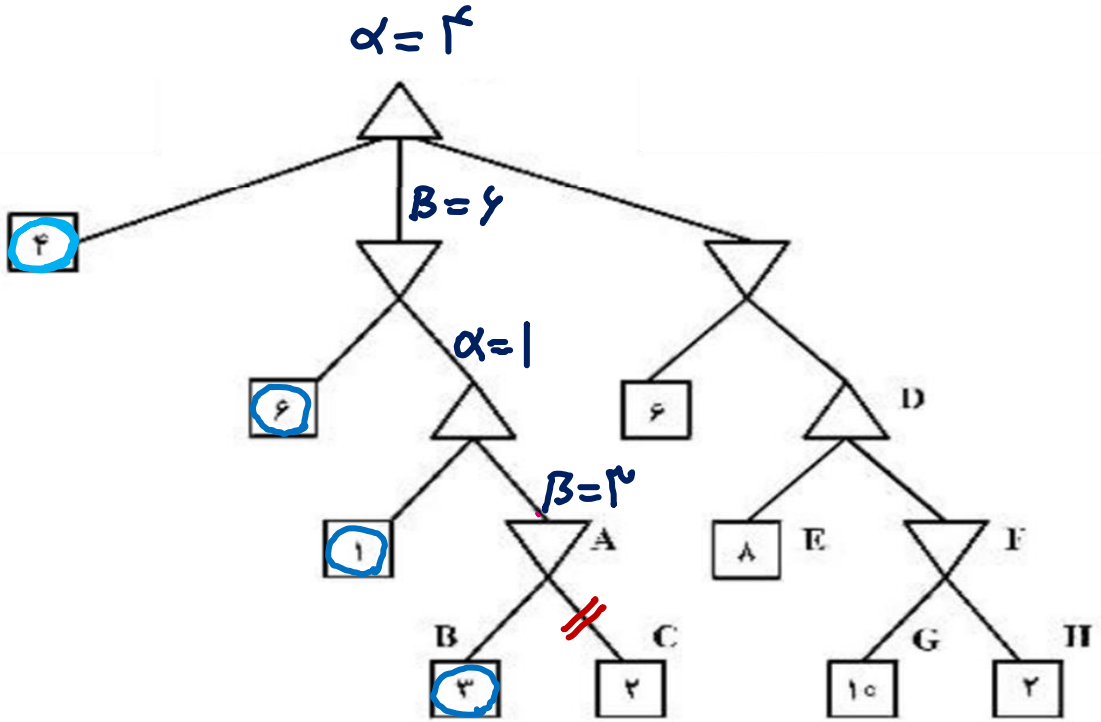

α=۴
B=۶
۴
α=۱
۶
۶
D
۱
A
۸
E
F
B
C
G
H
۳
۲
۱۰
۲


α=۴
β=۶
α=۱
β=+∞
۴
۶
۶
D
۱
A
۸
E
F
B
C
G
H
۳
۲
۱۰
۲

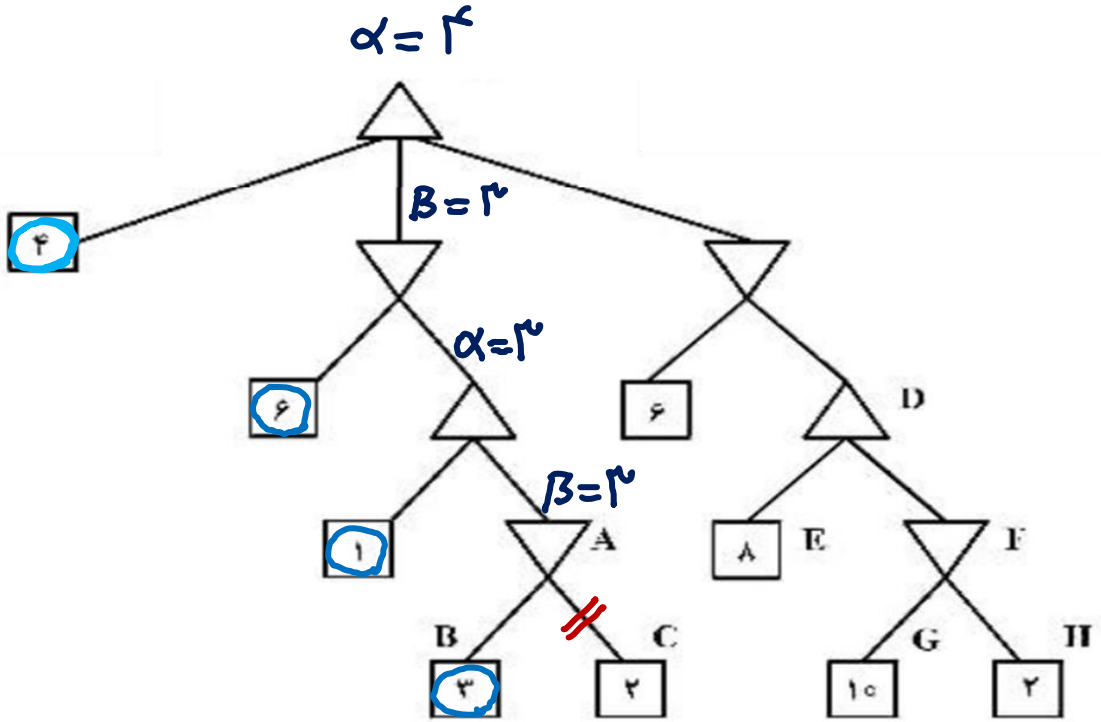


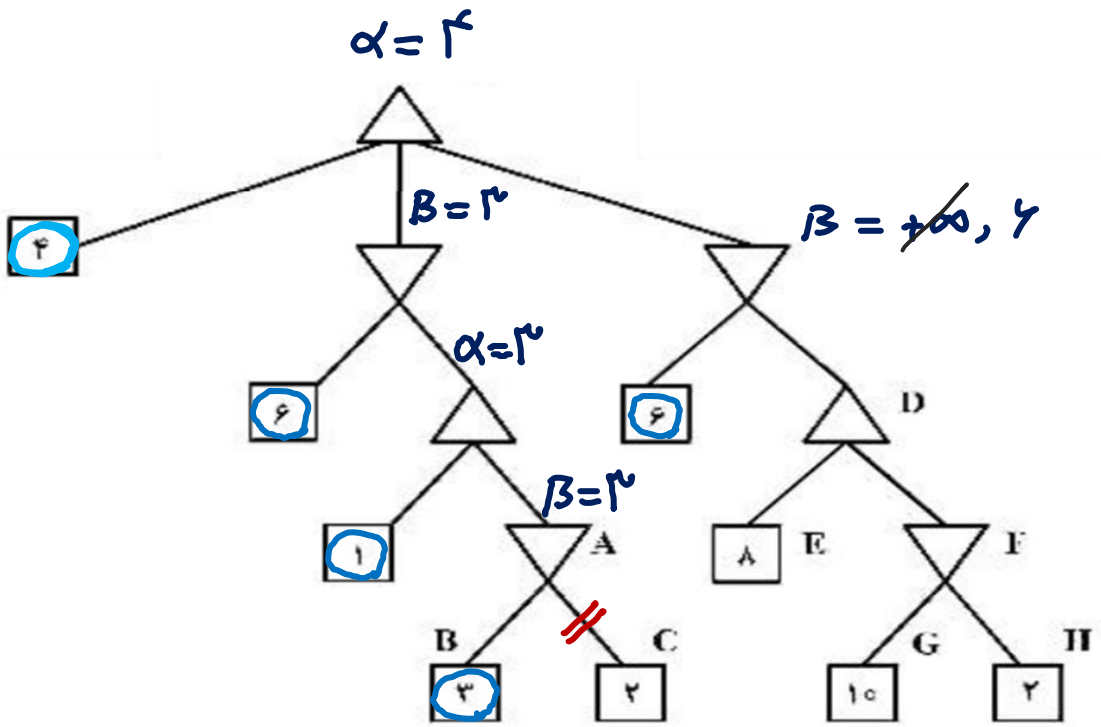
$\alpha \geq \beta$

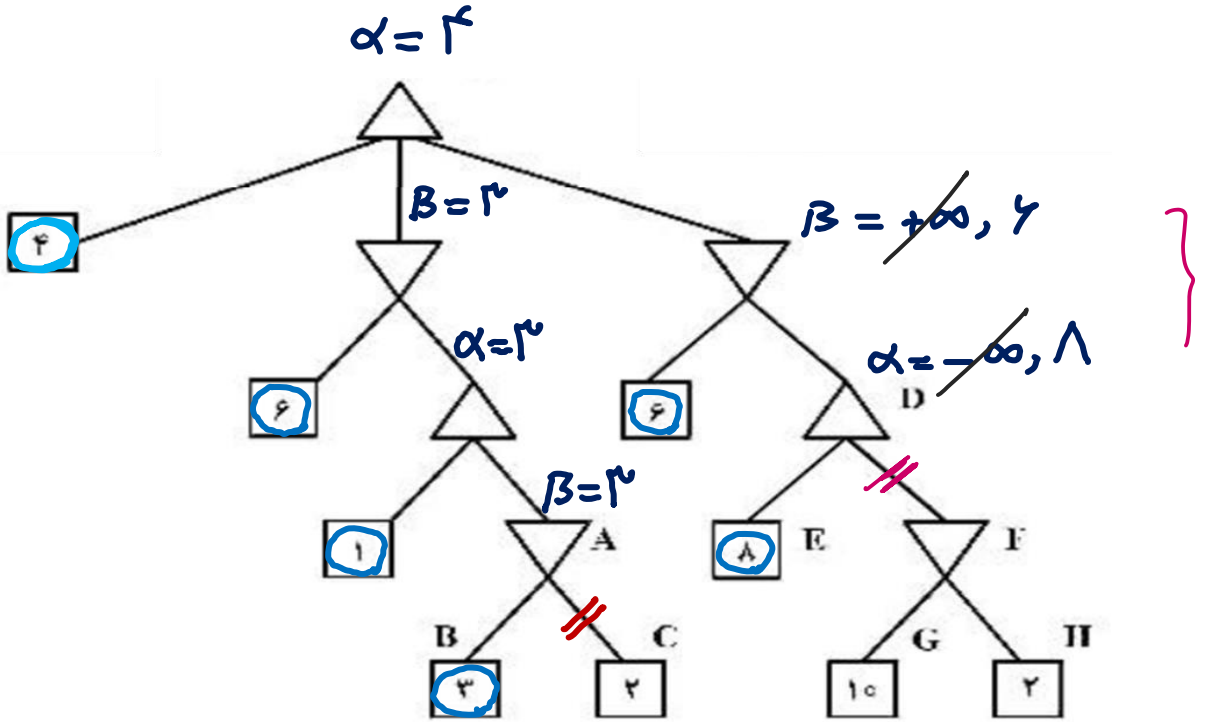
گره C هرس می‌شود.



در بازگشت، مقادیر گره‌ها به روزرسانی می‌شود.


α=۴
β=۴
α=۴
۴
۶
۶
D
β=۴
۱
A
۸
E
F
B
C
G
H
۳
۲
۱۰
۲


α=۴
β=۴
β = +∞, ۷
۴
α=۴
۶
۶
D
β=۴
۱
A
۸
E
F
B
C
G
H
۳
۲
۱۰
۲

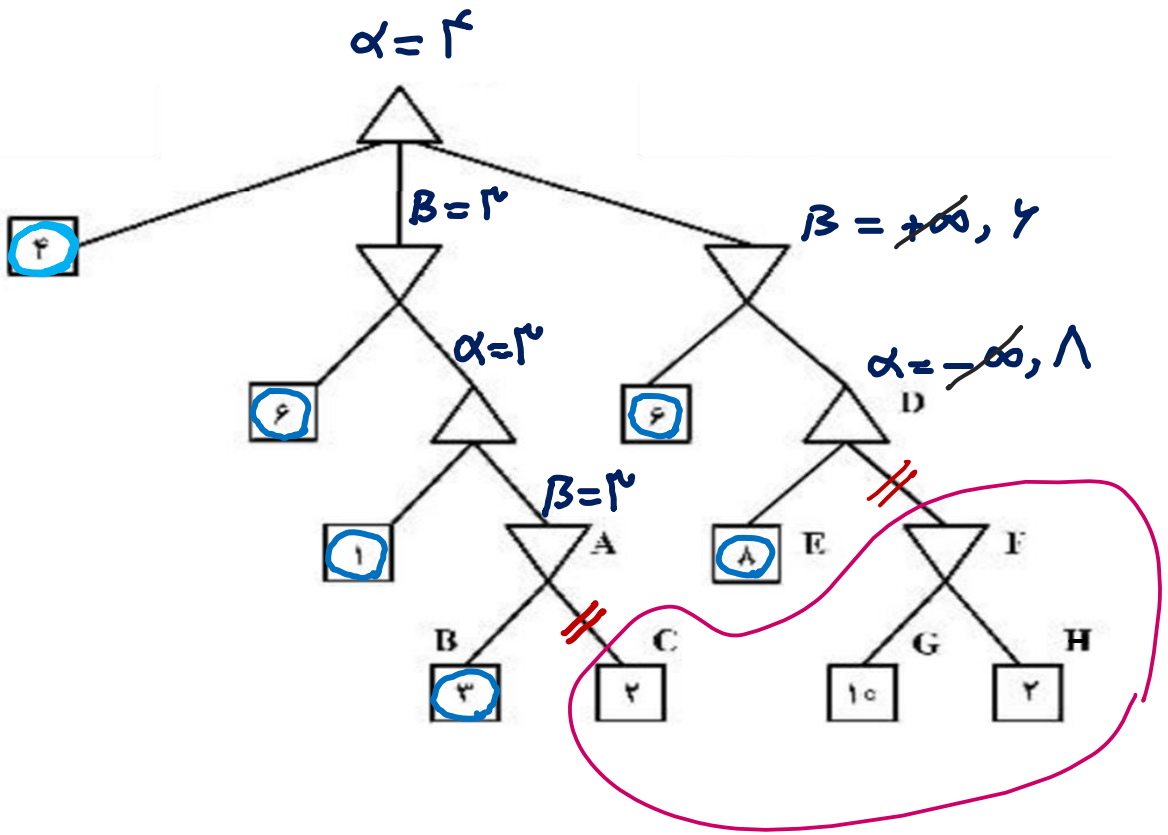


$\alpha \geq \beta$

گره F

یعنی F, G, H

نیز هرس می‌شوند.


α = ۴
β = ۴
β = +∞, ۷
α = ۴
α = −∞, ۸
۴
۶
۶
D
β = ۴
۱
A
۸
E
F
B
C
G
H
۳
۲
۱۰
۲

اثرات ترکیب ملاقات گره‌ای در هرس $k-3$:

روی این که چه گره‌ای هوس شوند اثر می‌گذارد.

روی تعداد هرس‌ها اثر می‌گذارد.

سیاست بهینه را می‌تواند تغییر دهد.

اما مقدار نهایی ریشه تغییر نمی‌کند.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

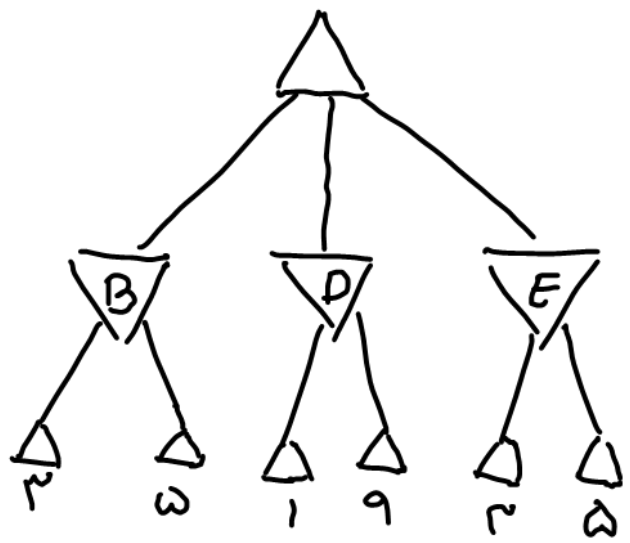
A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

بخش دوم جلسه ۶

The **minimax** search algorithm



روش Minimax:

چون به صورت عمقی پیاده‌سازی می‌شود:

مرتبه‌ی فضایی (فضای مورد نیاز حافظه) $O(bm)$

مرتبه زمانی $O(b^m)$

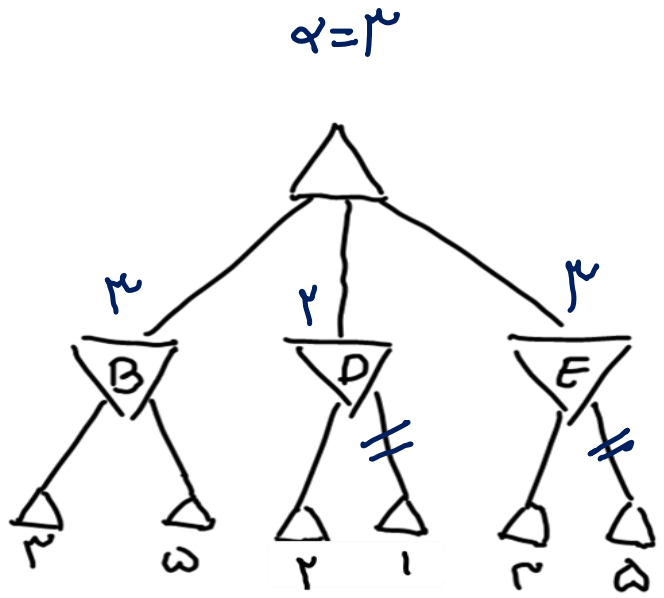
b = فاکتور انشعاب

m = ارتفاع درخت

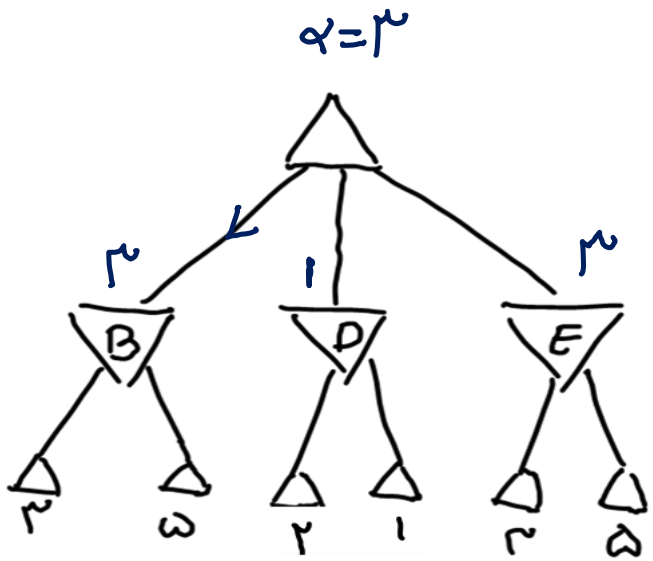
تفاوت اجرای minimax با هرس و بدون هرس: [بهتر بودن تغییر را نمی‌کند]

مقدار گره‌ای میانی ممکن است تغییر کند اما مقدار نهایی ریشه تغییر نمی‌کند. مثلاً در ملاقات از چپ به راست:

Minimax با هرس:



Minimax بدون هرس:



مثلاً مقدار نهایی گره D با هرس تغییر کرد. اما مقدار ریشه فرقی نکرد.

Alpha–Beta Pruning

Move ordering

اثراتِ ترتیبِ ملاقاتِ گره‌ها در هرس α-β :

روی این‌که چه گره‌ای هرس شوند اثر می‌گذارد.

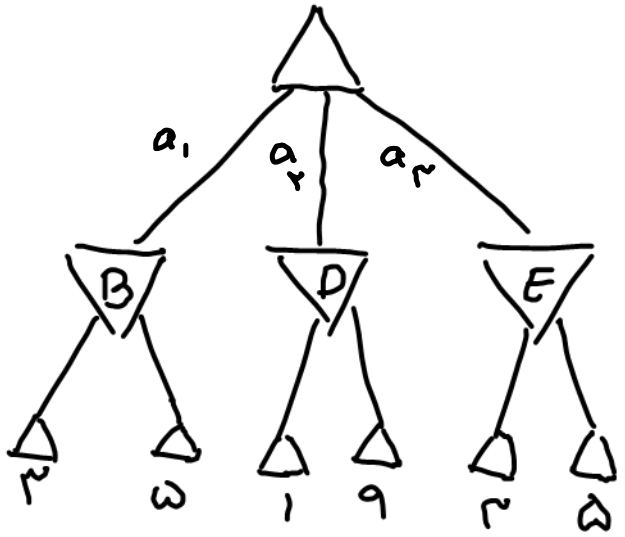
روی تعداد هرس‌ها اثر می‌گذارد.

سیاست بهینه را می‌تواند تغییر دهد.

اما مقدار نهایی ریشه تغییر نمی‌کند.

مسئله‌ی زیر را با هرس α-β حل کرده و بهترین سیاست را برای عامل حریص پیدا کنید.

یعنی از بین ۳ اقدام B, D, E کدام را انجام دهد؟

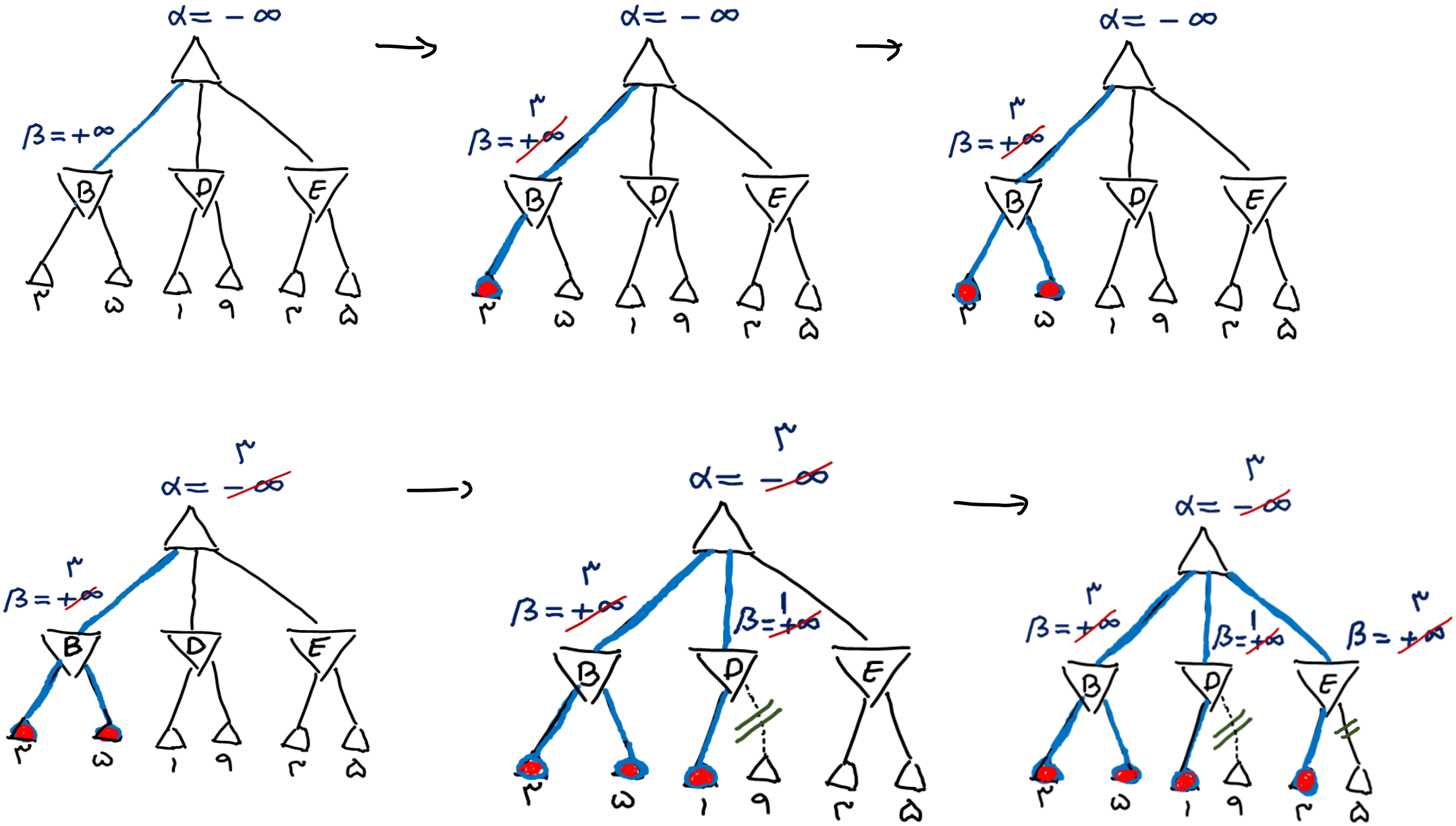


برای تمرین

یک بار از چپ به راست و یک بار از راست به چپ

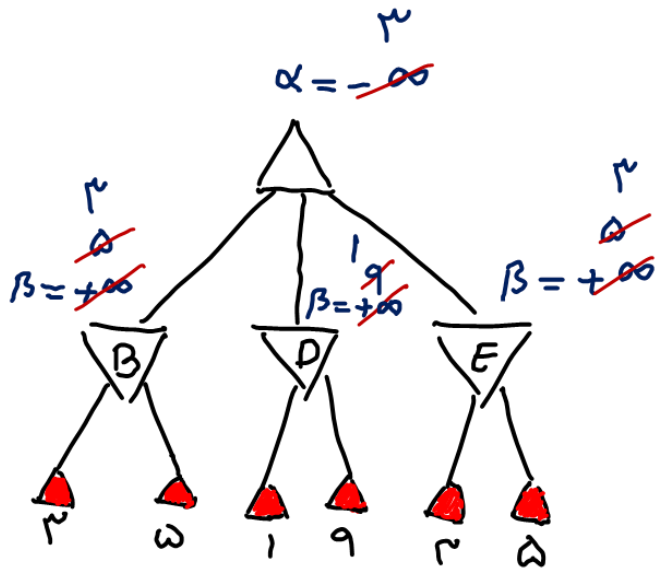
مساوات ها را انجام دهید:

از چپ به راست:


α = −∞
β = +∞
B
D
E
۲
۳
۱
۹
۲
۵
α = −∞
۳
β = +∞
B
D
E
۲
۳
۱
۹
۲
۵
α = −∞
۳
β = +∞
B
D
E
۲
۳
۱
۹
۲
۵
۳
α = −∞
۳
β = +∞
B
D
E
۲
۳
۱
۹
۲
۵
۳
α = −∞
۳
β = +∞
β = ۱ +∞
B
D
E
۲
۳
۱
۹
۲
۵
۳
α = −∞
۳
β = +∞
β = ۱ +∞
۳
β = +∞
B
D
E
۲
۳
۱
۹
۲
۵

درهمین مثال:

راست به چپ:

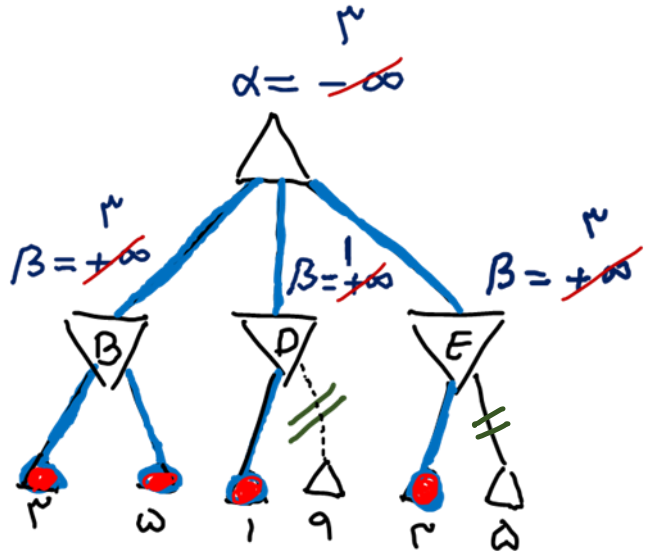


$\alpha = ۳$ مقدار بهینه است.

اقدام (سیاست) بهینه $a_۳$ است.

هیچ گره هرس نشد.

چپ به راست:



$\alpha = ۳$ مقدار بهینه است.

اقدام (سیاست) بهینه $a_۱$ است.

دو گره هرس شدند.

Alpha–Beta Pruning

مرتبه‌ی فضایی (فضای مورد نیاز حافظه) $O(bm)$

مرتبه‌ی زمانی:

بستگی به تعداد گره‌های هرس شده دارد

- بدترین حالت (هیچ هرسی) $O(b^m)$
- متوسط $O(b^{\frac{3}{4}m})$
- بهترین حالت (حداکثر هرس) $O(b^{\frac{1}{2}m})$

بیشترین تعداد هری با چه کریپی رخ می‌دهد:

جواب: فرزندان گره MAX: اولین ملاقات، بزرگترین باشه [مثلاً از بزرگ به کوچک]

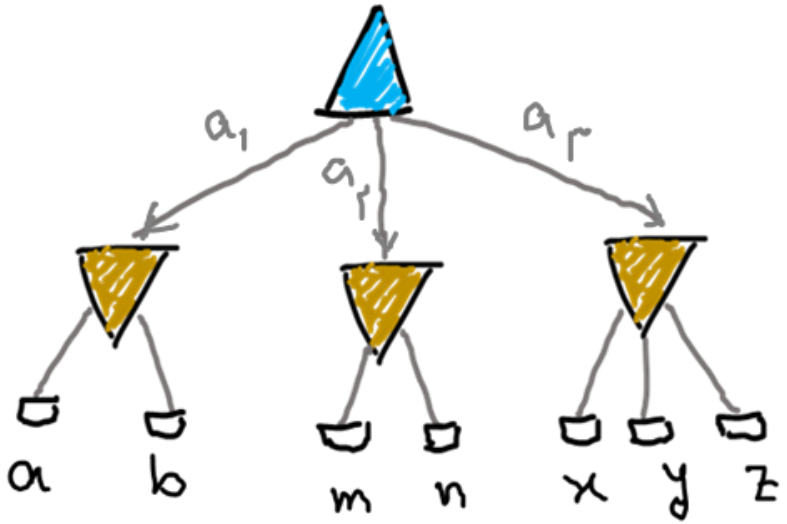
فرزندان گره MIN: اولین ملاقات کوچکترین باشه [مثلاً از کوچک به بزرگ]

مثال: در این درخت:

سود مندی پایانه‌ها: ۱, ۳, ۵, ۵, ۸, ۹, ۹ است.

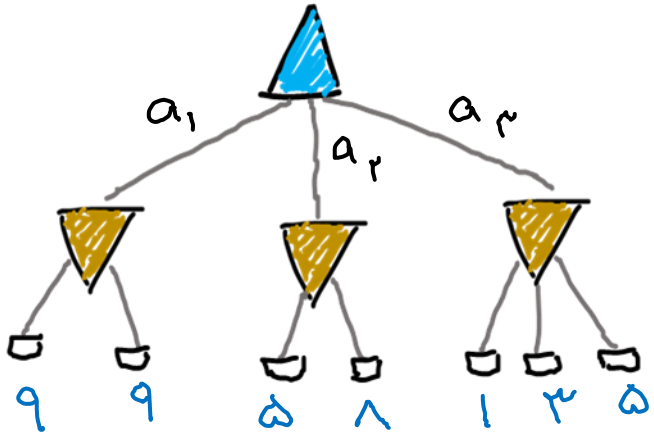
الف) در ملاقات از چپ به راست بیشترین هوس را خواهیم رخ داد.

ب) در ملاقات: اول a_2 سپس a_3 سپس a_1 را خواهیم بیشترین هوس رخ داد.

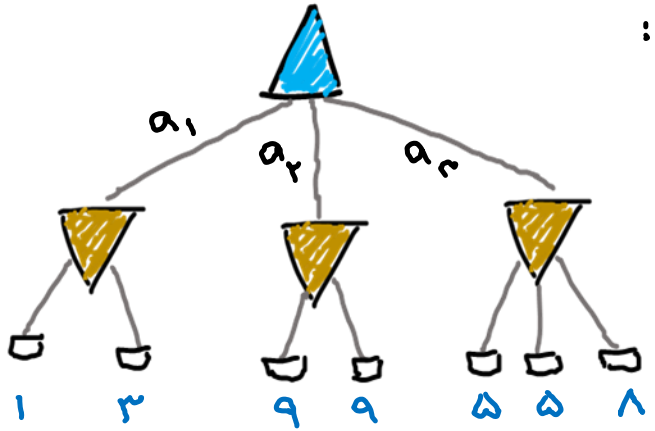


جواب‌ها: در این حالت‌ها بیشترین تعداد دوس را خواهیم داشت.

ترتیب الف:

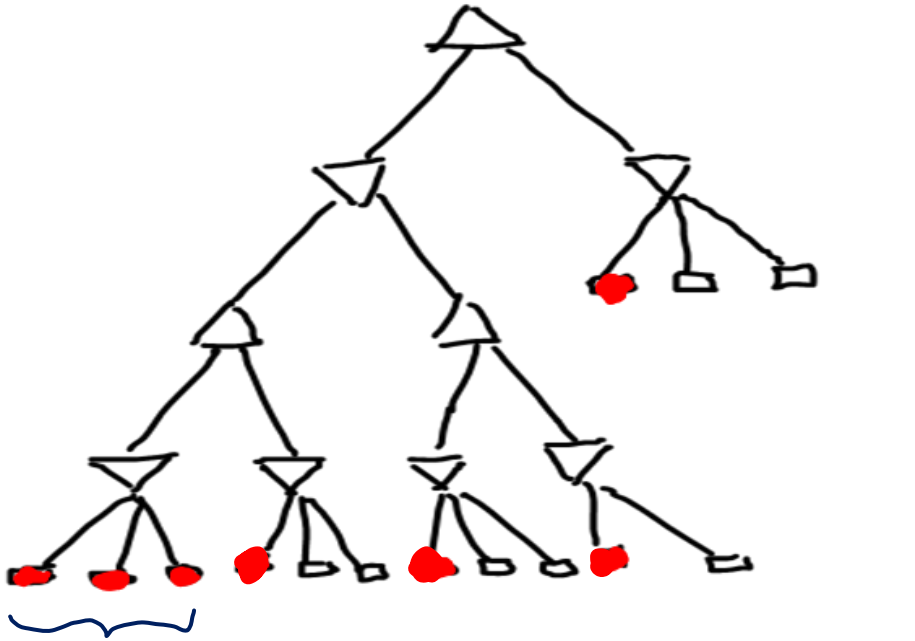


ترتیب ب:



هرس نشدنی ها در حالت $-\infty < \alpha < +\infty$:

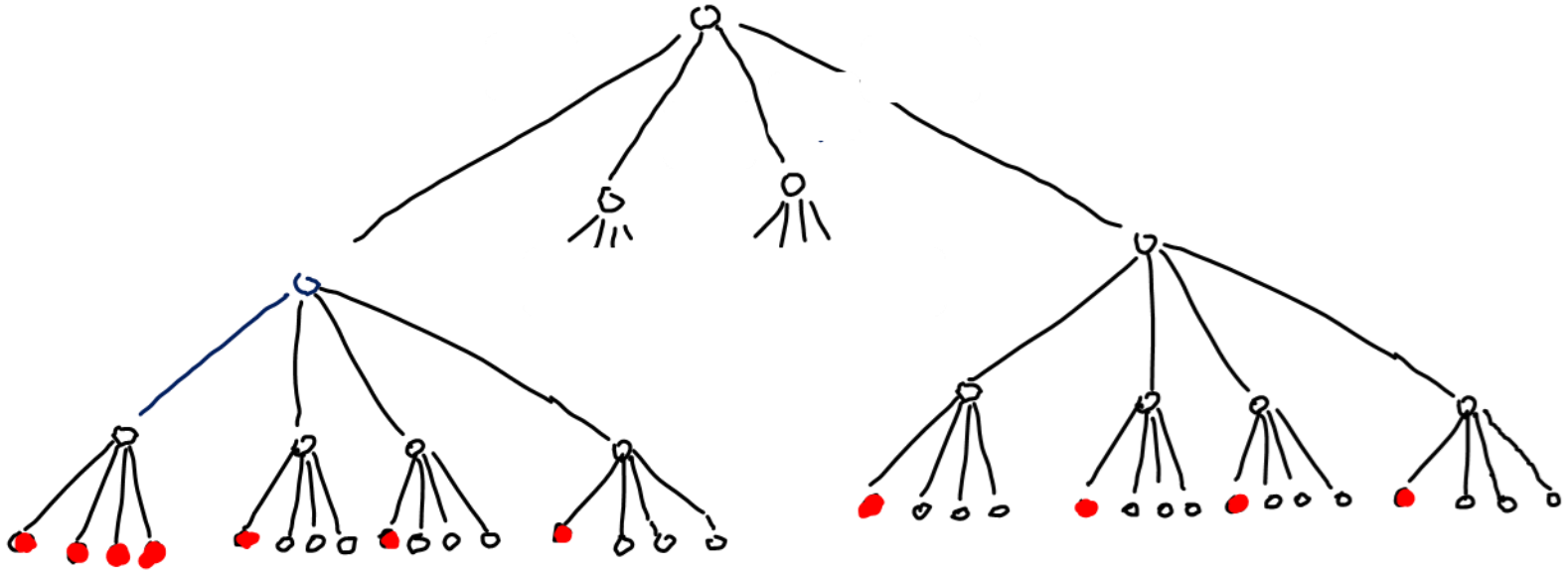
اگر $-\infty < \alpha < +\infty$ باشد و ترتیب فرزندان همان چپ به راست باشد، برخی از برگ ها قطعاً هرس نمی شوند سایر برگ ها ممکن است هرس شوند. (در بدترین حالت هیچ هرسی رخ نمی دهد)



۲۹ـ یک درخت جستجوی *minimax* را در نظر بگیرید که دارای عمق ۳ باشد و هر گره در آن دقیقاً ۴ فرزند داشته باشد (درخت ۶۴ گره برگ دارد). اگر از روش هرس آلفا ـ بتا در جستجو استفاده کنیم، حداکثر چه تعداد از گره‌های این درخت ممکن است هرس شود؟ (مهندسی کامپیوتر ـ دولتی ۹۷)

۱) ۴۸ ۲) ۴۵ ۳) ۳۵ ۴) ۳۲

حل :



۱۲ برگ قابل هرس

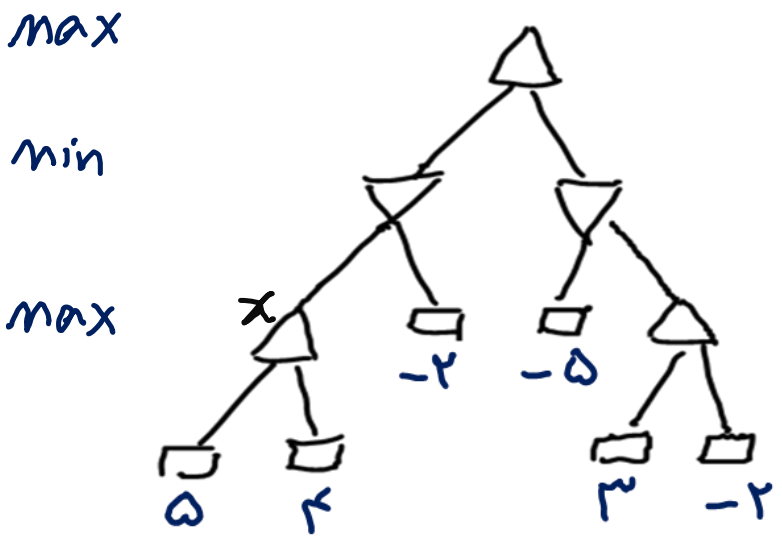
۹ برگ قابل هرس

$$3(12) + 9 = 45$$

وقتی سودمندی $b \geq u \geq a$ کران دارباشد، نوع دیگری از هرس نیز می‌تواند رخ دهد.

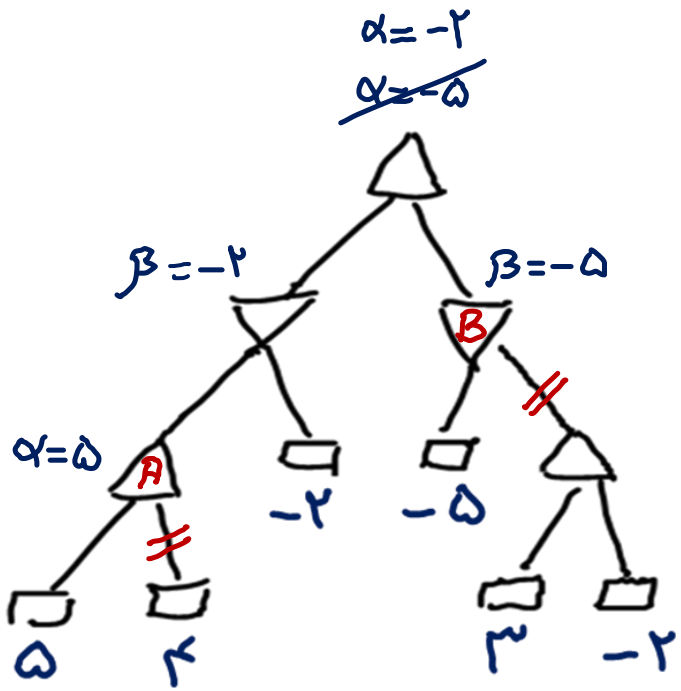
* هرس به دلیل رسیدن گره MAX به کران بالای سودمندی یا رسیدن MIN به کران پایین سودمندی.

درمثال زیر می‌دانیم که $5 \geq u \geq -5$ است.



در این حالت (سودمندی کران دار)

حتی از اولین خوشه‌ی سمت چپ نیز ممکن است هرس رخ دهد.



درگره A :

بعد از ملاقات اولین فرزند $\alpha = 5$

پس α به بیشترین حد ممکن رسید پس

ملاقات y لازم نیست هرس می‌شود.

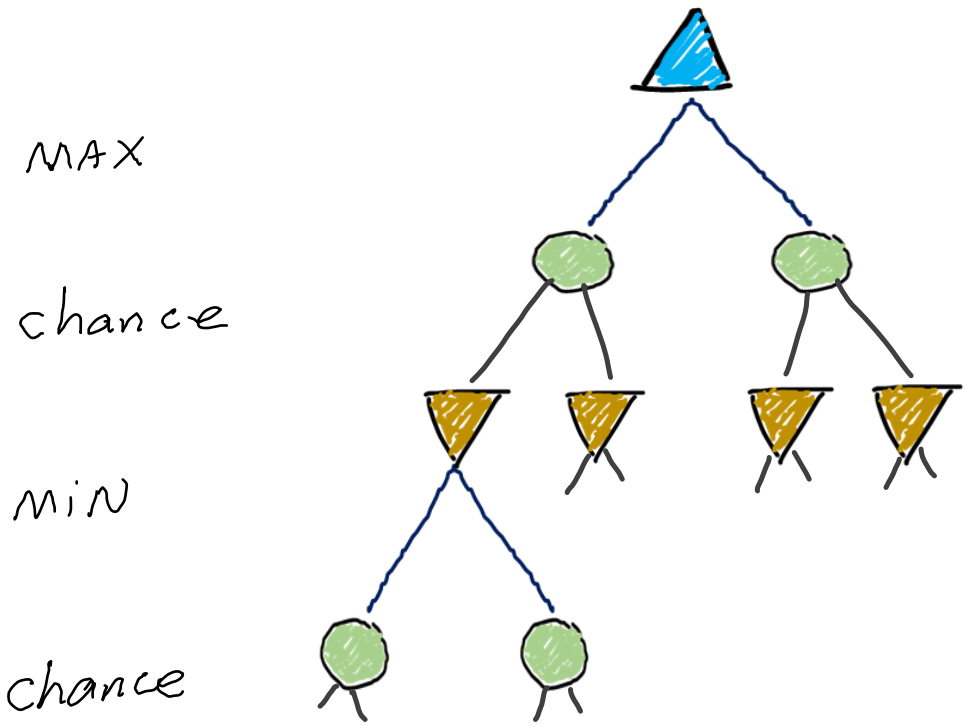
درگره B :

بعد از ملاقات اولین فرزند: $\beta = -5$

پس β به کمترین حد ممکن رسید پس

ملاقات w لازم نیست هرس می‌شود.

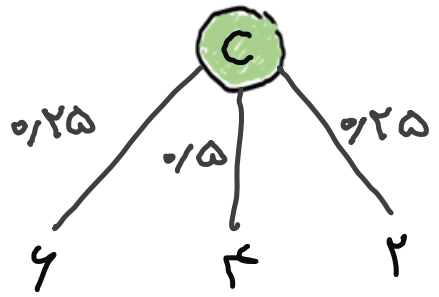
بازی‌های دارای عامل شانس:



علاوه بر گره‌های MAX و MIN

گره‌های شانس هم داریم.

chance

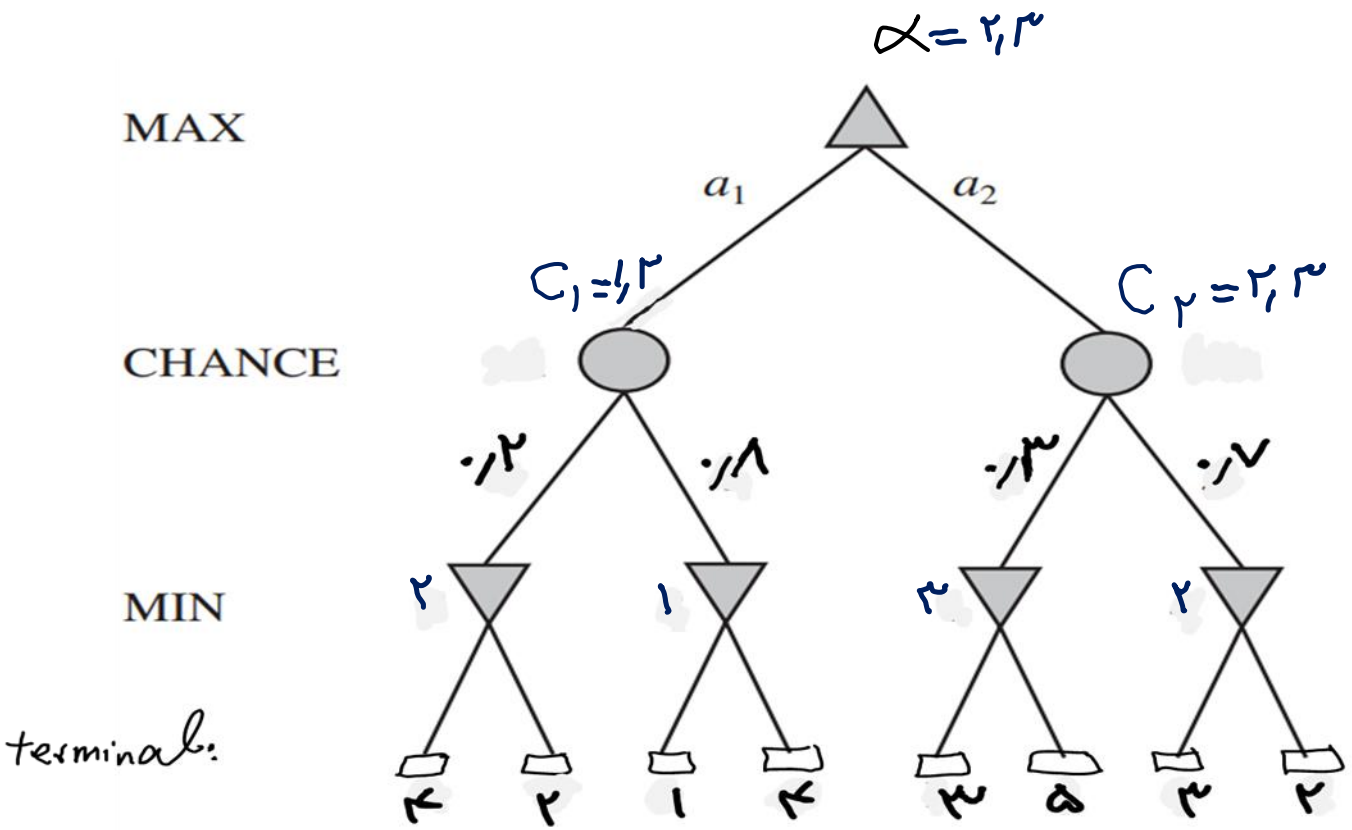


مقدار مورد انتظار گره C :

$$Exp(C) = ۰/۲۵\,(۶) + ۰/۵\,(۴) + ۰/۲۵\,(۲) = ۵$$

اجرای Minimax با گره‌شانسی:

(بدون استفاده از هرس)



محاسبه‌ی مقادیر گره‌های شانسی:

$C_1 = 0.2(2) + 0.8(1) = 1.2$

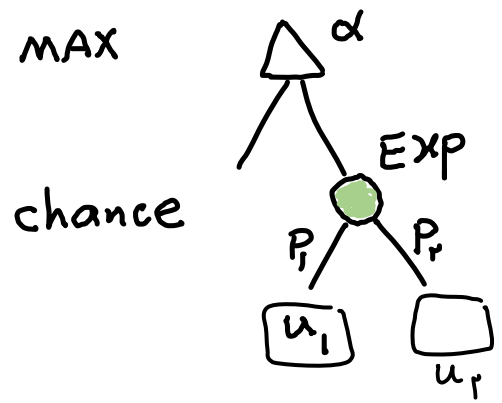
$C_2 = 0.3(3) + 0.7(2) = 2.3$

هرس $\alpha-\beta$ در بازی‌های دارای عامل شانسی:

* برای هرس با وجود عامل شانس باید u کران دار باشد: $a \leq u \leq b$ (شرط لازم)

(۱) هرگاه $b \sim MAX$ یا $a \sim MIN$ بر سر هرس انجام می‌شود.

(۲) و در مورد فرزندان گره شانس: طبق توضیحات آینده.

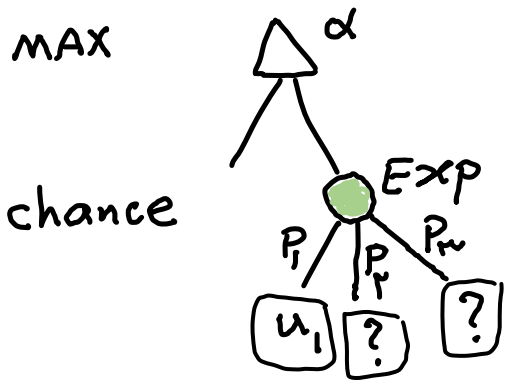


$a \leq u \leq b$

وقتی عامل شانس فرزند گره MAX است، به جای ؟ بیشترین مقدار ممکن را در نظر می‌گیریم. اگر $\max(EXP) < \alpha$ باشد معلوم است مقدار u_2 بی فایده است و u_2 هرس می‌شود.

$$EXP = P_1 u_1 + P_2 (?)$$

$\max Exp < \alpha \longrightarrow$ هرس



حاصل شانس فرزند گره MAX

است، به جای ؟ بیشترین مقدار ممکن

را در نظر می‌گیریم. اگر $\max(Exp) < \alpha$

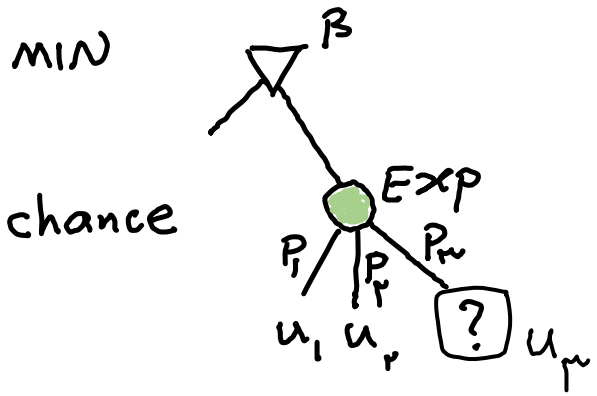
باشه معلوم است مقدار u_2 و u_3 بی‌فایده

است u_2 و u_3 هرس می‌شوند.

$$Exp = P_1 u_1 + P_2(?) + P_3(?)$$

$\max Exp < \alpha \longrightarrow$ هرس u_2 و u_3

$a \le u \le b$



$$Exp = P_1 u_1 + P_2 u_2 + P_3 (?)$$

$\min Exp > \beta \rightarrow$ حرس u_3

مثلاً بین از مقدار حدود u_2 و u_3:

وقتی حاصل شانس فرزند گره MIN

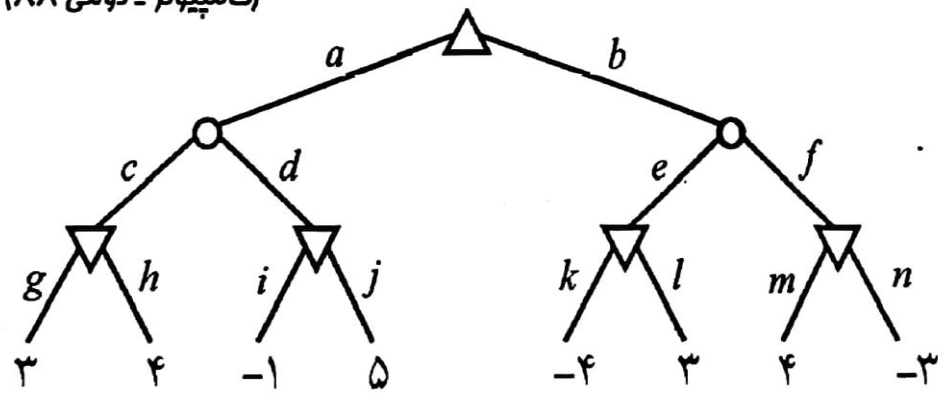
است، به جای ؟ بیشترین مقدار ممکن

را در نظر می‌گیریم. اگر $\min(Exp) > \beta$

باشد معلوم است مقدار u_3 بی فایده

است و u_3 حرس می‌شود.

۵ـفرض کنید در یک بازی، هر بازیکن ابتدا یک سکه سالم پرتاب کند. اگر شیر آمد، شاخه سمت چپ و اگر خط آمد شاخه سمت راست درخت بازی مقابل را انجام می‌دهد. درخت بازی داده شده به فرض اینکه امتیاز بازیکن‌ها در بازه [۵,۵-] است، اگر هرس آلفا - بتا اجرا شود، کدام شاخه‌ها حذف خواهند شد؟ (کامپیوتر ـ دولتی ۸۸)



۱) *l-f*

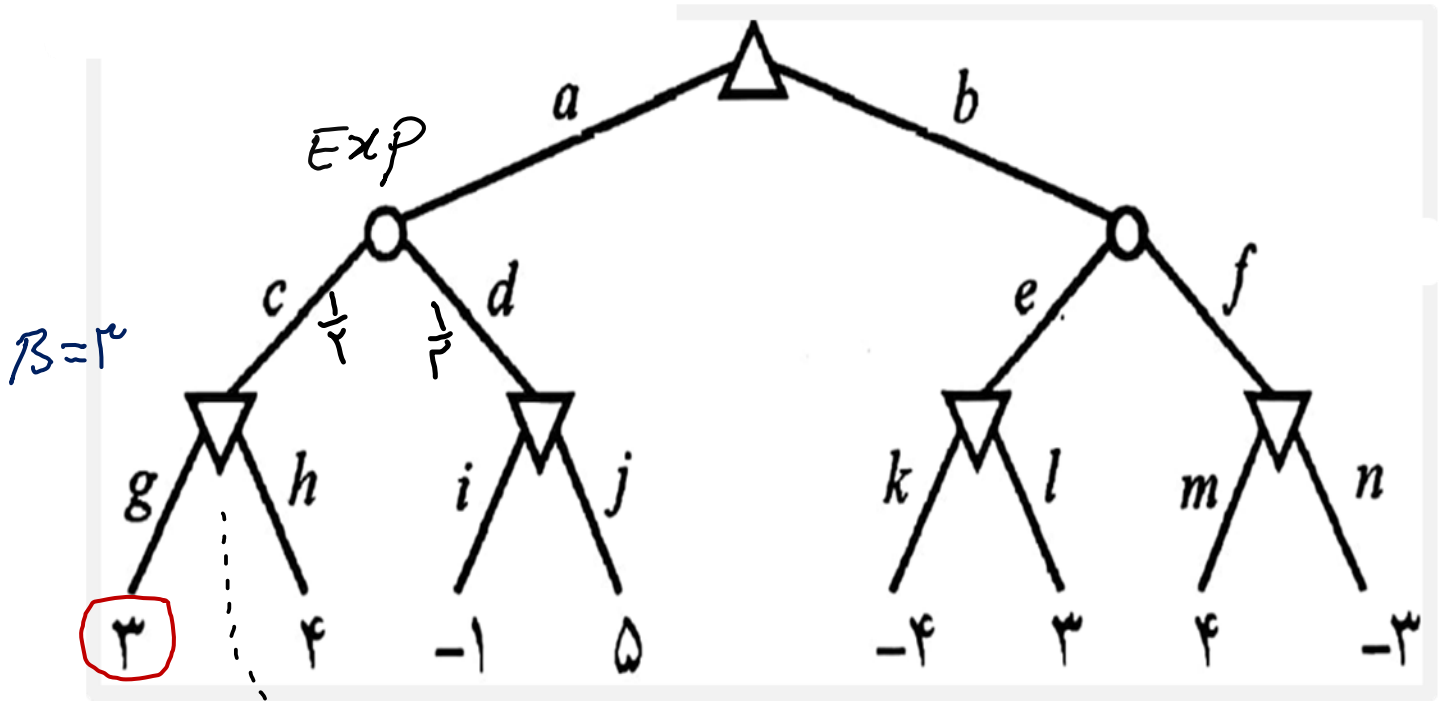
۲) *l-n*

۳) *j-l-n*

۴) هیچ هرسی رخ نخواهد داد

$\alpha = -\infty$

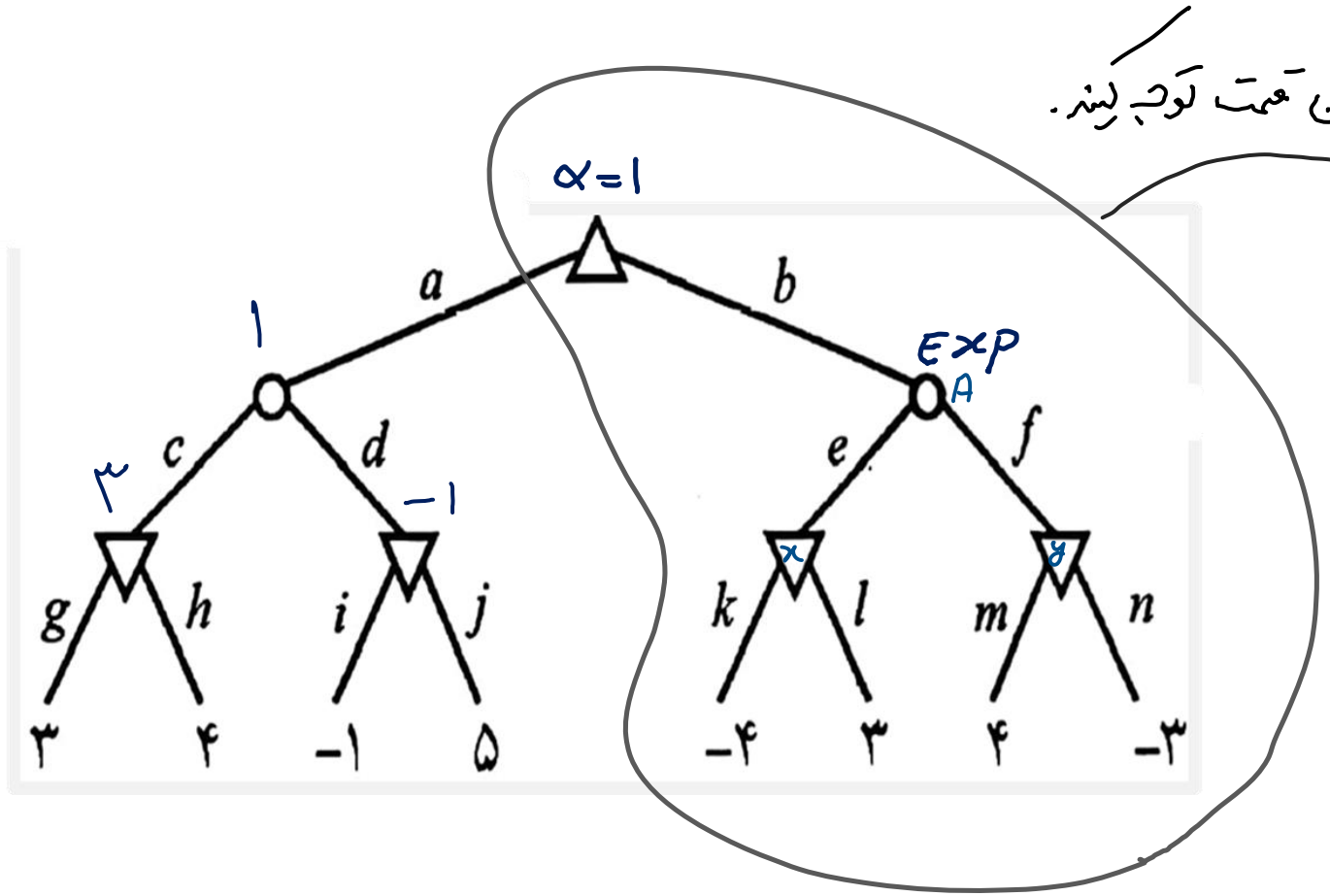
$-\infty \le u \le +\infty$

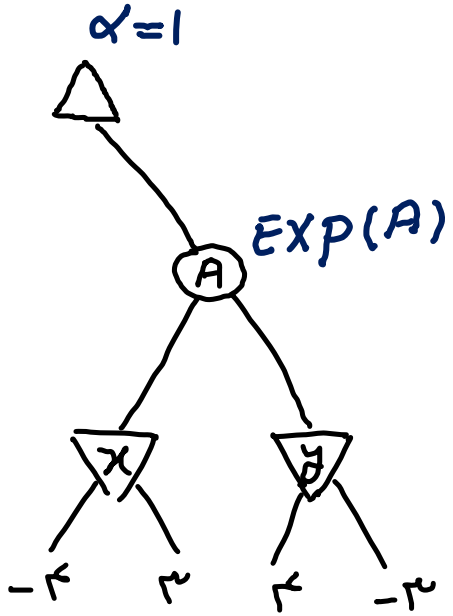


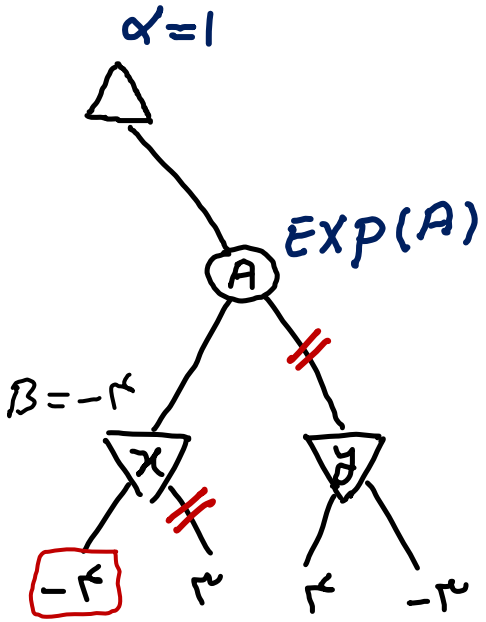
$\beta = +\infty$

$\max Exp = \frac{1}{2}(۳) + \frac{1}{2}(۵) = ۴$ → h هرس نمی‌شود باید ملاقات شود.

به صورتی درخت را اول تا وقتی α هنوز -۵ است هرس نخواهد.


به این قسمت توجه کنید.
α=1
a
b
۱
EXP
A
c
d
e
f
۳
-۱
x
y
g
h
i
j
k
l
m
n
۳
۴
-۱
۵
-۴
۳
۴
-۳


α=1
EXP(A)
A
x
y
-۴
۳
۴
-۳



پس از صلاقات اولین فرزند گره x عددی

در x داریم $\beta = -۴$.

امکان هرس را در گره شانس بررسی می‌کنیم:

$$\max Exp(A) = \frac{1}{۲}(-۴) + \frac{1}{۲}(۵) = \frac{1}{۲}$$

$$\Rightarrow \quad \max Exp(A) < \alpha$$

$\Rightarrow$ سایر فرزندان وی خانه کوس می‌شوند.

[توجه کنید، در این لحظه
در x $\beta = -۴$ است و امکان ندارد
β از مقدار فعلی اش بیشتر شود.
مقدار y هم با توجه به $-۵ \le u \le ۵$
حداکثر ۵ است.]

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

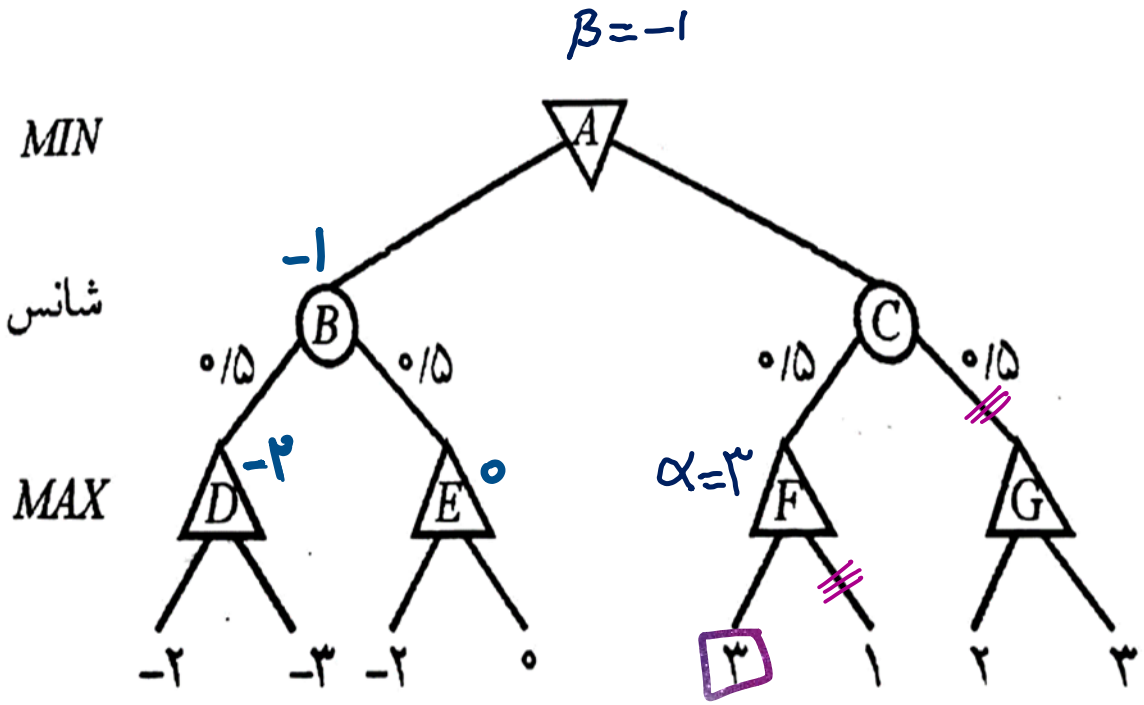
A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۷ فصل ۵

مثال : $u \in [-۳, +۳]$



درگره F : پس از ملاقات اولین فرزند $\alpha = ۳$
به بیشترین کران سودمندی رسید پس
فرزند دوم هرس شد.

درگره C :

$$\min Exp(C) = \frac{۱}{۲}(۳) + \frac{۱}{۲}(-۳) = ۰$$

پس $\min EXP(C) \geq \beta$ و G هرس
می شود.

بررسی چالش‌های جستجوی رقابتی و پیاده‌سازی گره‌های بازی در عمل:

حالات تکراری

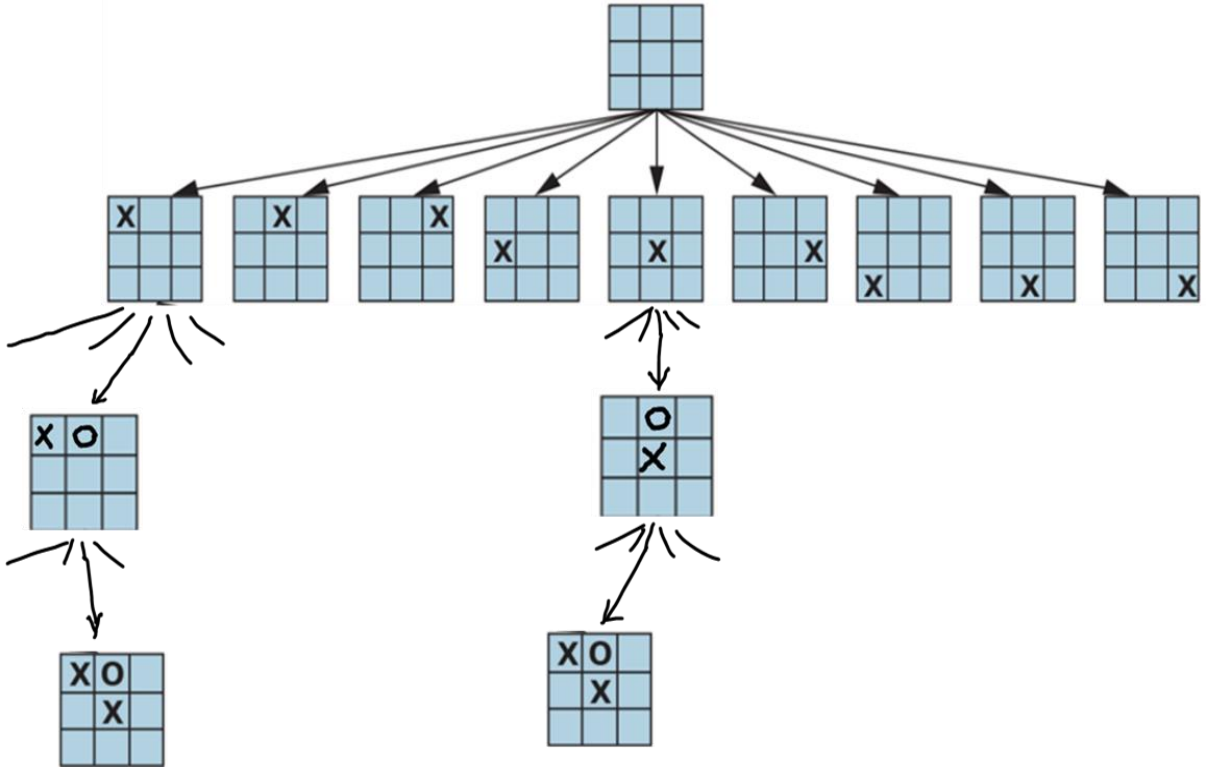
عمق زیاد پایانی‌ها

تابع ارزیابی

اثر افق

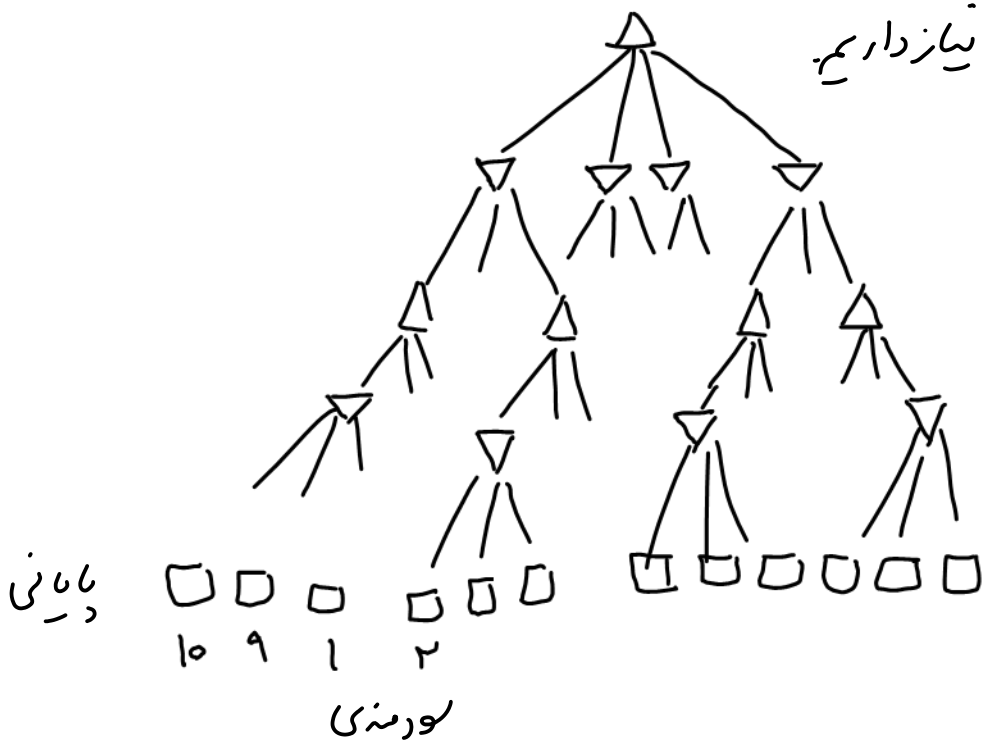
هرس پیش رو

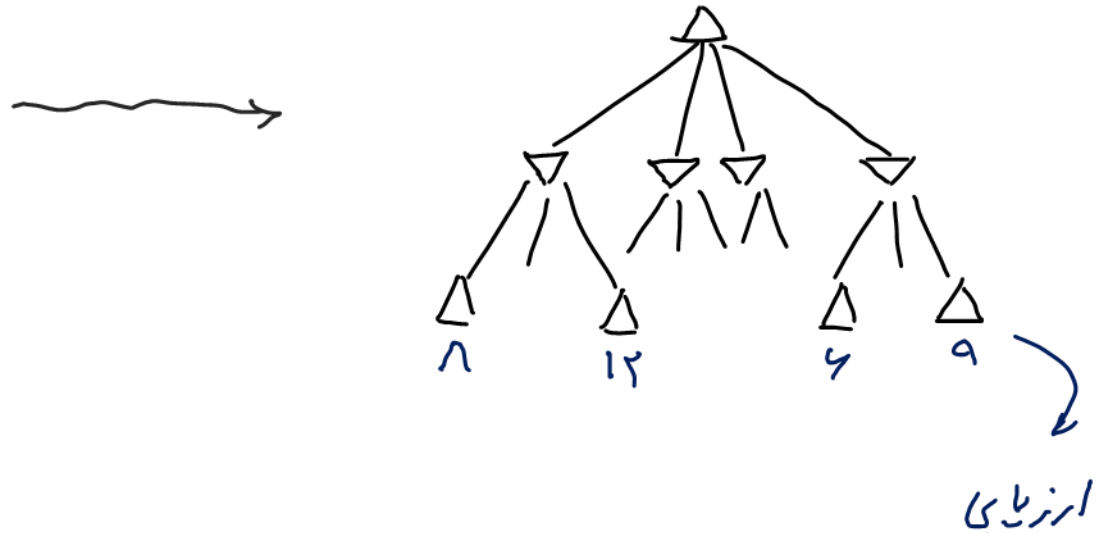
transposition table



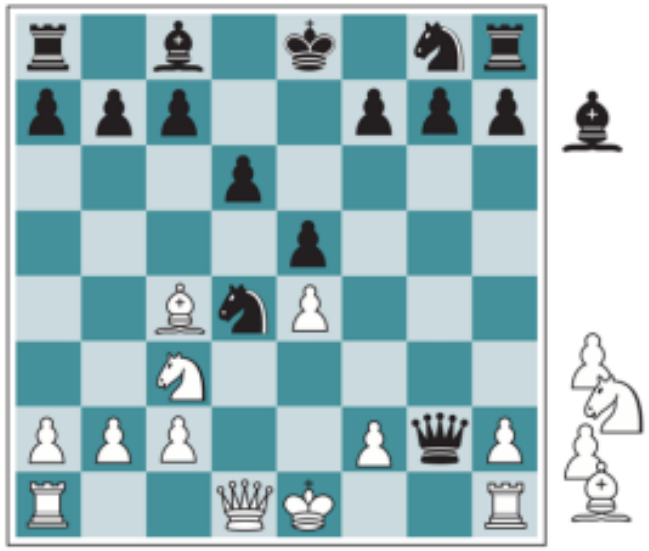
در عمل دو حالت از بازی چندین بار
در درخت کولید می شود. برای جلوگیری از
بررسی مجدد آن ها می توان حالات تکمیل شده
را در یک جدول درهم ساز (Hash) درج و
ذخیره کرد.

تابع ارزیاب: گاه (مانند شطرنج) حالت پایانی ممکن است در عمق بسیار زیاد باشد برای صرفه جویی در زمان درخت را تا عمق d بررسی می‌کنیم پس برای حالات غیر پایانی به تابع ارزیاب نیاز داریم.





مثال:



یک ارزیاب ساده از منیان برای مهره‌های شطرنج به صورت مجموع وزن‌دار از امتیاز هر نوع مهره قابل محاسبه است.

$$\text{EVAL}(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s) = \sum_{i=1}^{n} w_i f_i(s)\,,$$

* در اسلایدهای قبل به این موضوع پی برده‌ایم که آیا استفاده از مقادیر ارزیابی به جای مقادیر دقیق سودمندی، باعث نمی‌شود که الگوریتم minimax به جای سیاست بهینه، اقدام نادرست را انتخاب کند؟

در پایان به این جواب خواهیم رسید که:

الف: در بازی بدون عامل شانس: کافیست تابع ارزیاب، ترتیب سودمندی‌ها را رعایت کند تا باز هم سیاست بهینه انتخاب شود.

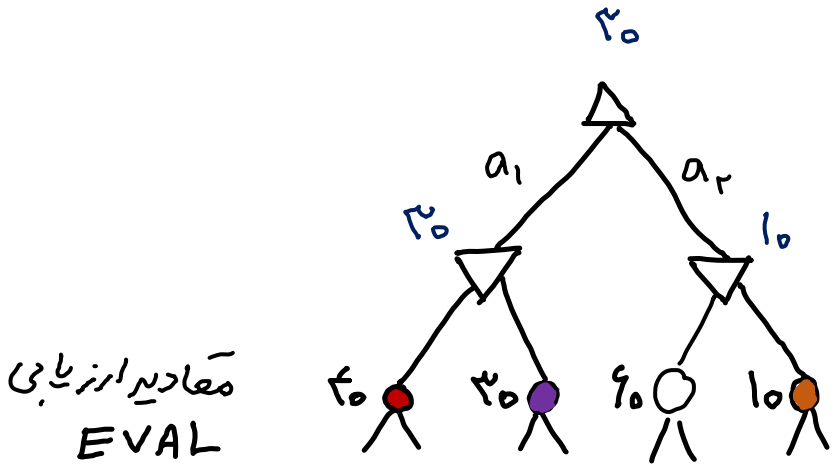
ب: در بازی با عامل شانس: کافیست تابع ارزیاب ترتیب را رعایت کند و رابطه‌ی خطی بین ارزیاب‌ها و سودمندی وجود داشته باشد تا باز هم سیاست بهینه انتخاب شود.

$$f = a\,u + b \qquad (a>0)$$

f: ارزیاب

u: سودمندی دقیق

بدون عامل شانس: کافیت ترتیب مقادیر رعایت شود:



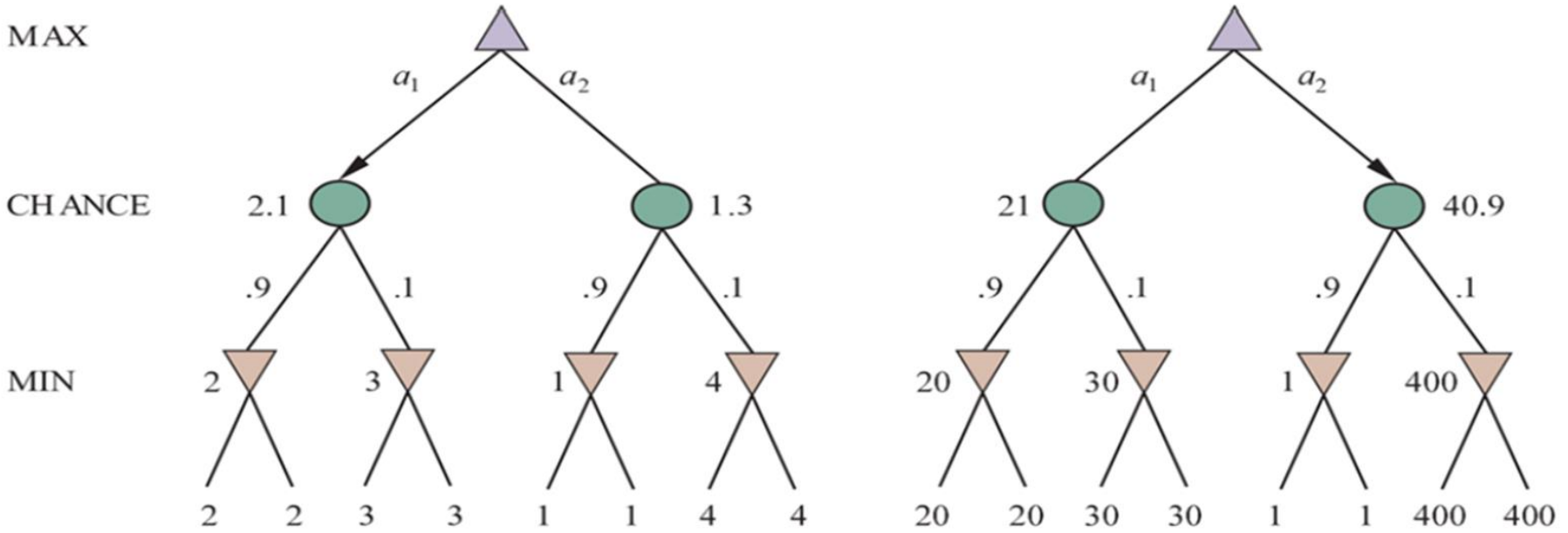




اجرای minimax درهردو درخت، باعث انتخاب اقدام بهینه a_1 می‌شود.

اما اگر عامل ت نسی داشته باشیم:

فقط حفظ ترتیب کافی نیست بلکه باید $f(u) = au + b$ باشد برای ثابت‌های $a > 0$ و $b \in \mathbb{R}$.



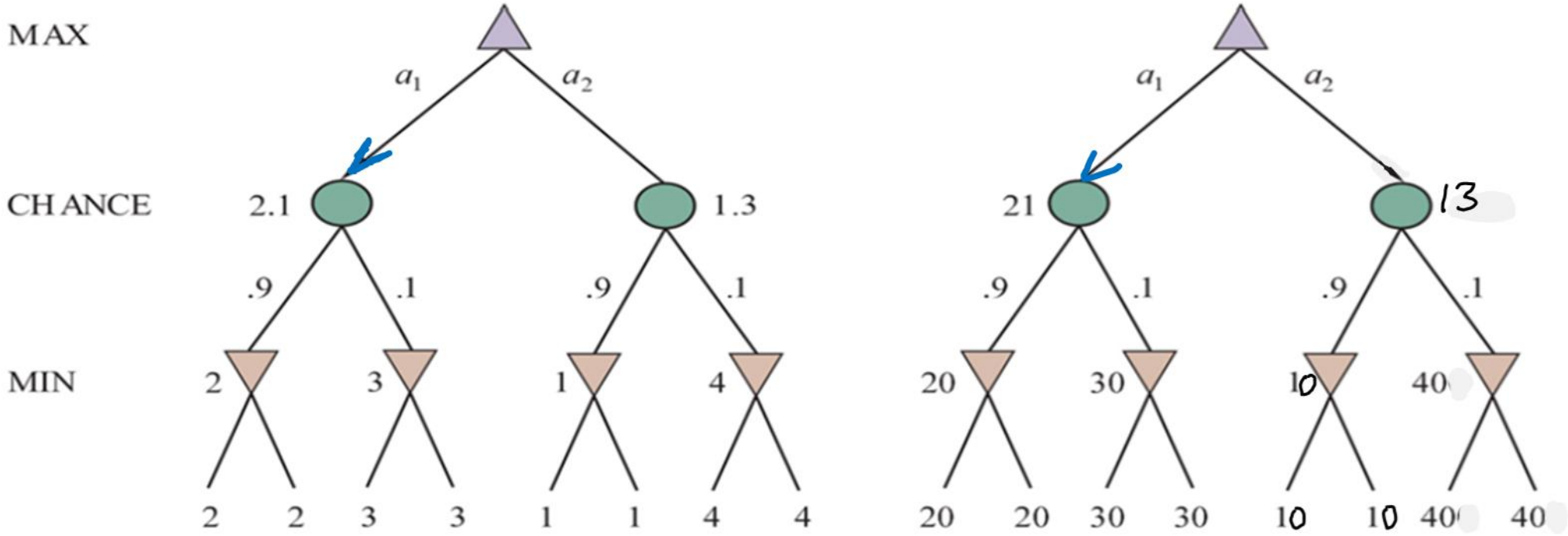
سه سودمندی ←

ارزیابی ←

با آن که تابع ارزیابی، ترتیب را رعایت کرده است،

اجرای minimax با مقادیر واقعی سودمندی، باعث انتخاب a_1 اما

با استفاده از ارزیابی‌ها باعث انتخاب a_2 می‌شود.



$u_i \longrightarrow f(u_i) = 10\, u_i$

اجرای minimax در هر دو درخت باعث انتخاب a_1 می‌شود.

(رابطه‌ی خطی $f = 10\,u$ بین ارزیابی و سودمندی وجود دارد.)

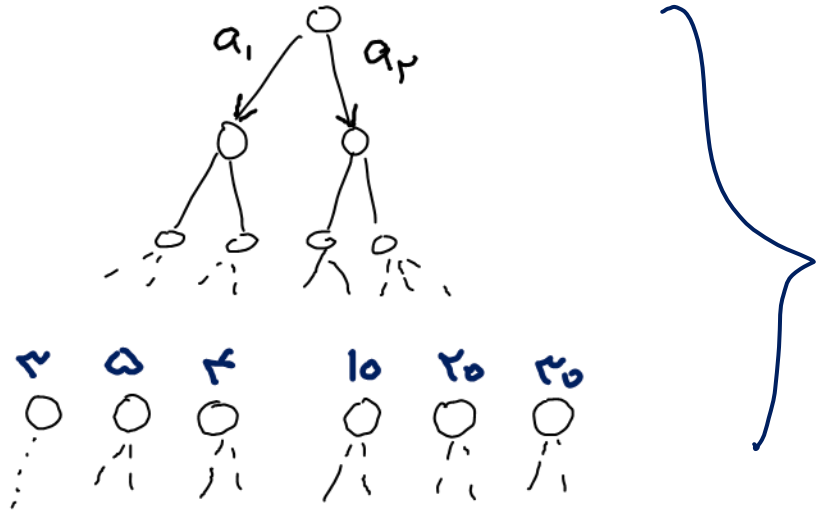
Horizon effect

وقتی درخت بازی را فقط تا یک عمق محدود d در نظر می‌گیریم ممکن است باعث خطا در انتخاب سیاست بهینه شود. مشکل اثر افق (افقِ دید) را با بیت تا حد امکان کنترل کنیم.

مشکل اثر افق

عمق در نظر گرفته شده برای جستجو

← افقِ دید.



پایانی‌ها



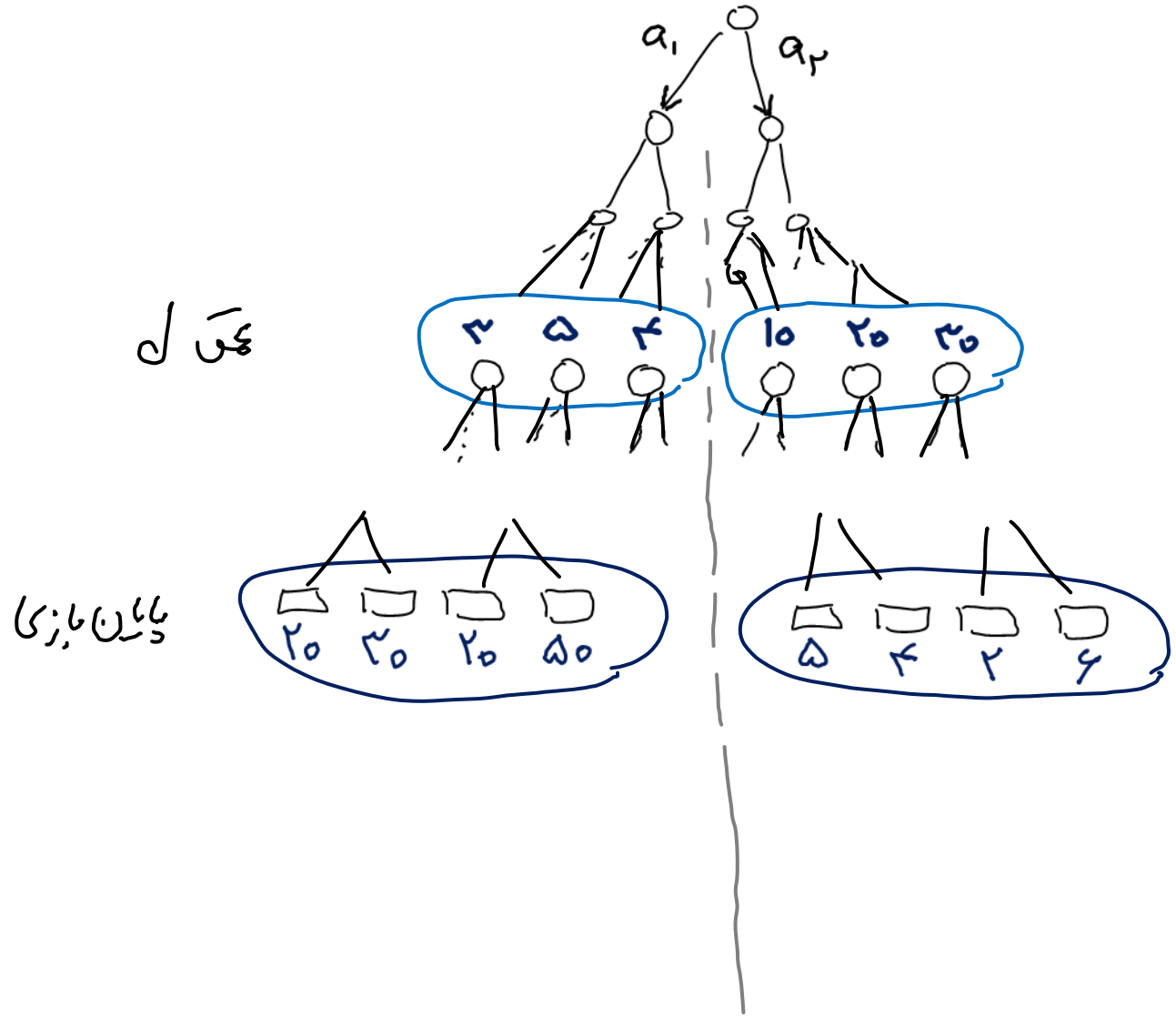
اگر بازی را تا انتها بررسی کنیم معلوم است که

انجام a_1 درنهایت به سود مندی بیشتری می‌رسد.

اما اگر تا عمق l درخت را رسم کنیم به نظر می‌رسد

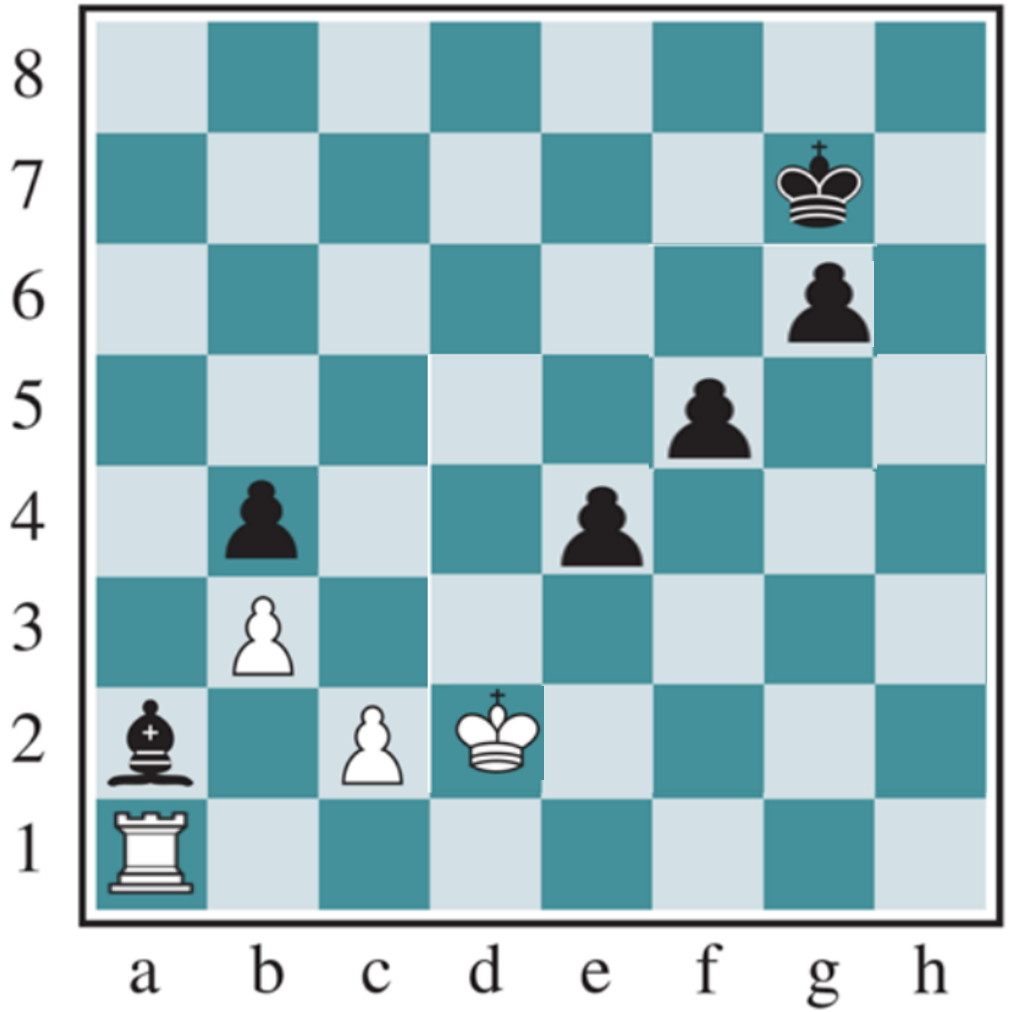
انجام a_2 بهتر است. این یکی از مشکلات

الگوریتم است.



Horizon effect

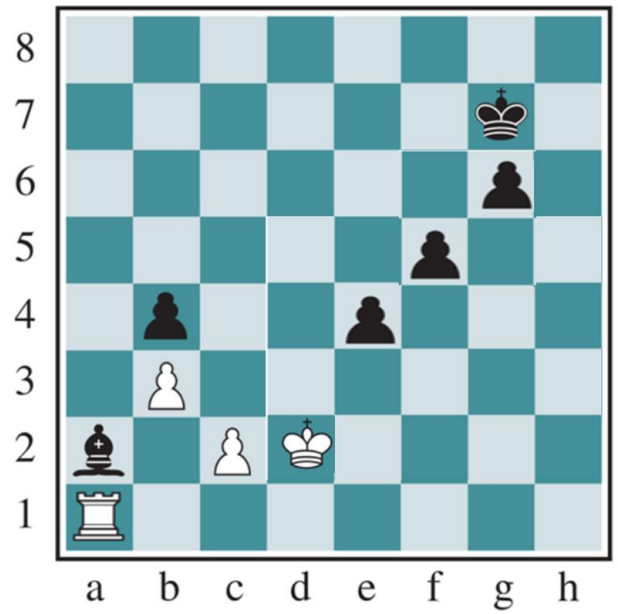
به تاخیر انداختن ضرری که قطعی شده است با دادن تلفات بیشتر:



[اتفاقی که در بازارهای مالی مانند فارکس یا بورس

نیز بسیار رخ می‌دهد]

در این مثال عامل مهره‌ی سیاه است:
MAX



اکنون نوبت مهره‌ی سیاه است. فیل سیاه در واقع
ازدست رفته است. بهتر است فیل را لااقل با یک
پیاده تعویض کند. (حرکت بهتر Fxb3
است که در ادامه حریف با
cxb3 جواب خواهد داد)

اما اگر مهره‌ی سیاه فقط دو یا سه حرکت بعدی را در نظر بگیرد. (عمق محدود) می‌تواند با کشی دادن شاه سفید با ۳ پیاده‌ای
و ازدست دادن آنها، موقتاً مهره‌ی فیل را روی صفحه حفظ کند. (اشتباه به دلیل اتفاق)

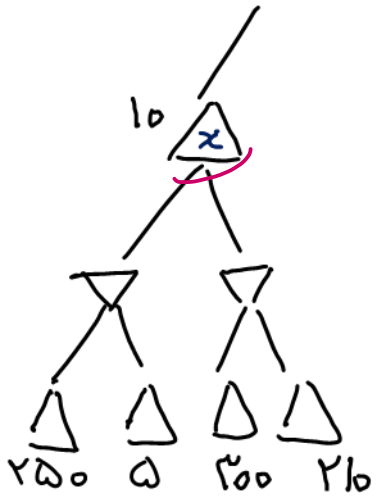
راهکارهایی برای کنترل خطاهای حاصل از محدودیت عمق:

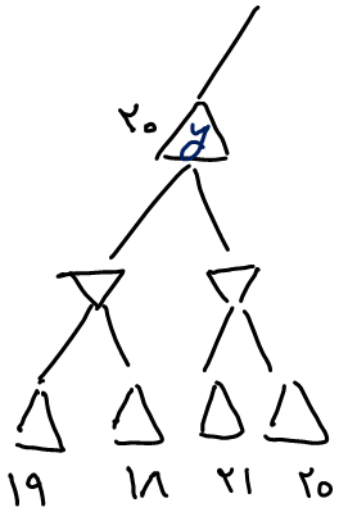
* استفاده از حالات stable :

حالات ایستا stable وضعیت‌هایی هستند که مقدار بسیار مهمی از آن‌ها تا چند حرکت آینده، تغییر چندانی نخواهد داشت (با احتمال زیاد)
حالات غیرایستا، (دینامیکی) حالاتی هستند که ممکن است به‌زودی بسیار مهمی آن‌ها به میزان زیادی تغییر کند.

راهکار: اگر حالات غیرایستا را نادیده گرفته و فقط برای حالات stable تابع ارزش داشته باشیم
می‌توانیم به بهینه بودن انتخاب‌هایمان با احتمال بیشتری اطمینان کنیم.

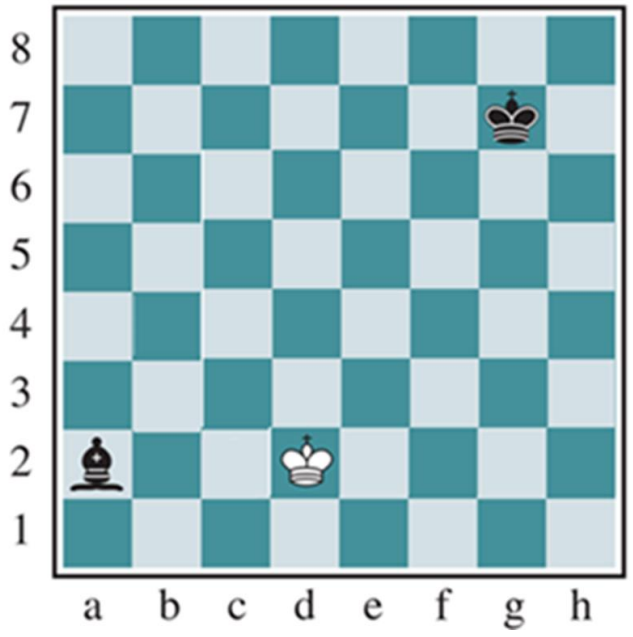
حالتِ x نیست stable





حالتِ y است stable.

مثال: در این مثال، مهره‌ی سفید، عامل است:

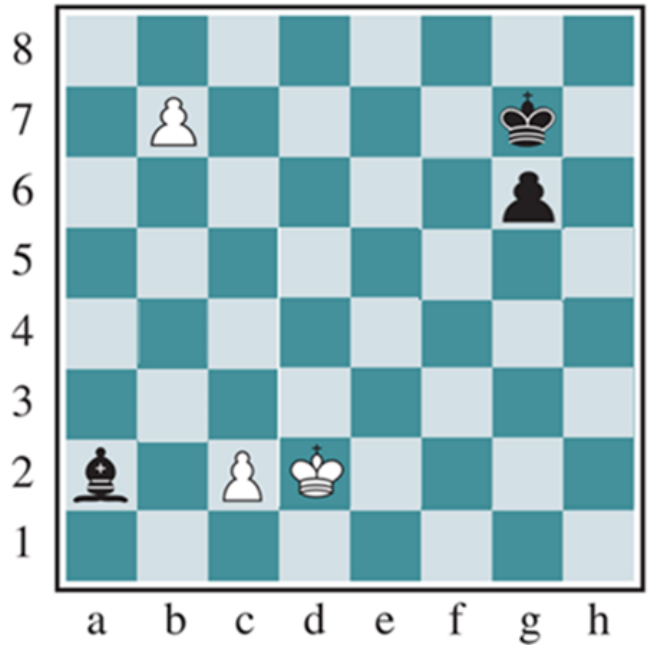


حالت:

stable

ایستا

در ادامه، هرکدام بازیکنان نسبت به هم تغییر جهانی نخواهد کرد.

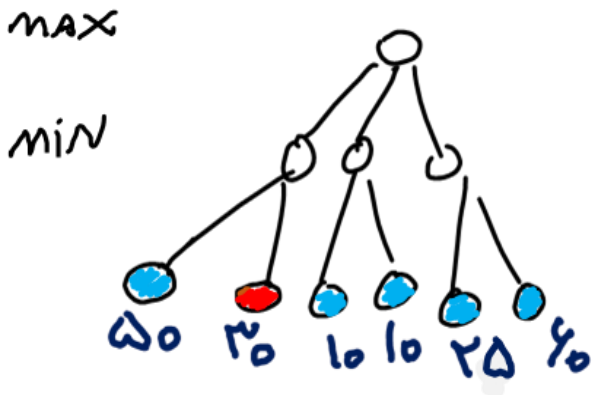


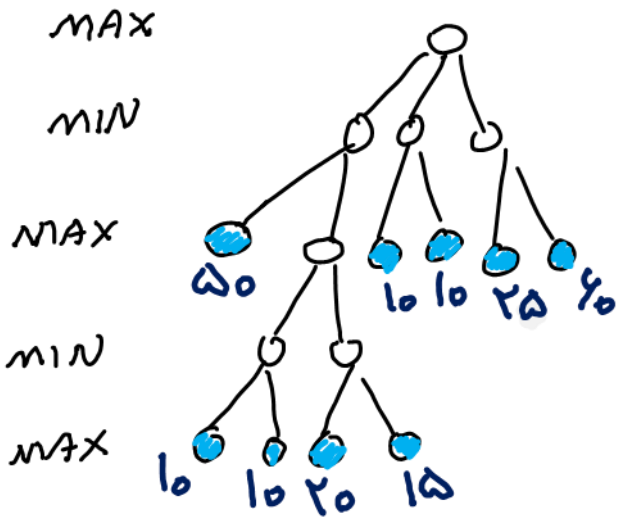
حالت غیرایستا: (ناپایدار)

ممکن است یک وقت بعد سرباز به وزیر تبدیل شود.

باید سعی کنیم از حالات پایدار به عنوان برگ ها در درخت با عمق محدود استفاده کنیم:

حالت غیر ایستا را آنقدر گسترش دهیم تا به حالات ایستا برسیم:

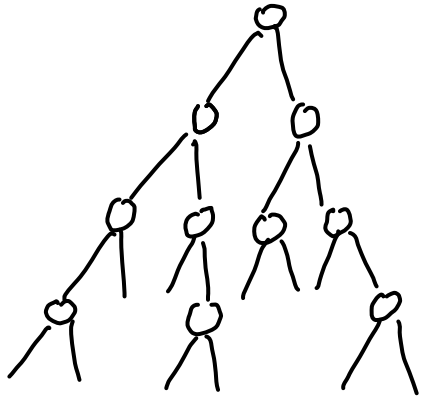


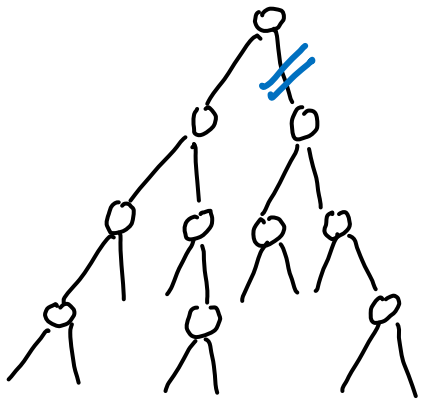


ایستا

غیرایستا

حالا از این درخت با عمق محدود استفاده کنیم.





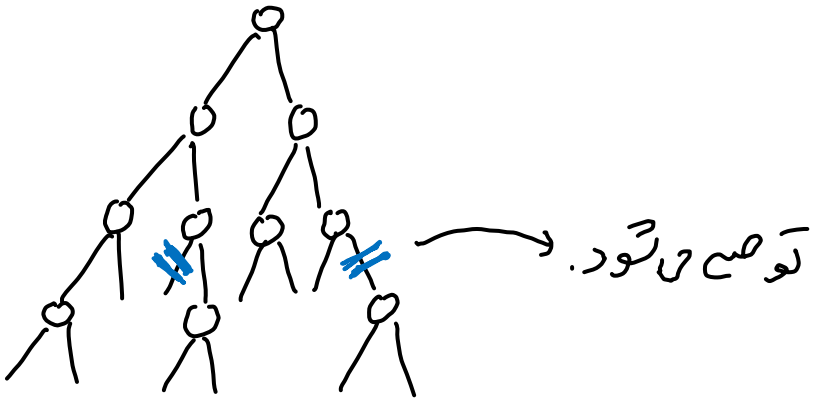
Forward pruning , beam search

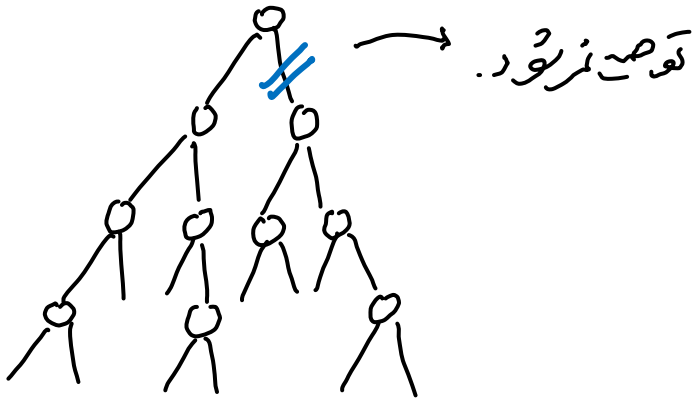
در هرس α-β ما هرس را از انتهای سرشاخه‌ها انجام می‌دهیم.

در هرس پیش‌رو (Forward) از همان ابتدا، برخی شاخه‌ها که امید کمتری
نیستند را هرس می‌کنیم تا فاکتور انشعاب کاهش یابد.

مثلاً در بازی شطرنج، اغلب اوقات مهره‌ی سفید با چند حرکت شناخته‌شده مثل
e4 یا d4 یا c4 بازی را شروع می‌کند. تعداد زیادی از
حرکات به عنوان شروع خوبی نیستند را کنار می‌گذارد.

در اغلب بازی ها، استفاده از هوس پیشرو در شروع کار (عمق کم) مناسب نیست زیرا ممکن است پایانی بهینه را از دست بدهیم. استفاده از این هوس در عمق بیشتر، مناسب‌تر است.





گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

بخش دوم از جلسه ۷ : فصل ۶

Chapter 6 Constraint Satisfaction Problems

CSP :

n = تعداد متغیرها

d = تعداد عناصر دامنه

$X = \{X_1, X_2, \dots, X_n\}$ مجموعه‌ی متغیرها:

$D \quad D_1 \quad D_2 \quad \dots \quad D_n$ مجموعه‌ی دامنه‌ها:

Constraint مجموعه‌ی محدودیت‌ها:

هدف: یک انتساب کامل سازگار.

یک انتساب (مقداردهی) کامل که: همه متغیرها مقداردهی کرده و شرایط داده شده را هم نقض نکند (سازگار باشد)

مثال:

متغیرها دامنه

$x_1, x_2, x_3, x_4 \in \{1, 2, \dots, 5\}$

محدودیت‌ها

$$x_1 + x_2 = 5$$

$$x_1 \leq 2$$

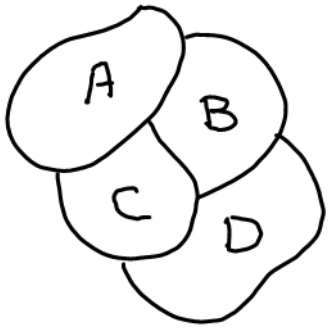
$$x_2 + x_3 = 4$$

$$x_3 x_4 = 3$$

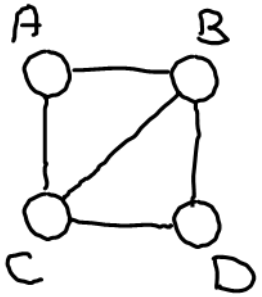
Example problem: Map coloring

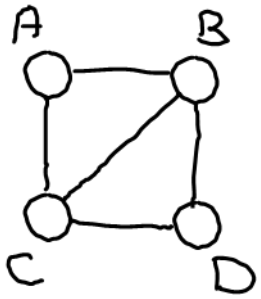
دامنه : { ● , ● , ● }

محدودیت : مجاورها همرنگ نباشند.

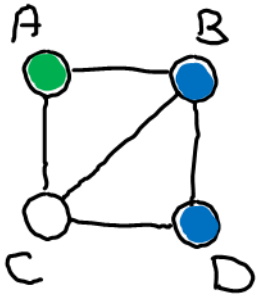


گراف :

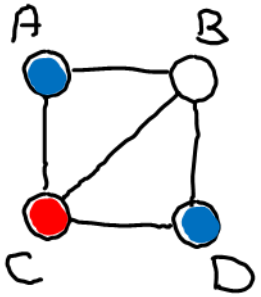




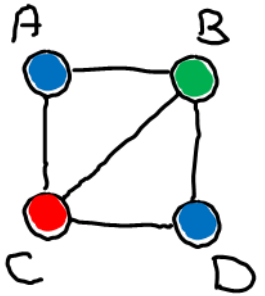
انتساب همگن
(سازگار است)



انتساب جزئی ناسازگار



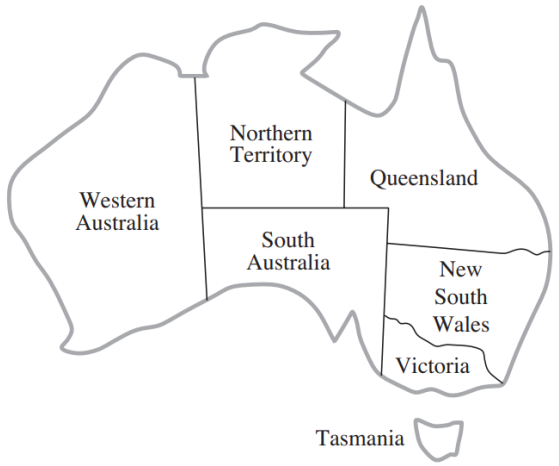
انتساب جزئی سازگار



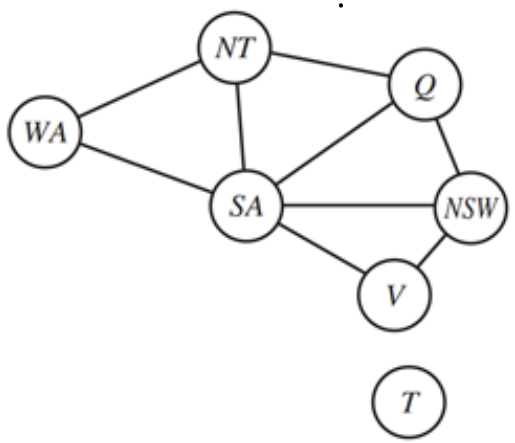
انتساب کامل سازگار

رنگ آمیزی نقشه‌ی استرالیا: (یک مثال از csp)

- Variables WA, NT, Q, NSW, V, SA, T
- Domains {red ,green ,blue}
- Constraints: متمایز بودن مجاورها



(از این مثال در جلسات بعد استفاده می‌کنیم)



انواع محدودیت‌ها

constraints

- **unary constraint,** (یکانی) $x_2 > 2$
- **binary constraint** (دوتایی) $x_1 + x_4 = 5$
- higher-order constraints. (چندتایی) $x_1 - x_2 + x_5 = 10$
- **global constraint.** (عمومی) $\text{AllDiF}(x_1, x_2, \ldots, x_n)$

(x) می‌توان مسائل CSP را به مسائل CSP فقط با محدودیت باینری (دوتایی) تبدیل کرد.

محدودیت یکانی ← اصلاح دامنه

محدودیت چندتایی ← معرفی متغیرهای جدید و آنبرگراف.

مثال: این CSP را به CSP باینری تبدیل کنیم:

x, y, z

$D = \{0, 1, \dots, 9\}$

$z = x + y$

$x \geq 1$

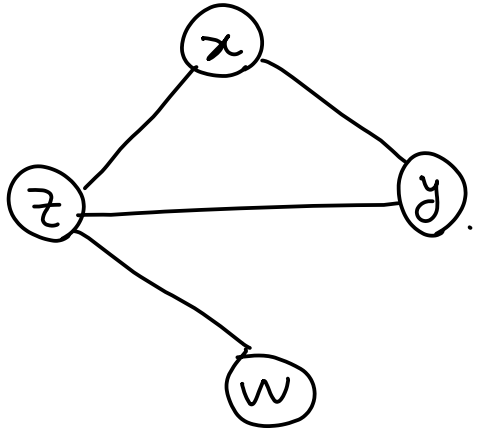
$x + z < 4$

$yz = 4$

$x < 2y$

حل:

با اصلاح دامنه‌ی x شرط $x \geq 1$ حذف می‌شود. با معرفی متغیر $W=(x,y)$ محدودیت $z=x+y$ حذف می‌شود.



x, y, z, w

$D_y = D_z = \{0,1,\ldots,9\}$

$D_x = \{1,2,\ldots,9\}$

$D_w = \{(1,0),(1,1),\ldots,(9,9)\}$

Z با مجموع مولفه‌های W برابر باشد

$x+z<4$

$yz=6$

$x<2y$

مدل سازی ناکارآمد:

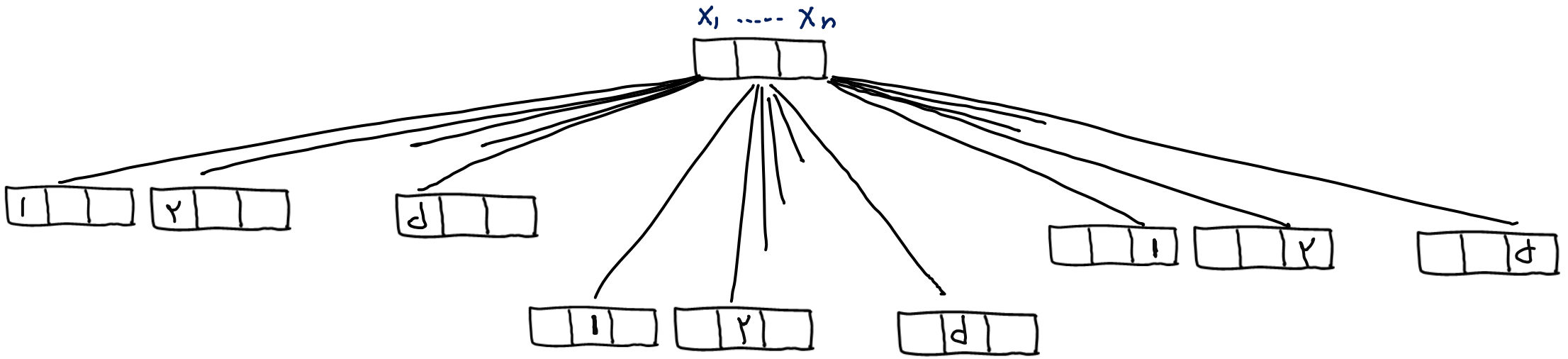
شروع: انتخاب عکس

گام اول: انتخاب یکی از متغیرها و یکی از مقادیر آن. [انتخاب هر متغیر و هر مقدار آن، یک شاخه از درخت باشد]

گام دوم: انتخاب یک متغیر دیگر و یکی از مقادیر آن. [انتخاب هر متغیر باقی مانده و هر مقدار آن، یک شاخه از زیر درخت باشد]

⋮

گام n ام: آخرین متغیر، مقادیر آن.



فاکتور انشعاب ریشه : n متغیر و هرکدام d مقدار دارند : nd

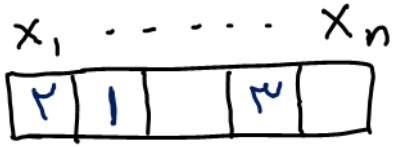
در لایه بعدی : $n-1$ متغیر و هرکدام d مقدار : $(n-1)d$

⋮ ⋮

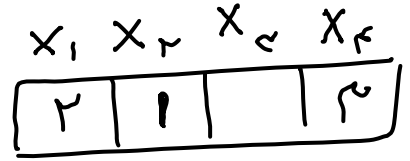
$$\text{تعداد برگ‌ها} = n! \, d^n = n d \times (n-1) d \times (n-2) d \cdots \times d$$

گره جواب: انتخاب کامل سازگار در عمق n قرار دارد. (در یکی از برگ‌ها)

نقاط ضعف این درخت جستجو: مرتبه زمانی بزرگ و نمایی ، گره‌های تکراری.



در $k!$ از گره‌ی دیده می‌شود. $\Rightarrow$ هر انتخاب جزئی k تایی

مثلاً گره 

را در $3!$ ترتیب می‌توان ایجاد کرد (می‌توان از x_1 شروع کرد یا از x_2 یا x_4)

اولین گام در مدل‌سازی: نوبت هر متغیر مشخص باشد. مثلاً ترتیب $X_1, X_2, \ldots, X_n$:

شیوه‌ی: انتساب بعدی

گام اول: مقداردهی به متغیر X_i. (مثلاً متغیر X_1)

گام دوم: مقداردهی به متغیر X_j. $(j \neq i)$ (مثلاً X_2)

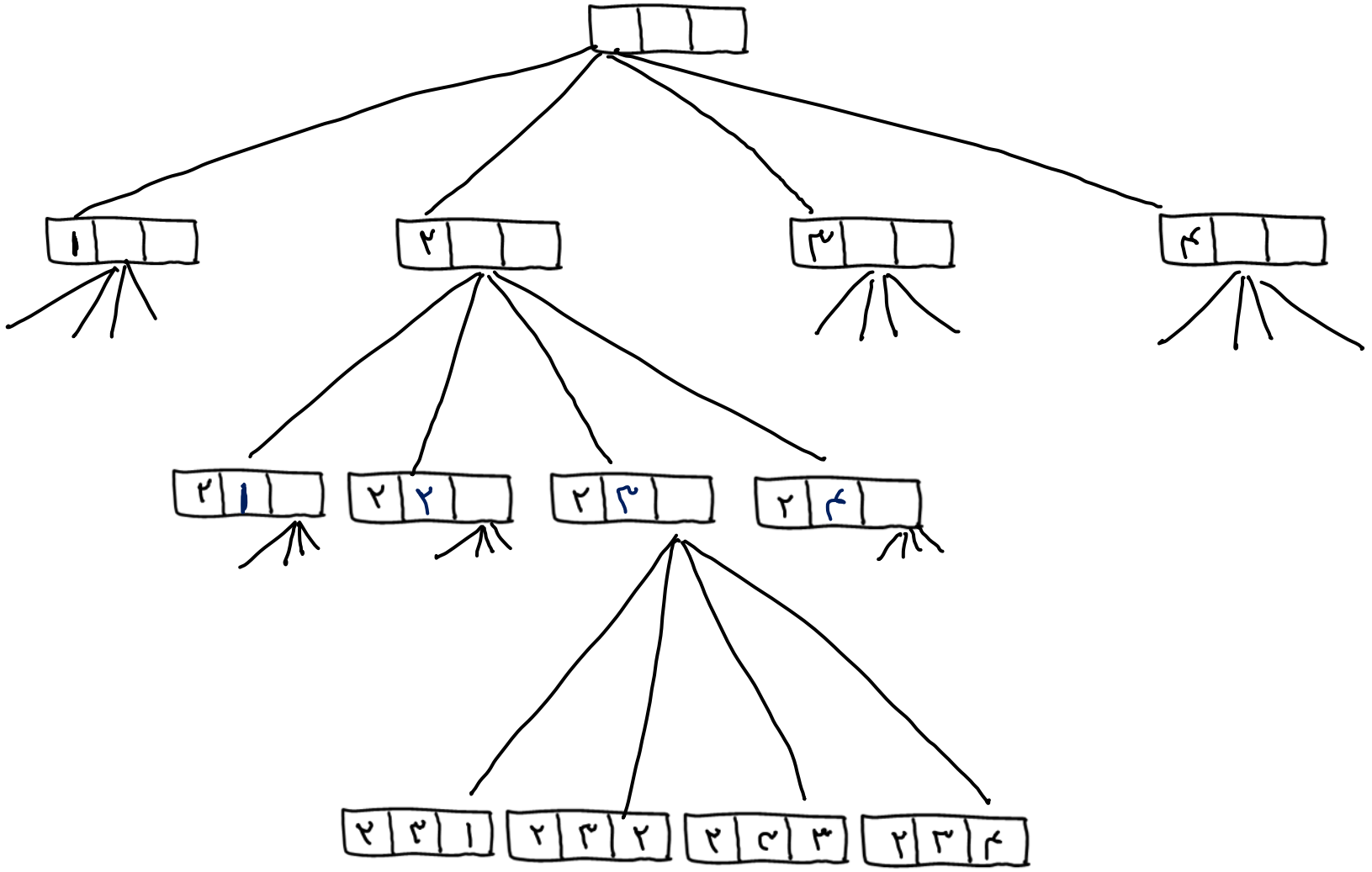
.
.
.

گام n ام: مقداردهی به متغیر X_n.

کاربرد:

حالات مقداردهی به X_1:

حالات مقداردهی به X_2:

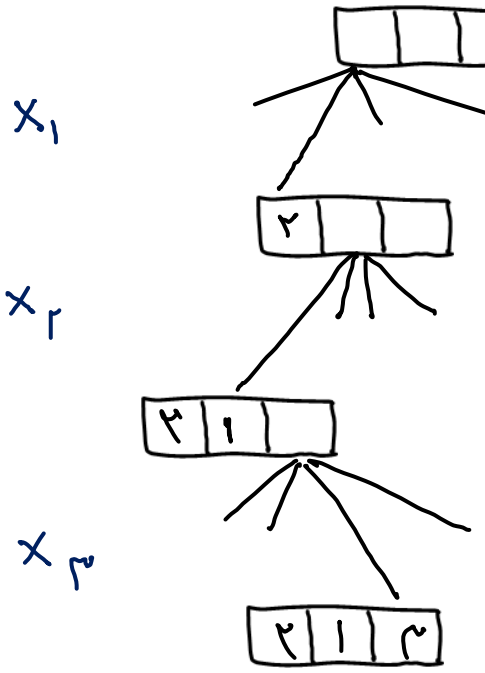


حالات مقداردهی به X_n:

در این روش:

$$\text{تعداد برگ‌ها} = d \times d \times \cdots \times d = d^n$$

و حالت تکراری نخواهیم داشت. مثلاً حالت ۲ ۱ ۳ فقط در یک گره دیده می‌شود.



BACKTRACKING-SEARCH

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return BACKTRACK({ }, csp)

function BACKTRACK(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if value is consistent with assignment then
      add {var = value} to assignment
      inferences ← INFERENCE(csp, var, value)
      if inferences ≠ failure then
        add inferences to assignment
        result ← BACKTRACK(assignment, csp)
        if result ≠ failure then
          return result
    remove {var = value} and inferences from assignment
  return failure
```

B.T جستجوی عقب‌گرد ساده

به شکل DFS جستجو می‌شود

هنگامه به ناسازگاری رسیدیم یعنی

یکی از محدودیت‌ها نقض شد،

به گره قبل (پدر) برمی‌گردیم.

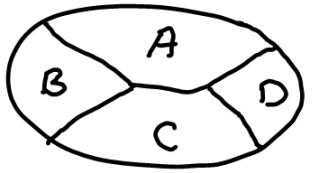
هدف در صورت وجود یکی از برگ‌ها

(در عمق n) است که انتخاب

کامل است و سازگار است.

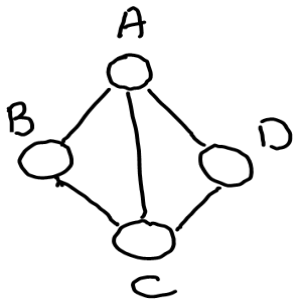
Example problem: Map coloring

- Variables A, B, C, D
- Domains {red ,green ,blue}
- Constraints: نواحی مجاور ، غیرهمرنگ باشند



$n = ۴$

$d = ۳$

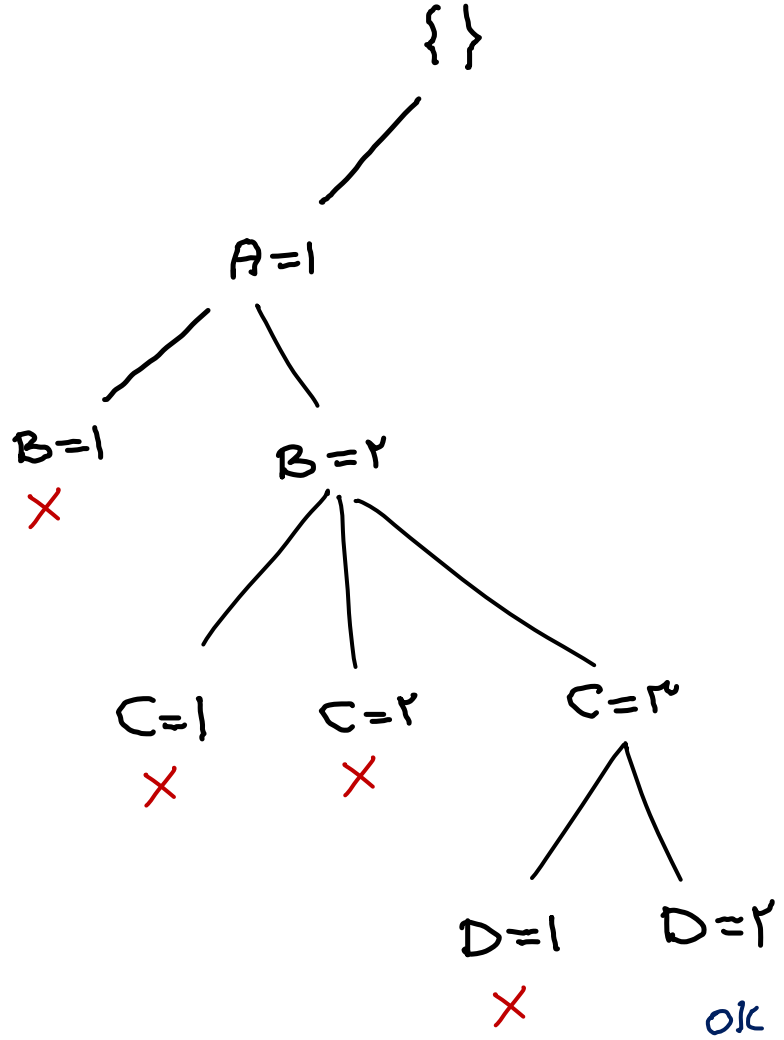


A

B

C

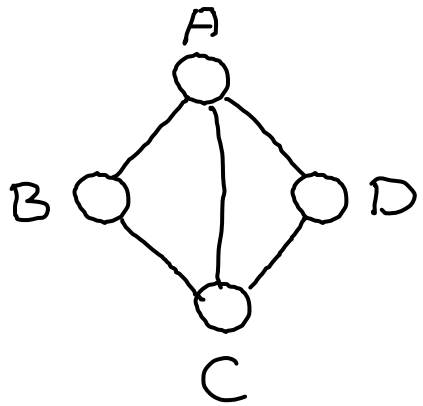
D



1 = Red

2 = Green

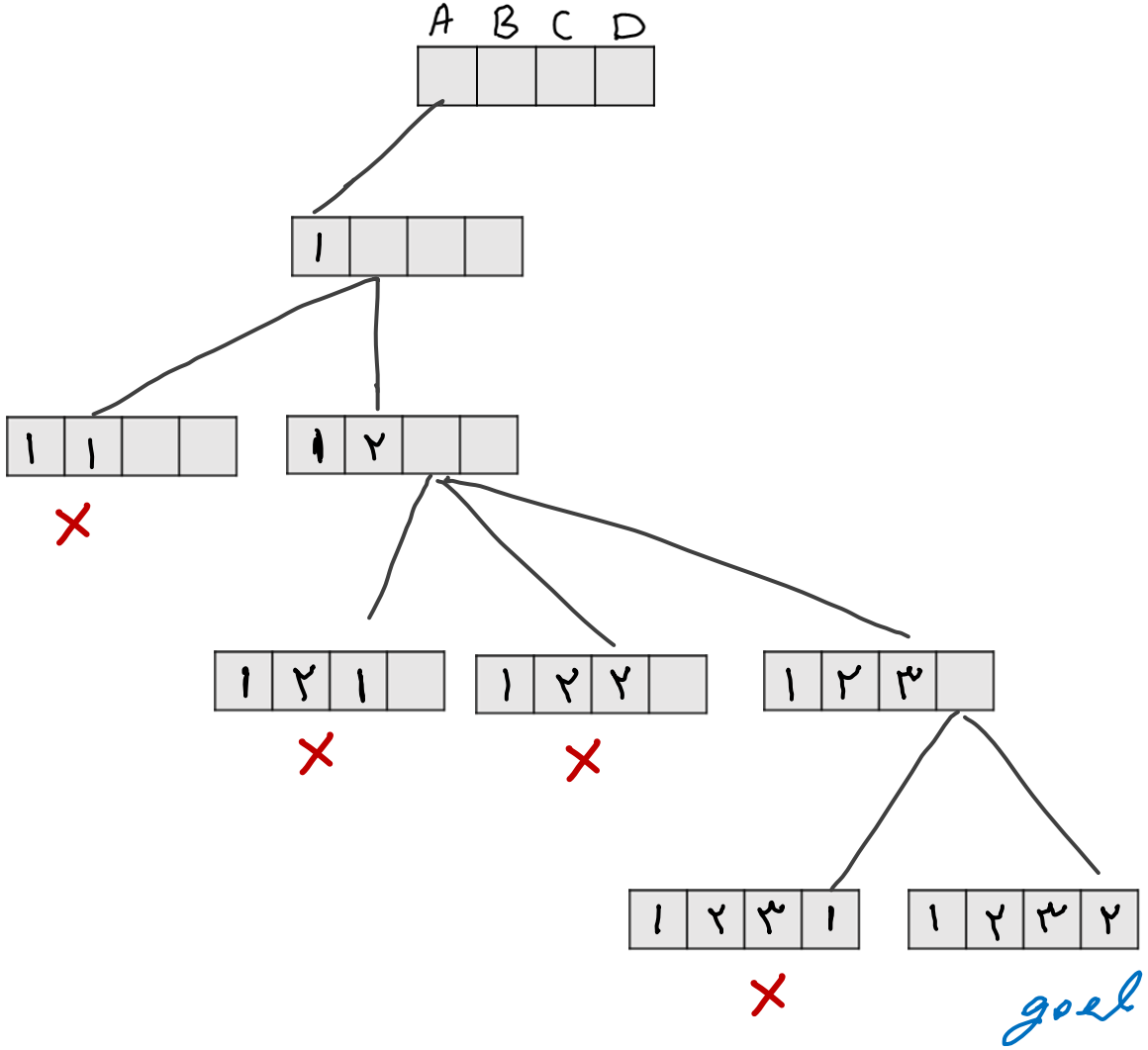
3 = Blue

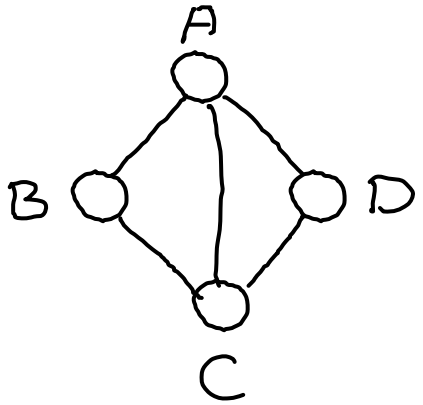


اولین انتساب کامل سازگاری که یافتیم جواب است.

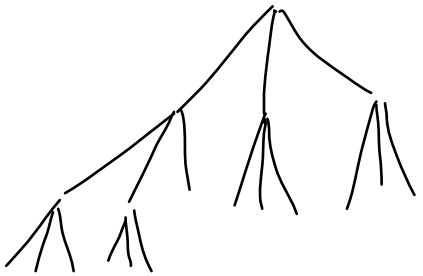
انتساب‌ها در درخت جستجوی عقب‌گرد مثال قبل

در واقع چنین است:





Variable and value ordering



$O(d^n)$ مرتبه‌ی زمانی BT

با انتخاب ترتیب مناسب متغیرها و ترتیب مناسب در مقداردهی ، می‌خواهیم زمان اجرا را تا حد امکان کاهش دهیم :

Variable and value ordering

هیوریستیک های مناب برای یافتن متغیر بعدی (ترتیب متغیرها) :

MRV

degree

هیوریستیک مناب برای ترتیب مقداردهی :

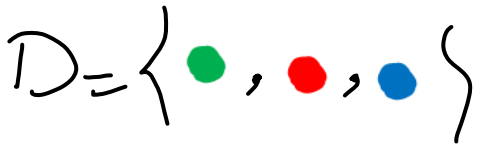
LCV

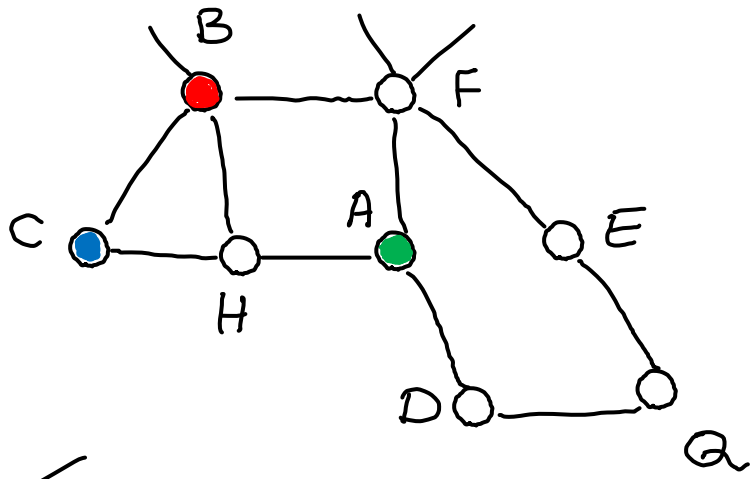
Most constrained variable (MCV)

minimum remaining values (MRV)

متغیر بعدی : آن که بیشترین محدودیت روی آن قرار گرفته

یعنی : کوچکترین دامنه‌ی مجاز برایش باقی مانده .





در این مثال دس از رنگ آمیزی A و B و C :

در دامنه‌ی F فقط یک رنگ مجاز باقی مانده

در دامنه D فقط دو رنگ مجاز باقی مانده

در دامنه‌ی E و G سه رنگ مجاز باقی مانده

در دامنه‌ی H صفر رنگ مجاز باقی مانده ← اولویت با H است .

$D = \{1, 2, 3\}$

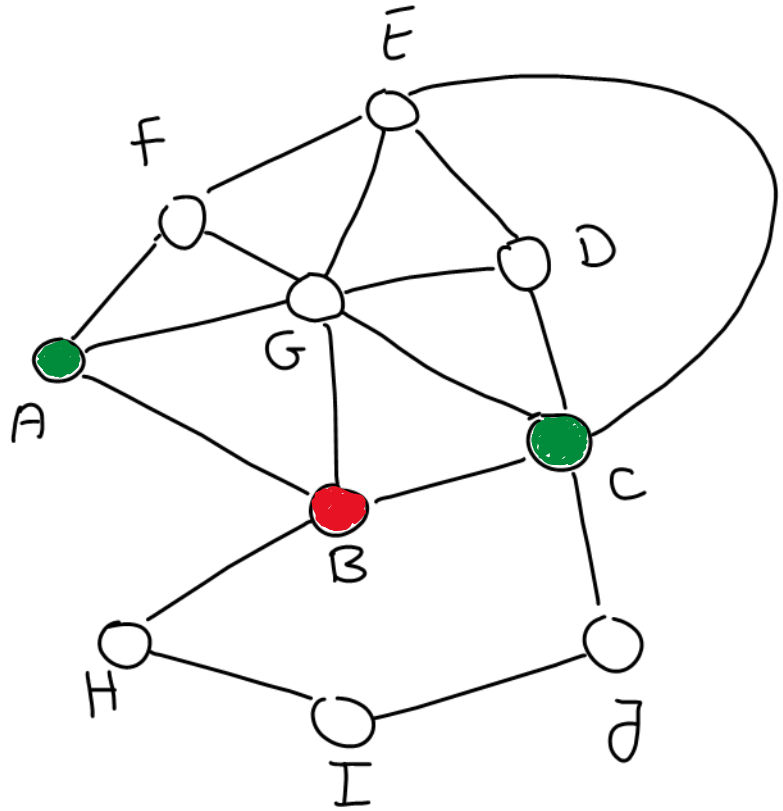
$A = 1$

$B = 2$

$C = 1$

?

G بیش از سایر گره‌ها محدودیت‌ه است. دامنه مجاز G اکنون تک‌عضوی است. پس طبق MRV متغیر G انتخاب می‌شود.



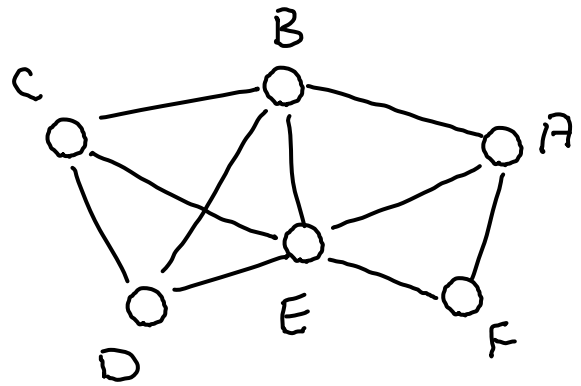
Most constraining variable

(most edges in graph)

degree.

اگر از نظر MRV گره‌ای یکسان بودند آن را که با گره‌های مقداردهی نشده‌ی بیشتری در ارتباط است انتخاب می‌کنیم. (هیوریستیک درجه‌ای)

$D = \{1, 2, 3\}$



در شروع: دامنه هم ۳ عضوی است. اما درجه‌ی E (از تمام گره‌های رنگ نشده) بیشتر از سایر گره‌هاست پس E انتخاب می‌شود.

انتخاب متغیر بعدی (ترکیب متغیر)

کمک می‌کنند مسیرهای منتهی به شکست زودتر شناخته شوند.

- MRV, MCV
- degree

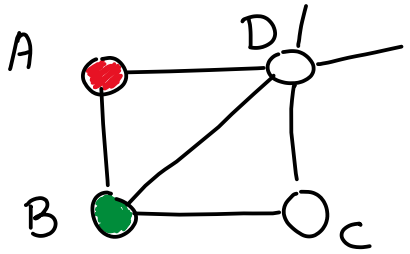
Least constraining value LCV

ترتیب مقادیر:

مثلاً هنگام مقداردهی به یک متغیر ابتدا از کدام مقدار آغاز کنیم؟

LCV می‌گوید مقداری را به متغیر فعلی بدهید که کمترین محدودیت را برای سایر گره‌های مقداردهی نشده به وجود آورد.

LCV ⟸ افزایش احتمال موفقیت. (یافتن موفقیت سریع‌تر)

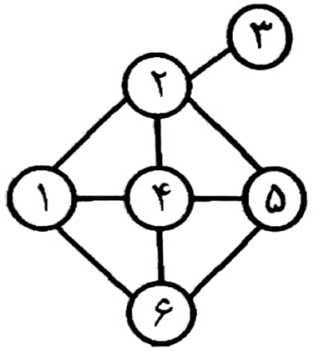


اگر C = ● برای D هیچ مقدار مجازی باقی نمی‌ماند.

اگر C = ● برای D یک مقدار مجاز باقی می‌ماند.

طبق LCV رنگ قرمز برای C انتخاب شود.

۱۳ـ در رنگ‌آمیزی گراف با روش ارضای محدودیت‌ها، پس از رنگ‌آمیزی نودهای یک و دو بهتر است کدام نود رنگ‌آمیزی شود؟ (مهندسی IT ـ دولتی ۹۴)

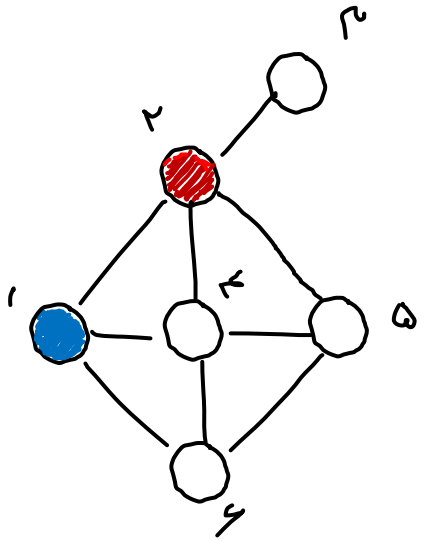


۱) ۴

۲) ۵

۳) ۶

۴) ۴ و ۵ با اولویت یکسان



مثلاً اگر دامنه متغیرها دراینجا شامل d رنگ باشه دراین مرحله:

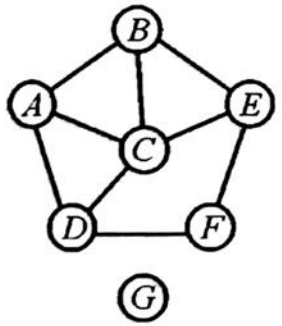
دامنه‌ی ۳ دارای d−۱ رنگ مجاز است.

دامنه‌ی ۵ و ۶ دارای d−۱ رنگ مجاز است.

دامنه‌ی ۴ دارای d−۲ رنگ مجاز است.

پس ④ انتخاب می‌شود.

۶ـ در یک مسئله می‌خواهیم نقشه را با سه رنگ *Red*(R)، *Green*(G) و *Blue*(B) رنگ‌آمیزی کنیم طوری که هیچ دو کشوری که با هم مرز مشترک دارند هم رنگ نشوند. اطلاعات نقشه را به صورت گراف محدودیت زیر نمایش داده‌ایم (وجود لبه بین دو گره نشانه وجود مرز مشترک بین دو کشور مربوط به گره‌هاست) براساس هیوریستیک‌های عمده مسائل *CSP* کدام گزینه زیر ترتیب بهتری برای دو کشوری که اول انتخاب می‌شوند است؟ (مکاترونیک ـ دولتی ۸۶)



۱) ۱ـ C و ۲ـ A

۲) ۱ـ C و ۲ـ F

۳) ۱ـ G و ۲ـ B

۴) ۱ـ G و ۲ـ F

در شروع همه دامنه‌ها ۳ عضوی‌اند. اما درجه‌ی C (ارتباط C با رنگ نشده‌ها) بیشتر است پس C انتخاب می‌شود. حالا گره‌های A, B, E, D هرکدام یک رنگ مجاز از دست داده‌اند و هرکدام از آن‌ها یا ۲ گره رنگ نشده رابطه دارد. پس گره دوم یکی از A, B, E, D است.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

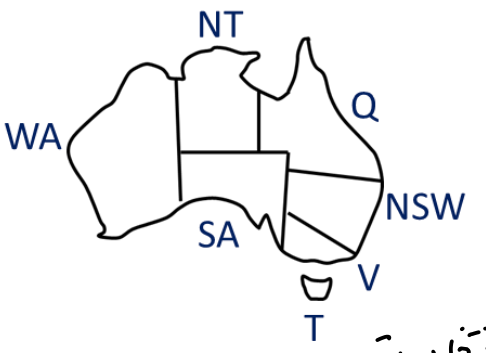
هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۸ فصل ۶

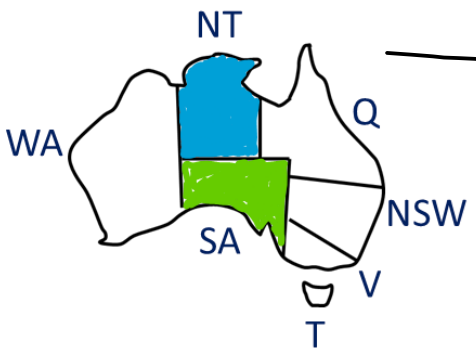
- Most constrained variable:
- Most constraining variable:

یادآوری جلسه قبل MRV: انتخاب متغیر با کمترین دامنه‌ی مجاز (بیشترین محدودشده)

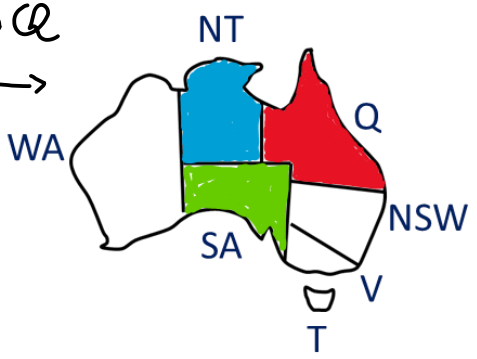




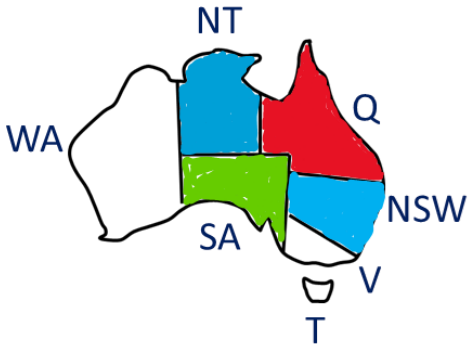
هم دامنه‌ها ۳ عضوی بودند SA انتخاب شد.

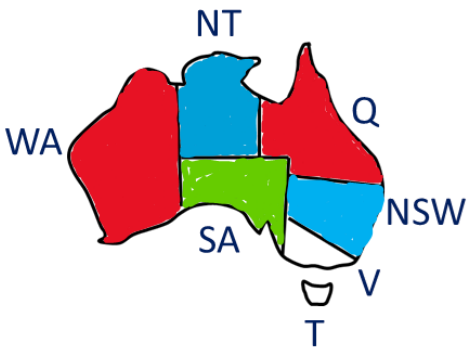


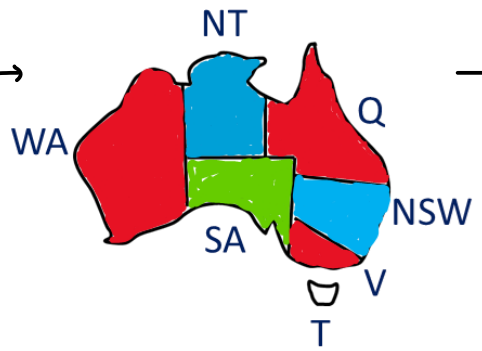
WA و Q خرد هستند

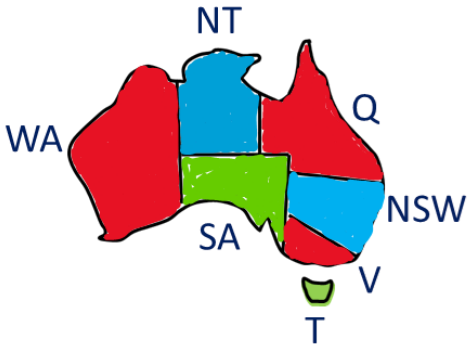


WA و NSW مناسب هستند



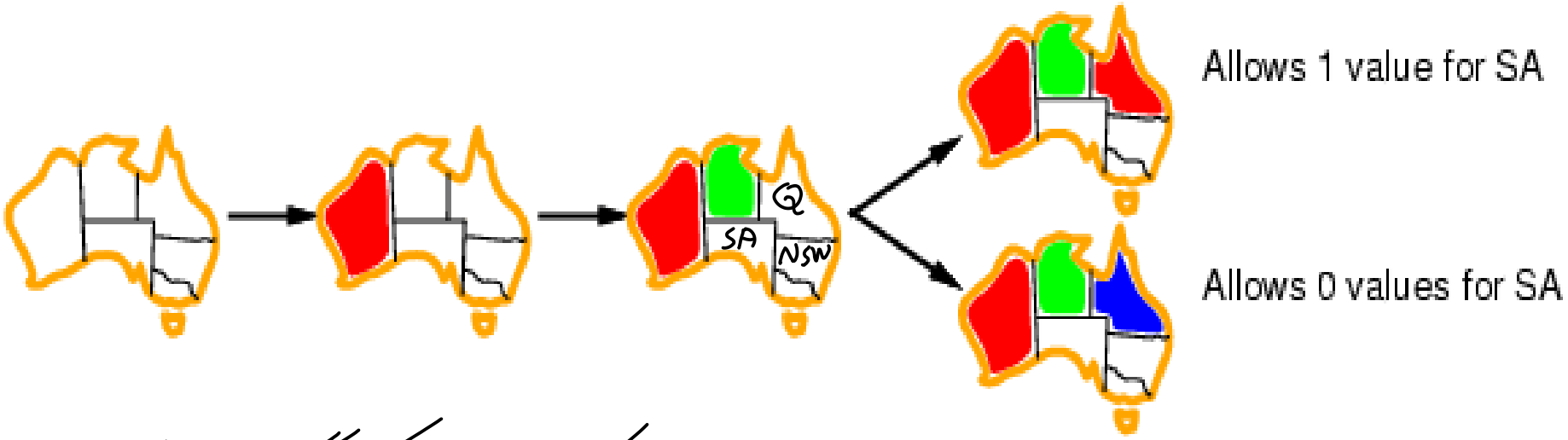






- Least constraining value:

مثال Least : (انتخاب مقداری که کمترین محدودیت را ایجاد کند.)



متغیر Q انتخاب شده. هنگام مقداردهی اگر با Q=Red شروع کنیم برای SA یک رنگ باقی می‌ماند

اگر با Q=blue شروع کنیم برای SA ، صفر رنگ باقی می‌ماند.

پس Q=Red انتخاب می‌شود.

Forward checking

بررسی پیش‌رو:

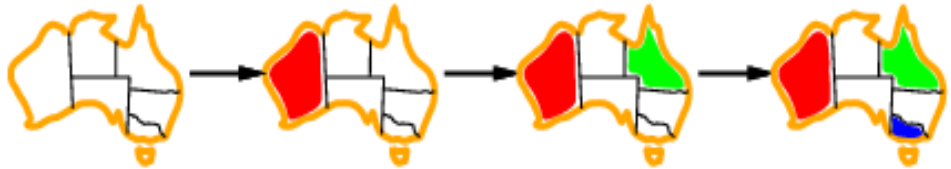
دس از مقداردهی به گره X، با توجه محدودیت‌های باینری بین X و سایر گره‌ها دامنه‌ی گره‌های همسایه‌ی X را محدود می‌کنیم. هرگاه دامنه‌ی یک گره تهی شد، ناسازگاری تشخیص داده می‌شود.

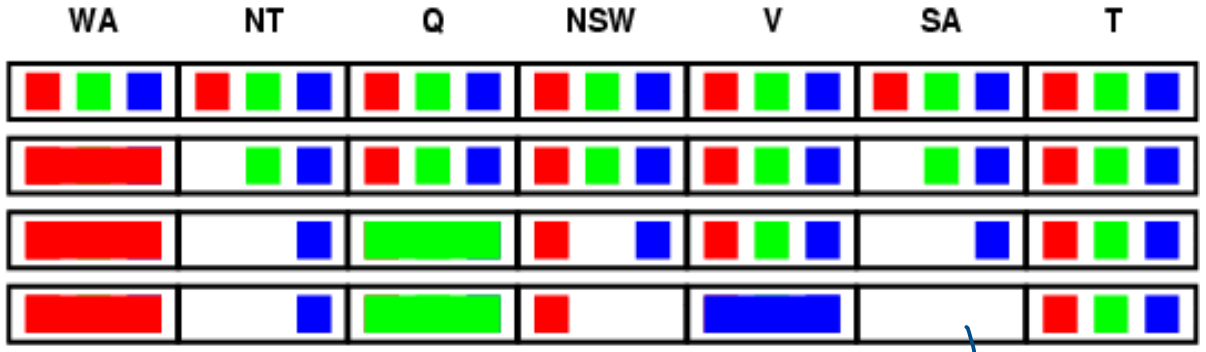
این باعث می‌شود که ناسازگاری را سریع‌تر از B.T تشخیص دهیم.

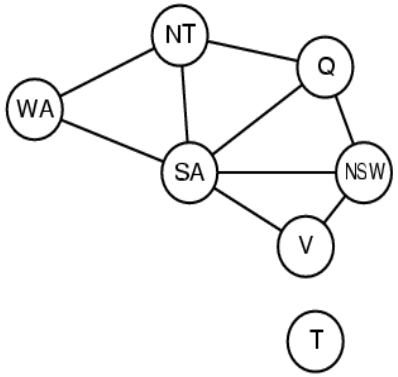
Forward checking











WA = Red

Q = green

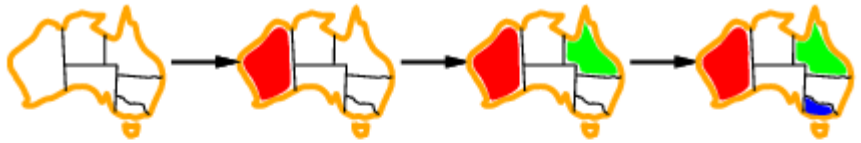
V = blue

بررسی پیش رو در همین مرحله متوجه ناسازگاری در SA می‌شود و عقب‌گرد می‌کند.

دامنه SA تهی شد.

: forward checking

می‌تواند محدودیت را از متغیرهای مقداردهی نشده به مقداردهی نشده اعمال کند.
این باعث می‌شود که زودتر از عقبگرد ساده (قبل از رسیدن به گره SA) ناسازگاری را تشخیص دهد.
اما با این حال نمی‌تواند وجود ناسازگاری بین یک جفت گره مقداردهی نشده را مشاهده کند.

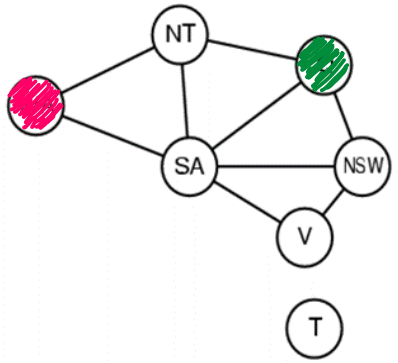




در همین جا ناسازگاری رخ داده اما forward checking تشخیص نداده.

از نظر f.c هنوز هیچ دامنه‌ای تهی نشده.

اما در واقع همین جا معلوم است که برای NT و SA هیچ مقداردهی سازگاری باقی نمانده.



Node consistency

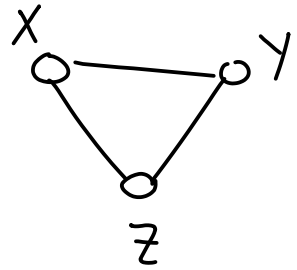
سازگاری گره : برای هر گره X دستکم یک مقدار در دامنه‌ی X باشد که محدودیت یکانی آن را برقرار کند.

1 ـ سازگاری

مثال : گراف محدودیت

سازگاری گره دارد.

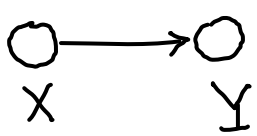
دامنه هیچ گرهی تهی نیست.



دامنه $D = \{blue\}$

Arc consistency

سازگاری کمان: ۲-سازگاری.

کمان 

، را سازگار گوییم اگر:

برای هر مقدار $x \in Dom(X)$ دستکم یک مقدار $y \in Dom(Y)$ موجود باشد که

انتساب $X=x, Y=y$ سازگار باشد.

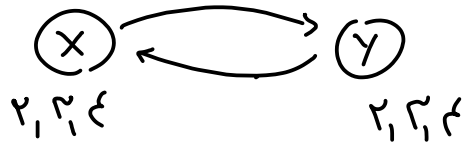
اگر همه کمان‌ها سازگار باشند، می‌گوییم آن CSP، یال‌سازگار است.

مثال

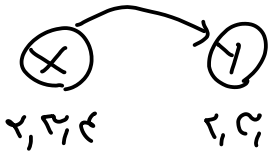
$X+Y \leq 5$

دامنه‌ها $= \{1, 2, 3, 4\}$

یال $x \to y$ در واقع دو کمان است:



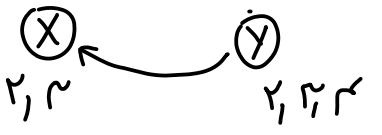
بررسی سازگاری



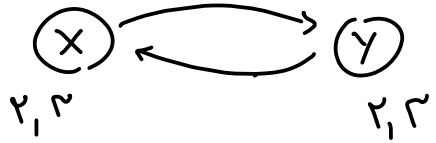
برای $x=2$ و $x=3$ می‌توان یک y سازگار یافت اما برای $x=4$ هیچ

مقداری سازگار برای y نداریم پس باید ۴ از دامنه x حذف شود.

حالا برای



باید $y=4$ از دامنه y حذف شود:

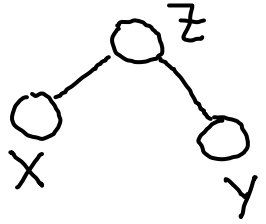


Path consistency

سازگاری مسیر:

۴- سازگاری:

برای هر ۳ تایی X, Y, Z (به ویژه آنهایی که مسیر



را می‌سازند)

در هر مقداردهی سازگار $(X=x,\ Y=y)$ یک مقدار $z \in Dom(Z)$ موجود باشد که

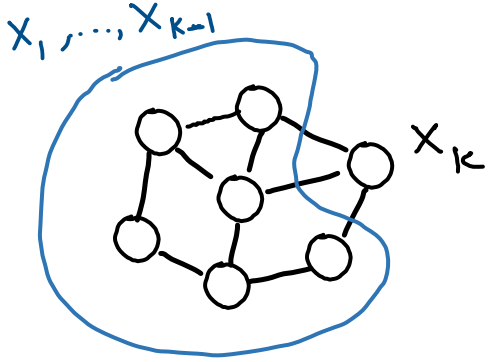
سازگار باشد.

K-consistency

k ـ سازگاری

به ازای هر انتخاب سازگار به متغیرهای $X_1, X_2, \ldots, X_{k-1}$ در کنار یک

مقدار سازگار برای X_k در دامنه‌ی آن وجود داشته باشد.



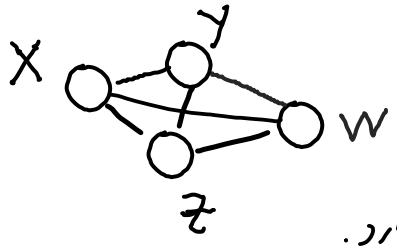
Kـ سازگاری یک گزاره شرطی است:

(انتساب سازگاری برای $X_1, \ldots, X_{k-1}$ موجود باشد) if $\Longrightarrow$ then (مقدار سازگاری برای X_k بتوان یافت.)

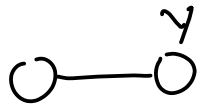
گاهی اوقات مقدمه اصلاً نمی‌تواند رخ دهد به همین دلیل گزاره شرطی معتبر می‌شود.

مثال: این گراف محدودیت در مسأله رنگ آمیزی؛ [شکل صحیح به این صورت است، در کلاس یال X–W رسم نشده بود]

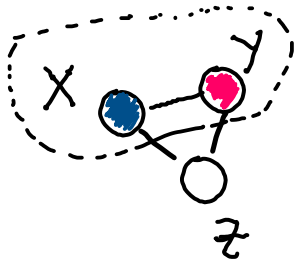
$Dom = \{ Red , Blue \}$



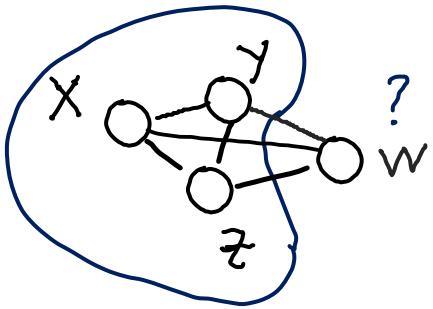
1 سازگاری : بله:

2 سازگاری : بله: مثلاً  به X هر مقداری بدهیم Y مقداری سازگار با آن دارد.

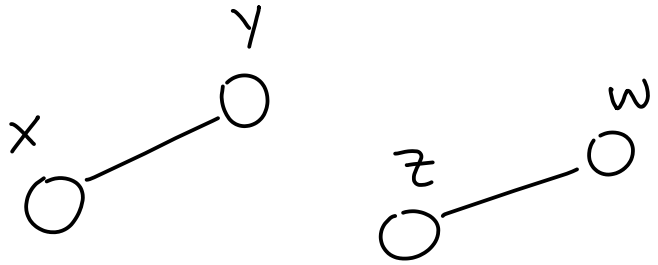
3 سازگاری : خیر: مثلاً در این ۳ تایی برای Z مقدار سازگار نداریم:



4 سازگاری بله: (به دلیل انتخابِ مقدم) اصلاً نمی‌توان به X,Y,Z مقدار سازگار داد.



مثال



$Dom = \{ Red , Blue \}$

حالا این مساله ۳ سازگار است.

مثال قبل: ۴ سازگار بود اما ۳ سازگار نبود.

پس: k سازگاری $\not\Longleftarrow$ k-1 سازگاری

اما: اگر $k, \ldots, 4, 3, 2, 1$ سازگاریا بشه می‌گوییم k سازگار قوی است.

k سازگار قوی باشه $\Longleftrightarrow$ ۱ و ۲ و ۳ و ... و k-1 و k سازگار است.

* در ادامه خواهیم دید که اگر مسأله ۲-سازگار باشد و (فاصله دور) آنگاه می‌توان انتخاب‌ها را در زمان $O(nd)$ انجام داد و جواب را یافت. توضیح کامل در بخش‌های بعدی

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

بخش دوم جلسه ۸

```
function AC-3( csp) returns false if an inconsistency is found and true otherwise
  inputs: csp, a binary CSP with components (X, D, C)
  local variables: queue, a queue of arcs, initially all the arcs in csp

  while queue is not empty do
    (X_i, X_j) ← REMOVE-FIRST(queue)
    if REVISE(csp, X_i, X_j) then
      if size of D_i = 0 then return false
      for each X_k in X_i.NEIGHBORS - {X_j} do
        add (X_k, X_i) to queue
  return true
```

```
function REVISE( csp, X_i, X_j) returns true iff we revise the domain of X_i
  revised ← false
  for each x in D_i do
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j then
      delete x from D_i
      revised ← true
  return revised
```

AC3:

هدف: تبدیل مساله به یک مساله‌ی
۲- سازگار

AC 3

هدف: سازگار کردن همه کمان‌ها. (ایجاد ۲-سازگاری)

۱) همه کمان‌ها به ترتیب دلخواه در صف قرار می‌گیرند.

۲) یال (X, Y) را از صف پرمی‌داریم. برای هر $x \in Dom(X)$ باید حداقل یک $y \in Dom(Y)$ باشد که با آن سازگار شود. در غیر این صورت x از دامنه X حذف می‌شود.

۳) اگر در مرحله (۲) دامنه‌ی X دچار تغییر شد، تمام یال‌هایی که به X می‌آیند باید دوباره در صف درج شوند.

۴) هرگاه صف خالی شود، اجرای AC3 به پایان می‌رسد.

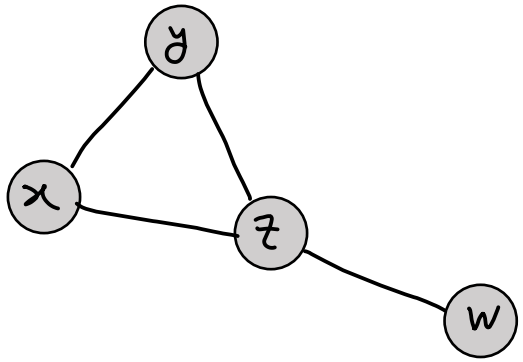
۵) اگر دامنه‌ی برخی از متغیرها تهی شود، یعنی مسأله جواب ندارد.

۶) هوپال، حداکثر یک بار و حداکثر d بار در صف قرار می‌گیرد و بررسی می‌شود.
سازگار کردن هوپال، حداکثر از مرتبه‌ی d^2 است.

مثال:

محدودیت‌ها:
$X < Z$
$X < Y$
$Y < Z$
$Z < W$

$Dom = \{1, 2, 3, 4, 5\}$



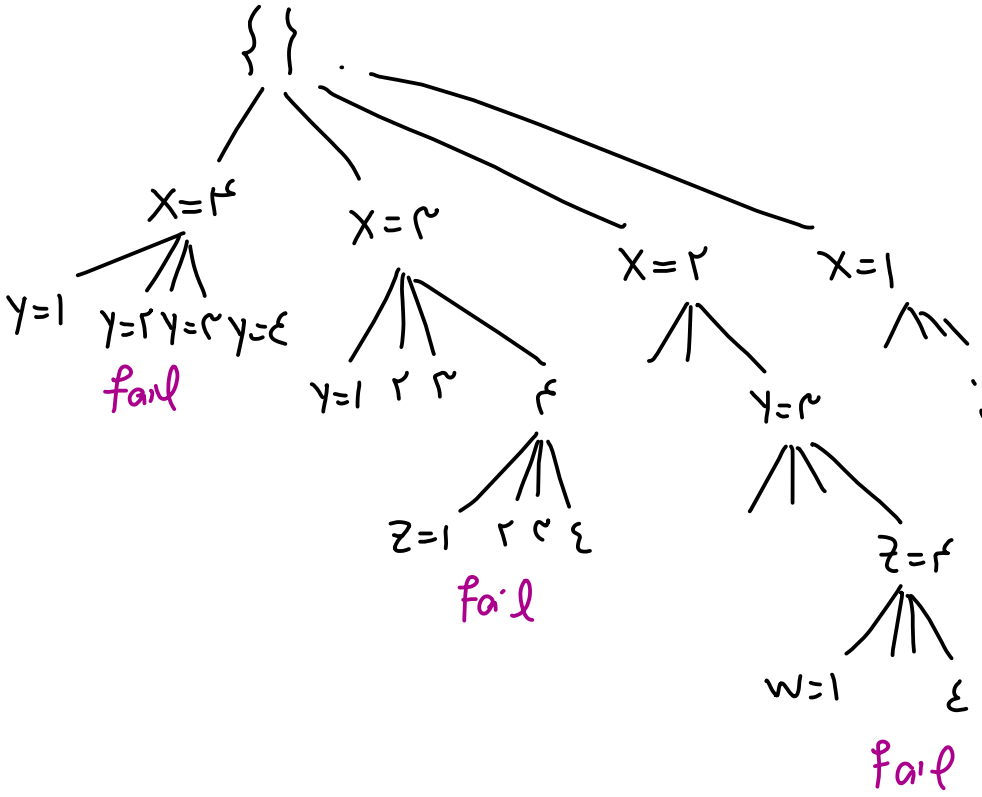
ابتدا بدون استفاده از AC3 اگر بخواهیم B.T اجرا کنیم

و مثلاً عامل از $X=۴$ شروع کند، بسیار طول می‌کشد تا جواب

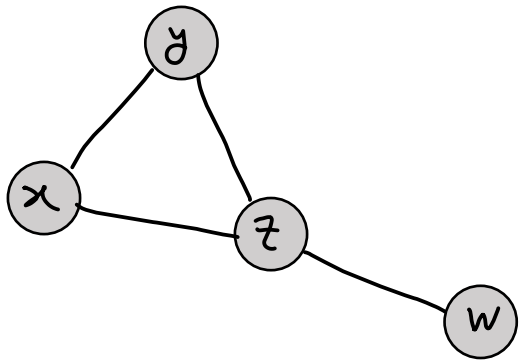
یافت شود. در شکل نشان داده‌ام چقدر طول می‌کشد تا

عامل متوجه شود $X=۴$ و $X=۳$ و $X=۲$

بن بست است.



مثال: اجرای AC3 :
پیش از B.T



محدودیت‌ها:
$X < Z$
$X < Y$
$Y < Z$
$Z < W$

$Dom = \{1, 2, 3, 4, 5\}$

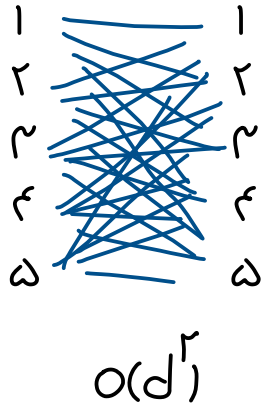
همه یال دوکمان است. به ترتیبی در صف Q درج می‌شوند مثلاً فرض کنیم:

$Q: \langle (X,Y)\ (Y,X)\ (X,Z)\ (Z,X)\ \ldots\ (Z,W)\ (W,Z)\ (Y,Z) \rangle$

ابتدای صف

مرحله اول

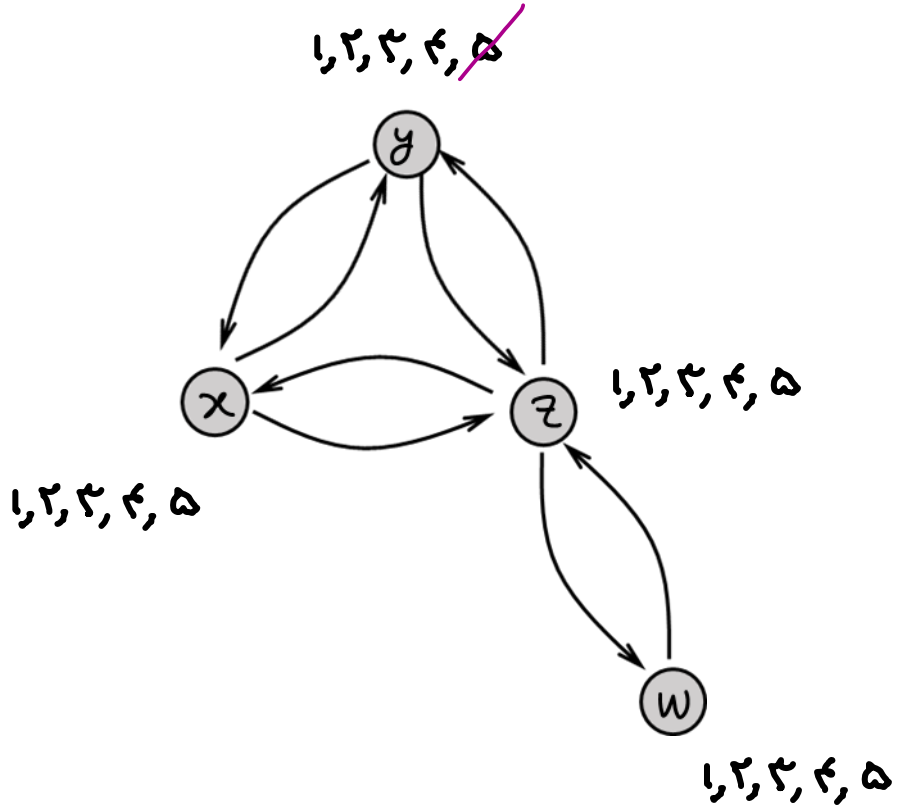
$z \to y$ از صف خارج می‌شود؛



سپس بازگذاری کمان $z \to y$. شرط: $(z > y)$

مقادیر دامنه y با مقادیر دامنه z بررسی می‌شوند.

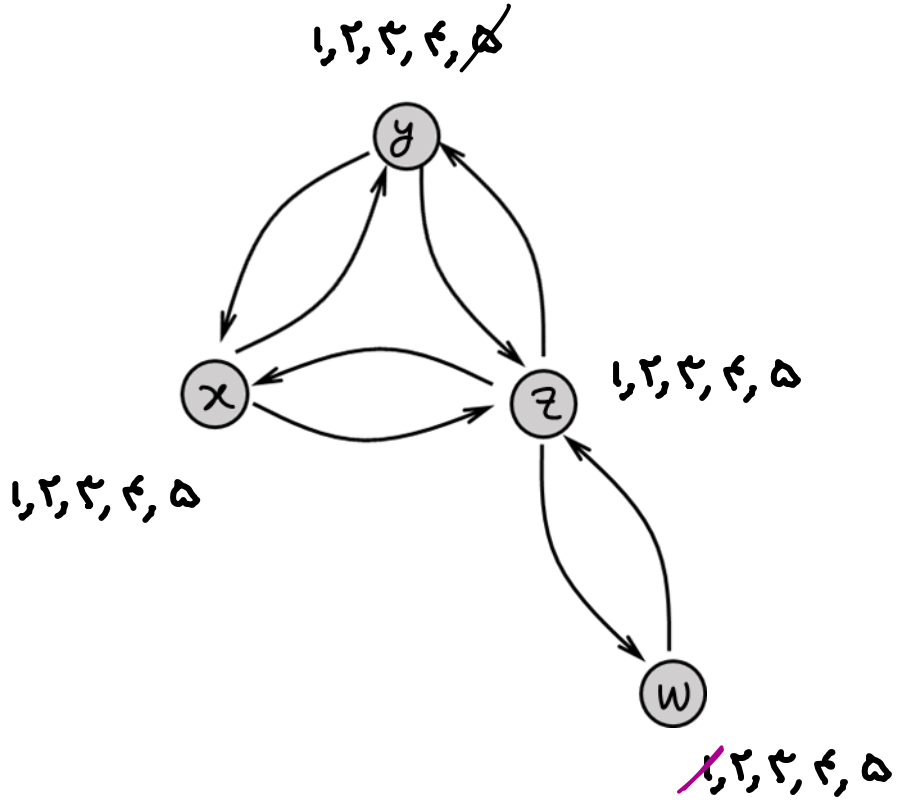
برای $y = 5$ هیچ مقدار بازگذاری در دامنه‌ی z نداریم پس ۵ از دامنه‌ی y حذف می‌شود. حال چون دامنه y تغییر یافته کمان‌های $y \to x$, $y \to z$ در انتهای صف درج می‌شوند.



Q:

$\langle (z,y) \quad (x,y) \quad (x,y) \quad (y,x) \quad (x,z) \quad (z,x) \quad \ldots \quad (z,w) \quad (w,z) \quad \square \rangle$

مرحله دوم



کمان $z \to w$ از صف خارج می‌شود.

بررسی کردن کمان $w \to z$. شرط: $(z < w)$

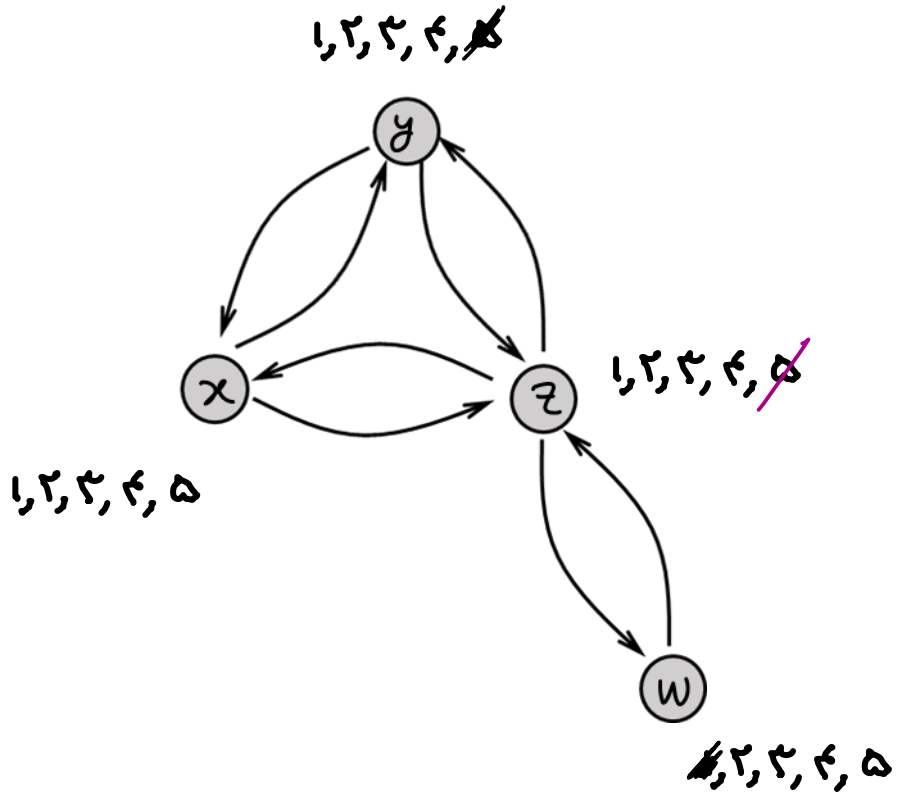
برای $w = 1$ هیچ مقداری برای z نداریم.

1 از دامنه w حذف می‌شود. حالا چون دامنه w تغییر کرد کمان $w \to z$ در صف درج می‌شود.

Q

$\langle (z,w)\ (z,y)\ (x,y)\ (x,y)\ (y,x)\ (x,z)\ (z,x)\ \ldots\ (z,w)\ \square\ \square \rangle$

مرحله سوم



کمان $w \to z$ از صف خارج می‌شود.

سبزکاری کردن کمان $w \to z$ شد: $(w > z)$

برای $z=5$ هیچ مقدار سبزکاری برای w نداریم.

۵ از دامنه z حذف می‌شود. حالا چون دامنه z تغییر کرد کمان‌های $x \to z$، $y \to z$، $w \to z$ در انتهای صف درج می‌شوند.

در نهایت

به همین ترتیب ادامه می‌دهیم تا آن‌جا که صف Q تهی شود.

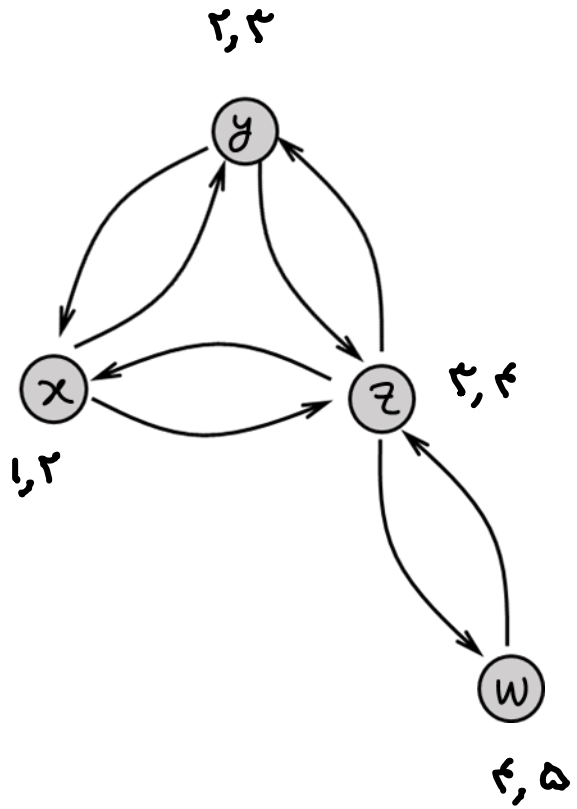
در پایان مساله CSP به یک مساله‌ی ۲-سازگار تبدیل می‌شود:

$x < y$

$z > x$

$z < y$

$w > z$



تخمین زمان اجرای AC3:

تعداد متغیرها n است پس حداکثر $\binom{n}{2} = O(n^2)$ یال داریم. هر یال دو کمان دارد پس تعداد کمان‌ها $O(n^2)$ است. سازگار کردن هر کمان از مرتبه $O(d^2)$ است. اما هر کمان ممکن است حداکثر d بار در صف درج شود پس:

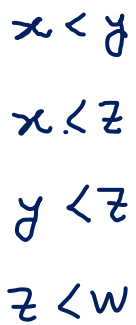
$$AC3: \quad O(n^2 \times d^2 \times d) = O(n^2 d^3)$$

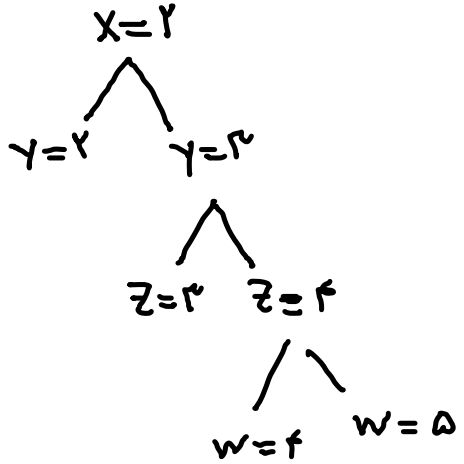
پس از اصلاح دامنه‌ها:

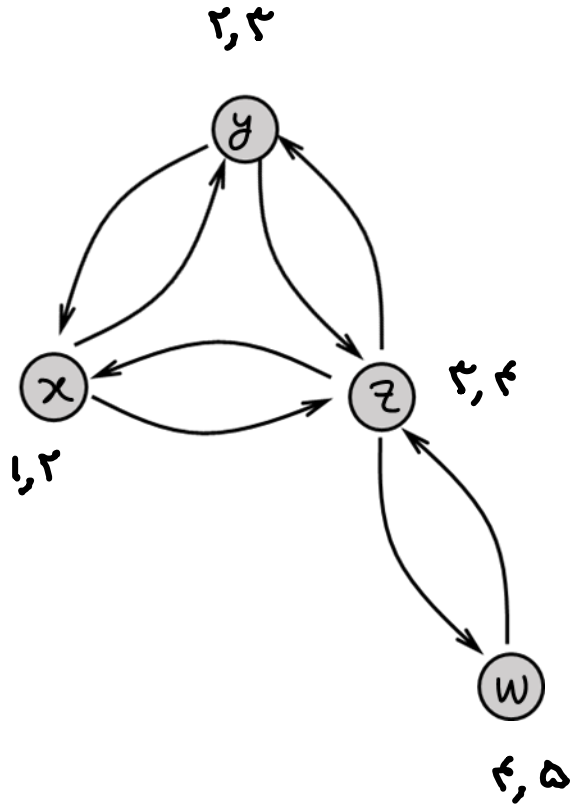
در این مثال پس از اجرای AC3 می‌توانیم

در $O(nd)$ جواب را با BT پیدا کنیم.

آیا در حالت کلی هم چنین است؟







خلاصه

AC3: $O(n(n-1) \times d \times d^2) = O(n^2 d^3)$

پس از AC3 : [درجه بعدی توضیح کامل برداده می‌شود]

یا فقط جواب پس از AC3

[اگر محدودیت ها باینری باشند و گراف فاقد دور باشد] $= O(nd)$

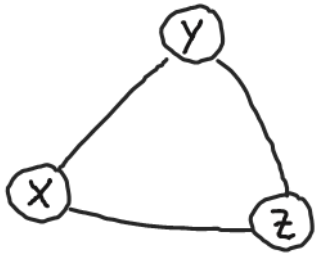
(درجه بعدی لحاظ می‌شود)

⊛ پس از اجرای AC3:

اگر دامنه یکی از متغیرها تهی شد ⟵ مساله جواب ندارد.

اگر دامنه هیچکدام تهی نشد ⟵ ممکن است مساله جواب داشته یا نداشته باشد
(البته اگر همه محدودیت ها فقط باینری باشند
و دور نداشته باشیم مساله حتماً جواب دارد.)

مثال

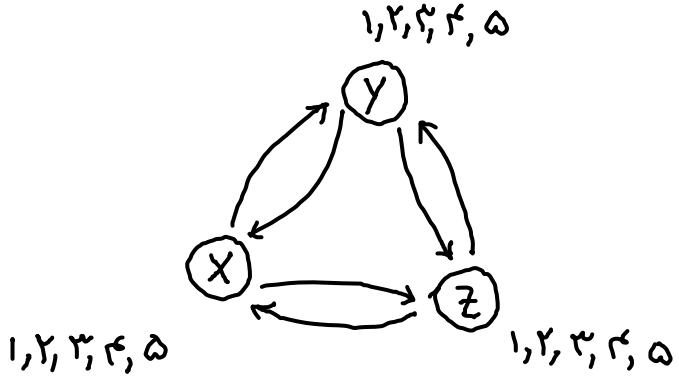


محدودیت‌ها:
$X<Y$
$Y<Z$
$Z<X$

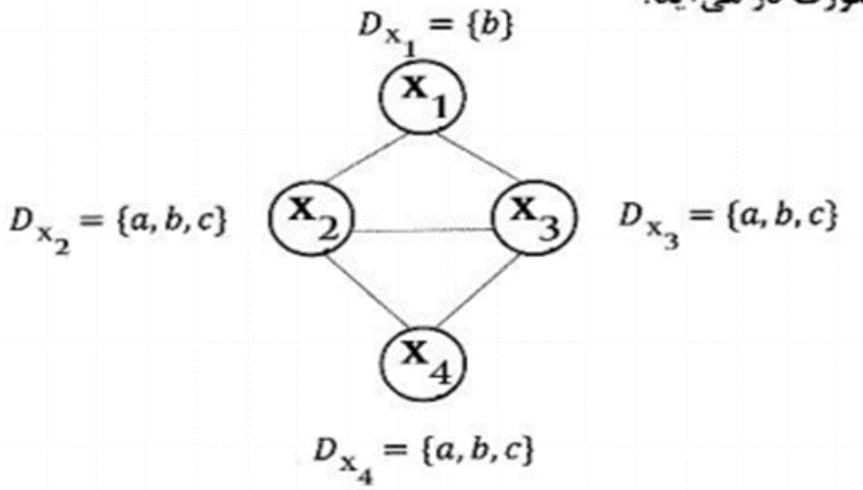
$Dom = \{1, 2, 3, 4, 5\}$

پس از اجرای AC3 دامنه‌ی از متغیرها کم می‌شود. (ما به جواب نداریم)

(بررسی کنید)



۸۷- در مسئله CSP با چهار متغیر X_1، X_2، X_3، X_4 و گراف محدودیت (که هر لینک آن محدودیت یکسان نبودن مقدار متغیرهای دو سر آن لینک را نشان می‌دهد) مشخص شده است. دامنهٔ متغیرها هم کنار آن‌ها نشان داده شده است. بعد از اعمال AC۳ دامنهٔ متغیرها به چه صورت در می‌آید؟



۱) $D_{X_1} = \{\}$
$D_{X_2} = \{a, b, c\}$
$D_{X_3} = \{a, b, c\}$
$D_{X_4} = \{a, b, c\}$

۲) $D_{X_1} = \{b\}$
$D_{X_2} = \{a, c\}$
$D_{X_3} = \{a, c\}$
$D_{X_4} = \{b\}$

۳) $D_{X_1} = \{b\}$
$D_{X_2} = \{a, c\}$
$D_{X_3} = \{a, c\}$
$D_{X_4} = \{a, b, c\}$

۴) خروجی بستگی به‌ترتیب گذاشتن آرک‌ها در صف دارد.

درمورد گزینه ۴:

توجه: اگر با اجرای AC3 دامنه‌ها تهی نشود

یعنی مساله نمی‌تواند ۲-سازگار شود،

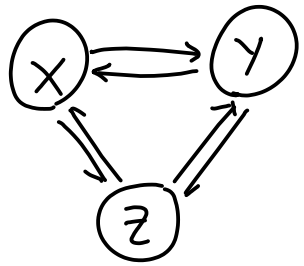
ترتیب قرارگرفتن کمان‌ها در صف مهم نیست. [مانند تست قبل]

اما اگر قرار باشد دامنه‌ی متغیرها تهی شود

این که کدام تهی می‌شود بستگی به ترتیب

بررسی کمان‌ها دارد. مثال: در مساله رنگ‌آمیزی با دامنه‌ی یک رنگی:

$D=\{1\}$



اگر اول کمان $Y \to X$ بررسی شود: $DomX = \phi$

اگر اول کمان $X \to Y$ بررسی شود: $DomY = \phi$.

جواب تست :

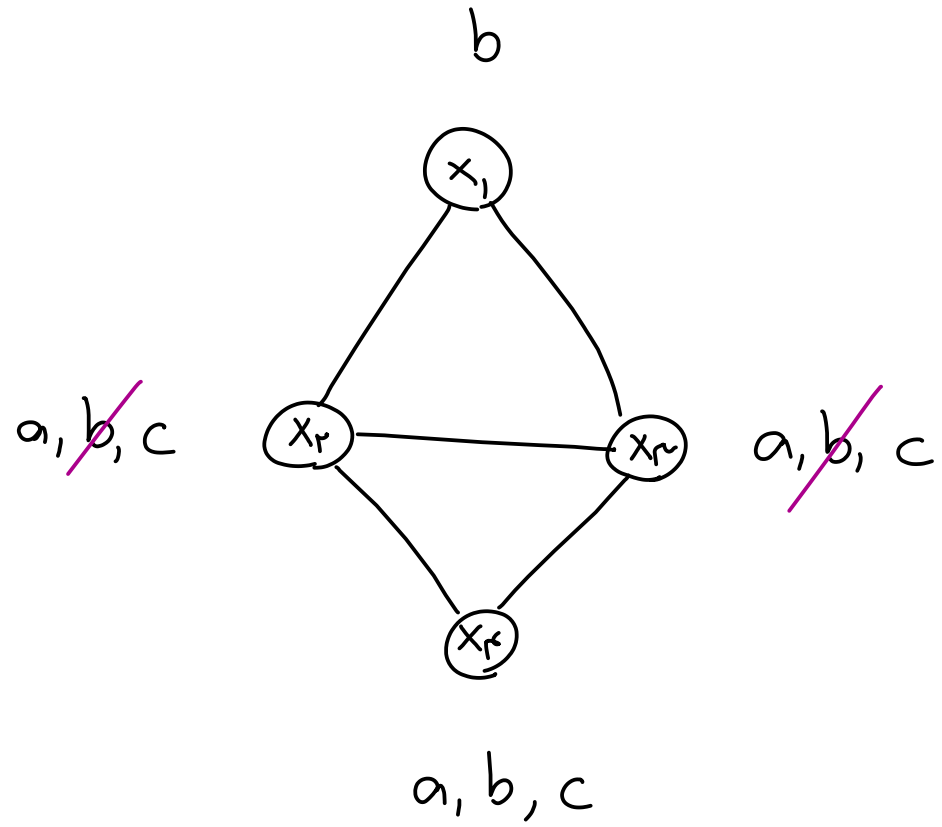
چون دامنه x_1 کوچک است همه را انتخاب می‌کنیم.

حال هر کدام که به x_1 وارد می‌شوند را سازگار کنیم.

b از دامنه x_3 و x_2 حذف می‌شود.

سایر یال‌ها هم سازگارند.

جواب گزینه ۳



گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

بخش سوم جلسه ۸

n = تعداد متغیرها

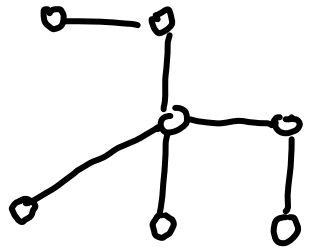
d = تعداد دامنه

معمولی	BT	$O(d^n)$
حالت کلی	AC3	$O(n^2 d^2 d)$
فاقد دور (درختی)	AC3	$O(nd^2)$

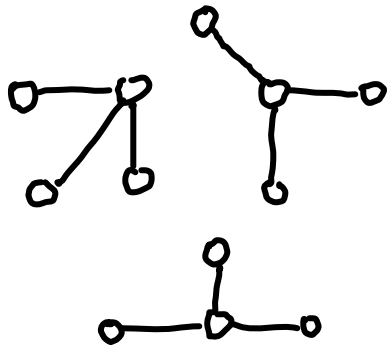
نتیجه: اگر گراف فاقد دور باشد:

AC3 + BT	$O(n^2 d^2 + nd) = O(n^2 d^2)$

گراف فاقد دور ← تعداد یال‌ها: $e < n$



$e = n-1$



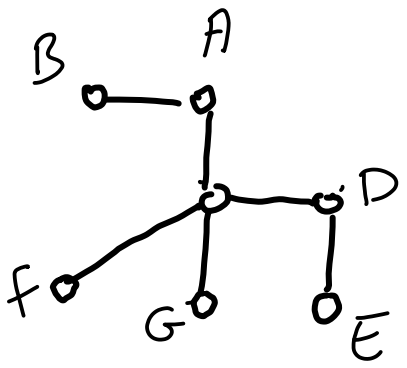
$e = n-2$

در جنگل یا درخت یا تعداد یال‌ها کمتر از n است پس کمان‌ها به جای $O(n^2)$ حداکثر $O(n)$ هستند. و در ضمن با ترکیب توپولوژیکی می‌توان کاری کرد که کمان دقیقاً یک بار در صف AC3 درج شود.

$O(nd^2)$

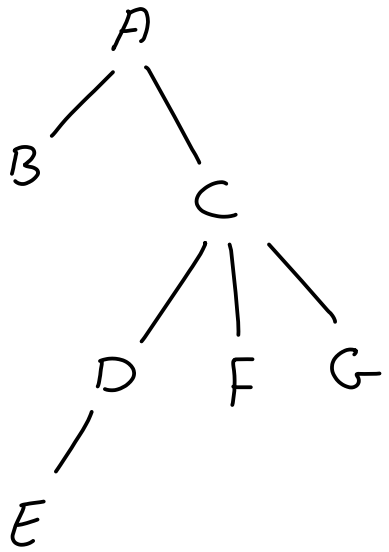
AC3

مثال ایجاد ترتیب توپولوژیک با DFS(A).



$e = n - 1$

DFS:

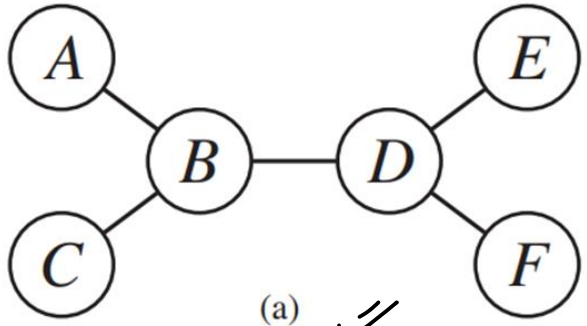


ایجاد لیست پیوندی با ترتیب لحظه‌ی خروج از گره‌ها:

مثلاً B اولین گره‌ای است که از آن خارج می‌شویم. سپس E، D، F، ----

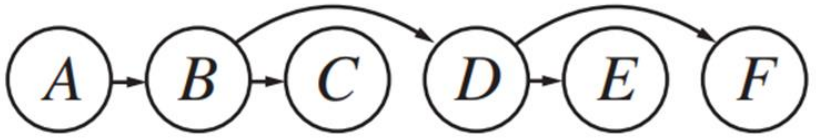


مثال:



(a)

گراف محدودیت فاقد دور



(b)

اجرای DFS مثلاً از A

دقت کنید که یال‌ها جهت دارشده‌اند

ترتیب توپولوژیک:



ترتیب مناسب قرارداد یال‌ها در ILP:

Q: (A,B) (B,C) (B,D) (D,E) (D,F)

حالا هر یال دقیقاً یک بار بررسی می‌شود.

$O(d^n)$: BT

$O(n^2 d^4)$: کلی

$O(n d^2)$: فاقد دور

: AC3

$O(nd)$: در گراف فاقد دور : AC3 یافتن جواب پس از

پس از اجرای AC3 روی گراف محدودیت فاقد دور:

- دامنه همگی داشته باشیم: مسئله جواب ندارد.
- دامنه همگی نداشته باشیم: می‌توان در $O(nd)$ جواب را یافت.

تبدیل گراف محدودیت به گراف فاقد دور

- استفاده از cut set
- استفاده از clique و max clique

در حالت کلی هر دو روش NP-Hard هستند اما هر کدام در حالات خاص می توانند زمان را به چند جمله‌ای کاهش دهند.

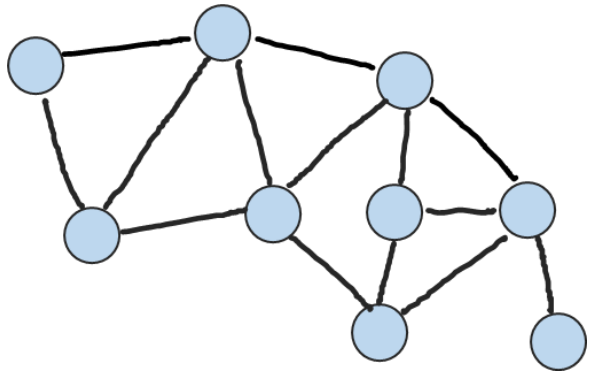
تبدیل گراف محدودیتی به حالت درختی:

۱) یک یا چند گره را حذف می کنیم

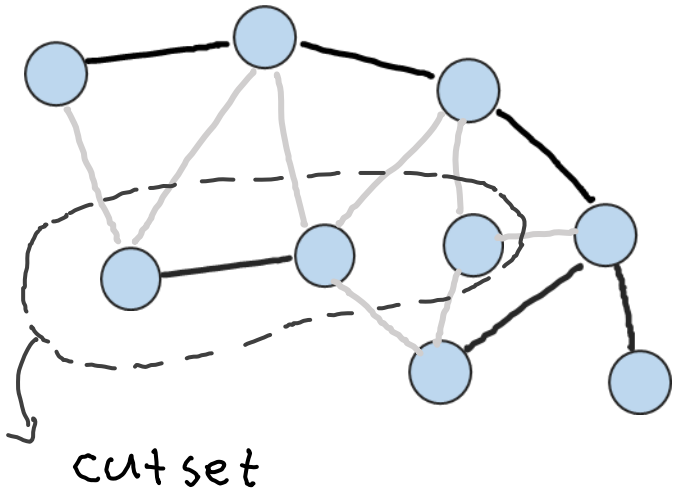
به این مجموعه از گره‌ها
cut-set

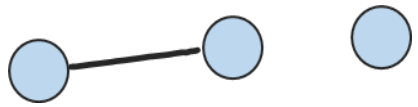
فرض کنید تعداد آنها c باشد.

[اگر بتوانیم مجموعه برشی از مرتبه
O(c) بیابیم این روش موفق است]

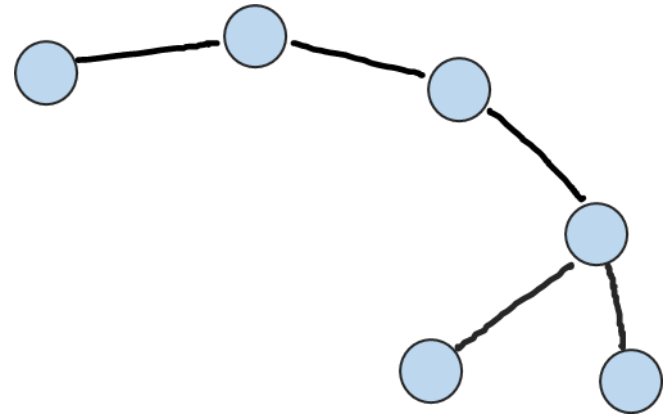


انتخاب مجموعه برشی





گراف فاقدور با $n-c$ گره



c گره دارای cut set

این مجموعه دارای d^c حالت انتخاب است.

گره های cut set حداکثر دارای d^c مقدار دهی هستند. برای هرکدام از آنها که سازگار است، جواب را برای سایر گره ها (گراف فاقدور با $n-c$ گره) بررسی می‌کنیم:

$$O\left(d^c \cdot (n-c)d^2\right)$$

d^c: انتخاب cutset

$(n-c)d^2$: AC3 گراف فاقدور

آیا زمان اجرا کاهش یافت؟

BT عادی: $O(d^n)$

استفاده از cutset: $O((n-c)d^{c+2})$ ← نتیجه: به شرط آن که بتوان cut set از مرتبه‌ی ثابت $O(c)$ پیدا کرد این روش زمان اجرا را به چنین جمله‌ای کاهش می‌دهد اما اگر مجموعه برشی مثلاً $n-3$ عضوی باشد باز هم زمان نمایی است.

(*) یافتن cutset کمینه یک مسأله‌ی NP-Hard است.

۸۷- در یک مسئله ارضای قیود دودویی، با دامنه محدود برای مقادیر مجاز متغیرها، می‌خواهیم صرفاً با سازگارکردن قیود، مسئله را در زمان چندجمله‌ای حل کنیم. کدام‌یک از موارد زیر درست است؟

۱) در همه گراف‌های قیود این کار امکان‌پذیر است.

۲) این کار تنها زمانی‌که گراف قیود کامل باشد، امکان‌پذیر است.

۳) تنها درصورتی‌که گراف قیود، درخت باشد، این کار امکان‌پذیر است.

۴) درصورتی‌که گراف قیود، تنها یک دور داشته‌باشد، این کار امکان‌پذیر است.

آی‌تی ۱۴۰۲

نه

$c=1$

جواب گزینه ۴

در حالت کلی بدون سازگارکردن BT $O(d^n)$

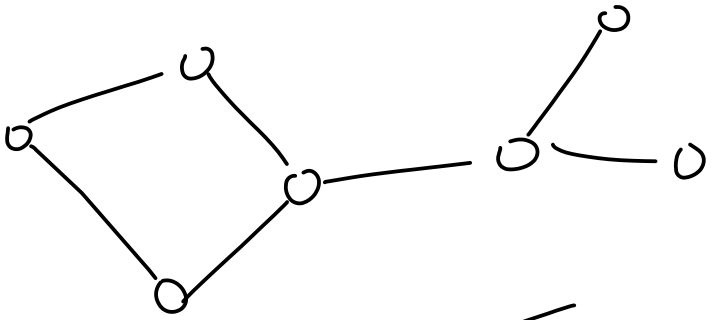
در حالت کلی با سازگارکردن (AC3) سپس یافتن جواب:

AC3: $O(n^2d^3) + O(d^n)$ ← احتمال وجود دور

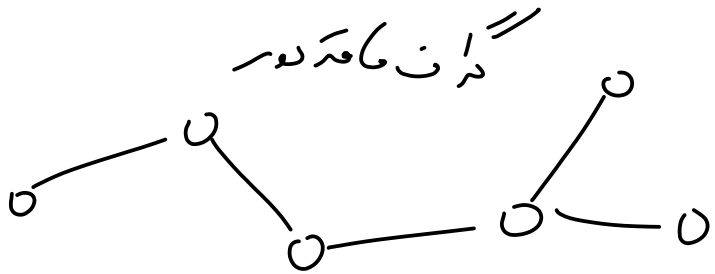
اگر فاقد دور باشد و AC3 را قبل از BT اجرا کنیم: $O(n^2d + nd)$

اگر فقط با حذف c گره بشود به درخت تبدیل کرد: $O((n-c)d^{c+2} + nd)$

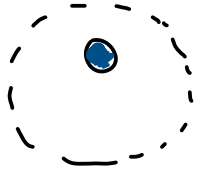
گزینه ۴



فقط یک دور



گراف حلقه دور



c=1

cutset

حالا AC3 سپس یافتن جواب با BT :

$$O((n-1)d^{l+1}) + O(nd)$$

روش دوم تبدیل گراف محدودیت به درخت:

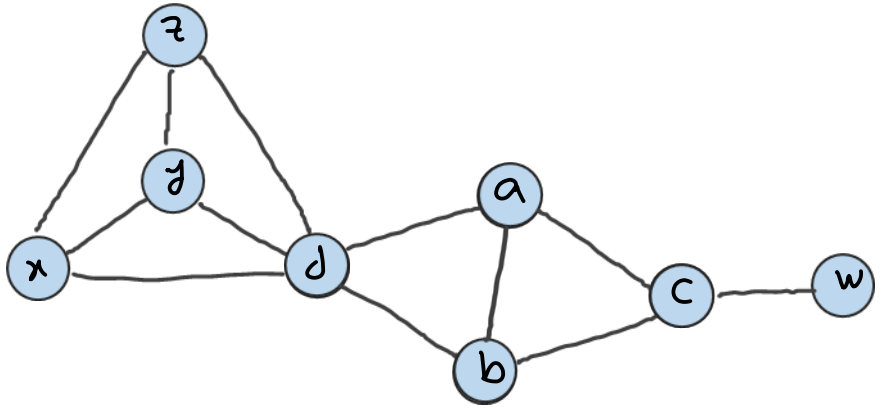
زیرگراف های کامل (خوشه) به ویژه خوشه‌های ماکزیمال (max clique) را در نظر بگیرید.

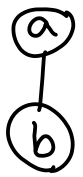
هرکدام از آن‌ها می‌توانند یک ابرگره super node در گراف جدید باشند.

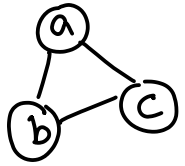
آن‌هایی که گره یا یال مشترک دارند، در گراف جدید باهم مجاورند.

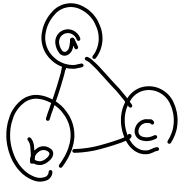
حالا بزرگترین درخت پوشا را به دست می‌آوریم.

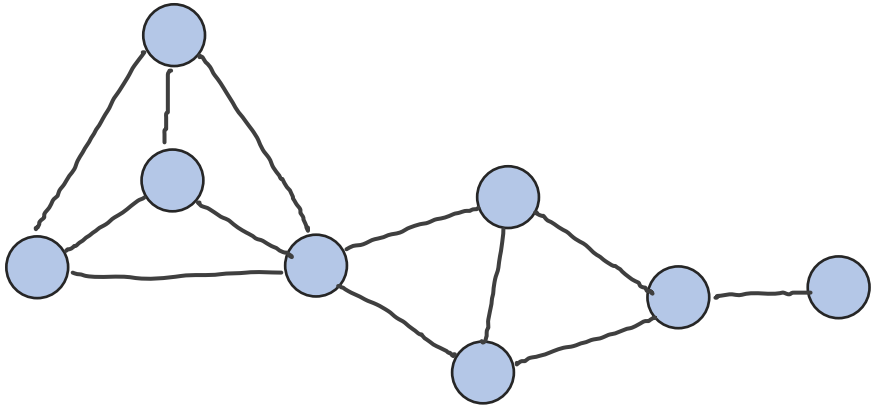
مثال استفاده از max clique ها:



مثلاً  یک خوشه است (زیرگراف کامل K_2)

K_3:  اما ماکسیمال نیست چون درون خوشه‌ی

 قرار دارد. اما ماکسیمال است.

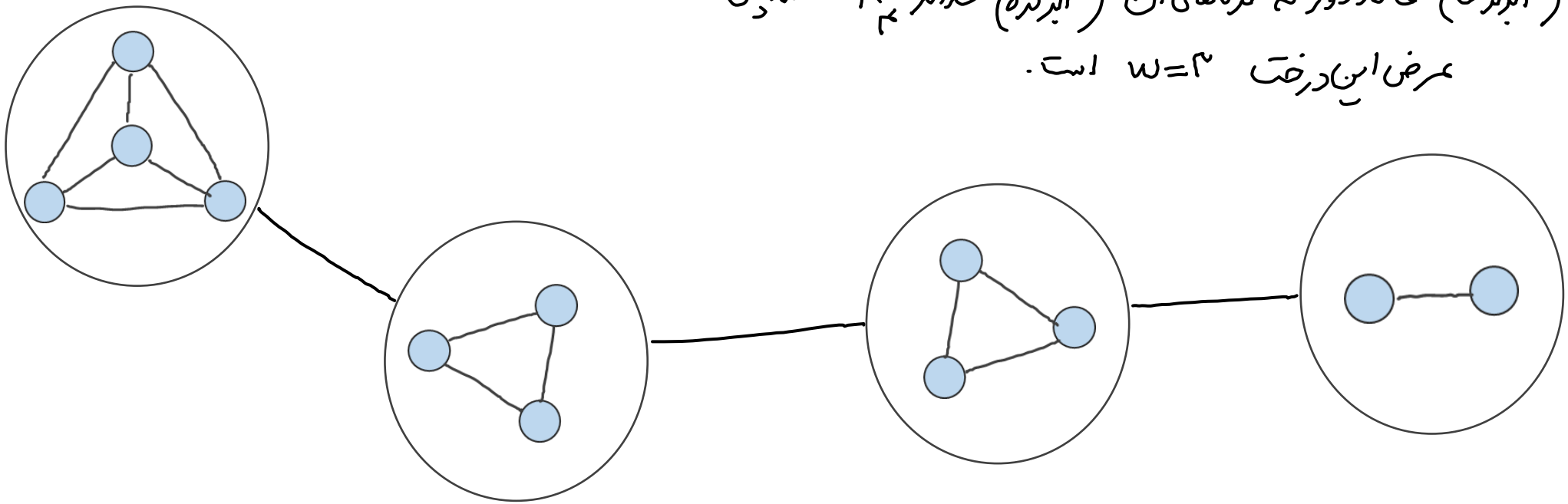


در این شکل اگر هر خوشه ماکسیمال را یک ابرگره بنامیم

درخت خواهیم داشت.

به یک گراف (ابرگراف) فاقد دور که گره‌های آن (ابرگره) حداکثر K_4 هستند

عرض این درخت $w=3$ است.



تکمیل این روش و

ادامه در جلسه ی ۹

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۹ فصل ۶

مرور و توضیح دوباره این مسائل:

آی‌تی ۱۴۰۲

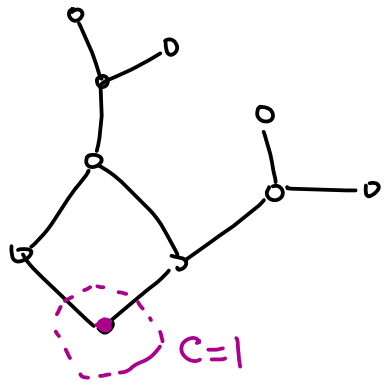
۸۷- در یک مسئله ارضای قیود دودویی، با دامنه محدود برای مقادیر مجاز متغیرها، می‌خواهیم صرفاً با سازگارکردن قیود، مسئله را در زمان چندجمله‌ای حل کنیم. کدام‌یک از موارد زیر درست است؟

غلط ۱) در همه گراف‌های قیود این کار امکان‌پذیر است.

غلط ۲) این کار تنها زمانی‌که گراف قیود کامل باشد، امکان‌پذیر است.

غلط ۳) تنها درصورتی‌که گراف قیود، درخت باشد، این کار امکان‌پذیر است.

True ۴) درصورتی‌که گراف قیود، تنها یک دور داشته‌باشد، این کار امکان‌پذیر است.



فقط یک دور ⇒ c=1

در حالت کلی: $\underbrace{n^2 d_x d^2}_{AC3} + \underbrace{O(d^n)}_{BT}$

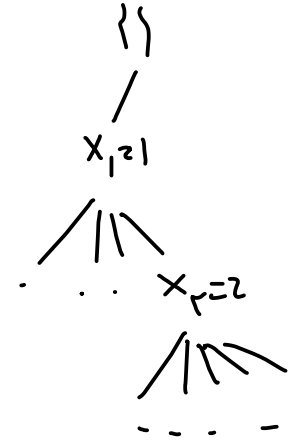
اگر گراف فاقد دور و csp باینری باشد:

$\underbrace{nd^2}_{AC3} + \underbrace{nd}_{\text{مقداردهی}} =$ چندجمله‌ای

اگر هم دور داشته باشد اما فقط تعداد متناهی

گره برشی بتوانه دورها را حذف کنه بازهم:

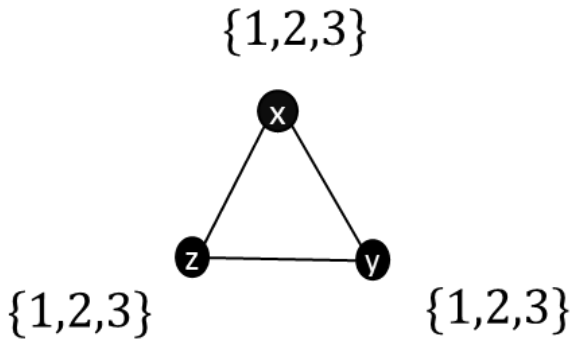
$(n-c)d^{c+2} + nd =$ چندجمله‌ای



اهمیت فاقه دور بودن را در مثال زیر نشان داده‌ایم:

همه محدودیت‌ها باینری هستند.

Binary CSP

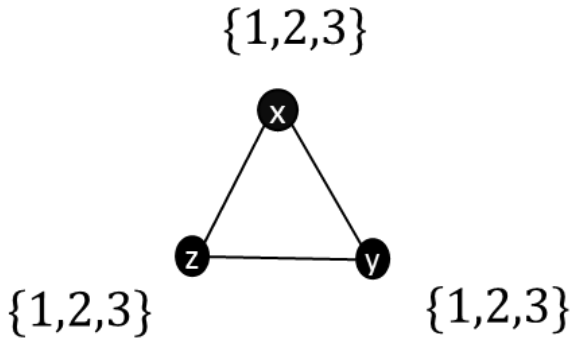


$$x + y = 3$$
$$z + y \; be \; odd$$
$$x + z \; be \; odd$$

در حال حاضر سازگاری کمان ندارد. مثلاً برای $x=3$ مقداری برای y نداریم. پس هم‌اکنون: $BT: O(d^n)$

حال AC3 را اجرا کنیم تا کمان سازگار شود:

Binary CSP



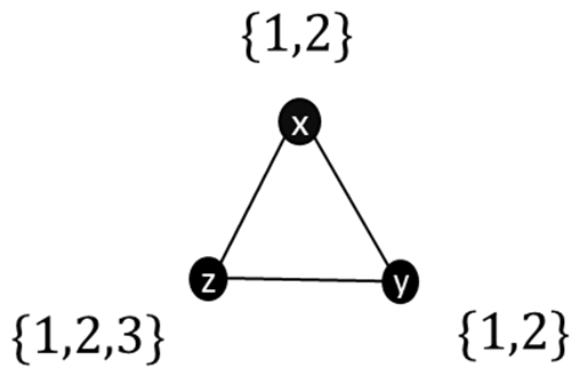
$x + y = 3$

$z + y$ be odd

$x + z$ be odd

$O(n^2 d^3)$: AC3 اجرای

AC3



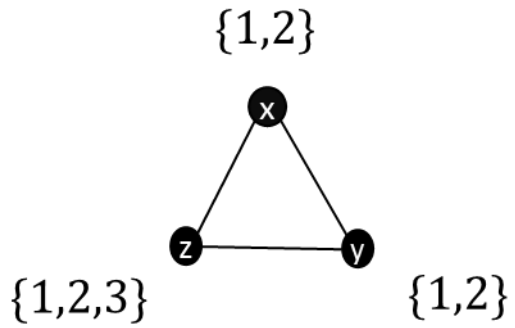
حالا یک csp باینری ۲ـ سازگار داریم.

$x + y = 3$

$z + y$ be odd

$x + z$ be odd

حالا آیا اجرای BT نهایی است؟ می بینیم که بازهم $O(d^n)$ است چون دور وجود دارد.

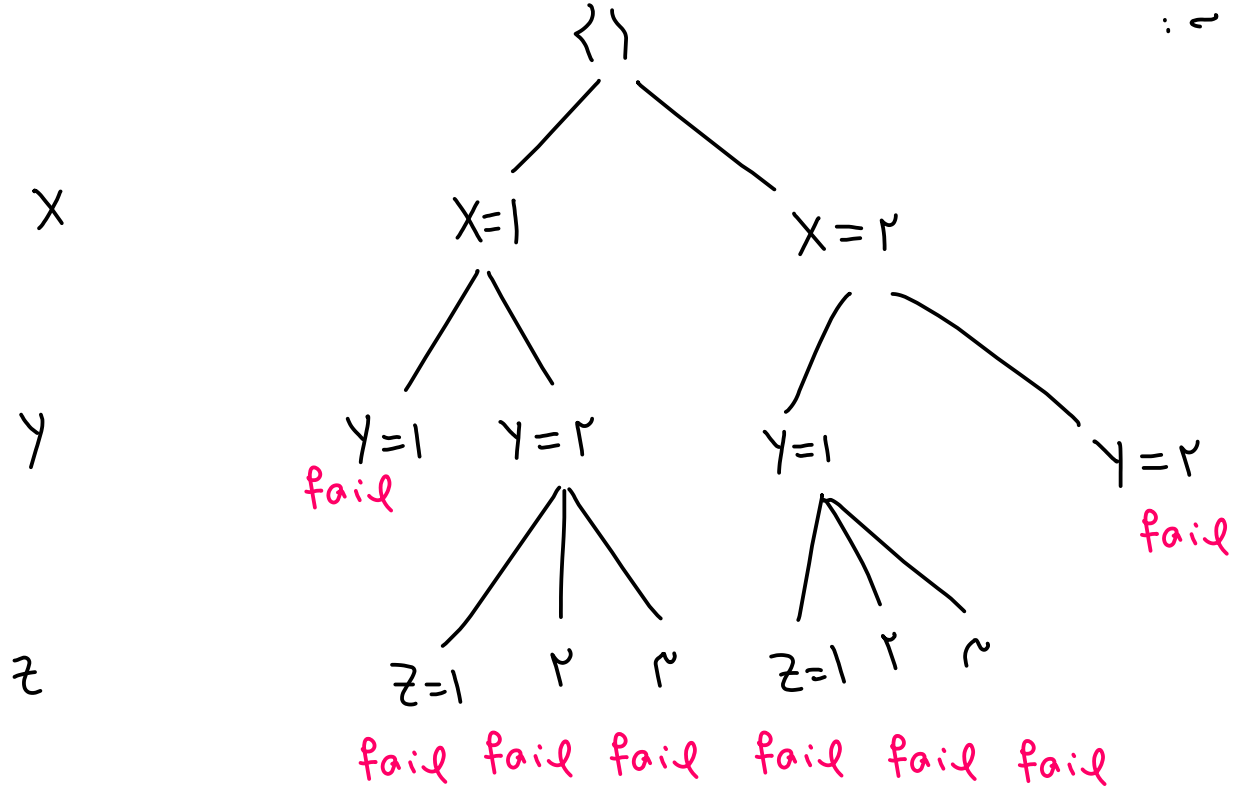


$x + y = 3$

$z + y \; be \; odd$

$x + z \; be \; odd$

BT



با آن که AC3 اجرا شده و مساله ۲-سازگار است
و دامنه هیچ‌کدام هم تهی نشده است می‌بینیم که:

① بازهم اجرای B.T برای $O(d^n)$ است
چون دور وجود دارد.

② جواب ندارد. (با آن که دامنه‌ها تهی نشده بود)

خلاصه در CSP های باینری

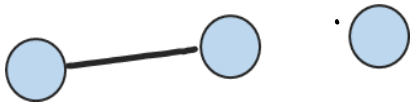
AC3 + BT

در حالت کلی: $n^2 d^3 + d^n$ (نمایی)

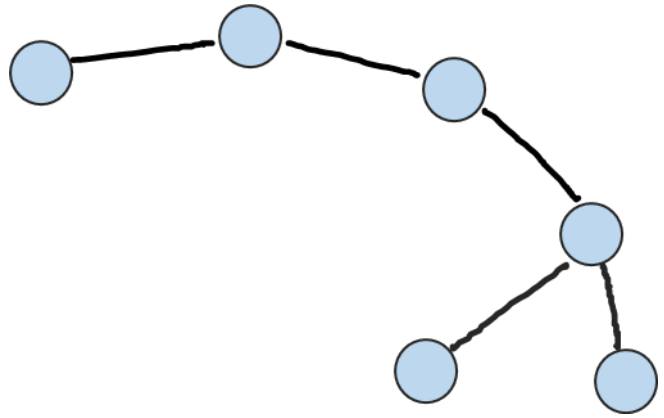
در گراف فاقد دور: $nd^2 + nd$ (چندجمله‌ای)

همچنین اگر گراف دارای دور را بتوان به شکل موفق به گراف فاقد دور تبدیل کرد باز هم زمان چندجمله‌ای به دست می‌آید.

یادآوری:



گراف فاقد دور با n-c گره



گراف دارای cut set با c گره

یادآوری

BT عادی: $O(d^n)$

استفاده از cutset: $O((n-c)d^{c+2})$

که اگر c ثابت باشد زمان چندجمله‌ای است

اما اگر مجموعه برش مثلاً $c = \frac{n}{2}$ باشد زمان بازهم نمایی است.

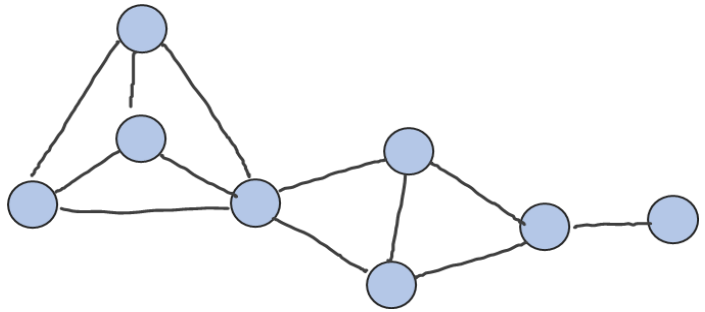
(*) یافتن cutset کمینه یک مسئله NP-Hard است.

روش دوم : استفاده از max clique ها

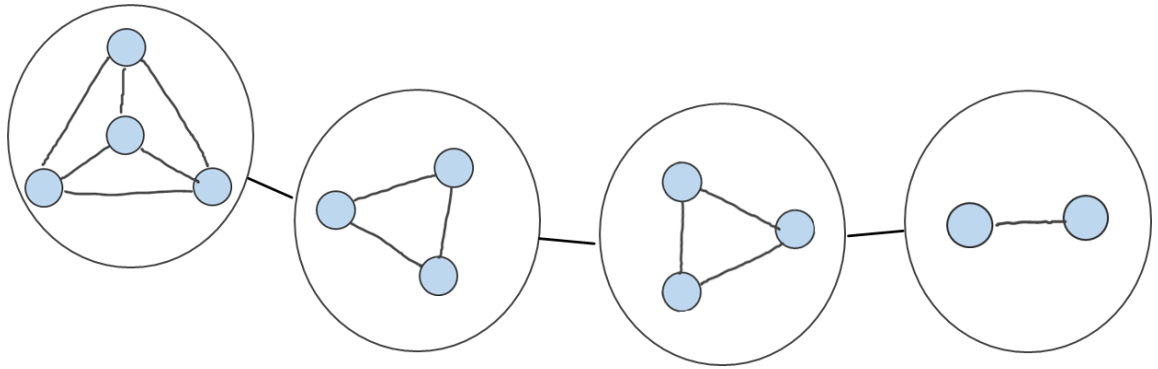
هم خوشه‌ی ماکیمال (max-clique) از گراف فضای حالت، تبدیل به یک متغیر جدید (ابرگره) در مساله‌ی جدید می‌شود. خوشه‌ای که $w+1$ گره دارد دارای دامنه‌ای d^{w+1} عضوی است.

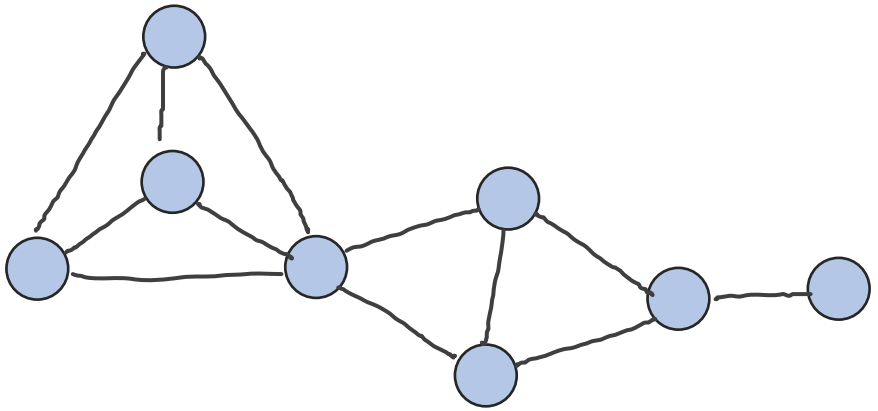
اگر دو خوشه‌ی ماکیمال دارای گره مشترک باشند به این معناست که در گراف جدید باهم رابطه دارند.

گراف مساله‌ی اول :



ابرگراف مساله‌ی دوم :



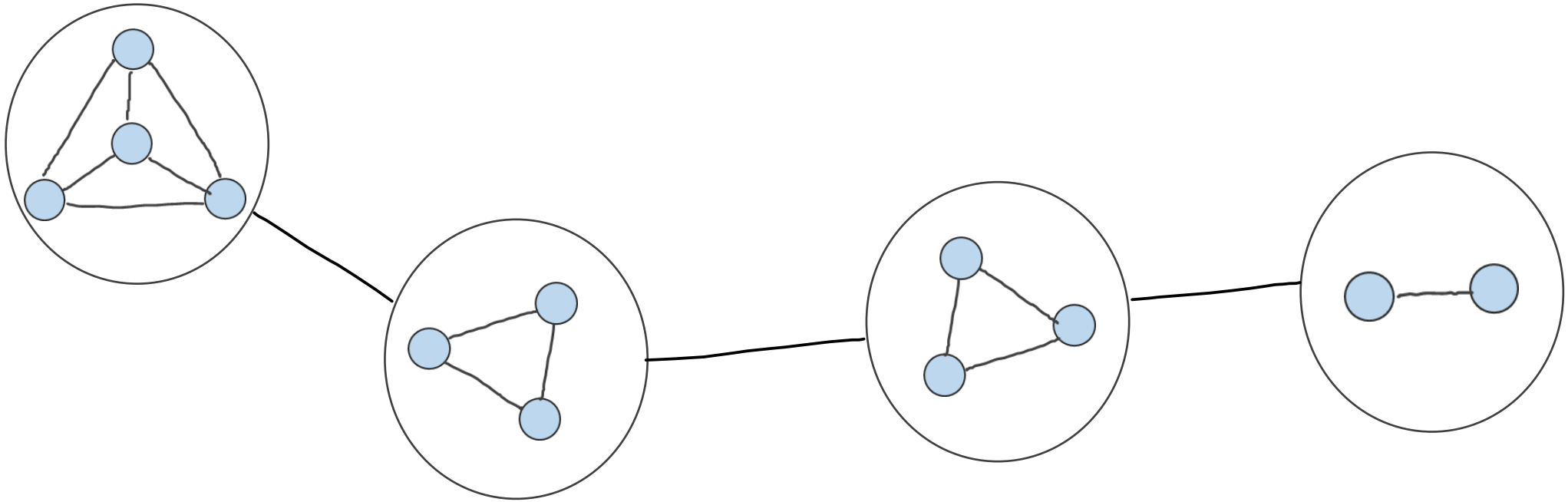


مسئله اصلی n متغیر داشت با دامنه d عضوی :

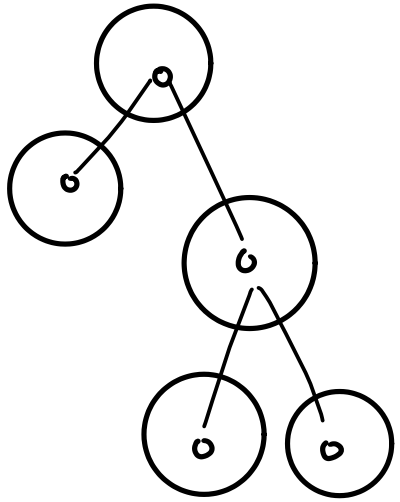
در مسئله‌ی جبهه اگر بزرگترین گره‌ها حداکثر $w+1$ گره داشته باشند دامنه‌ها حداکثر d^{w+1} عضوی است پس $|D| = d^{w+1}$.

w را عرض درخت در مسئله جبهه می نامیم. درمثال مقابل بزرگترین خوشه ۴ عضوی است

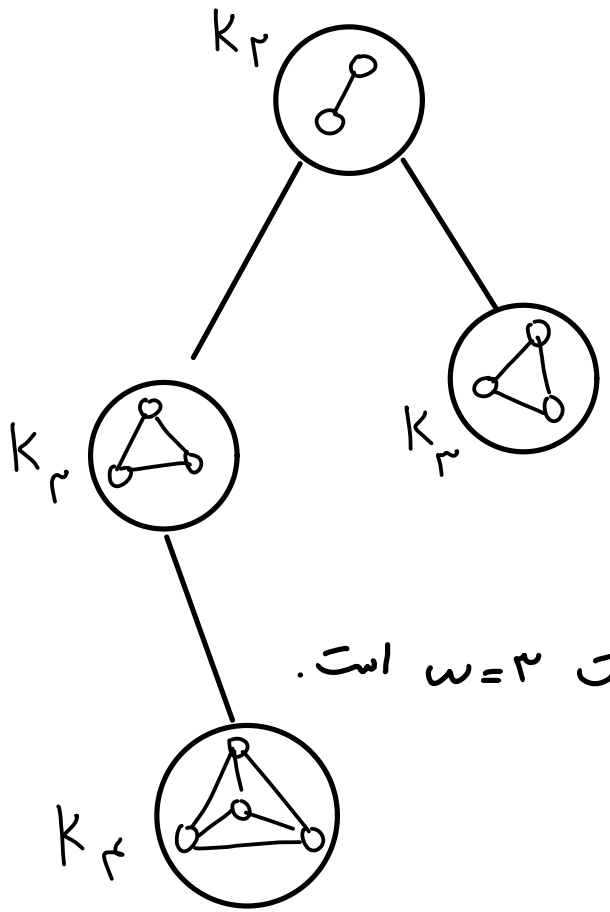
$$1+w = 4 \quad \Rightarrow \quad w = 3$$



مثال‌های توضیحی درباره عرض درخت.



در درخت باگره‌های معمولی، عرض $w=0$ است.



عرض این درخت $w=3$ است.

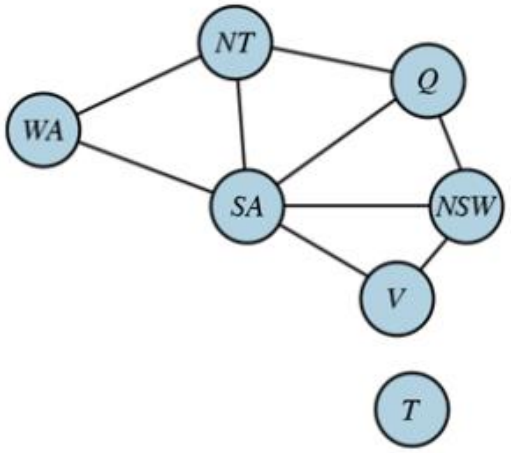
تحلیل این روش:

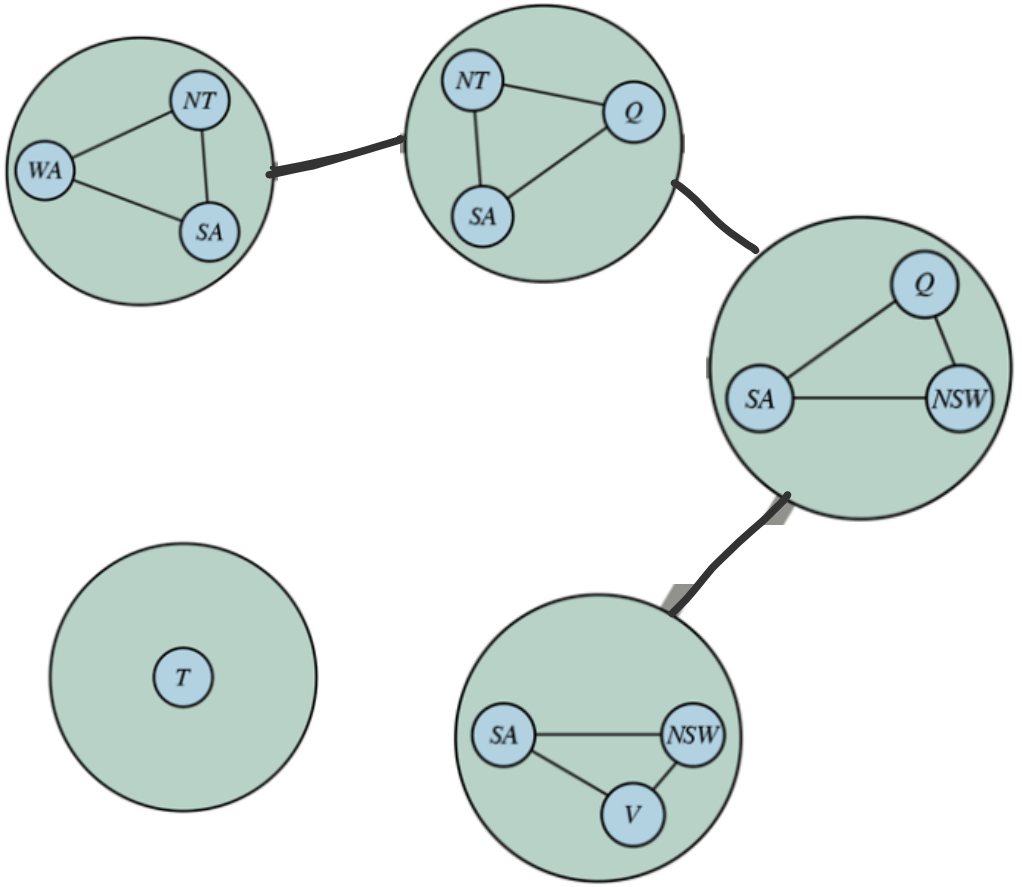
مسأله‌ی اصلی: $\Longrightarrow$ مسأله دوم:

n متغیر با دامنه حداکثر d عضوی

دارای n' متغیر (ابرگره) است. $(n' \leq n)$

دامنه‌ی آنها حداکثر $D = d^{1+w}$ عضوی است.





$$O(nd^{w+1})$$

tree width

هر ابرگره یک CSP کوچک است با $w+1$ متغیر

یافتن جواب‌های سازگار هر کدام آنها: $O(d^{w+1})$

تعداد زیرمسئله‌ها: n' (که $n' \leq n$)

$$n' D^{2} \leq n d^{2w+2}$$

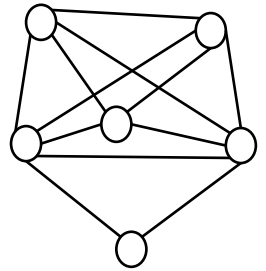
اجرای AC3+BT برای مسئله‌ی حبه (ابرگراف) چون فاقد دور است از مرتبه $n'D^2 + n'D$ یعنی $O(n'D^2)$

مرتبه‌زمانی حل CSP با این روش:

$$O(n d^{w+1} + n d^{2w+2})$$

$$O(n d^{2w+2})$$ بی همان حمله‌ی مهم است. همچنین بستگی به w دارد. اگر ثابت باشد مرتبه چندجمله‌ای به دست می‌آید.

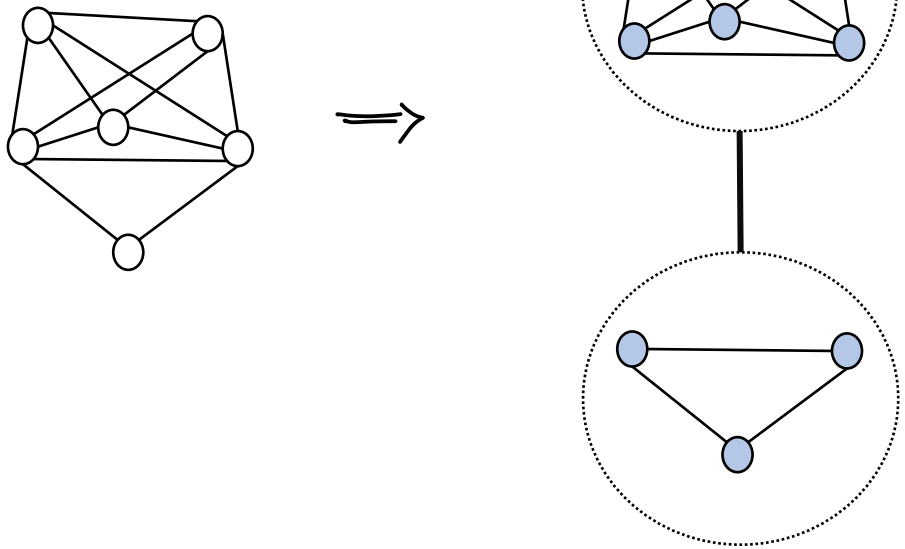
مثال: دراین csp باینری با n متغیر و دامنه d عضوی:



اگر به شکل مقابل خوشه بندی کنیم:

یک خوشه‌ی $n-1$ عضوی و

یک خوشه ۳ عضوی. دراین گراف حبابه $n'=2$ است. فقط ۲ ابرگره داریم.



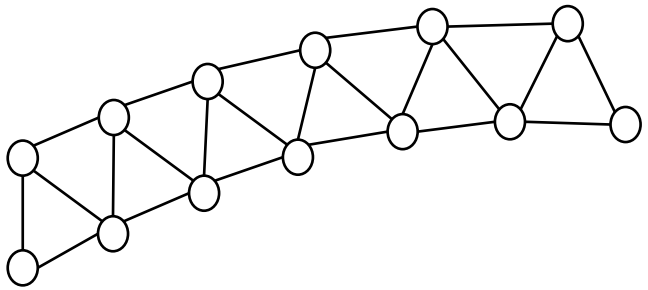
دراین صورت عرض $w=n-2$ است پس:

$$Time = O\left(n' d^{1+w} + n' d^{2w+2}\right) = O\left(2d^{n-1} + 2d^{2n-2}\right) = O(d^{2n})$$

(اصلاً خوب نیست)

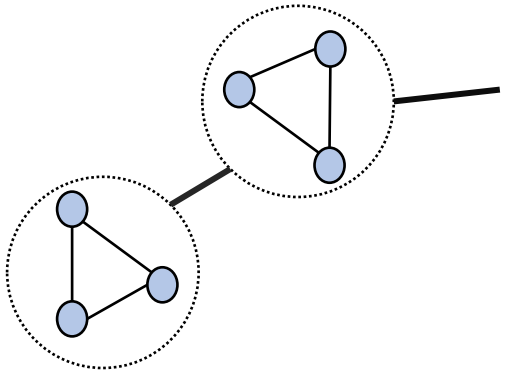
مثال دراین مساله روشی خوشه بندی موفق است:

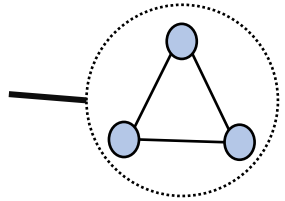
مساله اصلی



$\Rightarrow$

مساله‌ی دوم





تعداد ابرگره‌ها $n' \leq n$

عرض: $w = 2$

$$O(n d^{w+1} + n d^{2w+2}) = O(nd^3 + nd^6) = O(nd^6)$$

$n' \leq n$

چندجمله‌ای

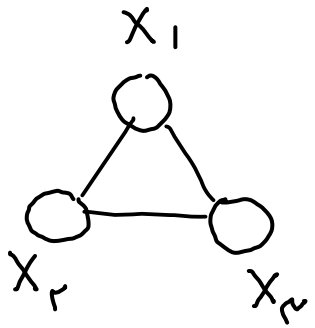
* تشخیص ناسازگاری برای محدودیت های از نوع چندتایی :

- محدودیت گلوبال مانند All Diff
- محدودیت منابع
- محدودیت حدود (کران ها)

(محدودیت متمایز بودن همه‌ی متغیرها) Alldiff constraints

در هر مرحله که تعداد متغیرهای باقی مانده از تعداد اعضای دامنه بیشتر باشد، ناسازگاری تشخیص داده می‌شود.

مثال: در این مثال تعداد متغیرها $n=3$ و تعداد دامنه $d=2$ است پس نمی‌توان محدودیت AllDiff را رعایت کرد.



$D=\{Red, Blue\}$

$d < n \Longrightarrow$ تشخیص ناسازگاری با محدودیت Alldiff

Resource constraints محدودیت منابع

$$X_1 + X_2 + \cdots + X_n \leq K$$ محدودیت منابع

تشخیص ناسازگاری و هرس کردن دامنه برای رعایت این محدودیت:

برای تشخیص ناسازگاری با این محدودیت، ابتدا عناصر دامنه را مرتب کنیم

عناصر دامنه را به ترتیب صعودی:

$$D = \{ d_1, d_2, \ldots, d_m \}$$

d_1: کوچک‌ترین — d_m: بزرگ‌ترین

نحوه بررسی ناسازگاری و اصلاح دامنه‌ها برای محدودیت منابع:

① $n d_1 > K \Longrightarrow$ مسأله جواب ندارد

② $(n-1) d_1 + d_m > K \Longrightarrow$ d_m از دامنه حذف شود.

در ① در واقع به هر متغیر کمترین مقدار ممکن را داده‌ایم.

در ② به یکی بیشترین و به دیگران کمترین مقدار را داده‌ایم.

پس از اصلاح دامنه، اقدام ② قابل تکرار است.

مثال: در مسئله $x_1 + x_2 + x_3 \leq 100$ با بررسی ناسازگاری‌ها دامنه‌ها را اصلاح کنید.

$D = \{35, 40, 50, 60\}$

پاسخ:

گام اول:

$D = \{35, 40, 50, 60\}$

$3 \times 30 = 90 < 100$

OK صدا جواب دارد.

گام دوم: ←

$2 \times 30 + 60 = 120 > 100$

60 از دامنه حذف شود

گام سوم: ←

$D = \{35, 40, 50\}$

$2 \times 35 + 50 = 110 > 100$

50 از دامنه حذف شود

پایان اصلاح دامنه ←

$D = \{35, 40\}$

مثال: ۱۶ پردازشگر داریم و ۴ ماموریت باید انجام شوند. تعداد پردازشگرهای لازم برای انجام ماموریت خام را X_i می‌نامیم. دامنه‌ی آن $D_{X_i} = \{5, 6, 7, 8\}$ است. محدودیت منابع و شرط سازگاری را بررسی کنید. دامنه را اصلاح کنید تا مساله سازگار شود. درصورت امکان یک جواب سازگار بیابید.

پاسخ:

محدودیت: فقط ۱۶ پردازشگر داریم پس $X_1 + X_2 + \cdots + X_4 \leq 16$

بررسی سازگاری:

$D = \{5, 6, 7, 8\}$

$n \times d_1 = 4 \times 5 = 20 \rightarrow n d_1 > 16$

$\rightarrow$ مساله ناسازگار است جواب ندارد.

مثال: ۲۱ پرداز شگر داریم و ۴ ماموریت باید انجام شوند. تعداد پردازشگرهای لازم برای انجام ماموریت خ‌ام را X_i می‌نامیم. و دامنه‌ی آن $D_{X_i} = \{5,6,7,8\}$ است. محدودیت منابع و شرط سازگاری را بررسی کنید. دامنه را اصلاح کنید تا مسئله سازگار شود. درصورت امکان یک جواب سازگار بیابید.

محدودیت:

$$X_1 + X_2 + \cdots + X_4 \leq 21$$

$$D = \{5,6,7,8\}$$

پاسخ:

① $n d_1 = 4 \times 5 = 20 \leq 21 \Rightarrow$ جواب دارد.

② $(n-1) d_1 + d_m = 3 \times 5 + 8 = 23 > 21 \Rightarrow$ ۸ از دامنه حذف شود:

$$D = \{5,6,7\}$$

ادامه:

$D=\{5,6,7\}$

$(n-1)d_1 + d_m = 3\times 5 + 7 = 22 > 21$ $\Rightarrow$ ۷ از دامنه حذف نمی‌شود:

$D=\{5,6\}$

حالا دیگر $(n-1)d_1 + d_m = 3\times 5 + 6 \leq 21$ است با این دامنه، سازگار است.

می‌توان جستجوی عقبگرد (BT) را آغاز کرد. ضمناً یک جواب سازگار آن است که همه X_iها برابر ۵ باشند.

Bounds propagation محدودیت گران

فرض کنید محدودیت‌های مسأله به این صورت باشد: $X_1 + X_2 = k$ (در حالت کلی $X_1 + X_2 + \cdots + X_t = k$)

و هر متغیر به یک کران بالا و یک کران پایین محدود شده باشد:

محدودیت گران $\begin{cases} X_1 \in [m_1, M_1] \\ X_2 \in [m_2, M_2] \end{cases}$

$X_1 + X_2 = k$ پس:

با قرار دادن کمترین مقدار ممکن، به جای X_1 می‌توان بیشترین مقدار ممکن برای X_2 را یافت و بالعکس.

با تکرار این ابزار می‌توان دامنه‌ی متغیرها را اصلاح کرد. (کوچک‌تر کرد)

مثال: مجموع مسافران دو پرواز ۴۲۰ نفراست: $F_1 + F_2 = ۴۲۰$.

ظرفیت اولین هواپیما ۱۶۵ نفراست. پس دامنه‌ی F_1 عبارت است از $D_1 = [۰, ۱۶۵]$

ظرفیت دومین هواپیما ۳۸۵ نفراست. پس دامنه‌ی F_2 عبارت است از $D_2 = [۰, ۳۸۵]$

با استفاده از محدودیت کران‌ها، دامنه‌ها را تا حد امکان اصلاح کنید.

$$D_1 = [0, 165] \quad \text{and} \quad D_2 = [0, 385] .$$

Now suppose we have the additional constraint that the two flights together must carry 420 people: $F_1 + F_2 = 420$. Propagating bounds constraints, we reduce the domains to

$$D_1 = [35, 165] \quad \text{and} \quad D_2 = [255, 385] .$$

پاسخ:

دامنه‌های اولیه: $D_1 = [0, 165]$ $\quad 0 \le F_1 \le 165$

$D_2 = [0, 385]$ $\quad 0 \le F_2 \le 385$

$F_1 + F_2 = 420$ $\xrightarrow{\max F_1 = 165}$ $165 + F_2 = 420$ $\xrightarrow{\min F_2}$ $F_2 = 420 - 165 = 255$

حال می‌دانیم $255 \le F_2$ پس دامنه‌ی آن اصلاح می‌شود:

$0 \le F_2 \le 385 \longrightarrow 255 \le F_2 \le 385$ $\quad D_2 = [255, 385]$

$255 \le F_2$

با تکرار این روش و قرار دادن $\max F_2 = 385$ داریم:

$F_1 + F_2 = 420$ $\xrightarrow{\max F_2 = 385}$ $F_1 + 385 = 420 \longrightarrow \min F_1 = 35$

$\longrightarrow D_1 = [35, 165]$

مثال: با استفاده از محدودیت کران‌ها، دامنه‌ها را اصلاح کنید.

$$X_1+X_2+X_3=200$$

$$D_1=[0,100]$$

$$D_2=[0,90]$$

$$D_3=[0,110]$$

پاسخ: به جای X_1 و X_2 کران بالای آنها را قرار دهیم، یک کران پایین برای X_3 به دست می‌آید:

$$X_1+X_2+X_3=200 \longrightarrow 100+90+X_3=200 \longrightarrow \min X_3=10$$

حال به دامنه‌ی اولیه X_3 توجه کنیم، خواهیم داشت:

دامنه جدید X_3: $D_3=[10,110]$

به این فرایند کران ادامه داد.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

ادامه جلسه ۹ فصل ۶

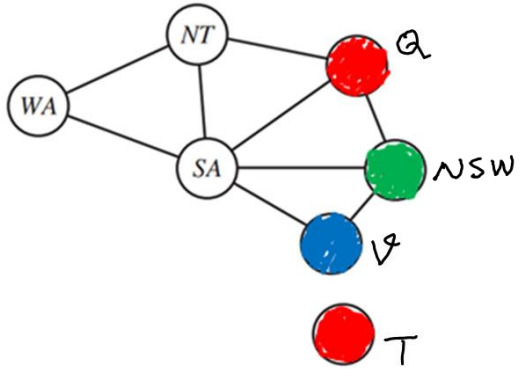
یک اشکال در الگوریتم عقبگرد ساده:

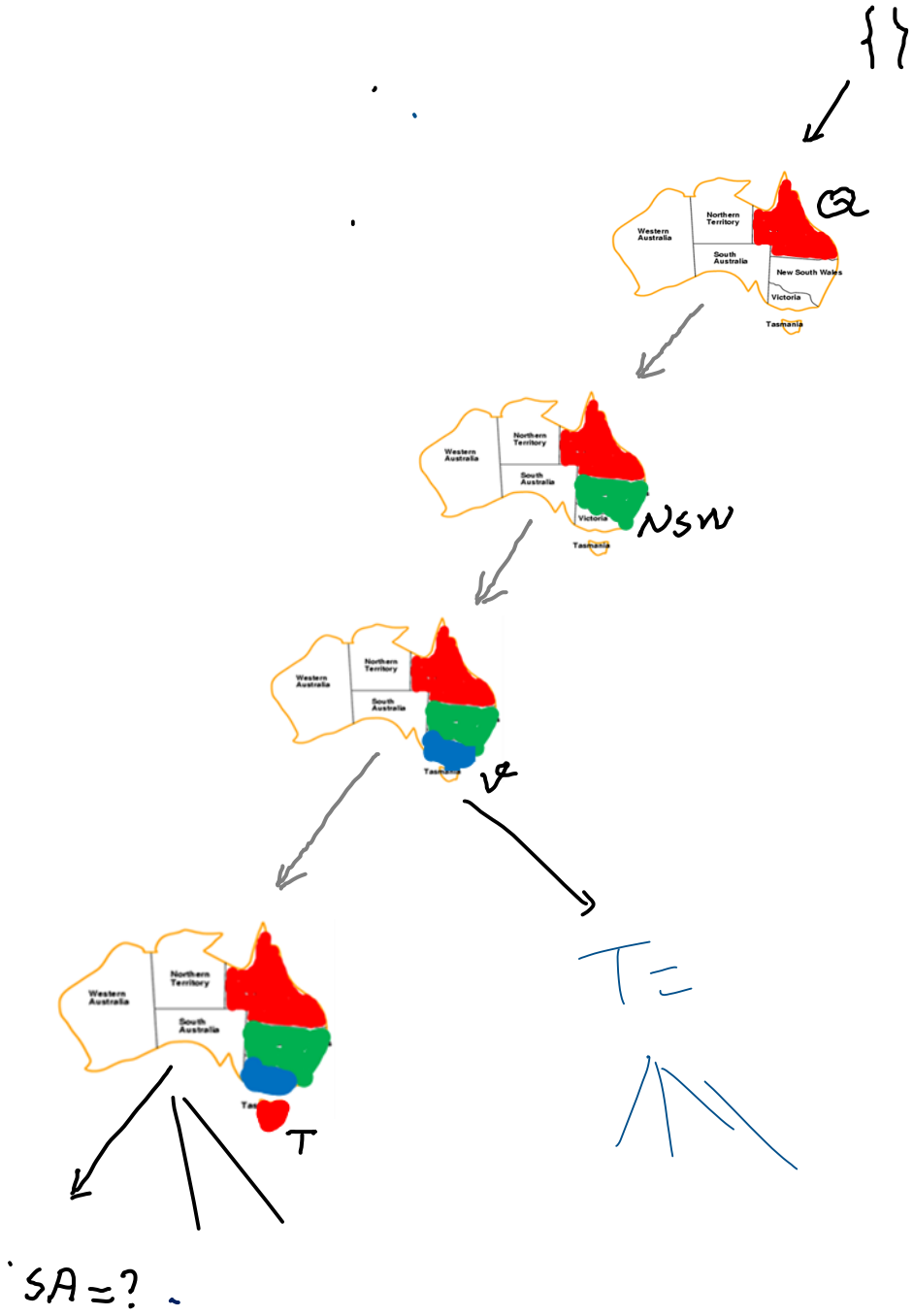
هرگاه به بن‌بست می‌رسد فقط یک گام به عقب برمی‌گردد. درحالی که شاید مشکل و ناسازگاری از چند قدم قبل ایجاد شده باشد.

ابتدا این اشکال را شرح می‌دهیم سپس برای حل آن روشهای عقبگرد هوشمندانه‌تری را مطرح می‌کنیم.

مثال :

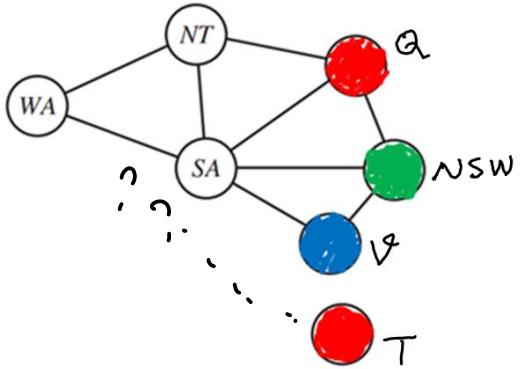
$$\{Q = red, NSW = green, V = blue, T = red\}.$$





در SA ناسازگاری رخ داده. اما این ناسازگاری ربطی به مقدار T ندارد. عقبگرد ساده فوراً متوجه این نمی‌شود. و ابتدا هم مقادیر T و SA را امتحان می‌کند.

یک راه حل: پرش به عقب با مجموعه برخورد.



conflict set (SA)

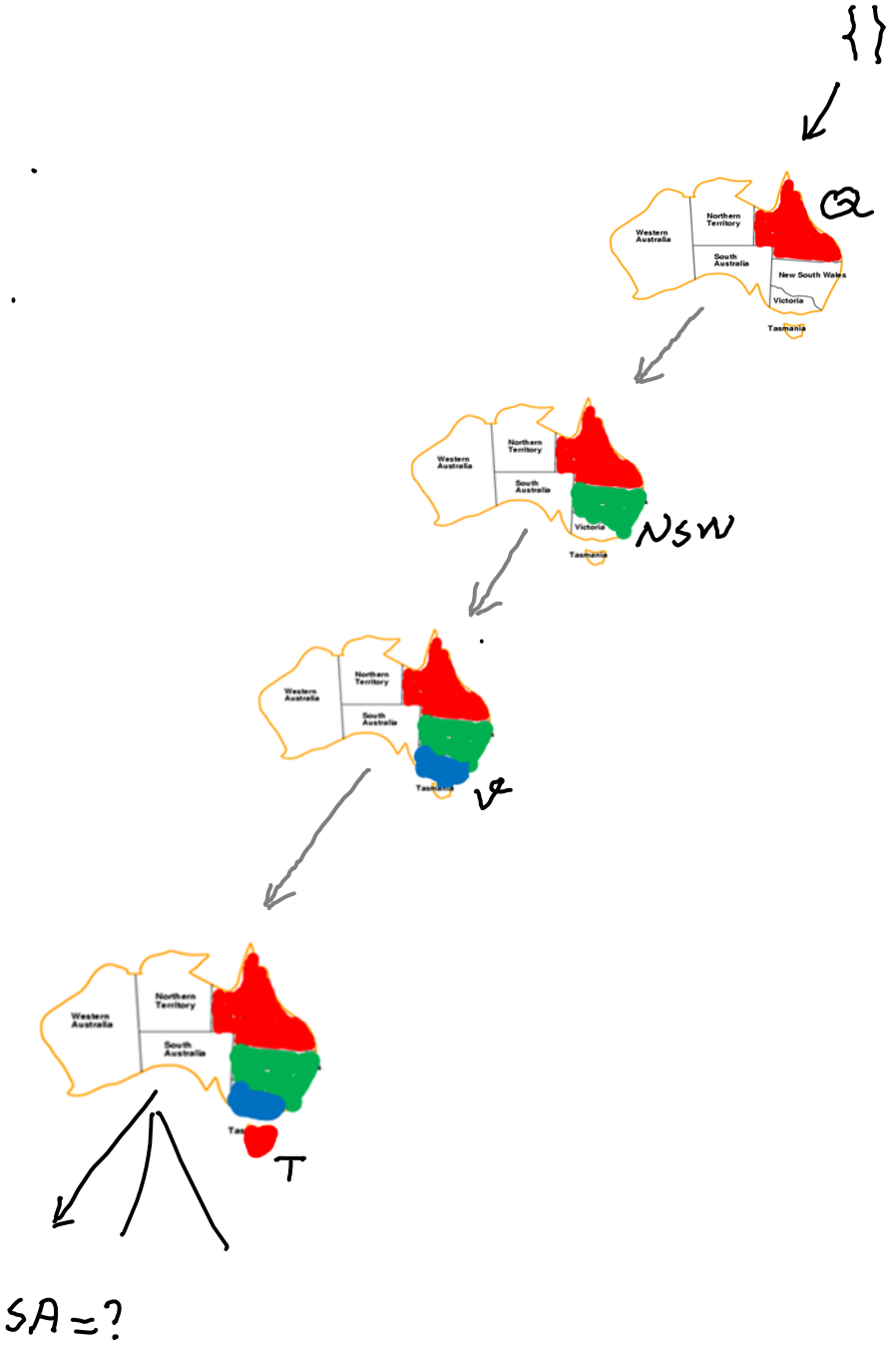
{}

{Q}

{Q, NSW}

{Q, NSW, V}

{Q, NSW, V}



[مجموعه‌ی برخورد SA برابر است با همه‌ی رنگ‌های مقداردهی شده‌ی SA تا آن مرحله.]

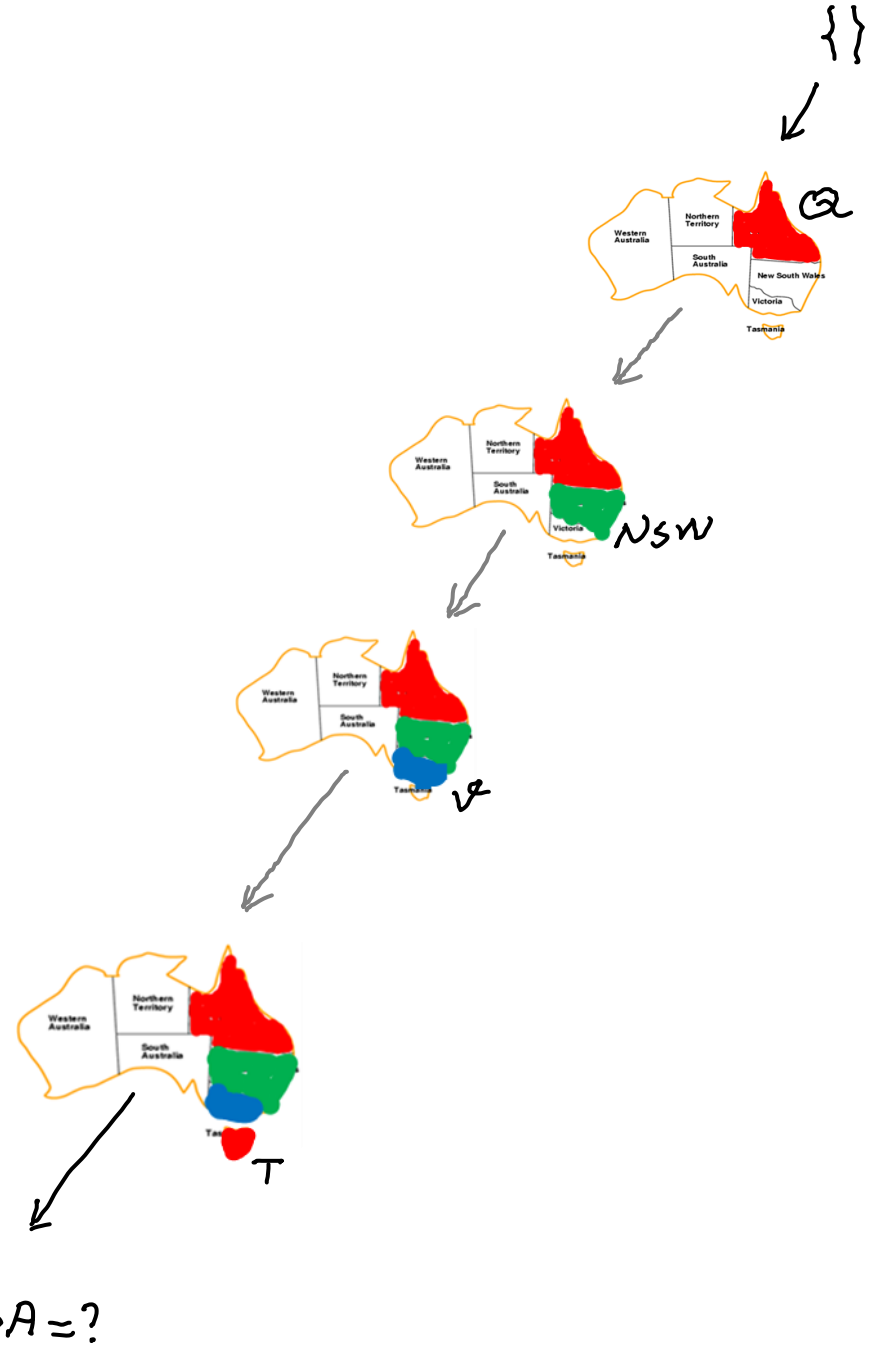
Back jumping:

وقتی از SA برمی‌گردد به عمیق‌ترین گره موجود در conflict(SA) برمی‌گردد.

Back jumping

در این مثل
به مجموعه‌ای برخورد (تعارض) کرد SA
نگاه کرده به گره‌ای از آن که عمیق‌تر بازگشت
برگشت کرده.

در این روش وقتی در SA به
ناسازگاری مربع به T برمی‌گردیم:



conflict set(sA)

{}

{Q}

{Q, NSW}

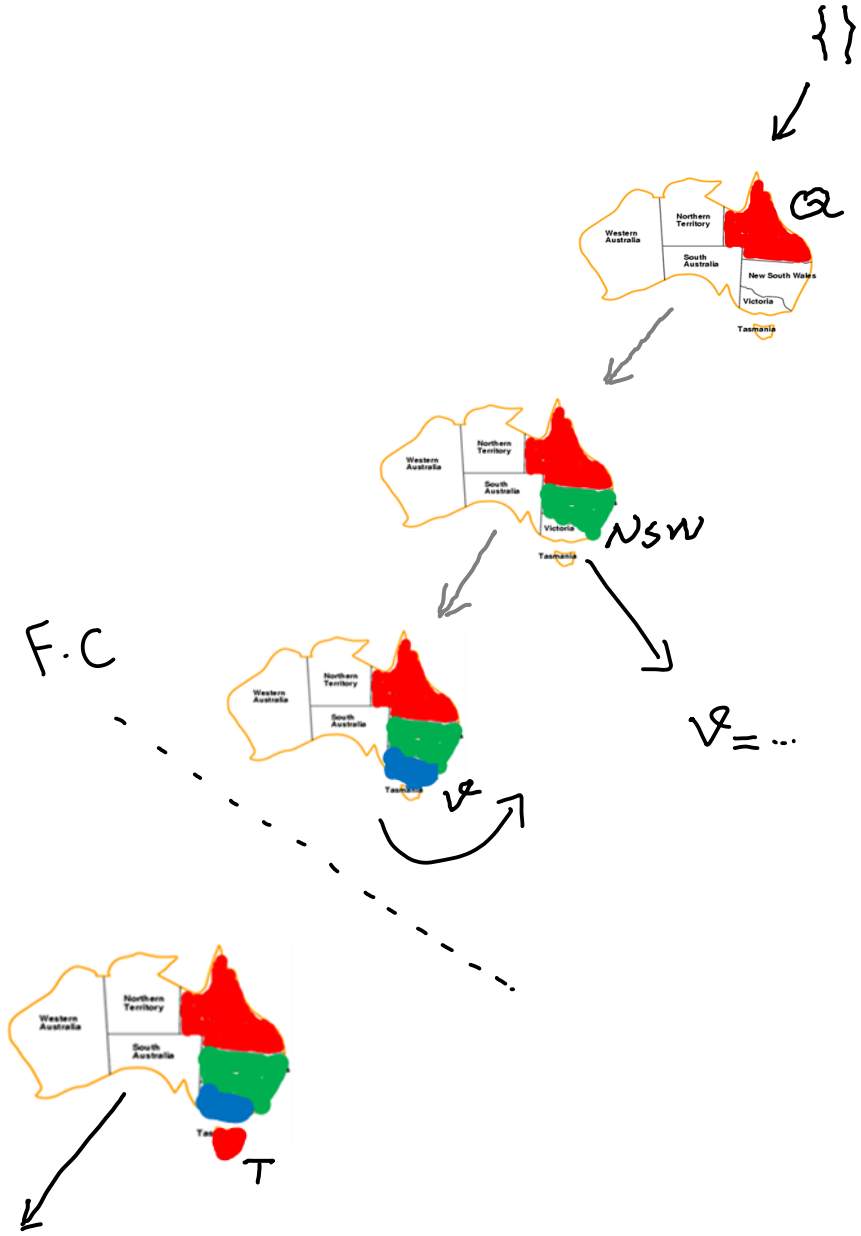
{Q, NSW, V}

{Q, NSW, T}

اما forward cheking قوی‌تر از Back jumping است.

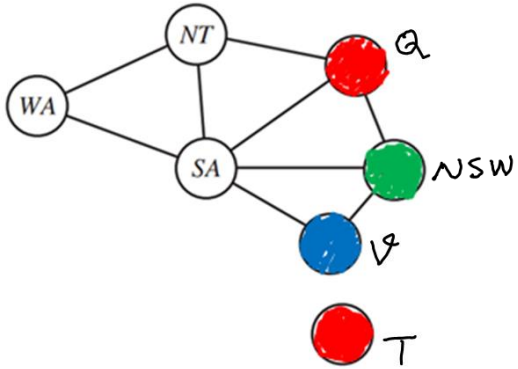
یعنی حتی سریع‌تر ناسازگاری را تشخیص می‌دهد.

در مثال قبل وقتی به ۷ مقدار آبی می‌دهیم دامنه SA تهی می‌شود از همین جا مقدار ۷ را تغییر می‌دهد (پیش از رسیدن به SA جلوی ناسازگاری گرفته شد)



قدم بعدی از مجموعه برخورد عمل می‌کند: Forward checking

وقتی به V آبی را دهم دامنه SA تهی می‌شود و از همان جا برمی‌گردیم.



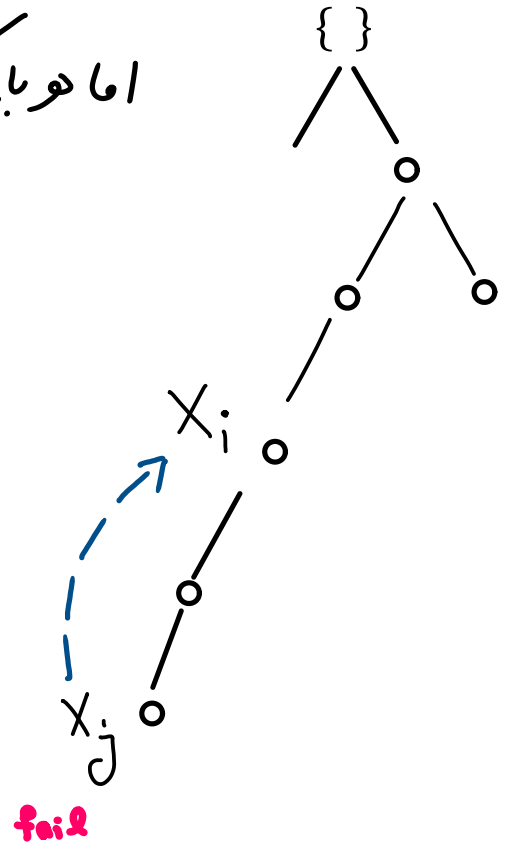
تا اینجا به نظر می‌رسه ایده‌ی مجموعه تعارض و پرش به عقب خیلی موفق نیست.
اما با کمی تغییر به نسخه‌ی قوی‌ای می‌رسیم به نام (پرش به عقب با هدایت برخورد)

Conflict-directed backjumping پرش به عقب با هدایت برخورد

در شروع، مجموعه تعارض‌های هر گره همان همسایه‌های مقدار دهی شده هستند.

اما هر بار که از X_j به X_i برمی‌گردیم این به روز رسانی انجام می‌شود:

$$conf(X_i) \leftarrow conf(X_i) \cup conf(X_j) - \{X_i\}$$



مثال راس

6.8

Consider the graph with 8 nodes A1, A2, A3, A4, H, T, F1, F2.
Ai is connected to Ai+1 for all i, each Ai is connected to H, H is connected to T, and T is connected to each Fi.

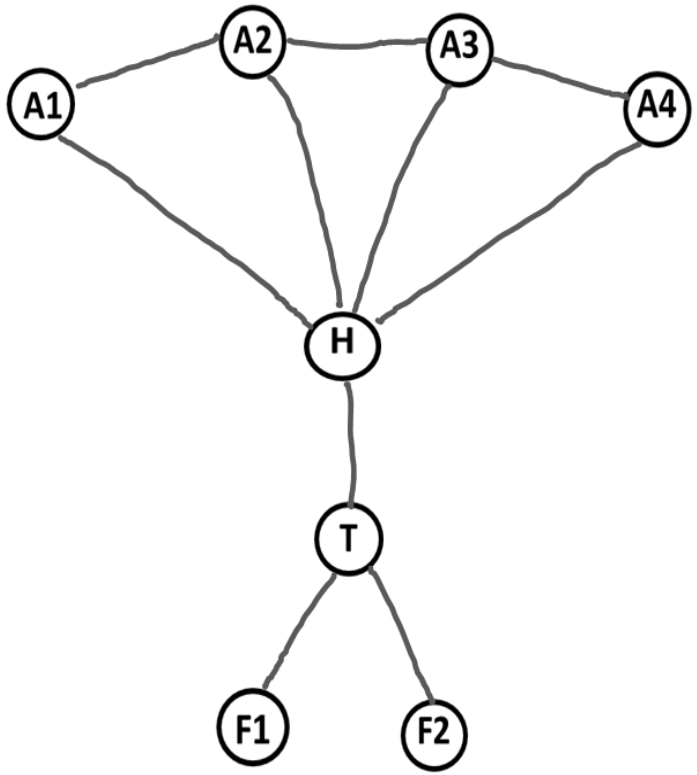
Find a 3-coloring of this graph by hand using the following strategy:

backtracking with conflict-directed backjumping,

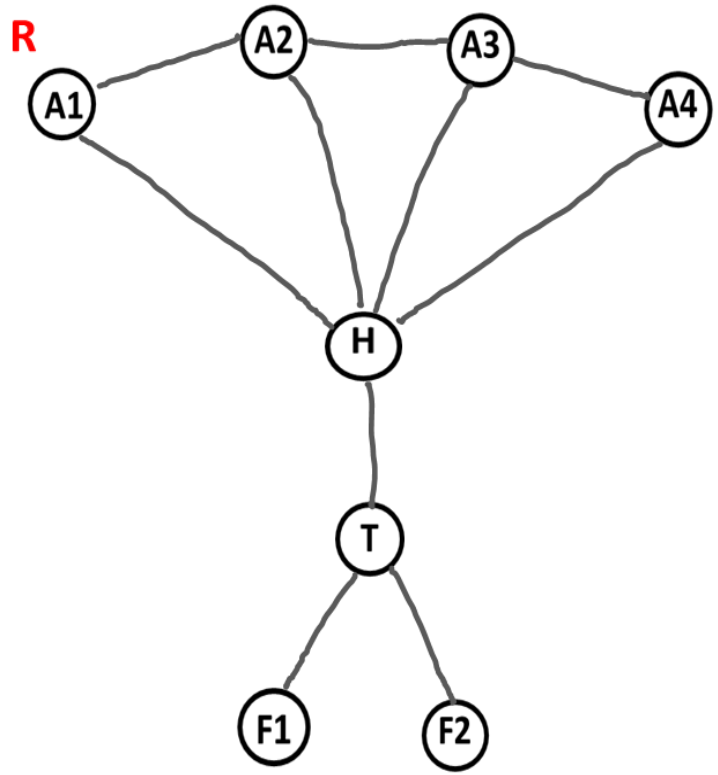
the variable order: **A1, H, A4, F1, A2, F2, A3, T,** ترتیب متغیرها

and the value order: **R, G, B.** ترتیب مقداردهی:

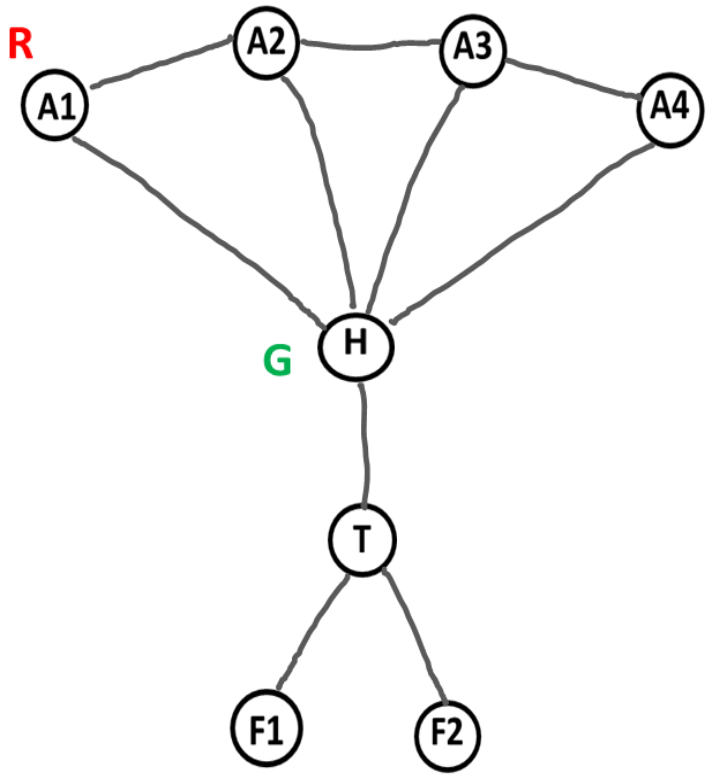
درس شروع کار: (انتخابی)



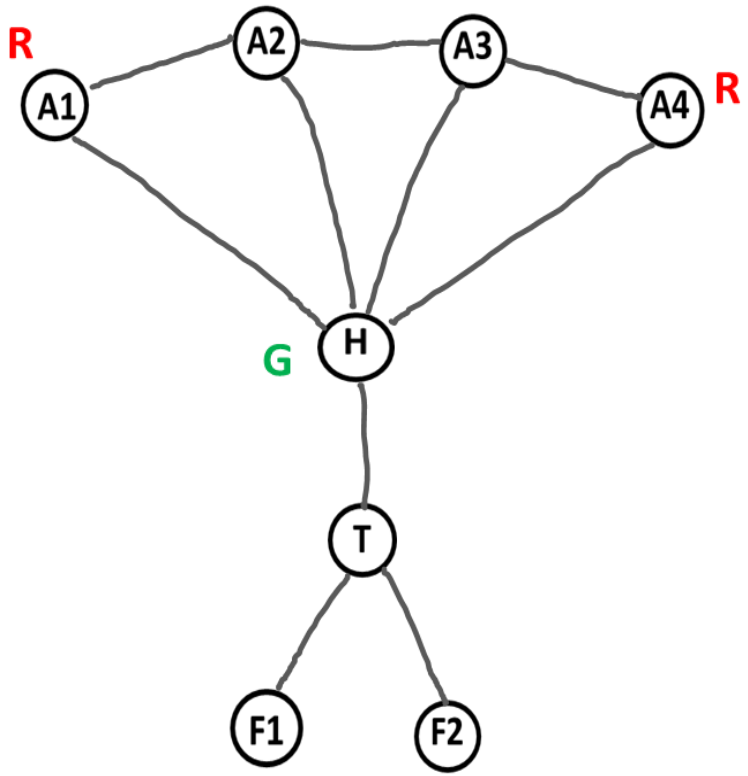
متغیرها	مقادیر	conflict set
A1		
H		
A4		
F1		
A2		
F2		
A3		
T		



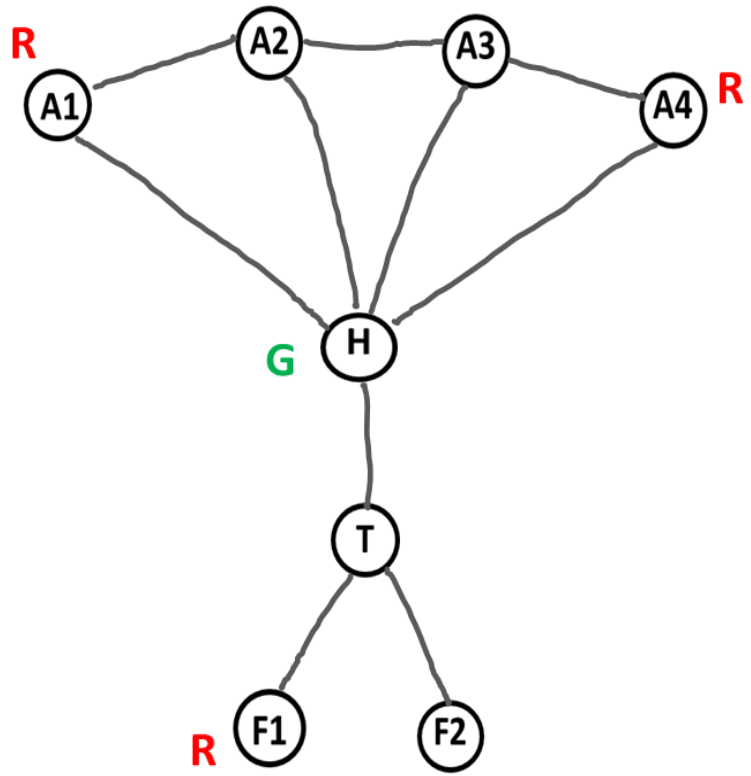
	معبّر	conflict set
A1	R	
H		A_1
A4		
F1		
A2		A_1
F2		
A3		
T		



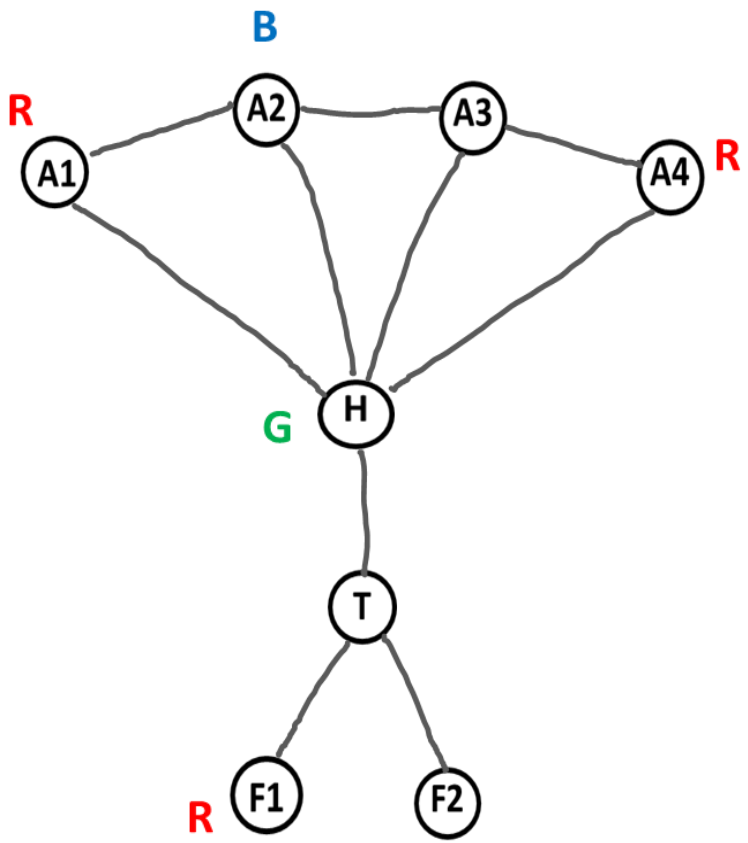
		conflict set
A1	R	
H	~~R~~ G	A_1
A4		H
F1		
A2		A_1 , H
F2		
A3		H
T		H



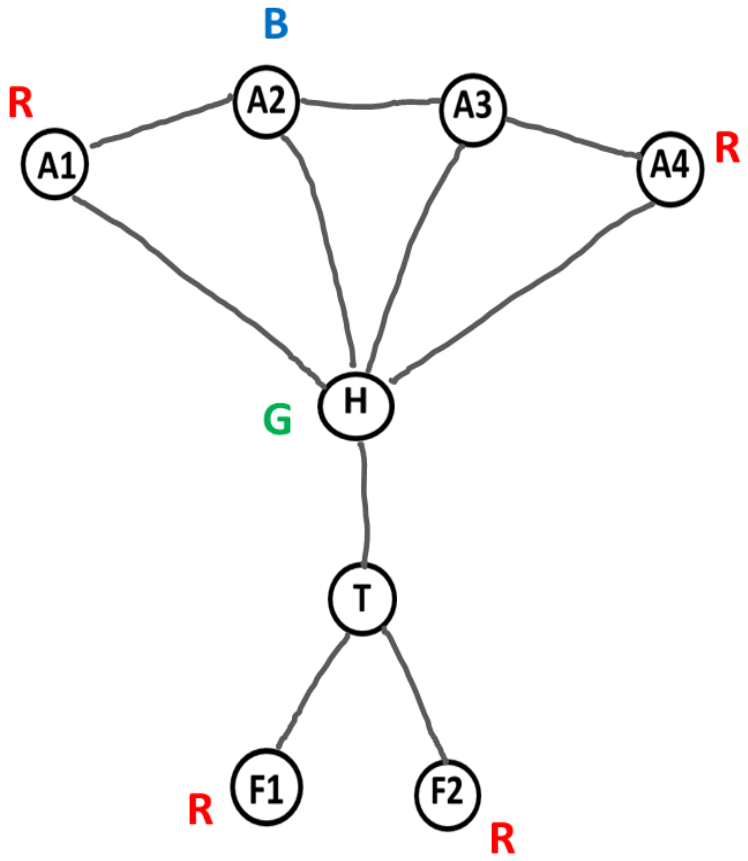
		conflict set
A1	R	
H	~~R~~ G	A_1
A4	R	H
F1		
A2		A_1 , H
F2		
A3		H , A_4
T		H



		conflict set
A1	R	
H	~~R~~ G	A_1
A4	R	H
F1	R	
A2		A_1 , H
F2		
A3		H , A_f
T		H , F_1

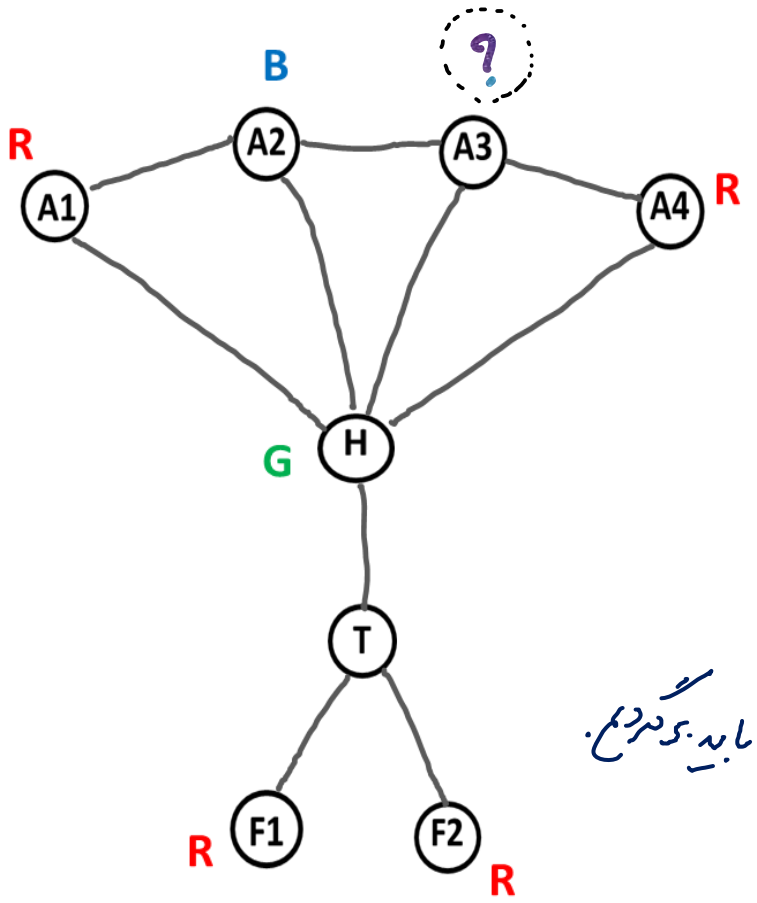


		conflict set
A1	R	
H	~~R~~ G	A_1
A4	R	H
F1	R	
A2	~~R~~ ~~G~~ B	A_1 , H
F2		
A3		H , A_4 , A_7
T		H , F_1



		conflict set
A1	R	
H	~~R~~ G	A_1
A4	R	H
F1	R	
A2	~~R~~ ~~G~~ B	A_1 , H
F2	R	
A3		H , A_4 , A_7
T		H , F_1 , F_2

در A_3 به ناسازگاری می‌رسیم باید طبق $conf(A_3)$ به A_2 برگردیم:

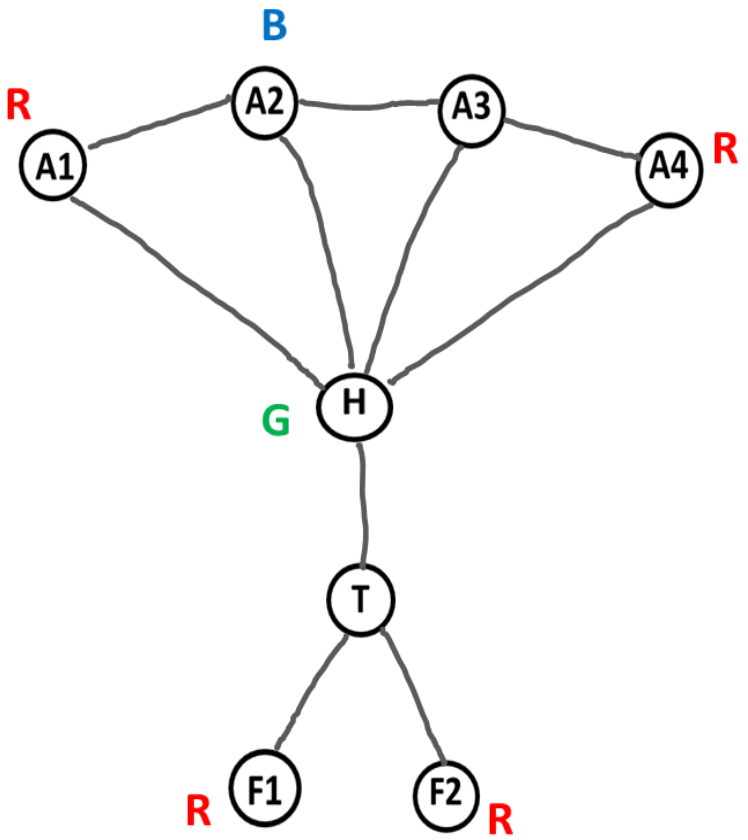


		conflict set
A1	R	
H	~~R~~ G	A_1
A4	R	H
F1	R	
A2	~~R~~ ~~G~~ B	A_1 , H
F2	R	
A3	~~R~~ ~~G~~ ~~B~~	H , A_4 , A_2
T		H , F_1 , F_2

ناسازگاری بابد برگردیم.

ادامه بعدی بازگشت به A_2 :

در بازگشت به A_2 مجموعه برخورد A_4 به برخورد A_2 اضافه می‌شود. (به جز خود A_2)

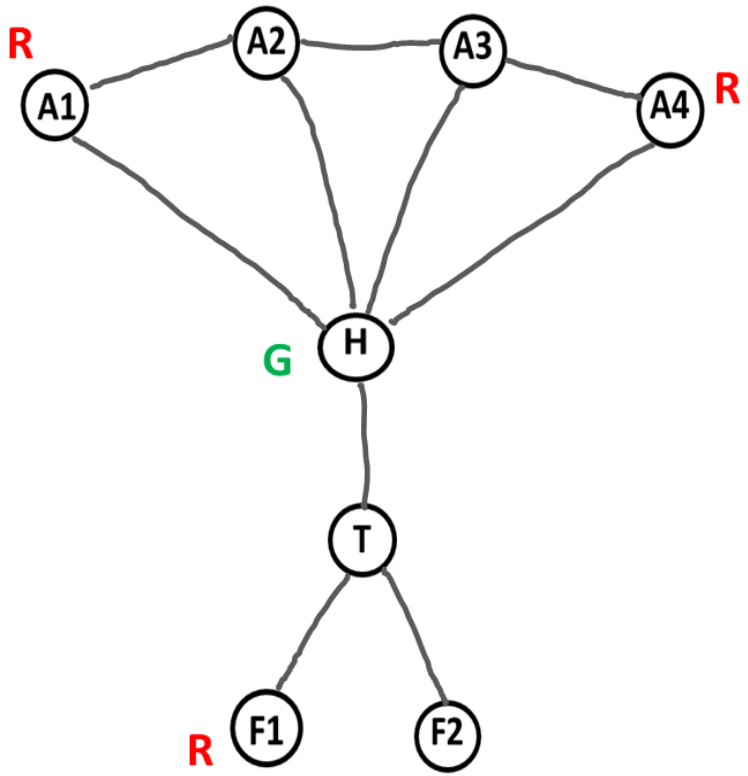


		conflict set
A1	R	
H	~~R~~ G	A_1
A4	R	H
F1	R	
A2	~~R~~ ~~G~~ B	A_1 , H
F2		
A3		H , A_4 , A_2
T		H , F_1 , F_2

#

$$conf(A_2) = conf(A_2) \cup conf(A_4) - \{A_2\}$$

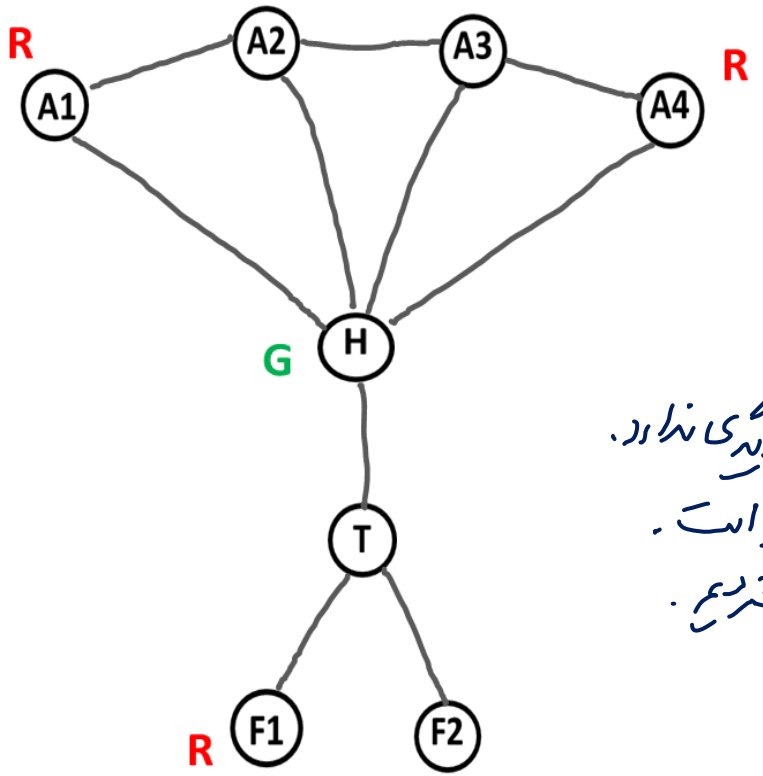
بازگشت به A_2 انجام شد: (F_2 و A_4 دیگر مقداردهی نشده اند)



		conflict set
A1	R	
H	~~R~~ G	A_1
A4	R	H
F1	R	
A2	~~R~~ ~~G~~ ~~B~~	A_1 , H , A_4
F2		
A3		H , A_4
T		H , F_1

در اسلایدِ بعدی کار را از A_2 ادامه می دهیم.

در A_2 هم ناسازگاری داریم. (دو رنگ قبلاً امتحان شده‌اند)



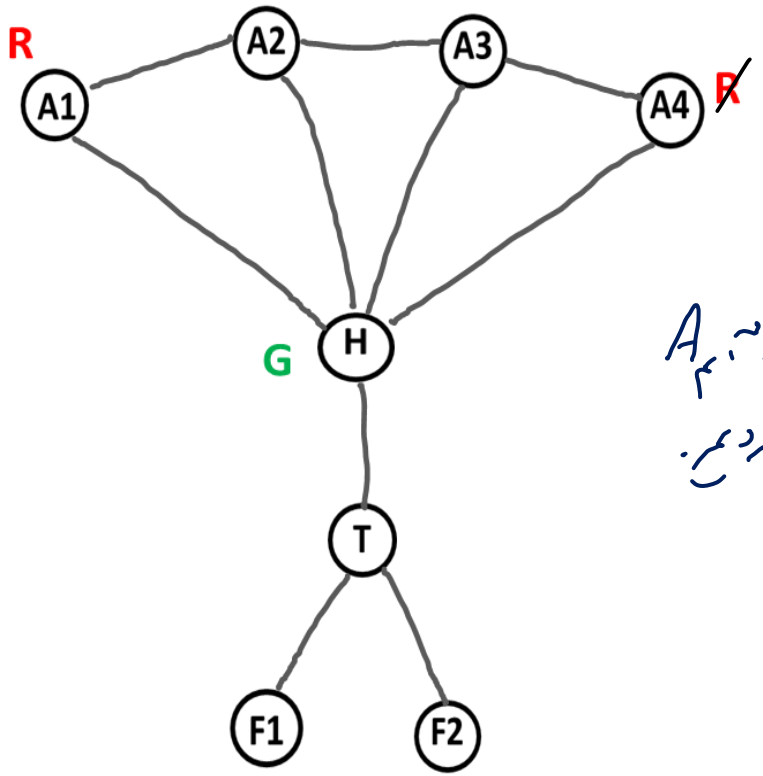
		conflict set
A1	R	
H	~~R~~ G	A_1
A4	R	H
F1	R	
A2	~~R~~ ~~G~~ ~~B~~ .	A_1 , H , A_4
F2		
A3		H , A_4 , A_2
T		H , F_1

A_2 مقداردهی ندارد. ناسازگار است. باید برگردیم.

طبق $conf(A_2)$ باید به A_4 برگردیم.

$$Conf(A_4) = Conf(A_4) \cup Conf(A_1) - \{A_4\}$$

بازنشست A_4

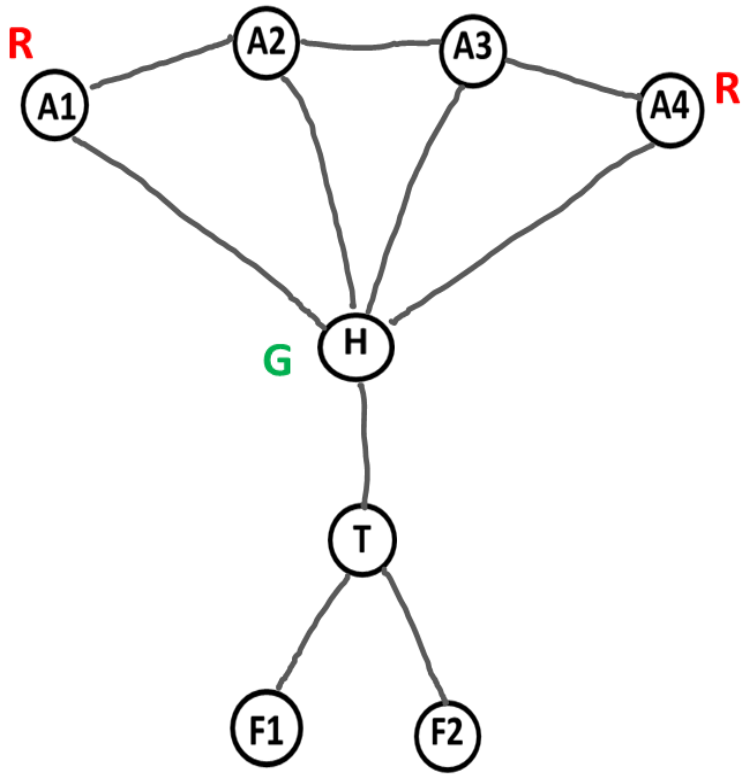


		conflict set
A1	R	
H	~~R~~ G	A_1
A4	~~R~~	H , A_1
F1		
A2		A_1 , H , A_4
F2		
A3		H , A_4 , A_2
T		H , F_1

A_4 باید بیرون برود

#

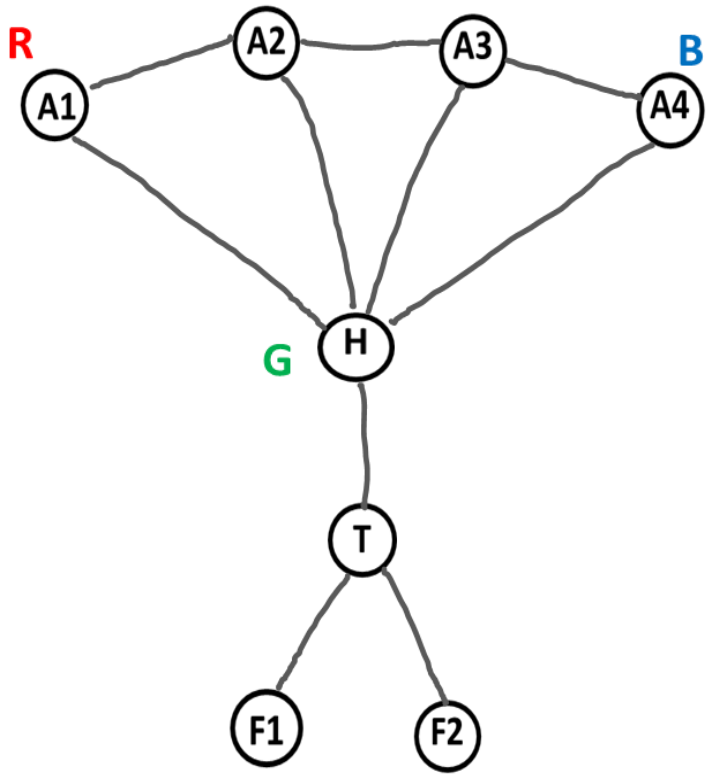
در ضمن دقت کنید گره های بعد از A_4 دیگر مقداری ندارند :



		conflict set
A1	R	
H	~~R~~ G	A_1
A4	~~R~~ ?	H , A_1
F1		
A2		A_1 , H , A_4
F2		
A3		H , A_4
T		H

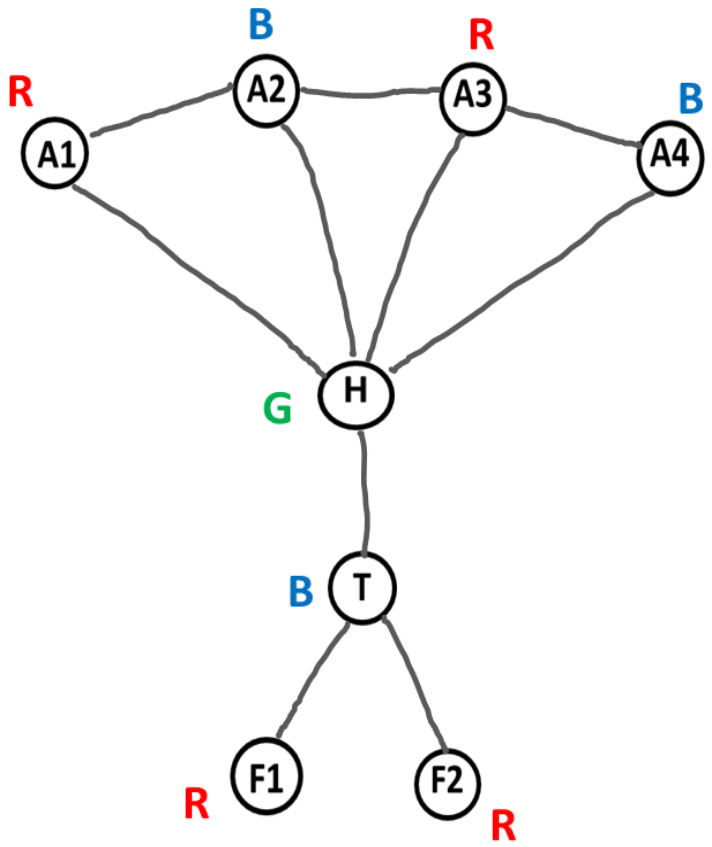
اصلاح برخوردها

ادامه کار



		conflict set
A1	R	
H	~~R~~ G	A_1
A4	~~R~~ ~~G~~ B	H , A_1
F1		
A2		A_1 , H , A_4
F2		
A3		H , A_4
T		H

دریافتی‌ها:



		conflict set
A1	R	
H	~~R~~ G	
A4	~~R~~ ~~G~~ B	
F1	R	
A2	~~R~~ ~~G~~ B	
F2	R	
A3	R	
T	~~R~~ ~~G~~ B	

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

ادامه جلسه ۹ فصل ۶

مرور چند مثال از esp های معروف.

Sudoku

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

سودوکو:

متغیرها: $X_{11}, \ldots, X_{99}$

(یا متغیرها ممکن است خانه‌های خالی باشند)

دامنه: $\{1,2,3,4,5,6,7,8,9\}$

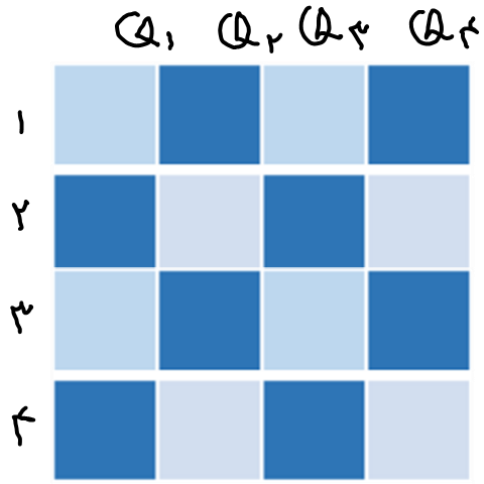
محدودیت: AllDiff در هر سطر.

AllDiff در هر ستون.

AllDiff در هر مربع کوچک.

	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

یک انتساب کامل سازگار (جواب)



مسأله n وزیر - مثلاً ۴-وزیر

متغیرها: $Q_1, Q_2, \dots, Q_n$

دامنه: $D = \{1, 2, 3, \dots, n\}$

Q_i = شماره سطر وزیر ستون i است.

محدودیت ها: برای هر جفت از وزیرها (تعداد جفت ها $\binom{n}{2}$ است)

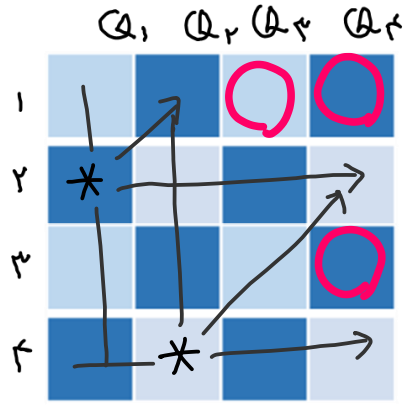
دو شرط داریم:

سطری برخورد نکنند $\quad i \neq j \implies Q_i \neq Q_j$

قطری برخورد نکنند $\quad i \neq j \implies |Q_i - Q_j| \neq |j - i|$

مثال: اگر در ۴-وزیر $Q_1=2$ ، $Q_2=4$

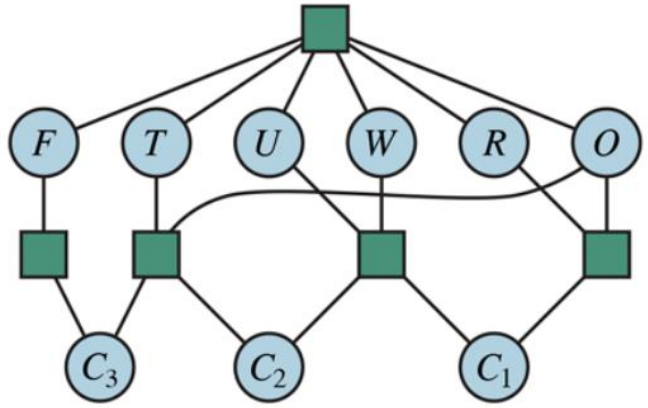
باشد کدام متغیر طبق MRV انتخاب می‌شود.



پاسخ: دامنه مجاز Q_3 شامل $\{1\}$ است دامنه مجاز Q_4 شامل $\{1,3\}$ است. طبق MRV: Q_3 انتخاب می‌شود.

cryptarithmetic puzzles. اعداد رمزی

$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$



متغیر اصلی — متغیر کمکی

- Variables: F ,T ,U ,W ,R ,O, — X_1 ,X_2 ,X_3
- Domains: {0,1,2,3,4,5,6,7,8,9} — {0,1}
- Constraints: Alldiff (F,T,U,W,R,O)
  - $O + O = R + 10 \cdot X_1$
  - $X_1 + W + W = U + 10 \cdot X_2$
  - $X_2 + T + T = O + 10 \cdot X_3$
  - $X_3 = F,\ T \neq 0,\ F \neq 0$

دامنه — محدودیت — دامنه

$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$

هر حرف نماینده یک رقم است.

محدودیت AllDiff داریم.

دامنه‌ها $\{0,1,\dots,9\}$

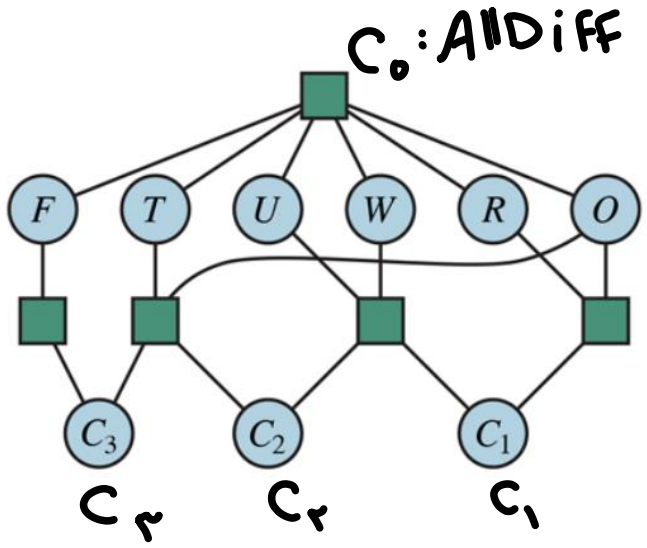
اما در عمل برای اعمال دو رقمی،

متغیر کمکی لازم داریم. مثلاً

$$\text{یا: } \begin{cases} O+O=R \\ O+O=10+R \end{cases} \Longrightarrow O+O=R+10x_1$$

که $x_1=0$ یا 1

$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$



برای CSP با محدودیت
حهای جبه‌گانه به جای گراف معمولی
از ابرگراف استفاده می‌شود.

هر محدودیت ممکن است به گره
تبدیل شود. و متغیرهای که
در آن شرکت دارند
به آن متصل شوند.

- Variables: F ,T ,U ,W ,R ,O, X_1 ,X_2 ,X_3
- Domains: {0,1,2,3,4,5,6,7,8,9} {0,1}
- Constraints: Alldiff (F,T,U,W,R,O)
  - $O + O = R + 10 \cdot X_1$
  - $X_1 + W + W = U + 10 \cdot X_2$
  - $X_2 + T + T = O + 10 \cdot X_3$
  - $X_3 = F,\ T \neq 0,\ F \neq 0$

C₀
C۱
C۲
C۳

Example:

مسئله‌ی ارضاءپذیری (SAT) را می‌توان به صورت یک CSP مطرح کرد:

$$\phi \equiv (P_1 \vee \sim P_2) \wedge (\sim P_1 \vee P_3)$$

محدودیت‌ها: می‌خواهیم ϕ ارزش true پیدا کنیم که معادل است با دو شرط زیر:

گزاره‌ی $(P_1 \vee \sim P_2)$ ارزش true داشته باشد.

گزاره‌ی $(\sim P_1 \vee P_3)$ ارزش true داشته باشد.

متغیرها: P_1, P_2, P_3 متغیرهای منطقی هستند

دامنه‌ها: دامنه آن‌ها $D = \{\text{false}, \text{true}\}$ یا همان $D = \{0,1\}$ است.

مقدمه و توضیح برای فصل بعدی

$$\underbrace{P \;,\; q \rightarrow P \;,\; q \vee \sim P}_{KB} \overset{?}{\models} q$$

استنتاج $KB \models \alpha$ درست است

اگر و تنها اگر:

$KB \wedge \neg \alpha$ تناقض (ناسازگاری) (ارضاءِ ناپذیر) باشد.

مسایل مثل یک CSP بود.

① ② ③

$\{ A \vee B \vee \neg C \;,\; \neg A \vee \neg B \;,\; \neg A \vee B \vee \neg C \}$

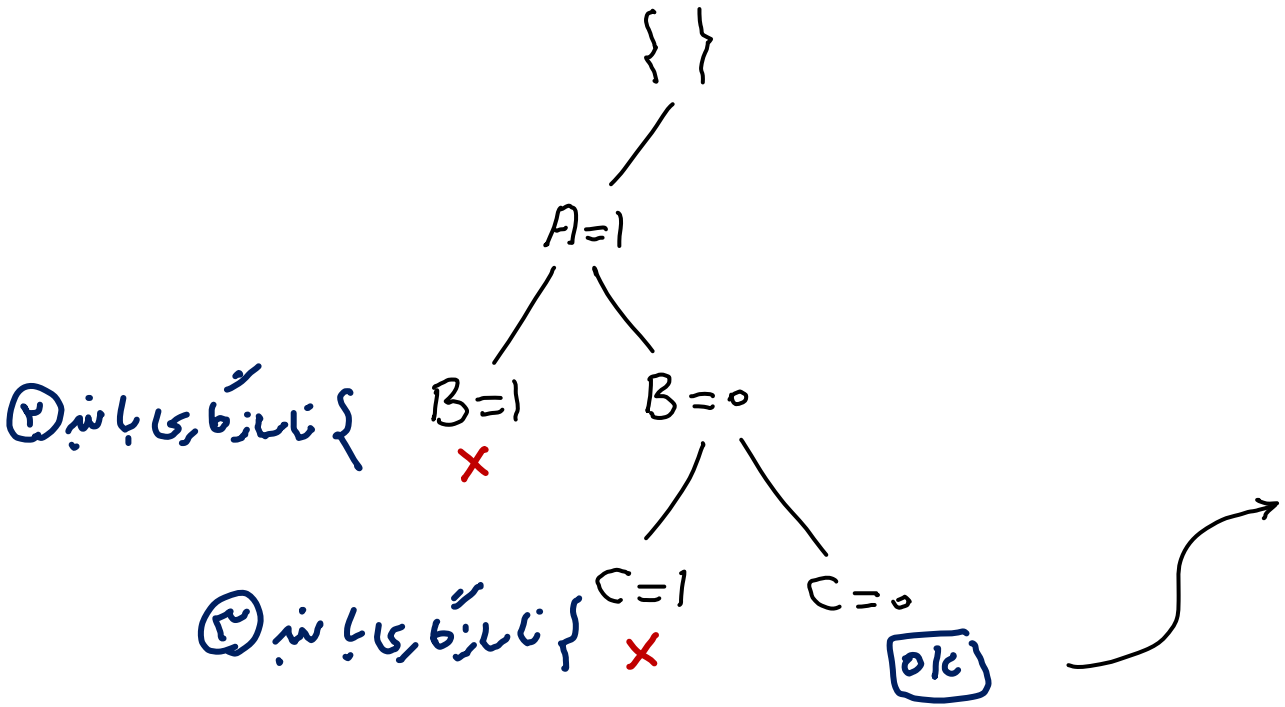
متغیرها: A, B, C

دامنه‌ها $D = \{0,1\}$ یا همان $\{false, true\}$

محدودیت‌ها: ① و ② و ③ همگی True باشند.

مثال: مسئله SAT را به روش DPLL (به صورت یک CSP) حل کنیم.

آیا این مجموعه ارضاپذیر است؟ $\{ A \vee B \vee \neg C \ (1) \ , \ \neg A \vee \neg B \ (2) \ , \ \neg A \vee B \vee \neg C \ (3) \}$



پس این مجموعه ارضاپذیر است و یکی از مدل‌های آن $(A=1, B=0, C=0)$ است.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

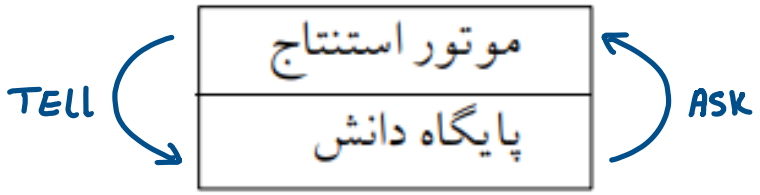
ابوالفضل گیلک
گروه بابان

جلسه ۱۰ فصل ۷

Chapter 7
Logical Agents

function KB-AGENT(*percept*) **returns** an *action*
  persistent: *KB*, a knowledge base
    t, a counter, initially 0, indicating time

  TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
  action ← ASK(*KB*, MAKE-ACTION-QUERY(*t*))
  TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
  $t \leftarrow t + 1$
  return *action*



مؤلفه‌های عامل مبتنی بر دانش

پایگاه دانش می‌تواند شامل دانش پیش زمینه[۴] (دانش اولیه) و دانشی باشد که عامل به مرور یاد می‌گیرد.

بر روی KB دو عمل انجام می‌شود:

۱ـ $TELL$: جمله‌ای را به KB اضافه می‌کند.

۲ـ ASK : اطلاعاتی را از KB استخراج می‌کند.

منطق گزاره‌ای دوارزشی (هوگزاره ارزش 0 یا 1 دارد) ← (موضوع مورد مطالعه ما در این فصل)

منطق گزاره‌ای چندارزشی (مثلاً ارزش 0 ، $\frac{1}{2}$ ، 1)

منطق فازی : ارزش‌ها در بازه‌ی [0،1] تغییر می‌کند.

منطق مرتبه اول یا FOL : استفاده از متغیرها و سورهای کلی و جزئی.

توضیحات تمرین



نوشتن این دانش به این معنی است که اگر $P(k)$ برقرار باشد $P(k+1)$ هم برقرار می‌شود.

در منطق گزاره‌ای ۵ جمله لازم است

$$P(1) \rightarrow P(2)$$

$$P(2) \rightarrow P(3)$$

$$\vdots$$

$$P(5) \rightarrow P(6)$$

در FOL یک جمله لازم است:

$$\forall i \in \{1,2,3,4,5\} \quad P(i) \rightarrow P(i+1)$$

زبان منطق گزاره ای :

عملگرهای رسمی : $\sim, \wedge, \vee, \rightarrow$

جمله اتمی : یک گزاره ساده بدون عملگر منطقی.

عملگرهایی که می‌توان تولید کرد :

حملات مرکب : $\sim P$ ، $P \vee q$ ، $P \rightarrow \sim q$ ، ...

$\leftrightarrow$, XOR, NAND, NOR, ..

ثابت‌های منطقی : T و F (همواره درست و همواره نادرست)

دامنه متغیرها (ارزش‌ها) : هر متغیر منطقی یکی از مقادیر $\{0, 1\}$ یا همان $\{false, true\}$ دارد.

لیترال‌ها : P ، $\sim P$ ، .. (لیترال مثبت و لیترال منفی)

بند (clause) : ترکیب فصلی لیترال‌ها : $\sim P_1 \vee P_2 \vee P_3$ ، P_1 ، $\sim P_2 \vee P_3$

یادآوری: (نحوه‌ی ارزش‌گذاری به گزاره‌های مرکب)

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

اگر ارزش گزاره‌ها را با 1 و 0 نشان دهیم $v_p \in \{0,1\}$ آنگاه:

$$v_{\sim p} = 1 - v_p$$

$$v_{p \wedge q} = v_p \cdot v_q$$

$$v_{p \vee q} = v_p + v_q - v_p v_q$$

$p \to q$ وقتی معتبر است که: $v_p \le v_q$

$p \leftrightarrow q$ وقتی معتبر است که $v_p = v_q$

گزاره‌ای H_1 و H_2 هم‌ارز (معادلند) اگر در هر حالت (درخصوص از جدول ارزشی) مقدار یکسان داشته باشند.

یک هم‌ارزی پرکاربرد این است:

$$P \rightarrow Q$$

$$\sim P \vee Q$$

$$\sim Q \rightarrow \sim P$$

} معادلند

فرم کلازال (بند): $\sim P \vee Q$

مثال

$p \wedge q \longrightarrow R$

معادل

$\sim(p \wedge q) \vee R$

معادل

$\sim p \vee \sim q \vee R$

(فرم کلازال)

هر گزاره مرکب با n متغیر، دارای 2^n حالت درستی (سطر در جدول) است. ارزش گزاره در هر سطر می‌تواند 0 یا 1 باشد (2 حالت)

پس تعداد کل گزاره‌های غیرهم‌ارز با n متغیر $2^{(2^n)}$ است.

p , q, r	p ∧ q	p ∨ q	p ⇒ q	p ∧ r ⇒ q	p ∨ -p	p ∧ -p
000	0	0	1	1	۱	۰
001	0	1	1	1	۱	۰
010	0	0	1	1	۱	۰
011	0	1	1	1	۱	۰
100	0	1	0	1	۱	۰
101	0	1	0	0	۱	۰
110	1	1	1	1	۱	۰
111	1	1	1	1	۱	۰

2^n

تاتولوژی $\top$

[تناقض F
ناسازگار ،
ارضاءِ ناپذیر
همواره نادرست]

مدل‌های α : مجموعه همه‌ی مدل‌هایی که در آن‌ها ارزش α، true است:

$$M(\alpha) = \{\text{تمام حالاتی که در آن‌ها ارزش } \alpha \text{ است True}\}$$

برای یک KB:

$$KB = \{S_1, \dots, S_n\}$$

$$M(KB) = M(S_1 \wedge S_2 \wedge \dots \wedge S_n) = M(S_1) \cap M(S_2) \cap \dots \cap M(S_n)$$

گزاره‌های منطقی

ارضاء پذیر: حداقل در یک مدل (در یک سطر) ارزش true دارد. $M(\alpha) \neq \{\}$

- تائولوژی: همواره درست. T — $|M(T)| = 2^n$
- ارضاءِ پذیر اما غیر تائولوژی.

ارضاءِ ناپذیر F: در هیچ سطری ارزش true ندارد. مجموعه مدل‌هایش تهی است. $M(F) = \{\}$

مثال: مجموعه‌ی مدل‌های $H: p \wedge q$ در حالی که ۳ اتم p, q, r را داشته باشیم: مجموعه‌ای دو عضوی است:

p , q, r	p ∧ q	p ∨ q	p ⇒ q	p ∧ r ⇒ q	p ∨ -p	p ∧ -p
000	0	0	1	1	۱	۰
001	0	1	1	1	۱	۰
010	0	0	1	1	۱	۰
011	0	1	1	1	۱	۰
100	0	1	0	1	۱	۰
101	0	1	0	0	۱	۰
→ 110	1	1	1	1	۱	۰
→ 111	1	1	1	1	۱	۰

H

$$M(p \wedge q) = \{ p \wedge q \wedge \sim r \ , \ p \wedge q \wedge r \}$$

توجه: استنتاج $A \models B$ درست است، اگر و تنها اگر $M(A) \subseteq M(B)$، اگر و تنها اگر گزاره‌ی شرطی $A \to B$ یک تاتولوژی باشد.

معادل هستند:

- $A \models B$ استنتاج صحیح است
- $M(A) \subseteq M(B)$ هر مدل A یک مدل B است
- $(A \to B) \cong T$ گزاره شرطی $A \to B$ تاتولوژی است.
- $|M(A \to B)| = 2^n$

$$\overset{KB}{p \wedge q} \models \overset{\alpha}{(p \wedge r \rightarrow q)}$$

تمرین : آیا این استنتاج درست است؟

پاسخ: در جدول ارزش، مجموعه مدل‌های $KB: p \wedge q$ و مدل‌های $\alpha: p \wedge r \rightarrow q$ را می‌یابیم. اگر $M(KB) \subseteq M(\alpha)$ باشد استنتاج درست است.

می بینیم که $M(p \wedge q) \subseteq M(p \wedge r \rightarrow q)$ پس استنتاج صحیح است:

$p \wedge q \models (p \wedge r \rightarrow q)$

p , q, r	p ∧ q	p ∨ q	p ⇒ q	p ∧ r ⇒ q	p ∨ -p	p ∧ -p
000	0	0	1	1	۱	۰
001	0	1	1	1	۱	۰
010	0	0	1	1	۱	۰
011	0	1	1	1	۱	۰
100	0	1	0	1	۱	۰
101	0	1	0	0	۱	۰
110	1	1	1	1	۱	۰
111	1	1	1	1	۱	۰

روشی استفاده از جدول ارزش (TT) از مرتبه‌ی نمایی است، حداکثر 2^n: $\Omega(2^n)$

تمرین تکمیلی جلسه:

$KB \models \alpha$

$(p \wedge q \rightarrow r) \models (p \rightarrow r)$ آیا این استنتاج درست است؟

پاسخ: خیر، باتوجه به جدول مجموعه مدل‌های KB زیرمجموعه مدل‌های α نیست:

p , q, r	p ∧ q	p ∨ q	p ⇒ q	p ∧ r ⇒ q	p ∨ -p	p ∧ -p
000	0	0	1	1	۱	۰
001	0	1	1	1	۱	۰
010	0	0	1	1	۱	۰
011	0	1	1	1	۱	۰
100	0	1	0	1	۱	۰
101	0	1	0	0	۱	۰
110	1	1	1	1	۱	۰
111	1	1	1	1	۱	۰

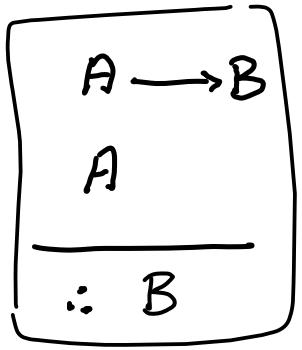
گزاره ارضاءِ ناپذیر: $M(\alpha) = \phi$

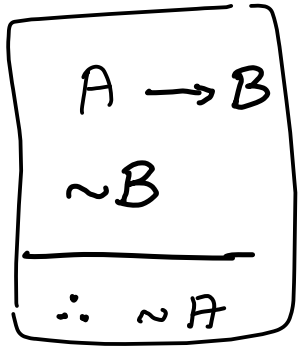
گزاره تاتولوژی (رانگر) valid: $|M(\alpha)| = 2^n$

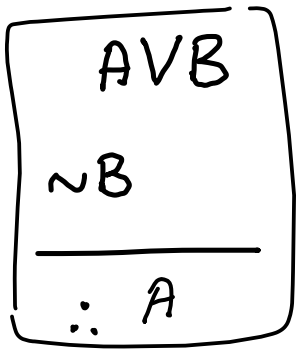
گزاره ارضاءِ پذیر: $M(\alpha) \neq \phi$

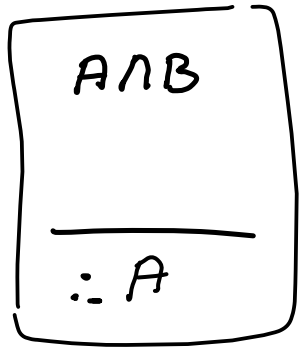
یادآوری چند قانون استنتاج طبیعی:

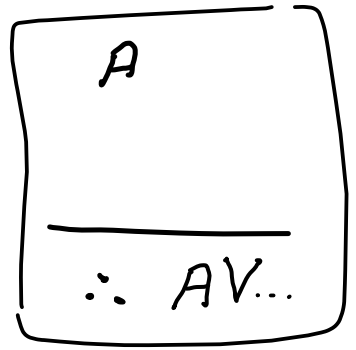
MP:











این ۳ قاعده دراصل
باهم معادل هستند زیرا
$A \to B$ همان
$\sim A \vee B$ است.

$$\frac{A \to B \quad A}{\therefore B} \qquad \frac{A \to B \quad \sim B}{\therefore \sim A} \qquad \frac{A \vee B \quad \sim B}{\therefore A}$$

مثال از کاربردِ این قواعد :

$$\frac{A \vee B \vee D \qquad \sim A}{B \vee D}$$

در ضمن می دانیم $A \vee B \vee D$ را می توان مثلاً به این فرم نوشت: $\sim A \longrightarrow B \vee D$

پس استنتاج بالا در اصل همان MP است :

$$\frac{\sim A \longrightarrow B \vee D \qquad \sim A}{B \vee D}$$

نکته پرکاربرد :

گزاره‌ی شرطی

$$A \longrightarrow \Big(B \longrightarrow (C \longrightarrow D) \Big)$$

معادل است با :

$$A \wedge B \wedge C \longrightarrow D$$

یعنی :

$$\begin{array}{l} A \\ B \\ C \\ \hline \therefore D \end{array}$$

یادآوری: استنتاج

P_1
P_2
$\vdots$
P_5

$\therefore P_6 \rightarrow P_7$

معادل است با:

P_6
P_1
P_2
$\vdots$
P_5

$\therefore P_7$

همه این‌ها معادل هستند:

$$A \rightarrow (B \rightarrow (C \rightarrow D))$$

$$\frac{A}{B \rightarrow (C \rightarrow D)}$$

$$A \wedge B \rightarrow (C \rightarrow D)$$

$$\frac{\begin{array}{l} A \\ B \end{array}}{C \rightarrow D}$$

$$A \wedge B \wedge C \rightarrow D$$

$$\frac{\begin{array}{l} A \\ B \\ C \end{array}}{\therefore D}$$

۱۳۳- در صورتی‌که بخواهیم با استفاده از روش رزولوشن (Resolution) نوع عبارت گزاره‌ای زیر را تعیین کنیم، کدام مورد در خصوص نوع این عبارت درست است؟

$$(P \Rightarrow (Q \Rightarrow R)) \Rightarrow ((P \Rightarrow Q) \Rightarrow (P \Rightarrow R))$$

۱) غیرقابل ارضاء (Unsatisfiable) است.
۲) ارضاپذیر (Satisfiable) است.
۳) نامعتبر (Invalid) است.
۴) معتبر (Valid) است.

جواب: گزاره داده شده تائولوژی است پس ارضاء پذیر هم هست. اما هیچ‌کدام جواب گزینه ۴ است.

گزاره شرطی: $(P \to (Q \to R)) \to ((P \to Q) \to (P \to R))$

معادل است با این ادعا که:

$$\frac{P \to (Q \to R) \quad (P \to Q)}{P \to R}$$

معادل است با این ادعا که:

$$\frac{P \to (Q \to R) \quad (P \to Q) \quad P}{R}$$

با کمی دقت درستی این استنتاج با mp برقرار است. پس گزاره داده شده valid، تائولوژی است.

$\alpha \longleftrightarrow \beta$ و $\sim\alpha \longleftrightarrow \sim\beta$ معادل هستند.

دودوی آنها فقط وقتی معتبر هستند که $v_\alpha = v_\beta$.

$$(\alpha \longleftrightarrow \beta)$$

$$\equiv (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$$

$$\equiv (\sim\beta \rightarrow \sim\alpha) \wedge (\sim\alpha \rightarrow \sim\beta)$$

$$\equiv (\sim\alpha \longleftrightarrow \sim\beta)$$

۹۱- فرض کنید ϕ_1 و ϕ_2 دو عبارت در منطق گزاره‌ای باشند. اگر عبارت $(\phi_1 \leftrightarrow \phi_2)$ ارضاپذیر (satisfiable) باشد، کدام مورد درست است؟

۱) عبارت $(\phi_1 \rightarrow \neg\phi_2)$ ارضاناپذیر است.

۲) عبارت $(\neg\phi_1 \leftrightarrow \neg\phi_2)$ ارضاپذیر است.

۳) عبارت‌های ϕ_1 و ϕ_2 هم‌ارز هستند.

۴) هر سه گزینه فوق صحیح هستند.

$\phi_1 \longleftrightarrow \phi_2$

$\sim\phi_1 \longleftrightarrow \sim\phi_2$

این‌ها معادل هستند

وقتی یکی ارضاپذیر باشد دیگری هم هست.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

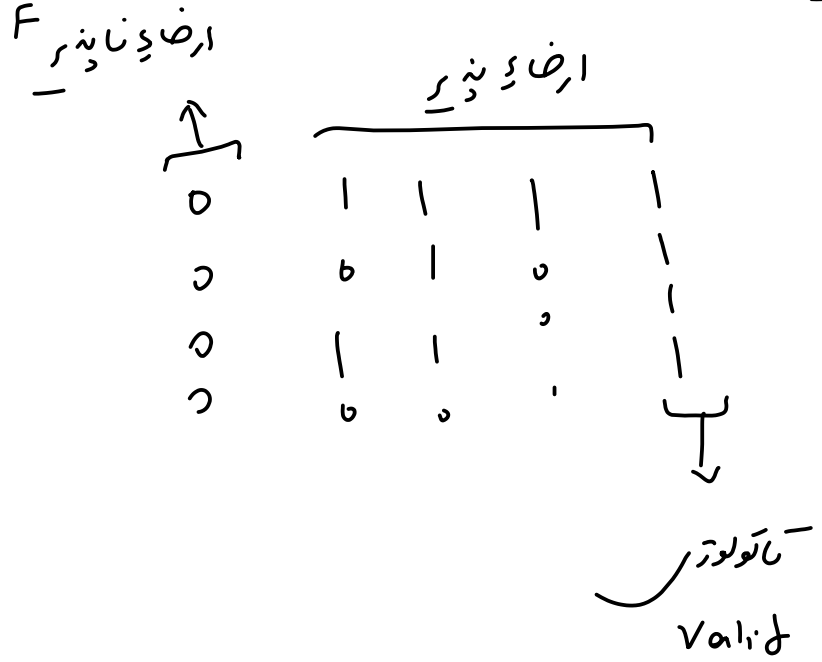
A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

ادامه جلسه ۱۰ فصل ۷

مرور بخش قبل



p	q	p ∧ q
F	F	F
F	T	F
T	F	F
T	T	T

ارضای پذیر

KB

استنتاج درست است اگر و تنها اگر $\{P_1, P_2, \ldots, P_n\} \models \alpha$

این گزاره شرطی، تاکولوژی باشد $P_1 \wedge P_2 \wedge \ldots \wedge P_n \longrightarrow \alpha$

چند هم ارزی مهم :

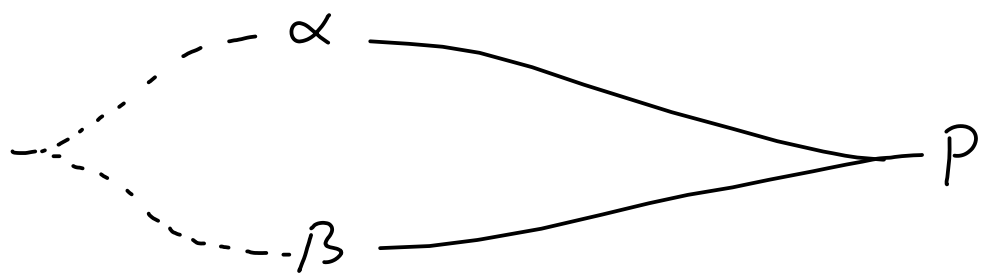
$$(P \rightarrow \alpha \wedge \beta) \equiv (P \rightarrow \alpha) \wedge (P \rightarrow \beta)$$

$$(P \rightarrow \alpha \vee \beta) \equiv (P \rightarrow \alpha) \vee (P \rightarrow \beta)$$

$$(\alpha \wedge \beta \rightarrow P) \equiv (\alpha \rightarrow P) \vee (\beta \rightarrow P)$$

$$(\alpha \vee \beta \rightarrow P) \equiv (\alpha \rightarrow P) \wedge (\beta \rightarrow P)$$

توجیه و توضیح با یک شکل:



از هم‌ارزی بررسی (α یا β) به p می‌رسیم:

$$\alpha \vee \beta \longrightarrow p$$

یعنی آن‌که: $(\alpha \to p)$ و $(\beta \to p)$.

$$\left[\alpha \vee \beta \longrightarrow p \right] \equiv \left[(\alpha \to p) \wedge (\beta \to p) \right]$$

مثال : این گزاره شرطی را به چند گزاره شرطی ساده تر تبدیل کنیم:

$$p \vee q \rightarrow R \wedge S$$

$$\equiv (p \vee q \rightarrow R) \wedge (p \vee q \rightarrow S)$$

$$\equiv (P \rightarrow R) \wedge (q \rightarrow R) \wedge (P \rightarrow S) \wedge (q \rightarrow S)$$

به وضوح می دانیم این ها درست هستند:

① ok $A \wedge B$ ارزش True داشته باشد آنگاه A ارزش True دارد و B ارزش True دارد.

② ok $A \vee B$ ارزش True داشته باشد آنگاه A ارزش True دارد یا B ارزش True دارد.

اما آیا این ها هم درست هستند؟

③ ok $A \wedge B$ تاتولوژی باشد آنگاه A تاتولوژی است و B تاتولوژی است

④ $A \vee B$ تاتولوژی باشد آنگاه A تاتولوژی است یا B تاتولوژی است

غلط است

مراقب این نتیجه‌گیری غلط باشید:

$A \vee B$ تاتولوژی باشد آنگاه A تاتولوژی است یا B تاتولوژی است.

مثلاً ممکن است جدول ارزش چنین باشد:

A	B	A∨B
۰	۱	۱
۱	۱	۱
۰	۱	۱
۱	۰	۱

آیا این ادعا درست است؟

$$(KB \models \alpha \vee \beta) \overset{?}{\Rightarrow} (KB \models \alpha) \vee (KB \models \beta)$$

پاسخ: اسنبه! با یک مثال نقض ثابت می‌شود که این ادعا غلط است. پس با بررسی دقیق‌تر علت نادرستی را ثابت می‌کنیم.

مثال

$m \in \{1, 2, 3, 4, \dots\}$

KB: { m عدد طبیعی است }

A: m زوج است

B: m فرد است

در این مثال:

$KB \models A \vee B$

درست

اما:

$KB \not\models A$

غلط

$KB \not\models B$

غلط

آیا این ادعا درست است؟

$$(KB \models \alpha \vee \beta) \overset{?}{\Rightarrow} (KB \models \alpha) \vee (KB \models \beta)$$

علت نادرستی:

طبق فرض:

$$KB \models \alpha \vee \beta$$

پس:

$$KB \rightarrow \alpha \vee \beta$$

تاتولوژی است.

پس:

$$(KB \rightarrow \alpha) \vee (KB \rightarrow \beta)$$

تاتولوژی است.

نمی‌توان نتیجه گرفت

$$(KB \rightarrow \alpha) \vee (KB \rightarrow \beta)$$

تاتولوژی است.

$$(KB \rightarrow \alpha)$$

تاتولوژی است.

$$(KB \rightarrow \beta)$$

تاتولوژی است.

ازاین‌که $(A) \vee (B)$ تاتولوژی است نمی‌توان نتیجه گرفت که یکی از آن‌ها تاتولوژی است.

تمرین: آیا این ادعا درست است؟

$$(KB \models \alpha \wedge \beta) \overset{?}{\Rightarrow} (KB \models \alpha) \wedge (KB \models \beta)$$

پاسخ: بله درست است:

طبق فرض:

$$KB \models \alpha \wedge \beta$$

پس:

$$KB \rightarrow \alpha \wedge \beta$$

تائولوژی است.

پس:

$$(KB \rightarrow \alpha) \wedge (KB \rightarrow \beta)$$

تائولوژی است.

می‌توان نتیجه گرفت

$$(KB \rightarrow \alpha) \wedge (KB \rightarrow \beta)$$

تائولوژی است.

تائولوژی است.

آیا این ادعا درست است؟

$(KB \models \alpha \vee \beta) \quad \overset{?}{\Leftarrow} \quad (KB \models \alpha) \vee (KB \models \beta)$

پاسخ: به بررسی نشان می‌دهیم درست است:

طبق فرض $(KB \models \alpha) \vee (KB \models \beta)$

پس: $(KB \rightarrow \beta)$ تاتولوژی یا $(KB \rightarrow \alpha)$ تاتولوژی است.

وقتی حداقل یکی از جملات A ، یا B تاتولوژی باشند حتماً $A \vee B$ هم تاتولوژی است

پس $(KB \rightarrow \alpha) \vee (KB \rightarrow \beta)$ تاتولوژی است

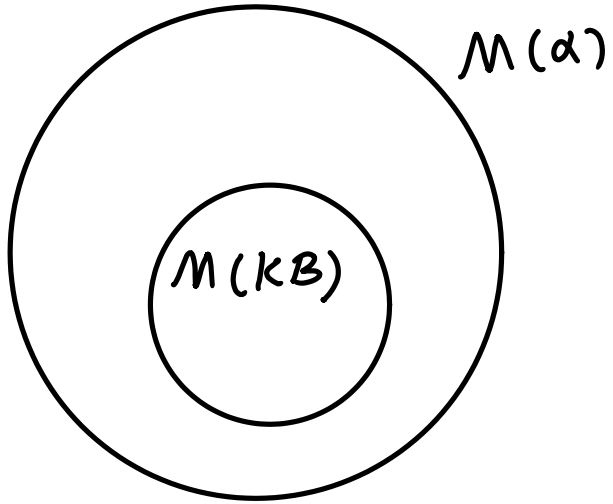
که معادل است با $KB \rightarrow \alpha \vee \beta$

پس $KB \rightarrow \alpha \vee \beta$ تاتولوژی است

پس $KB \models \alpha \vee \beta$.

روشی استفاده از نظریت مجموعه ها در بررسی این نوع سوالات :

می دانیم استنتاج $KB \models \alpha$ برقرار است اگر و تنها اگر $M(KB) \subseteq M(\alpha)$



اگر فرض کنیم

$$A = M(\alpha)$$
$$B = M(\beta)$$

آنگاه:

$$M(\alpha \wedge \beta) = A \cap B$$

$$M(\alpha \vee \beta) = A \cup B$$

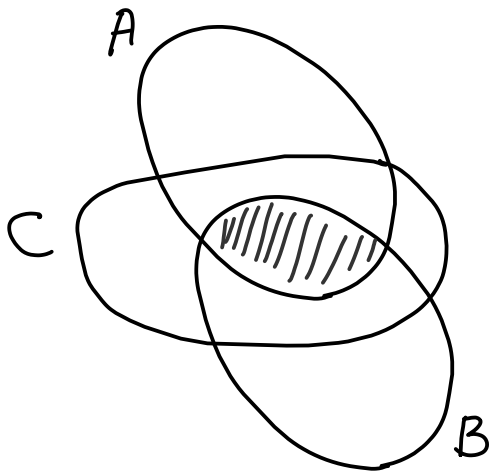
تمرین: آیا این نتیجه‌گیری درست است؟

$$(\alpha \wedge \beta \models \gamma) \overset{?}{\Longrightarrow} (\alpha \models \gamma) \vee (\beta \models \gamma)$$

پاسخ: (خیر) فرض کنیم $m(\alpha) = A$, $m(\beta) = B$, $m(\gamma) = C$ حالا: طبق فرض $\alpha \wedge \beta \models \gamma$ پس $A \cap B \subseteq C$

آیا از این که $A \cap B \subseteq C$ می‌توان نتیجه گرفت که $(A \subseteq C) \overset{\text{یا}}{\vee} (B \subseteq C)$ ؟

طبق شکل زیر نمی‌توان نتیجه گرفت.



در این شکل $A \cap B \subseteq C$

اما A و B هیچ‌کدام زیرمجموعه C نیستند.

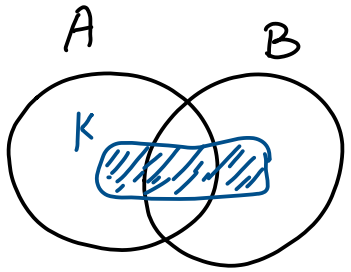
حل سوالات قبل با مجموعه‌ها

$(KB \models \alpha \vee \beta) \overset{?}{\Longrightarrow} (KB \models \alpha) \vee (KB \models \beta)$

آیا این ادعا درست است؟

ترجمه:

$K \subseteq A \cup B \overset{?}{\Longrightarrow} (K \subseteq A) \text{ یا } (K \subseteq B)$



نادرست بودن این نتیجه‌گیری طبق شکل واضح است. در این شکل $K \subseteq A \cup B$ اما زیرمجموعه هیچکدام از A و B نیست.

حل سوالات قبل با مجموعه‌ها

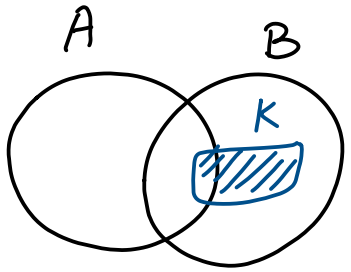
آیا این ادعا درست است؟

$$(KB \models \alpha \vee \beta) \overset{?}{\Longleftarrow} (KB \models \alpha) \vee (KB \models \beta)$$

ترجمه:

$$K \subseteq A \cup B \overset{?}{\Longleftarrow} (K \subseteq A) \text{ یا } (K \subseteq B)$$

بله درست بودن این نتیجه‌گیری واضح است.



خاصیت کنشوایی پایگاه‌های دانش منطقی:

$$\begin{cases} KB_1 \subseteq KB_2 \\ KB_1 \models \alpha \end{cases} \Rightarrow KB_2 \models \alpha$$

(پایگاه‌های دانش در حالت کلی این ویژگی را ندارند. پایگاه دانش گزاره‌های منطقی این ویژگی را دارند.)

$KB_1 \subseteq KB_2$

خاصیت کنشوایی یعنی آن که اگر KB_1 جمله‌ای را نتیجه دهد، KB_2 هم آن را نتیجه می‌دهد.

در پایگاه‌ای دانش منطقی : (منطق‌گزاره‌ای)

if $KB_1 \subseteq KB_2$ then

if $KB_1 \models \alpha$ $\Rightarrow$ $KB_2 \models \alpha$

(اما در حالت کلی چنین نیست)

مثلاً در پایگاه‌های دانش مبتنی بر احتمال، مبتنی بر منطق فازی، این ویژگی برقرار نیست.

← دیسی ممکن است خاصیت کلنوایی را نداشته باشد.

۱۸ـ اگر KB_1 یک پایگاه دانش دلخواه و b یک جمله دلخواه باشد. با فرض اینکه افزودن b به پایگاه KB_1 منجر به ایجاد پایگاه KB_2 می‌شود، اگر بدانیم جمله a از KB_1 قابل نتیجه‌گیری است، کدام‌یک از گزینه‌های زیر درست است؟ (مهندسی کامپیوتر ـ دولتی ۹۱ و مهندسی IT ـ دولتی ۹۳)

۱) $a \wedge b$ از KB_1 قابل نتیجه‌گیری است.

۲) a از KB_2 قابل نتیجه‌گیری است.

۳) $a \wedge b$ از KB_2 قابل نتیجه‌گیری است.

۴) $a \vee b$ هم از KB_1 و هم از KB_2 قابل نتیجه‌گیری است. ✓

از صورت سوال می‌توان اطمینان داشت که: $KB_1 \models a$.

همچنین b عضو KB_2 است پس $KB_2 \models b$

اما نمی‌توان با اطمینان گفت که KB_2 هم a را می‌دهد. چون خاصیت کلنوایی‌اش برقرار نباشد.

حالا می‌دانیم که چه از a و چه از b می‌توان دربتی $a \vee b$ را نتیجه گرفت.

نکاتی درباره شمارش تعداد مدل‌ها:

اولاً تعداد کل اتم‌ها (متغیرهای منطقی) در تعداد مدل‌هایی که برمی‌گزارد اثر می‌گذارد. حواستان که در آن جمله ظاهر نشده باشد

تعداد مدل‌ها را دو برابر می‌کند.

تعداد مدل های $p \vee q$:

در اولین جدول ۳ و در دومین جدول ۶ است.

جدول دوم البته ۲ را هم دارد.

جدول 2

r	p	q	$p \vee q$
۰	۰	۰	۰
۰	۰	۱	۱
۰	۱	۰	۱
۰	۱	۱	۱
۱	۰	۰	۰
۱	۰	۱	۱
۱	۱	۰	۱
۱	۱	۱	۱

جدول 1

p	q	$p \vee q$
۰	۰	۰
۰	۱	۱
۱	۰	۱
۱	۱	۱

اگر n اتم داشته باشیم:

$$|m(p_1 \vee p_2 \vee \dots \vee p_n)| = 2^n - 1$$

$$|m(p_1 \wedge p_2 \wedge \dots \wedge p_n)| = 1$$

نکته: به ازای هر اتم اضافی که در ترکیب دیده نشده باشد، جواب ۲ برابر می‌شود.

اگر K تا متغیر در گزاره غایب باشند در 2^K ضرب می‌شود.

تعداد مدل‌های $A \to B$ و $A \vee B$, $A \wedge B$ وقتی A و B جمله‌ی مرکب هستند.

قانون 1 طوری بازنویسی کنید که متغیر تکراری نداشته باشیم هر متغیر حداکثر یک بار ظاهر شود.

قانون 2 اگر K اتم غایب هستند تعداد مدل‌ها را در 2^K ضرب کنید.

قانون 3

$$\left|M(A \to B)\right| = M(A)M(B) + (2^n - M(A))\,2^m$$

متغیرهای متمایز $\begin{cases} A : n \text{ متغیره} \\ B : m \text{ متغیره} \end{cases}$

۹۳- برای سه جمله داده شده $\{(A\Leftrightarrow B\Leftrightarrow C),(D\vee C),(A\wedge B)\vee(B\wedge C)\}$ و با در نظر گرفتن مقـادیر ممکـن برای چهار متغیر گزاره‌ای $\{A,B,C,D\}$، (به ترتیب از چپ به راست) چند مدل (Model) وجود دارد؟

۱) $\{۶،۳،۸\}$

۲) $\{۶،۱۲،۴\}$

۳) $\{۱۲،۳،۸\}$

۴) $\{۸،۱۲،۴\}$

پاسخ: گزینه ۲

حل:

$$(A \wedge B) \vee (B \wedge C) \equiv B \wedge (A \vee C)$$

B: یک متغیره، در یک حالت درست است

$|M(B)| = 1$

$(A \vee C)$: ترکیب OR دو متغیره

$|M(A \vee C)| = 2^2 - 1 = 3$

$$M(B \wedge (A \vee C)) = (1)(3) \times 2^1 = 6$$

2^1: یک متغیر D غایبات

$$|M(DVC)| = (2^2-1)\cdot 2^2 = 12$$

(2^2-1): ترکیب OR با دوتا

2^2: دو متغیر غایب
A و B

$$A \leftrightarrow B \leftrightarrow C$$

$(A \leftrightarrow B) \wedge (B \leftrightarrow C)$ درواقع منظورش آن است که

اما خلاصه نوشته است:

$$A \leftrightarrow B \leftrightarrow C$$

برای درستی آن، A و B و C یا باید هر ۳ true باشند یا باید هر ۳ false باشند

پس ۲ حالت دارد اما یک متغیر غایب D داردیس:

$$\text{جواب} = ۲ \times ۲^{۱} = ۴$$

A	B	C	D
۰	۰	۰	?
۱	۱	۱	?

تمرین تعداد مدل های $H: (A \to \sim B) \wedge (A \to D)$ وقتی ۴ متغیر $\{A, B, C, D\}$ داریم

چیزی است؟

پاسخ: بازنویسی تا حواتم حداکثر یک بار دیده شود.

$$H: \quad A \to (\sim B \wedge D)$$

A: $n=1$ اتم

$(\sim B \wedge D)$: $m=2$ اتم

$$m(A)\,m(\sim B \wedge D) + (2^n - m(A))\,2^m = (1)(1) + (1)(2^2) = 5$$

یک اتم غایب هم داریم:

$$\text{جواب} = 5 \times 2^1 = 10$$

اللایه تکمیلی: جواب ۱۰ به دست آمده

درمثال قبل این ۱۰ مدل

هستند. دراین ۱۰ مدل

$A \rightarrow \sim B \wedge D$

صادق است.

A	B	D	C
۱	۰	۱	۱ یا ۰
۰	۰	۰	۱ یا ۰
۰	۰	۱	۱ یا ۰
۰	۱	۰	۱ یا ۰
۰	۱	۱	۱ یا ۰

جنبری

n اتم m اتم

$$A(P_1, \cdots, P_n) \longrightarrow B(P_{n+1}, \cdots, P_{n+m})$$

غایب

$$\left[m(A)\, m(B) + (2^n - m(A))\, 2^m \right] * 2^K$$

دوستی:

$A(P_1, \cdots, P_n) \longleftrightarrow B(P_{n+1}, \cdots, P_{n+m})$

تعداد n تایی — تعداد m تایی

$$\left[m(A)\, m(B) + (2^{n} - m(A))(2^{m} - m(B)) \right] * 2^{K}$$

غایب

حالت خاصی از دو شرطی: $m=1$

$m=1$ ام n ام

$$A(P_1, \dots, P_n) \longleftrightarrow B(P_{n+1})$$

اگر در ترکیب $\longleftrightarrow$ در یک سمت $m=1$ ام و در سمت دیگر n ام

داشتیم:

$$\text{تعداد حالت‌ها} = 2^n * 2^{k^{\text{غایب}}}$$

• با توجه به این‌که هر مدل (Model) برای یک گزاره، عبارت است از یک تفسیر (interpretation) از آن گزاره، که به آن گزاره ارزش "درست" (true) می‌دهد، عبارت $\Big(\big(\big(\big((A \Rightarrow B) \wedge C\big) \Leftrightarrow D\big) \vee E\Big) \Leftrightarrow F\Big)$ برای مجموعه متغیرهای گزاره‌ای $\{A, B, C, D, E, F\}$ چند مدل دارد؟

۱) ۶۳

۲) ۳۲ ✓

۳) ۱۶

۴) ۲

جواب $= 2^5 * 2^0 = 2^5 = 32$

اتم غایب نداریم.

یک متغیر F $\Leftrightarrow$ (ترکیبی از ۵ متغیر)

گزاره ای مانند $P \leftrightarrow Q \leftrightarrow R$ یک خلاصه نویسی غیر رسمی از جمله زیر است:

$$(P \leftrightarrow Q) \wedge (Q \leftrightarrow R)$$

این جمله وقتی درست است یا همه ی عبارات P و Q و R ارزش 1 داشته باشند یا همه ارزش 0 داشته باشند پس:

$$M(P \leftrightarrow Q \leftrightarrow R) = M_P M_Q M_R + (2^k - M_P)(2^m - M_Q)(2^l - M_R)$$

P: k اتم
Q: m اتم
R: l اتم

پس اتم های غایب را هم در نظر بگیرید.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۱۱ فصل ۷

* هر گزاره را می‌توان در فرم CNF و DNF نوشت.

DNF : مجموعی از حاصلضرب‌ها

DNF: $(\sim P \wedge \sim Q \wedge \sim R) \vee (\sim P \wedge \sim Q) \vee (R \wedge P) \vee (\sim R)$

CNF حاصلضربی از حاصل جمع‌ها

CNF: $(\sim P \vee R \vee Q) \wedge (\sim R \vee P) \wedge (\sim Q \vee P)$

(اگر در هر دو ترم‌ها حاضر باشند آنها را PDNF و PCNF می‌نامند.)

یادآوری: با استفاده از جدول ارزش می‌توان فرم‌های PDNF و PCNF را نوشت.

مثال: فرم‌های نرمال گزاره H را بنویسید: $H: P_1 \leftrightarrow P_2$

	P_1, P_2	$H: P_1 \leftrightarrow P_2$
۰	۰ ۰	۱
۱	۰ ۱	۰
۲	۱ ۰	۰
۳	۱ ۱	۱

PDNF: سطرهایی که ارزش H، 1 شده است:

$$H \equiv \underbrace{(\sim P_1 \wedge \sim P_2)}_{\text{رخ دادن سطر صفر}} \vee \underbrace{(P_1 \wedge P_2)}_{\text{رخ دادن سطر ۳}} = \sum (\underbrace{۰}_{\text{سطر}}, \underbrace{۳}_{\text{سطر}})$$

PCNF: نقیض سطرهایی که ارزش H در آنها ۰ شده است:

$$H \equiv \underbrace{(P_1 \vee \sim P_2)}_{\text{رخ ندادن سطر ۲}} \wedge \underbrace{(\sim P_1 \vee P_2)}_{\text{و رخ ندادن سطر ۱}} = \prod (\underbrace{۱}_{\text{سطر}}, \underbrace{۲}_{\text{سطر}})$$

	P_1, P_2	$H: P_1 \leftrightarrow P_2$
۰	۰ ۰	۱
۱	۰ ۱	۰
۲	۱ ۰	۰
۳	۱ ۱	۱

طبق جدول ارزش: برای آنکه H نخ دهد باید سطر صفرم یا سطر سوم رخ دهند

یعنی $(\sim P_1 \wedge \sim P_2)$ یا $(P_1 \wedge P_2)$

یعنی: $(\sim P_1 \wedge \sim P_2) \vee (P_1 \wedge P_2)$:PDNF

	P_1, P_2	$H: P_1 \leftrightarrow P_2$
۰	۰ ۰	۱
۱	۰ ۱	۰
۲	۱ ۰	۰
۳	۱ ۱	۱

طبق جدول ارزش: برای آنکه H نخ دهد باید سطرهای یک و دو نخ ندهند.

یعنی $\sim(\sim P_1 \wedge P_2)$ و $\sim(P_1 \wedge \sim P_2)$

یعنی: $(\sim P_1 \vee P_2) \wedge (P_1 \vee \sim P_2)$:PCNF

مثال: فرم‌های نرمال گزاره H را بنویسید:

$$H: P \to Q \wedge \sim R$$

	P	Q	R	$P \to Q \wedge \sim R$
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

DNF:

$$H \equiv 0 + 1 + 2 + 3 + 4 = \sum(0,1,2,3,4)$$

$$\equiv (\sim P \wedge \sim Q \wedge \sim R) \vee (\sim P \wedge \sim Q \wedge R) \vee (\sim P \wedge Q \wedge \sim R) \vee \dots \vee (P \wedge Q \wedge \sim R)$$

CNF:

حاصلضرب $\prod(5,6,7)$

$$H \equiv (\sim P \vee Q \vee R) \wedge (\sim P \vee Q \vee \sim R) \wedge (\sim P \vee \sim Q \vee \sim R)$$

روش دیگر نوشتن فرم CNF (مجموعه‌ای از نه!) استفاده از هم‌ارزی‌های منطق است.

ترتیب اقدامات برای نوشتن در فرم نرمال عطفی CNF (فرم کلازال)

① تبدیل $A \longleftrightarrow B$ به $(A \to B) \wedge (B \to A)$

② تبدیل $A \to B$ به $\sim A \vee B$.

③ دمورگان. مثلاً $\sim(p \vee q) \equiv \sim p \wedge \sim q$

④ توزیع $\vee$ روی $\wedge$.

$$(p \wedge q) \vee R \equiv (p \vee R) \wedge (q \vee R)$$

مثال: برای $p \leftrightarrow R \vee Q$ فرم کلازال (مجموعه‌ای از بندها) بنویسید.

پاسخ:

$$\equiv (p \rightarrow R \vee Q) \wedge (R \vee Q \rightarrow p)$$

$$\equiv [\sim p \vee (R \vee Q)] \wedge [\sim (R \vee Q) \vee p]$$

$$\equiv (\sim p \vee R \vee Q) \wedge (\sim R \wedge \sim Q) \vee p$$

$$\equiv \underset{①}{(\sim p \vee R \vee Q)} \wedge \underset{②}{(\sim R \vee p)} \wedge \underset{③}{(\sim Q \vee p)} \equiv \{①, ②, ③\}$$

مثال: فرم CNF را برای گزاره‌ی $H: P \to Q \wedge \sim R$ با این روش بنویسیم:

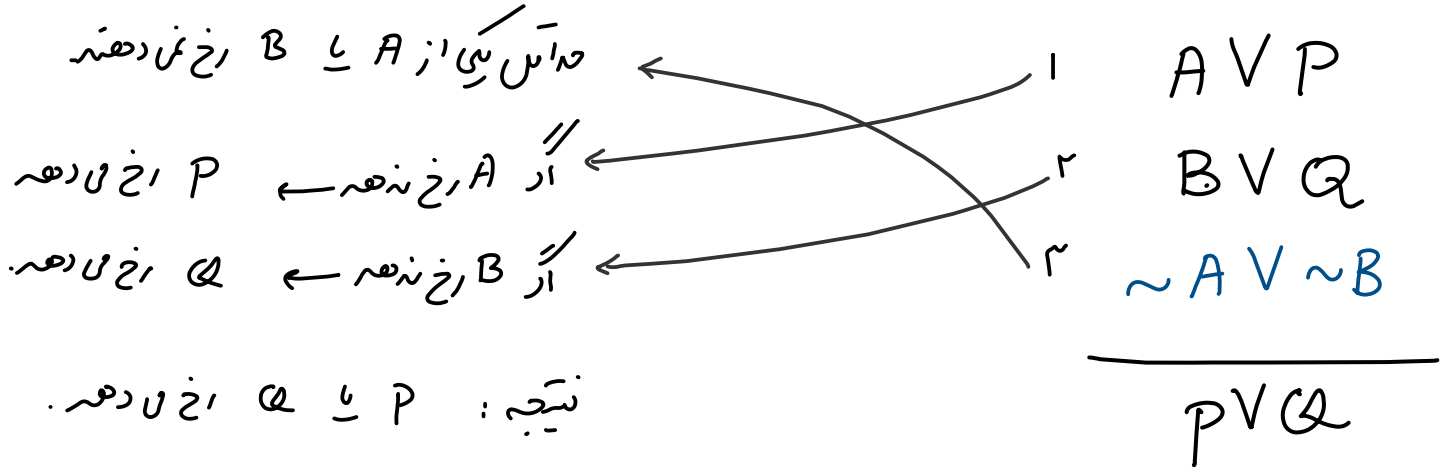
$$H: P \to Q \wedge \sim R$$

$$\simeq \sim P \vee (Q \wedge \sim R)$$

$$\simeq \underbrace{(\sim P \vee Q)}_{\text{بند ①}} \wedge \underbrace{(\sim P \vee \sim R)}_{\text{بند ②}} \quad = \{ \sim p \vee Q \ , \ \sim p \vee \sim R \}$$

دقت کنید: همین مثال را با جدول حل کردیم، به فرم PCNF نوشته شد.
با هم ارزها به فرم CNF نوشته شدن به نوشتن به CNF ممکن است به چند حالت قابل انجام باشد.

مفهوم قاعده رزولوشن:



در حالتی که B همان ~A باشد فرض سوم خود بخود برقرار است.

به حالت معروف کَ رزولوشن می‌رسیم:

رزولوشن: یک جفت نه داریم که در یکی P و در دیگری $\sim P$ قرار دارد. آنها قابل ادغام (ترکیب-تحلیل) هستند:

$$\frac{A \vee P \qquad \sim A \vee Q}{P \vee Q}$$

$$\frac{p \vee \sim l \vee m \qquad \sim p \vee k \vee q}{\sim l \vee m \vee k \vee q}$$

l_i و P_j نقیض هم باشند:

$$l_1 \vee l_2 \vee \dots \vee l_i \vee \dots \vee l_m$$

$$P_1 \vee P_2 \vee \dots \vee P_j \vee \dots \vee P_n$$

$$l_1 \vee l_2 \vee \dots l_{i-1} \vee l_{i+1} \vee \dots \vee l_m \vee P_1 \vee P_2 \vee \dots P_{j-1} \vee P_{j+1} \vee \dots \vee P_n$$

نتیجی که با حذف l_i و P_j و تجمیع دو بند قبل به دست می‌آید.

احه‌ای غلط : نباید در یک ترکیب ، دو حقت اتم نقیض را حذف کنیم.

$p \vee q \vee R \qquad \sim p \vee \sim q \vee K$

$R \vee K$ نتیجه غلط:

اثبات غلط بودن:

فرض کنید:

$R = 0$
$K = 0$
$P = 1$
$q = 0$

$$\{ p \vee q \vee R \;,\; \sim p \vee \sim q \vee K \} \overset{?}{\models} R \vee K$$

$1 \; 0 \; 0 \quad 0 \; 1 \; 0 \quad 0 \; 0$

$1 \qquad 1 \qquad 0$

برهان خلف

معنای برهان خلف آن است که برای اثبات $KB \models \alpha$

موقتاً $\neg\alpha$ را به پایگاه دانش اضافه می‌کنیم $KB \wedge \neg\alpha$ منجر به تناقض می‌شود.

$$KB \models \alpha \longleftrightarrow KB \wedge \neg\alpha \models F$$

در الگوریتم رزولوشن از برهان خلف استفاده می‌شود.

A resolution algorithm

function PL-RESOLUTION(KB, α) **returns** *true* or *false*

inputs: KB, the knowledge base, a sentence in propositional logic
α, the query, a sentence in propositional logic

$clauses \leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
$new \leftarrow \{\,\}$
loop do
  for each pair of clauses C_i, C_j **in** $clauses$ **do**
    $resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)
    if $resolvents$ contains the empty clause **then return** *true*
    $new \leftarrow new \cup resolvents$
  if $new \subseteq clauses$ **then return** *false*
  $clauses \leftarrow clauses \cup new$

الگوریتم رزولوشن:

ورودی: KB و α

فرم: KB و $\neg\alpha$ باید CNF باشند.

(هر جمله‌ی مرکب دارای فرم CNF هستند)

اجرا: روی $KB \wedge \neg\alpha$ شروع به تحلیل می‌کنیم.

در خاتمه:
- نتیجه‌ی به دست آمد: $KB \models \alpha$
- نتیجه‌ها تکراری شوند: $KB \not\models \alpha$

الگوریتم حتماً متوقف می‌شود و کامل است.

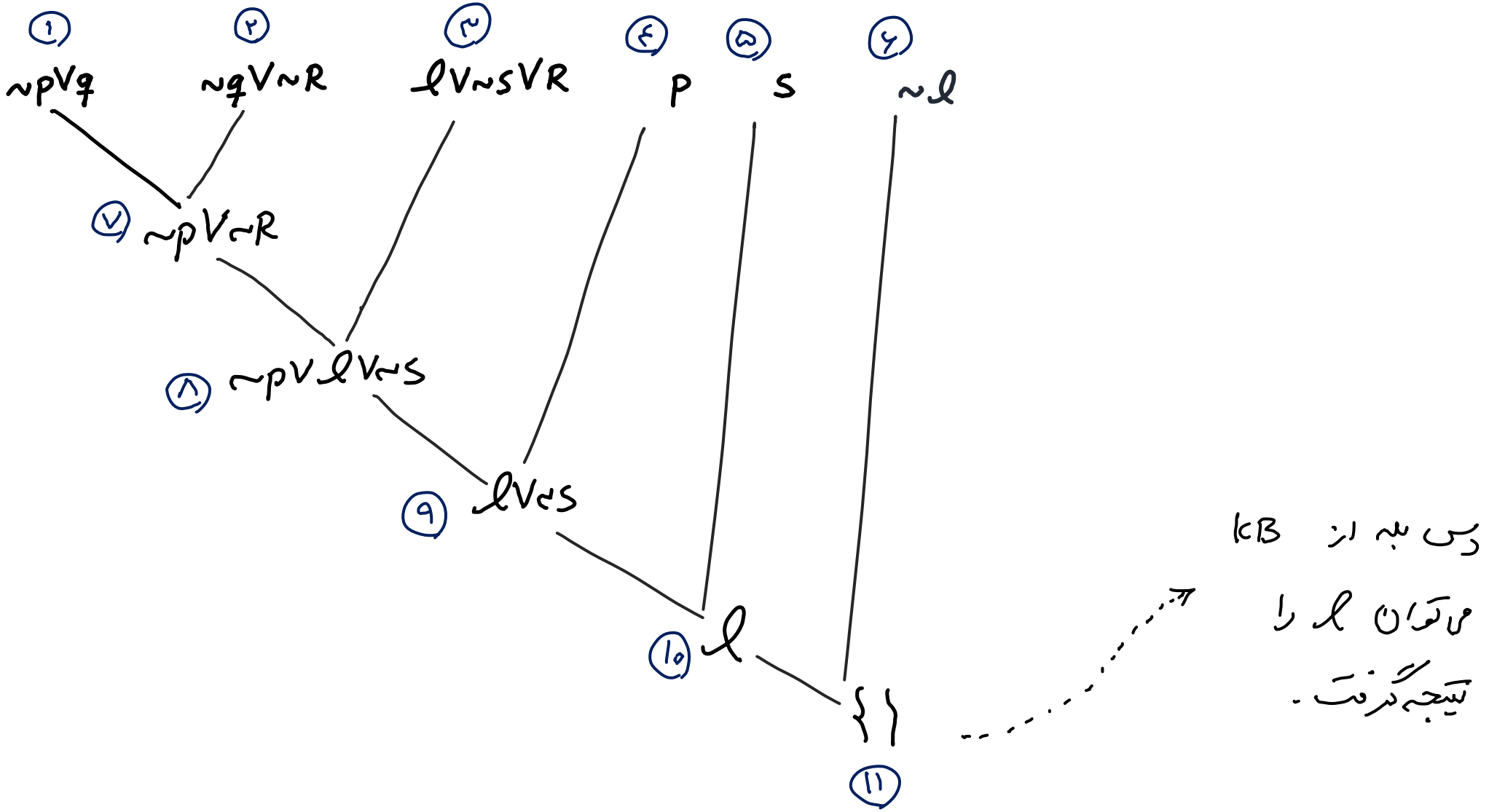
مثال: آیا استنتاج درست است؟

$$\{P \to q \ , \ \neg q \vee \neg R \ , \ \neg l \wedge S \to R \ , \ P \wedge S\} \stackrel{?}{\models} l$$

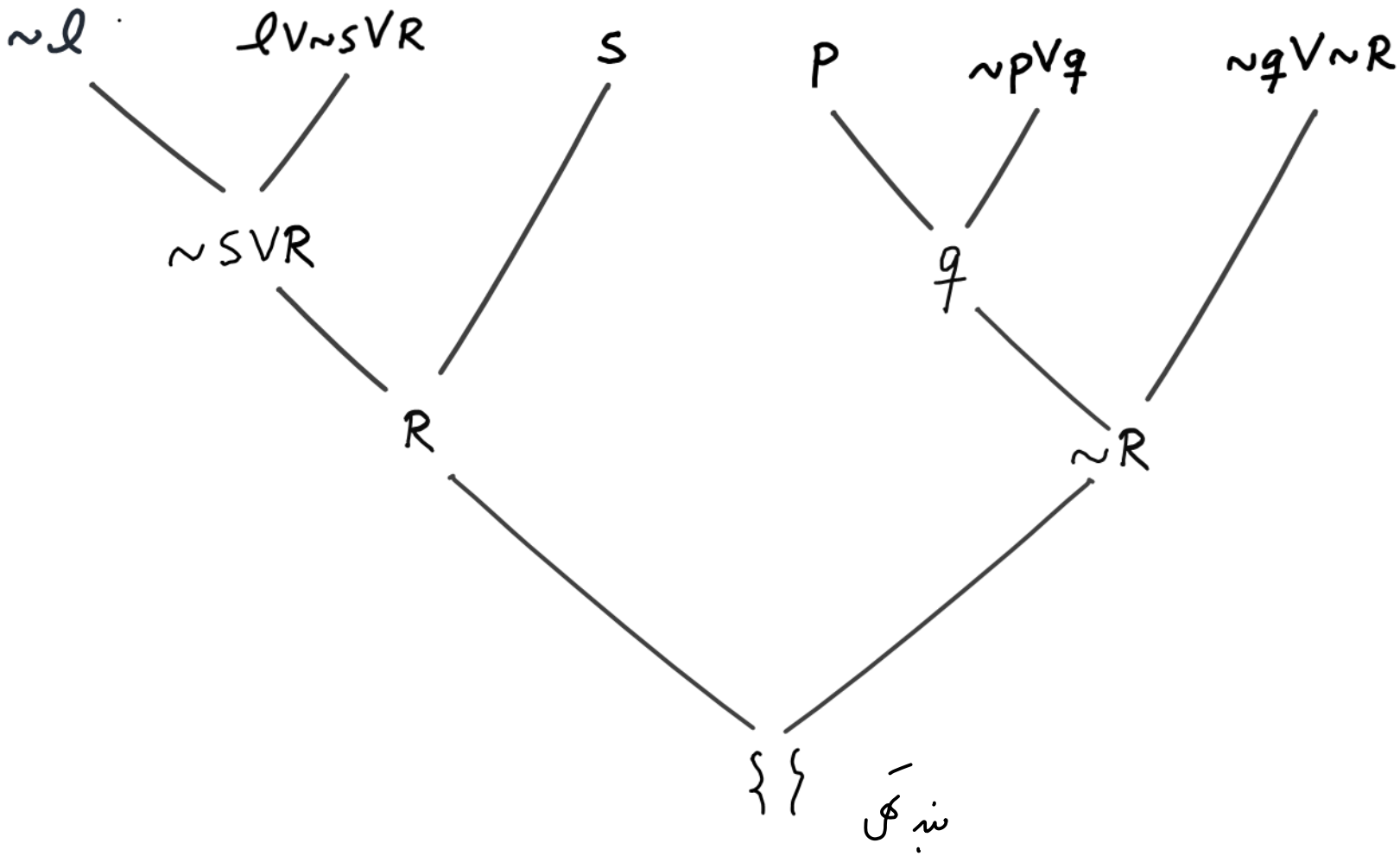
ابتدا $\neg l$ و KB هم به صورت CNF ، مجموعه‌ای از بندها نوشته شوند.

$$\neg P \vee q \ , \ \neg q \vee \neg R \ , \ l \vee \neg S \vee R \ , \ P \ , \ S \ , \ \neg l$$

اکنون سئوال این است که آیا از $KB \wedge \neg l$ به تناقض می‌رسیم یا خیر؟


۱ ~pVq
۲ ~qV~R
۳ qV~sVR
۴ p
۵ s
۶ ~q
۷ ~pV~R
۸ ~pVqV~s
۹ qV~s
۱۰ q
۱۱ {}
چس نمی‌ناله از KB
می‌توان q را
نتیجه گرفت.

درهمین مثال می توانیم این طور عمل کنیم



مثال آیا این استنتاج درست است یعنی α از KB استنتاج می‌شود یا خیر؟

$$\{\sim p \vee q \vee \sim R \;,\; p \leftrightarrow R \;,\; q \rightarrow S \;,\; \sim q\} \models \underbrace{\sim S \wedge \sim q}_{\alpha}$$

ابتدا $\sim\alpha = S \vee q$ که یک نبداست را به KB اضافه کنیم.

سپس می‌بینیم که با ادامه‌ی تکمیل به جایی می‌رسیم که نبدهای به دست نمی‌آید و

هیچ نبدیدی هم به دست نمی‌آید: $new = \{\}$ [Fixed point نقطه‌ی ثابت]

پس $KB \not\models \alpha$.

۱	۲	۳	۴	۵	۶	۷
$\sim p \vee q \vee \sim R$	$p \vee \sim R$	$\sim p \vee R$	$\sim q \vee s$	$\sim q$	$s \vee q$	s

چند مرحله از اجرا را در اینجا نشان داده‌ایم:

۵, ۶ ⟶ s ۷

۵, ۱ ⟶ $\sim p \vee \sim R$ ۸

۱, ۲ ⟶ $q \vee \sim R$ ۹

۲, ۸ ⟶ $\sim R$ ۱۰

۲, ۳ ⟶ $R \vee \sim R$ ۱۱

۴, ۹ ⟶ $\sim R \vee s$ ۱۲

۳, ۱۰ ⟶ $\sim p$ ۱۳

با ادامه‌ی این الگوریتم به مرحله‌ای می‌رسیم که در آن new نداریم یعنی هیچ جمله‌ی جدیدی تولید نمی‌شود و نتیجه هم به دست نیامده پس الگوریتم متوقف شده و false را برمی‌گرداند یعنی $KB \not\models \alpha$.

بستار رزولوشن یک مجموعه از نده‌ها:

$$S = \{ ① \quad ② \quad \dots \quad ⑥ \}$$

$$RC(S) = \{ ① \quad ② \quad \dots \quad ⑥ \quad ⑦ \quad ⑧ \quad \dots \quad ⑬ \quad \dots \}$$

مجموعه همه‌ی نده‌ایی که با رزولوشن می‌توان از S به وجود آورد را بستار رزولوشن S گوئیم.

Completeness of resolution

کامل بودن رزولوشن

resolution closure $RC(S)$

If a set of clauses is unsatisfiable, then the resolution closure of those clauses contains the empty clause.

اگر یک مجموعه از نبدها (مثلاً برای ما $KB \wedge \neg\alpha$) ناسازگار و ارضاءناپذیر باشد، حتماً با افزودن نبدهای حدید طبق رزولوشن در نهایت نبد تهی وارد تبار خواهد شد.

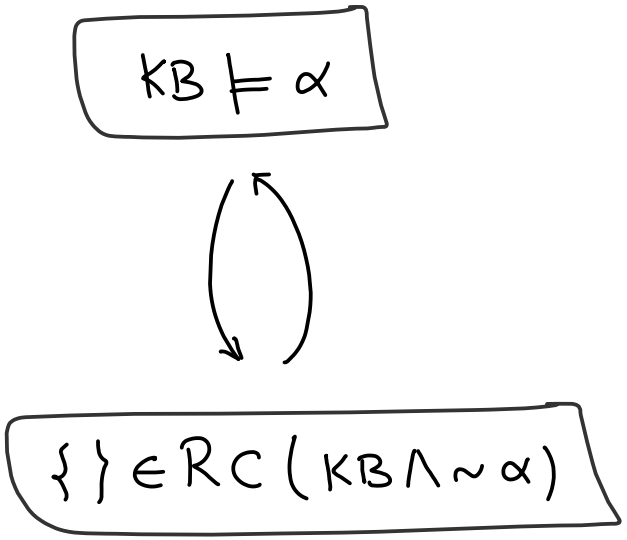
پس: اگر α از KB قابل استنتاج باشد، حتماً الگوریتم رزولوشن می‌تواند این استنتاج را انجام دهد.

کامل بودن الگوریتم رزولوشن:

α از KB استنتاج می‌شود

اگر و تنها اگر:

$KB \wedge \sim\alpha$ به تهی در بستار رزولوشن قرار داشته باشد.



دو اصطلاح

Input Resolution:

رزولوشن ورودی: در هر مرحله حداقل یکی از بندها جزء بندهای ورودی است. یعنی جزء $KB \wedge \neg\alpha$.

Linear Resolution:

رزولوشن خطی: در هر مرحله، حداقل یکی از بندها، جزء بندهای ورودی است یا آن که یکی از بندها از اجداد دیگری است.

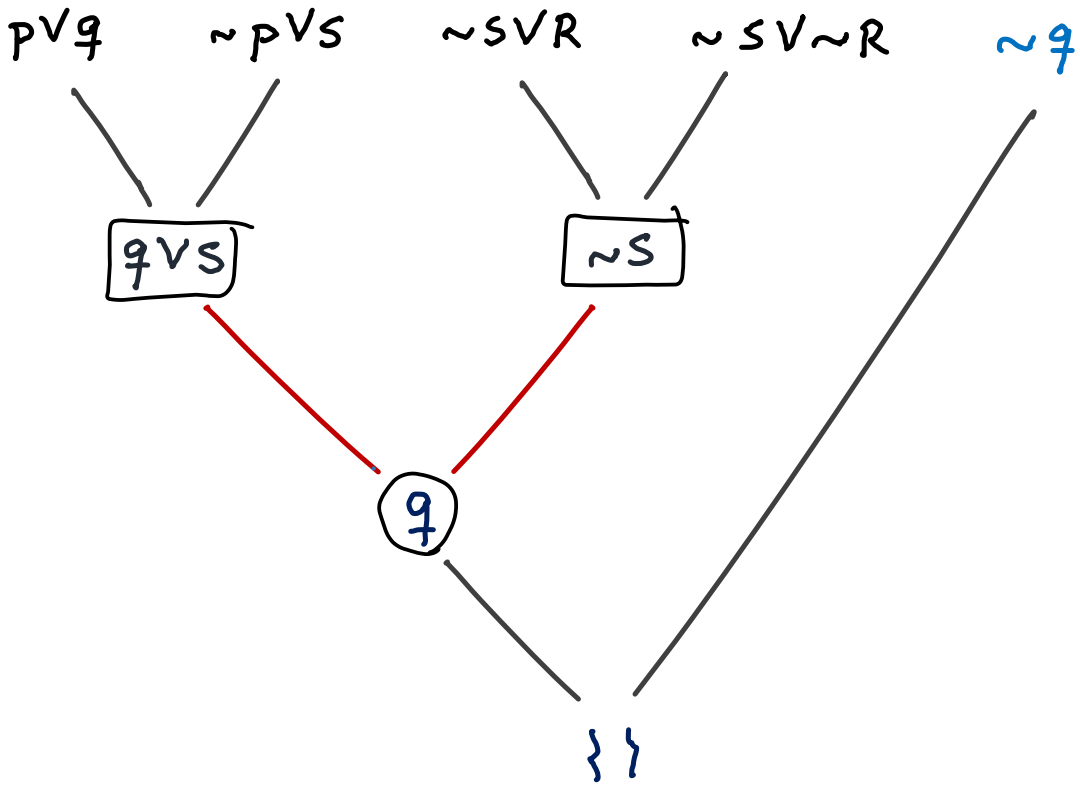
* رزولوشن‌های وروری و خطی ، هم قدرت با همان رزولوشن هستند یعنی اگر α قابل استنتاج باشه

حتماً α را استنتاج می‌کنند.

مثال:

$$\{ \sim p \to q \ , \ p \to s \ , \ s \to R \ , \ s \to \sim R \} \overset{?}{\models} q$$

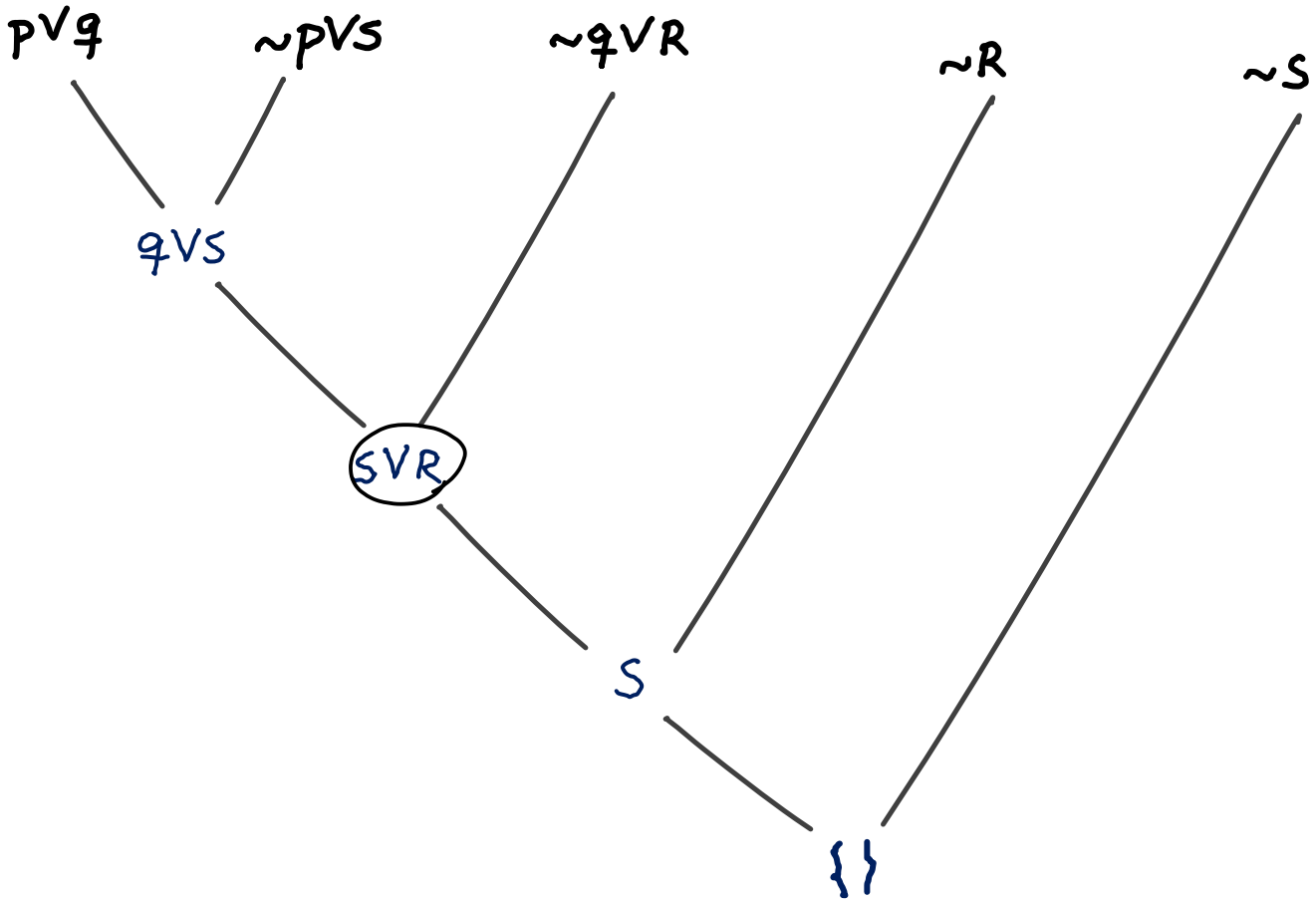
این مثال از هیچ کدام از انواع ورودی و خطی نیست:



مثال رزولوشن ورودی

Input Resolution:

$$\{ \sim p \to q \ , \ p \to s \ , \ q \to R \} \stackrel{?}{\models} s \vee R$$

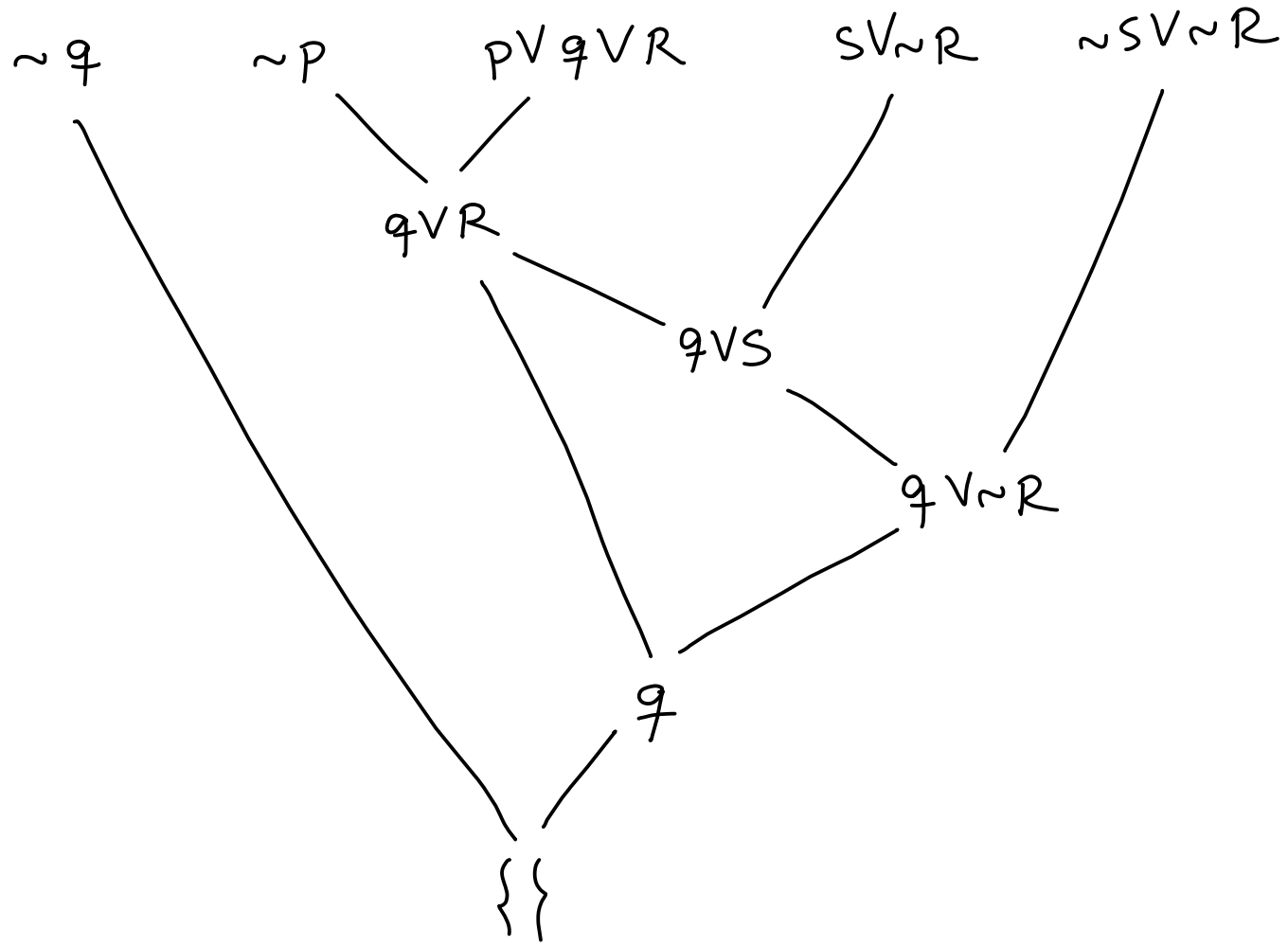


هر رزولوشن ورودی، رزولوشن خطی هم محسوب می‌شود.

مثال رزولوشن خطی

$\{ \sim P \ , \ P \vee q \vee R \ , \ s \vee \sim R \ , \ \sim s \vee \sim R \} \overset{?}{\models} q$

Linear Resolution :



ازمربته چنه جمله‌ای $(KB \stackrel{?}{\models} \alpha)$ این‌که درحالت کلی آیا ازولوشن و به طور کلی ساده‌ی استنتاج (این‌که درحالت کلی آیا

نیست. (ازرده‌ی $\underline{P}$ نیست). اما در مورد فرم‌های خاص مانند فرم ماکس، در الگوریتمی که در بخش

بعدی مطالعه می‌کنیم ممکن است زمان اجرا چنه جمله‌ای شود.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

ادامه جلسه ۱۱ فصل ۷

(clause) (بند) : ترکیب فصلی یک یا چند لیترال :

P , $\sim A$, $A \vee \sim B$, $\sim p_1 \vee p_2 \vee p_3$

معین $\begin{cases} P_1 \\ \sim P_1 \vee P_2 \\ \sim P_1 \vee \sim P_2 \vee P_3 \end{cases}$

$\begin{cases} \sim P_1 \\ \sim P_1 \vee \sim P_2 \\ \sim P_1 \vee \sim P_2 \vee \sim P_3 \end{cases}$ Horn

بند معین (definite clauses): بندی که دقیقاً یک لیترال مثبت دارد.

بند هورن (Horn clauses): بندی که حداکثر یک لیترال مثبت دارد.

فرم معین (definite form): گزاره‌ای که بتوان آن را به صورت یک بند معین، نوشت.

فرم هورن (Horn form): گزاره‌ای که بتوان آن را به صورت یک بند هورن، نوشت.

Horn clauses ,definite clauses,Horn forms ,definite forms

گزاره	بند	هورن	معین
P_1	P_1	✓	✓
$\sim P_1$	$\sim P_1$	✓	
$P_1 \rightarrow \sim P_2$	$\sim P_1 \vee \sim P_2$	✓	
$P_1 \rightarrow P_2$	$\sim P_1 \vee P_2$	✓	✓
$P_1 \wedge P_2 \rightarrow P_3$	$\sim P_1 \vee \sim P_2 \vee P_3$	✓	✓
$P_3 \rightarrow P_1 \vee P_2$	$P_1 \vee P_2 \vee \sim P_3$		

معادلند:

$A \rightarrow B$

$\sim A \vee B$

جملات به فرم طبیعی
و معادل های آنها

معادل هستند:

$\sim P_1 \vee \sim P_2 \vee P_3$

$P_1 \wedge P_2 \rightarrow P_3$ } معروف ترانت

$P_1 \rightarrow \sim P_2 \vee P_3$

$P_2 \rightarrow \sim P_1 \vee P_3$

$\sim P_3 \rightarrow \sim P_1 \vee \sim P_2$

معادل:

P_1

$\top \rightarrow P_1$

فرم معروف گزاره شرطی طبیعی:

$$l_1 \wedge l_2 \wedge \dots \wedge l_k \rightarrow l_{k+1}$$

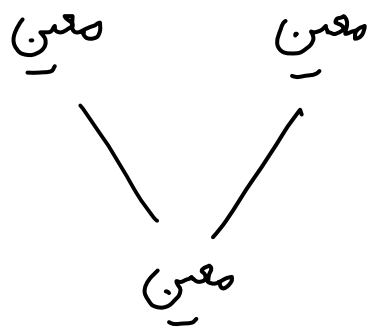
مثال:

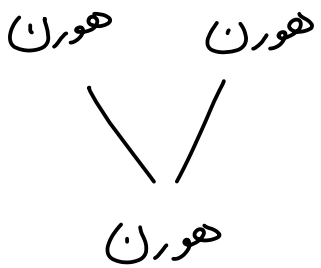
$$\begin{cases} p_1 \wedge p_2 \wedge p_3 \wedge p_4 \to p_5 \\ \sim p_1 \vee \sim p_2 \vee \sim p_3 \vee \sim p_4 \vee p_5 \end{cases}$$

$$\begin{cases} p_1 \wedge p_2 \to p_3 \\ \sim p_1 \vee \sim p_2 \vee p_3 \end{cases}$$

$$\begin{cases} p_1 \to p_1 \\ \sim p_1 \vee p_2 \end{cases}$$

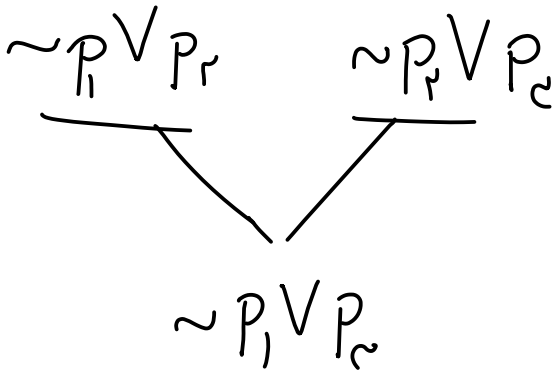
$$\begin{cases} p_1 \\ \top \to p_1 \end{cases}$$

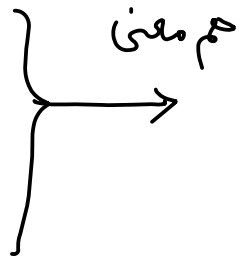




بندهای معین: نسبت به تکمیل، سنبه‌اند.

بندهای هورن: نسبت به تکمیل، سنبه‌اند.





$$\frac{P_1 \rightarrow P_2 \quad P_2 \rightarrow P_3}{P_1 \rightarrow P_3}$$

$$\frac{A \rightarrow B \quad B \rightarrow C}{A \rightarrow C}$$

رزونانس از هورن ها:

$\sim p_1 V \sim p_2$

$\sim p_3 V \sim p_2$

قابل رزونانس نیستند

$p_1 V \sim p_2 V \sim p_3$

$\sim p_1 V \sim p_4$

$\sim p_2 V \sim p_3 V \sim p_4$

رزونانس دو تا هورن یک تا هورن است.

Forward and backward chaining

$KB \models q$

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional definite clauses
          q, the query, a proposition symbol
  count ← a table, where count[c] is the number of symbols in c's premise
  inferred ← a table, where inferred[s] is initially false for all symbols
  agenda ← a queue of symbols, initially symbols known to be true in KB

  while agenda is not empty do
    p ← POP(agenda)
    if p = q then return true
    if inferred[p] = false then
      inferred[p] ← true
      for each clause c in KB where p is in c.PREMISE do
        decrement count[c]
        if count[c] = 0 then add c.CONCLUSION to agenda
  return false
```

Figure 7.15 The forward-chaining algorithm for propositional logic. The *agenda* keeps

ورودی برنامه:
KB: به صورت بندهای معین
q: یک لیترال مثبت.

خروجی برنامه: True یا False.

Time: $O(n)$

$n = |KB|$

Forward and backward chaining

جمله‌های معین:

اگر تک جمله نباشد می‌توان آن‌ها را به صورت $p_1 \wedge p_2 \wedge p_3 \wedge \dots \wedge p_k \rightarrow q$ نوشت.

حتی تک جمله‌ی p را می‌توان به صورت $T \rightarrow p$ نوشت.

اگر KB به صورت جمله‌های معین باشد، با این تکنیک می‌توان از زنجیرسازی برای استنتاج استفاده کرد و زمان اجرای آن خطی می‌شود.

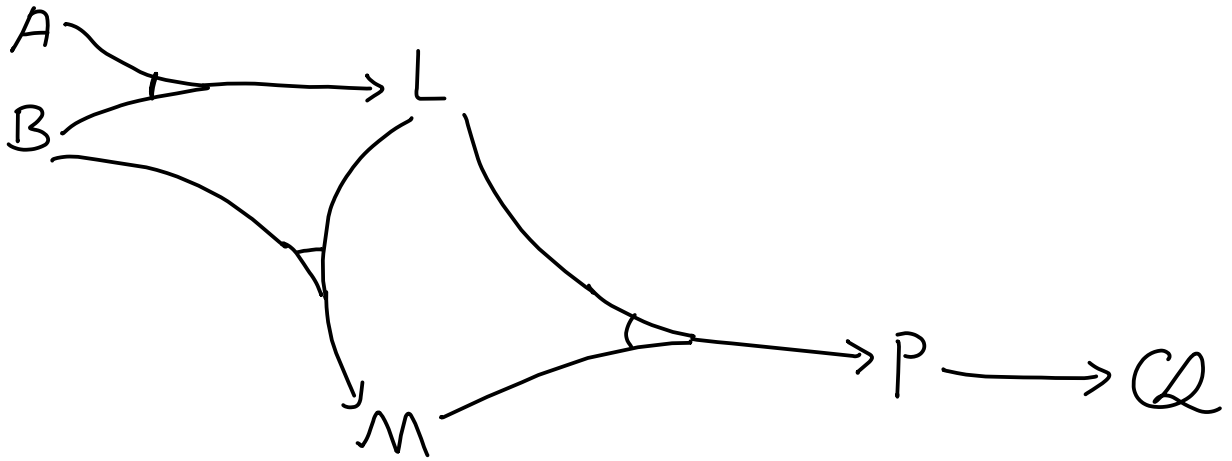
$$|KB| = n$$

Time: $O(n)$

گراف and-or برای زنجیره‌سازی رو به جلو:

KB
- قواعد شرطی (rule)
  - $P \Rightarrow Q$
  - $L \wedge M \Rightarrow P$
  - $B \wedge L \Rightarrow M$
  - $A \wedge P \Rightarrow L$
  - $A \wedge B \Rightarrow L$
- حقایق (fact)
  - A
  - B

$KB \overset{?}{\models} Q$



agenda (دستورکار): یک صف است. لیترال‌هایی که درست هستند اما هنوز پردازش نشده‌اند در آن قرار می‌گیرند.
(در واقع دستورکار را در هر لحظه تشخیص می‌دهد) در ابتدای کار لیترال‌هایی که مبنا هستند در KB در agenda قرار می‌گیرند. یعنی factها.

count: یک جدول است و در واقع به ما تشخیص می‌دهد برای هر گزاره شرطی هنوز چند لیترال از مقدمه پردازش نشده‌اند. وقتی برای یک گزاره شرطی c=0 شود، تالی آن نیز رفت و شود.

inferred: جدول ارزشی لیترال‌ها. در ابتدا برای همه False است. هر کدام که پردازش شد، True می‌شود.

This process continues **until** the **query q is added** **or** until **no further inferences** can be made.

شرط خاتمه زنجیرسازی رو به جلو :

- هدف به دست آید (ارزش q ، true شود)
- یا: هیچ قاعده جدیدی قابل تولید نباشد.

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

مثال: $KB \stackrel{?}{\models} Q$

مراحل الگوریتم F.C را نشان دهید.

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

$KB \stackrel{?}{\models} Q$

مقادیر اولیه:

	count
$P \rightarrow Q$	۱
$L \wedge M \rightarrow P$	۲
$B \wedge L \rightarrow M$	۲
$A \wedge B \rightarrow L$	۲

هیچ کدام از پیش شرط‌ها برقرار نشده‌اند.

	Inferred
P	false
Q	false
L	false
M	false
B	false
A	false

همه‌ی اتم‌ها false هستند.

Agenda
A
B

فقط fact‌ها در آن قرار دارند

استخراج A

pop

A=true

برای هرکدام که پیش شرط Ab را دارند: C = C-1

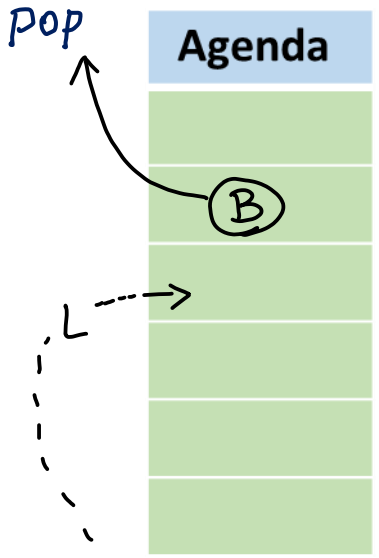
Agenda
A
B

	Inferred
P	false
Q	false
L	false
M	false
B	false
A	true

	count
$P \rightarrow Q$	۱
$L \wedge M \rightarrow P$	۲
$B \wedge L \rightarrow M$	۲
$A \wedge B \rightarrow L$	~~۲~~ ۱

هیچکدام به صفر نرسیدند

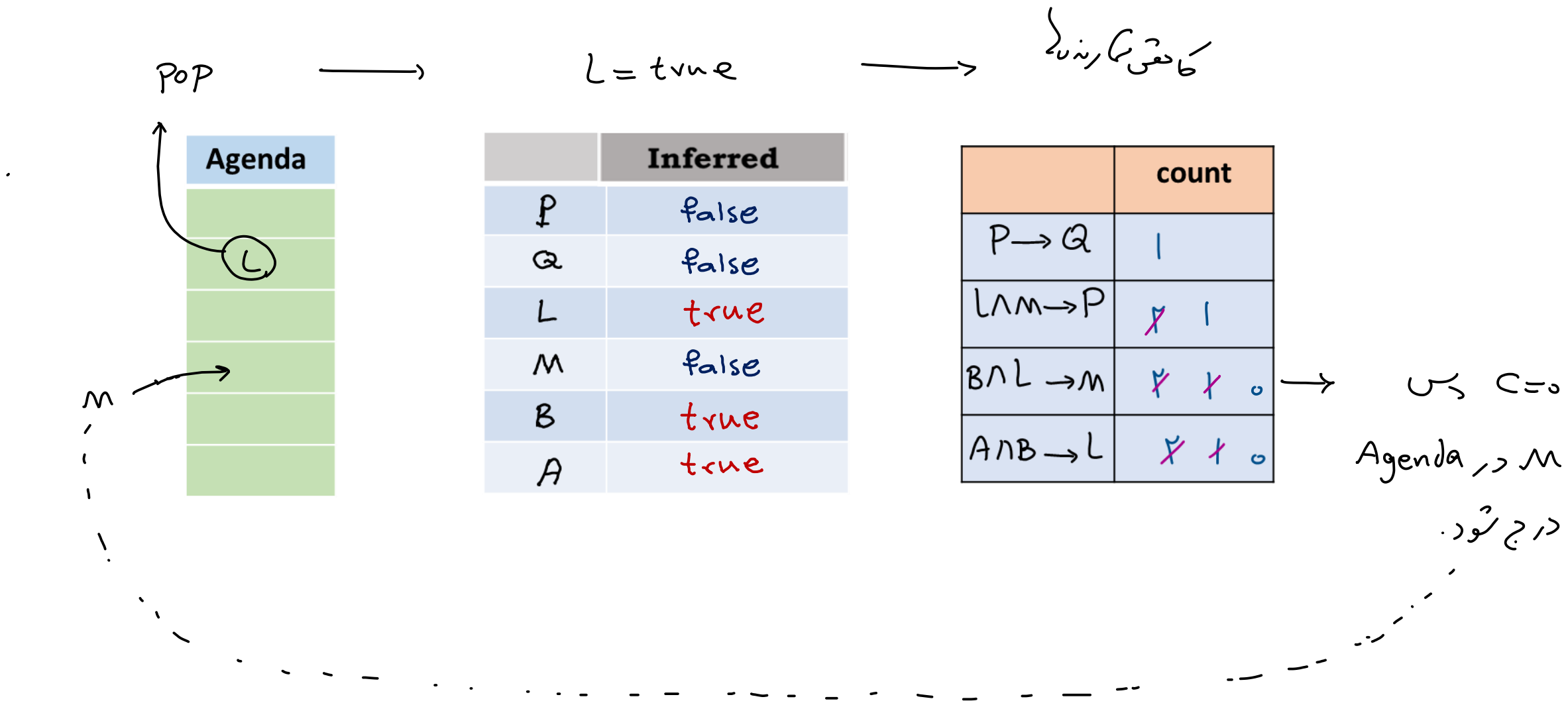
استخراج B ⟶ B = true ⟶ کاهش شمارنده‌ها

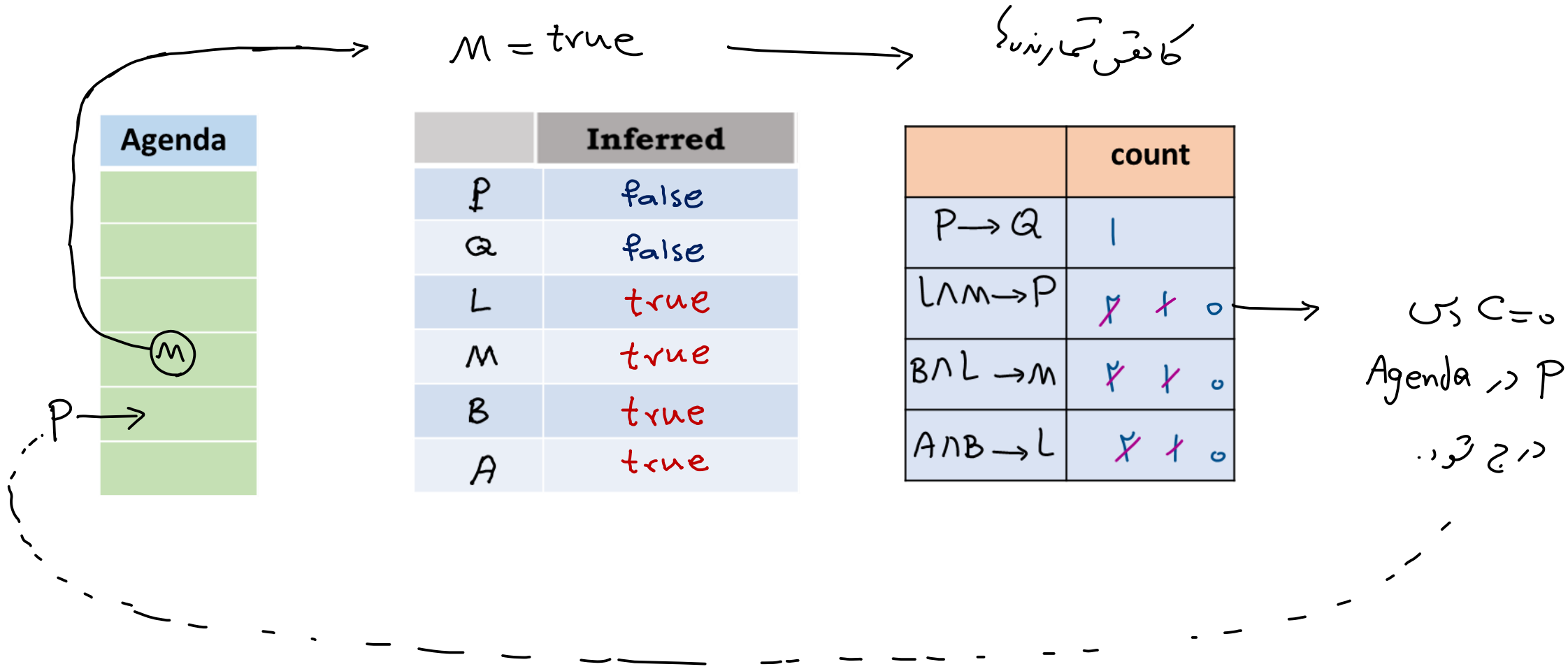


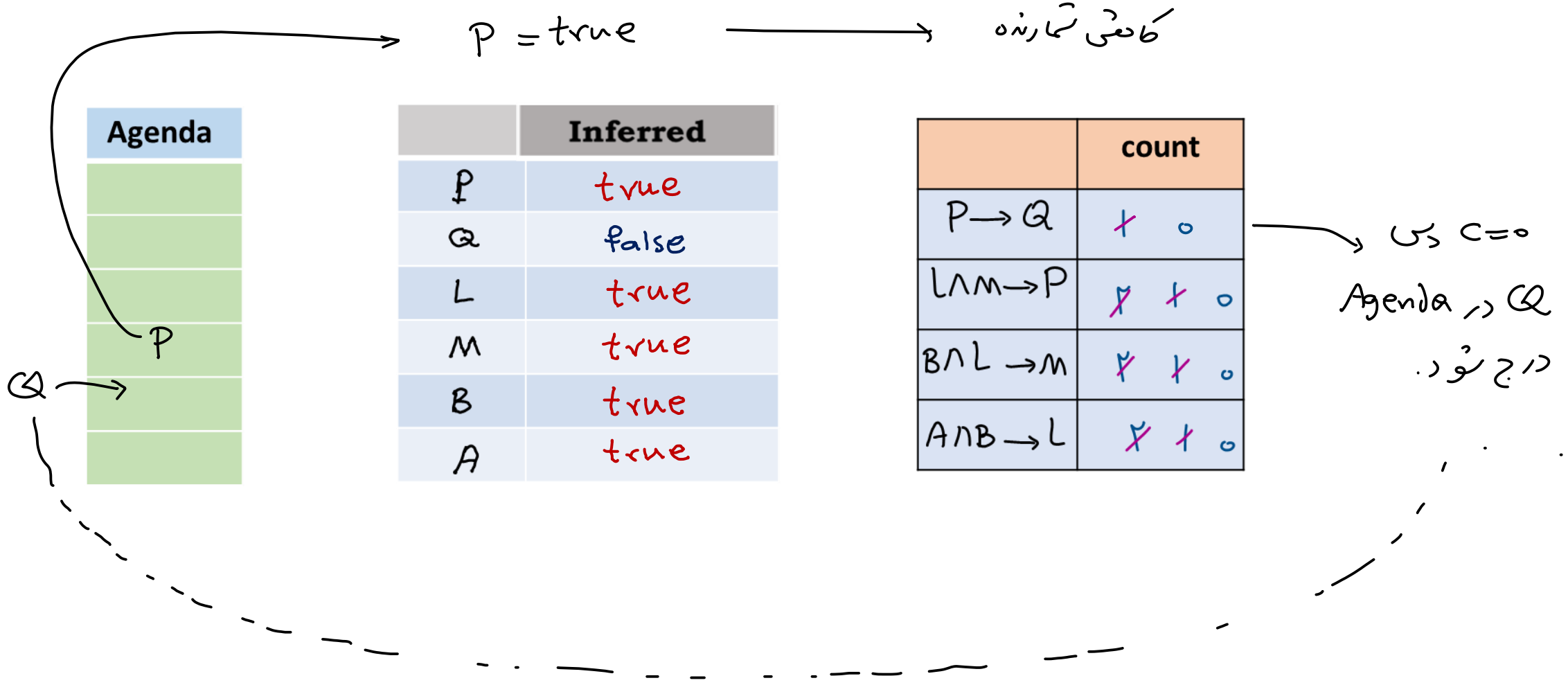
	Inferred
P	false
Q	false
L	false
M	false
B	true
A	true

	count
$P \rightarrow Q$	۱
$L \wedge M \rightarrow P$	۲
$B \wedge L \rightarrow M$	~~۲~~ ۱
$A \wedge B \rightarrow L$	~~۲~~ ۱ ۰

⟹ c=۰ شد، پس L در Agenda درج نشود.


pop
L = true
کاهش می‌یابد
Agenda
L
M
Inferred
P false
Q false
L true
M false
B true
A true
count
P → Q 1
L ∧ M → P ✗ 1
B ∧ L → M ✗ ✗ 0
A ∧ B → L ✗ ✗ 0
بنابراین C=0
M در Agenda
درج نمی‌شود.


M = true
کاهش شمارنده
Agenda
M
P
Inferred
P false
Q false
L true
M true
B true
A true
count
P → Q 1
L ∧ M → P ۲ ۱ ۰
B ∧ L → M ۲ ۱ ۰
A ∧ B → L ۲ ۱ ۰
اگر C=۰
P در Agenda
درج شود.


P = true
وقتی تمام شد
Agenda
P
Q
Inferred
P true
Q false
L true
M true
B true
A true
count
P → Q 1 0
L ∧ M → P 2 1 0
B ∧ L → M 2 1 0
A ∧ B → L 2 1 0
بی c==0
Agenda از Q
درج شود.

چون هدف استخراج شده است پس کار تمام است → Return: True

POP

Agenda
Q

	Inferred
P	true
Q	false
L	true
M	true
B	true
A	true

	count
$P \rightarrow Q$	✗ 0
$L \wedge M \rightarrow P$	✗ ✗ 0
$B \wedge L \rightarrow M$	✗ ✗ 0
$A \wedge B \rightarrow L$	✗ ✗ 0

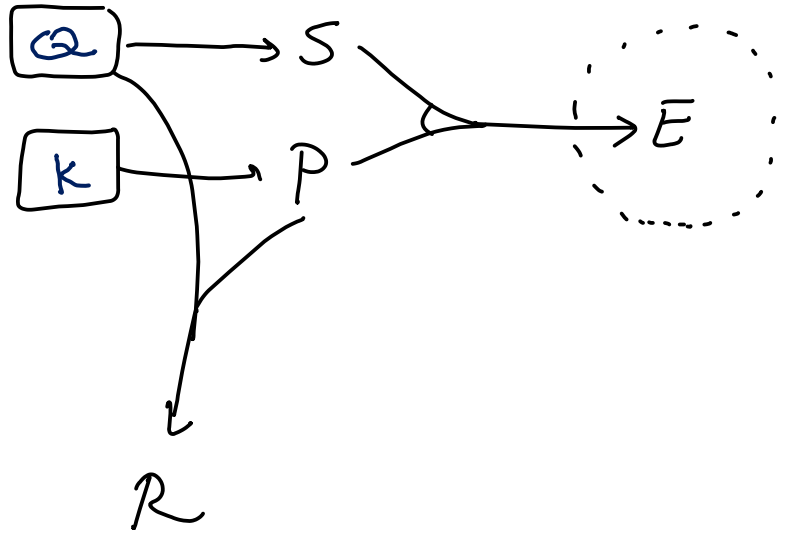
شکل زنجیره‌سازی forward : تولید نتایج غیرمرتبط

KB:

- $P \wedge Q \rightarrow R$
- $Q \rightarrow S$
- $K \rightarrow P$
- $S \wedge P \rightarrow E$
- Q
- K

$KB \overset{?}{\models} R$

R به دست آمد اما E هم که غیرمرتبط بود تولید شد.



برای آن که جملات غیر مرتبط کوئیک نشوند می‌توان از زنجیره‌سازی او به عقب استفاده کرد. (از هدف به سمت داده‌ها (fact) حرکت کنیم.

forward chaining : مبتنی بر داده‌ها. Data Driven

Backward chaining: مبتنی بر هدف. goal Driven (به صورت عمیق)

تک رشته‌ای مناسب (فرم ماشین) حرف زن: O(n) Time

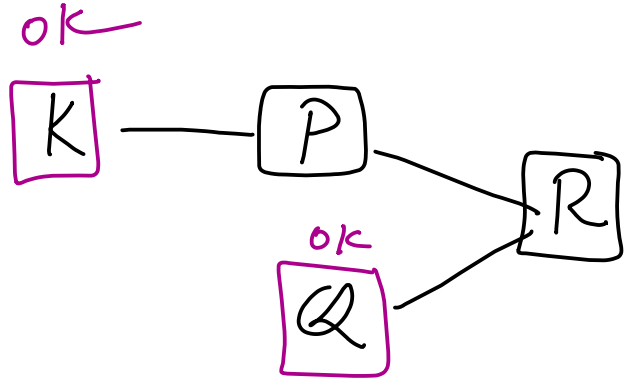
اما B.C در عمل سریعتر از F.C است چون نتایج غیر ضروری تولید نمی‌کند.

Backward chaining:

$$KB \overset{?}{\models} R$$

$$KB: \begin{cases} P \wedge Q \rightarrow R \\ Q \rightarrow S \\ K \rightarrow P \\ S \wedge P \rightarrow E \\ Q \\ K \end{cases}$$

از R شروع می‌کنیم: (به صورت DFS)



اگر برگ ها همه fact ها برسند استنتاج درست است: $KB \models R$.

درغیر این صورت $KB \not\models R$.

زنجیرسازی‌های Backward و forward هردو در منطق گزاره‌ای

برای پایگاه دانشی‌ای فرم معین و هدفی که لیترال مثبت باشد (مانند q)

الگوریتم‌های کاملی هستند. با آن که در عمل زنجیرسازی رو به عقب قدری سریع‌تر است

اما زمان اجرای هردو $O(n)$ است. $n = |KB|$.

* . در منطق مرتبه اول هنگامی که تابع‌ها تابعی داشته باشیم خواهید دید که شرایط تغییر می‌کند.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۱۲ فصل ۷

در این جلسه مروری بر مباحث جلسه‌ی قبل داشتیم. (به جزوه جلسه ۱۱ رجوع کنید)

FC BC are complete ⟶ زنجیره‌سازی‌های رو به جلو و رو به عقب کامل هستند.

[البته دقت کنید که هم جملات KB باید فرم خاصی باشند و هم هدف، باید نتایج مثبت مانند q باشد. برای این نوع جملات کامل هستند]

FC BC : O(n)

↓

در شرایط گفته شده

خودو O(n) هستند.

$n = |KB|$

FOL? ⟶ در منطق مرتبه اول شرایط کاملا تفاوت دارد.

خواهیم خواند در فصل ۹

SAT

مسئله ارضاپذیری: آیا مدلی وجود دارد که در آن تمام جملات مجموعه‌ی K ارزش True داشته باشند؟

مجموعه‌ی ارضاپذیر را مجموعه‌ی سازگار نیز می‌گوییم.

SAT

مثال آیا این مجموعه ارضاءِ پذیر است؟

$$K = \{ p \to q \ , \ \sim qVS \ , \ pVS \ , \ \sim S \wedge R \}$$

مقادیر اجباری:

می‌خواهیم $\sim S \wedge R$ درست باشد پس: $S=0 \ , \ R=1$

می‌خواهیم pVS درست باشد پس: $p=1$

می‌خواهیم $\sim qVS$ درست باشد پس $q=0$

می‌خواهیم $p \to q$ درست باشد: اما $p=1$ ، $q=0$ و در نتیجه ارزش $p \to q$ نمی‌تواند true باشد. شکست خوردیم.

مجموعه K ارضاءِ ناپذیر است.

مجموعه‌ی K ارضاءِ ناپذیر است $\Longleftrightarrow$ از K یک تناقض به دست می‌آید.

در مثال قبل:

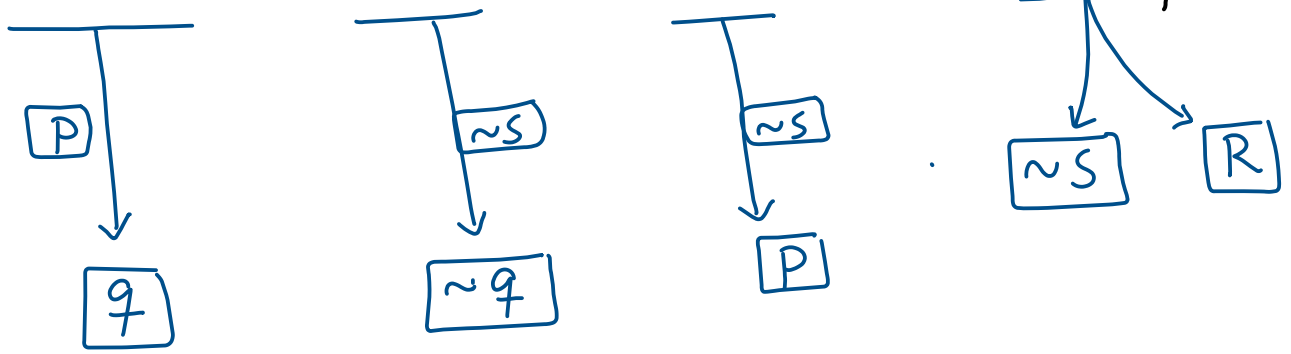
می‌توانیم از رزولوشن روی K استفاده کنیم (البته ابتدا جمله را به فرم CNF تبدیل کنید)

اگر با رزولوشن به تهی رسیدیم، به معنای ارضاءِ ناپذیری K است. (به عنوان تمرین انجام دهید)

روش دیگر نشان دادن ارضاء ناپذیری: نشان دهیم این مجموعه باعث تولید تناقض می‌شوند.

از راست به چپ مصداق کنید:

$$K = \{ p \to q \;,\; \sim q \vee s \;,\; p \vee s \;,\; \sim s \wedge R \}$$



تناقض به دست آمد $q \wedge \sim q$

همه‌ی این جملات، معنای یکسانی دارند:

K ارضاپذیر نیست ، K دارای مدل نیست ،

K ناسازگار است.

K به تناقض منجر می‌شود. ، K_1 معادل با تناقض است.

$K \models F$ $K \cong F$

ارتباط مسئله‌ی استنتاج با مسئله‌ی SAT:

$KB \models \alpha$ برقرار است.

$\Leftrightarrow$

$KB \rightarrow \alpha$ تautology است.

$\Leftrightarrow$

$\sim KB \vee \alpha$ تautology است.

$\Leftrightarrow$

$KB \wedge \sim \alpha$ تناقض (ناسازگار) است.

پاسخ مثبت به استنتاج $KB \models \alpha$ ؟

معادل است با:

پاسخ منفی به ارضاپذیری $KB \wedge \sim \alpha$

مثال: نشان دهید K ارضاپذیر است.

$$K = \{ \underbrace{p \vee q \vee R}_{①} , \underbrace{\sim q \vee p}_{②} , \underbrace{\sim p \vee \sim R}_{③} , \underbrace{R}_{④} \}$$

اگر بخواهیم همه جدول معتبر شود:

برای ④ ناچاریم: $R=1$

پس ④ و ① معتبر شده.

برای ③ ناچاریم $P=0$.

برای ② ناچاریم $q=0$.

پس: K ارضاپذیر است.

K دارای مدل های زیر است:

$(P, q, R, S) = (0, 0, 1, \text{dont care})$

مقدار S مهم نیست و میتواند 0 یا 1 باشد.

جدول ارزش TT-entails : مرتبه $\Omega(2^n)$

الگوریتم Davis-putnam (DPLL) : تبدیل SAT به یک CSP و حل آن با جستجوی عقبگرد و هم ابزارهای فصل ۶.

جستجوی محلی (WALK-SAT)

الگوریتم‌های حل مسئله‌ی SAT

الگوریتم دیویس پوتنام (DPLL) برای مسائل (SAT).

ساختارهای: حل مسائل SAT به صورت یک CSP. [جملات را در فرم CNF بنویسیم. هر بند یک محدودیت است.
دامنه متغیرها {0,1} یا معادل {false, true}
و هر P_i یک متغیر است.

مزیت‌ها و هیوریستیک‌ها نسبت به استفاده از جدول ارزش TTentails:

- خاتمه زود هنگام
- تک مخض
- بند واحد

مثال: مسئله‌ی SAT را به روش DPLL حل کنیم:

$$\{ A \vee B \vee \neg C \;,\; \neg A \vee \neg B \;,\; \neg A \vee B \vee \neg C \}$$

پاسخ: هرکدام از این بندها یک محدودیت برای ما هستند: آنها را ①، ②، ③ بنامید:

محدودیت‌ها:
- ① = True
- ② = True
- ③ = True

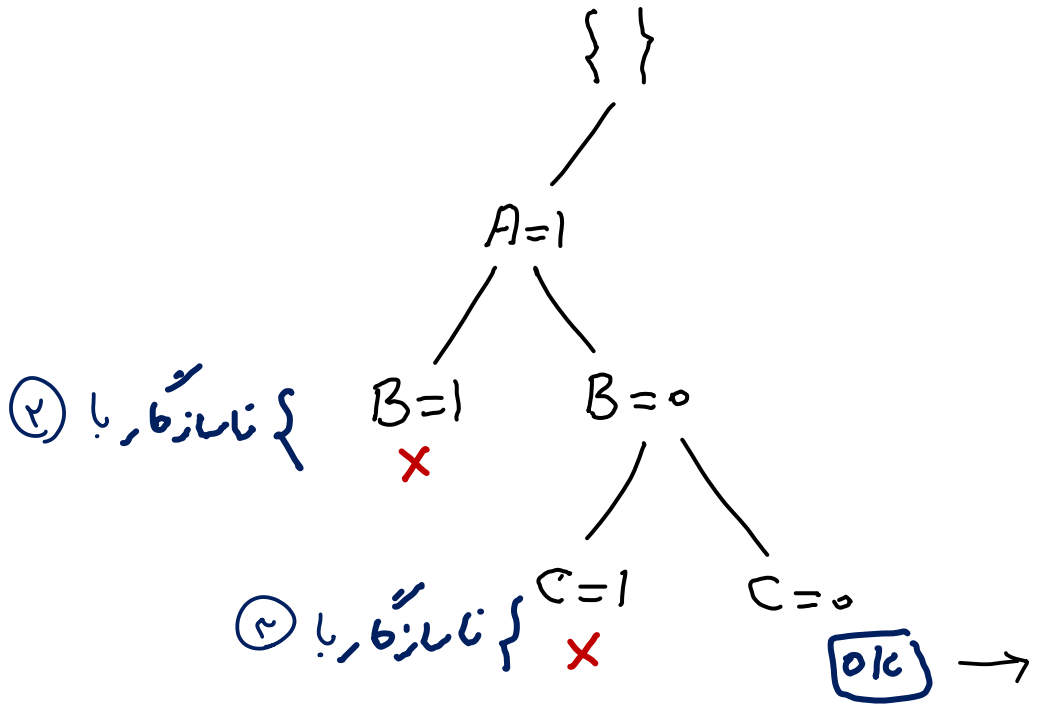
متغیرها همان اتم‌ها هستند: A, B, C

دامنه‌ی آنها: $\{0, 1\}$

① ② ③

$\{ A \vee B \vee \neg C \;,\; \neg A \vee \neg B \;,\; \neg A \vee B \vee \neg C \}$

حالا جستجوی عقبگرد (BT) با انتساب تهی شروع می‌شود.



پس:

$A=1, B=0, C=0$

یک انتساب کامل سازگار است.

مجموعه‌ی K ارضاء پذیر است.

(یک ویژگی خوب DPLL) خاتمه زودهنگام :

مثال: $\{ A \vee B \vee C \ , \ A \vee D \vee \neg B \}$

حل:



$n=4$ متغیر داریم اما لازم نیست که آن‌ها کامل پیدا کنیم. آنها جزئی $(A,B,C,D)=(1,\ldots,\ldots,\ldots)$

مجموعه K را سازگار (ارضاپذیر) می‌کند.

هیوریستیک برای Dpll
(آردین)

استفاده از نماد محض: در هر مرحله اگر در بین جمله محدودیت های باقی مانده، متغیر p فقط به شکل مثبت یا فقط به شکل منفی آمده باشد، نماد محض نام دارد.

نماد محض، یک روش برای انتخاب متغیر بعدی است.

مقدار مناسب برای نماد محض:

if p → p ≡ True

if ¬p → p ≡ false

مثال: با استفاده از آرومین نمادمحض، مسأله‌ی SAT را حل کنیم.

$$\{\ \underset{①}{A \vee \neg B}\ ,\ \underset{②}{\neg A \vee \neg B \vee D}\ ,\ \underset{③}{A \vee \neg D}\ ,\ \underset{④}{\neg B \vee \neg A}\ ,\ \underset{⑤}{A \vee E}\ \}$$

①، ②، ④ ارضا شده‌اند.
③ و ⑤ باقی مانده‌اند.

$A = true$ ← حال A نمادمحض است ← $B = false$ ← B نمادمحض است فقط به صورت $\neg B$ آمده است.

مدل جزئی یافت شده:

$$A=1\ ,\ B=0\ ,\ D=\dots\ ,\ E=\dots$$

{}
/
B=0
/
A=1

یک هیوریستیک دیگر DPLL

استفاده از بند واحد : در هر مرحله، اگر بندی داریم که هنوز ارضا نشده اما فقط یک متغیر مقداردهی نشده دارد ، بند واحد نام دارد.

مثال:

$$\{ AV\sim BV\sim D \ , \ B \ , \ DV\sim A, \ \sim BVD \}$$

$\longrightarrow$ $D=true$ $\longrightarrow$ حالا $\sim BVD$ بند واحد است $\longrightarrow$ $B=true$ $\longrightarrow$ B بند واحد است.

$\longrightarrow$ $AV\sim BV\sim D$ بند واحد است $\longrightarrow$ حالا $A=true$ $\longrightarrow$ OK

مجموعه سازگار است.

توجه: اگر پایگاه دانش فقط‌ش من سنبه‌های معین باشه، آنگاه DPLL کاری معادل با زنجیره‌سازی forward است و سنبه واحد در شروع کار همان fact ها در زنجیره‌سازی هستند.

درضمن قبل اگر هر سنبه معین را به گزاره شرطی با جملات مثبت تبدیل کنیم، این معادل بودن قابل مشاهده است:

مثال قبل:

$B \wedge D \rightarrow A$ $\quad$ B $\quad$ $A \rightarrow D$ $\quad$ $B \rightarrow D$

$\{ A \vee \sim B \vee \sim D \ , \ B \ , \ D \vee \sim A, \ \sim B \vee D \}$

$B \rightarrow D$

B بند واحد است. $\rightsquigarrow$ $B = true$ $\rightsquigarrow$ حالا $\sim B \vee D$ بند واحد است $\rightsquigarrow$ $D = true$ $\rightsquigarrow$

$\rightsquigarrow$ بند واحد است $A \vee \sim B \vee \sim D$ حالا $\rightsquigarrow$ $A = true$ $\rightsquigarrow$ (#)

$B \wedge D \rightarrow A$

خلاصه: برای نبع‌های معین، forward chaining و DPLL به شکل مثبتی از نبع واحد استفاده می‌کنند.

یک خلاصه

به جای $KB \models \alpha$?

روی SAT : $KB \wedge \neg \alpha$?

کار کنیم.

یک راه برای ثابت کردن ارضاءناپذیری، همان رزولوشن است.

$KB \wedge \neg \alpha$

↓ رزولوشن

ته

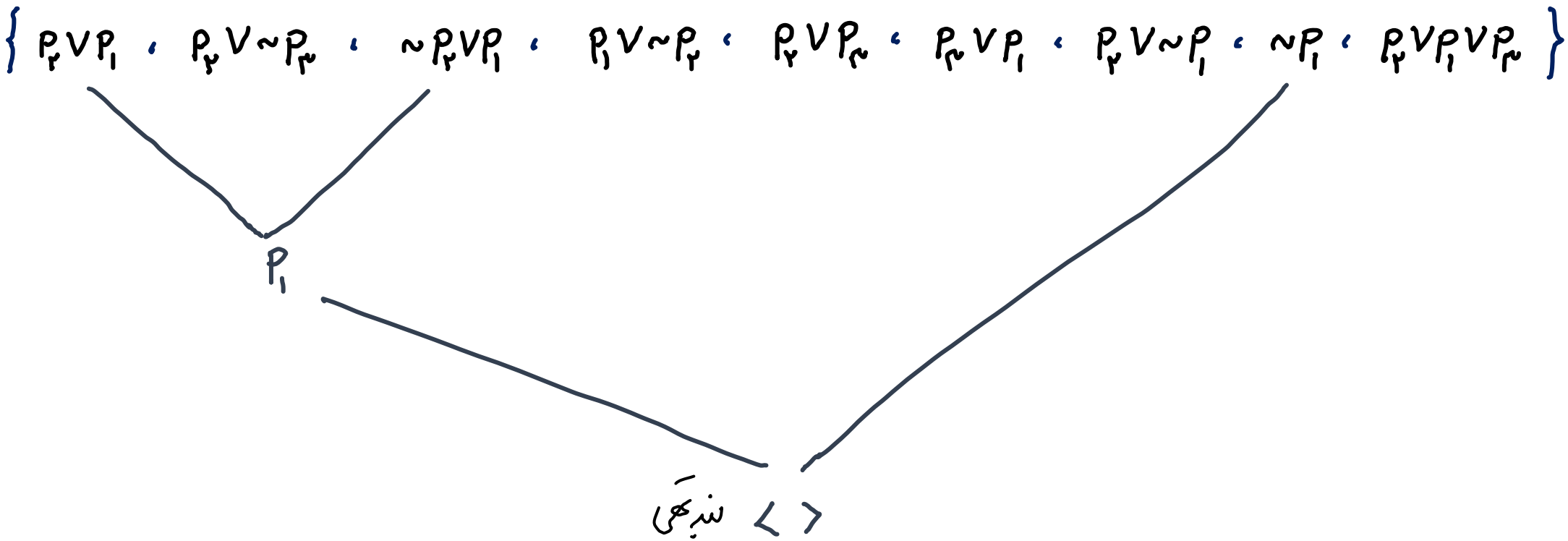
تمرین: مسأله‌ی SAT را در مورد این مجموعه حل کنید: (ارضاءپذیری این مجموعه را بررسی کنید)

$$\{ p_2 \vee p_1 \,،\, p_2 \vee \sim p_3 \,،\, \sim p_2 \vee p_1 \,،\, p_1 \vee \sim p_2 \,،\, p_2 \vee p_3 \,،\, p_3 \vee p_1 \,،\, p_2 \vee \sim p_1 \,،\, \sim p_1 \,،\, p_2 \vee p_1 \vee p_3 \}$$

پاسخ:

همه جمله‌ها نیز هستند. از رزولوشن استفاده کنیم اگر [illegible] به دست آمد مجموعه ارضاءناپذیر است.

حل: اثبات ناسازگاری بودن با استفاده از رزولوشن:

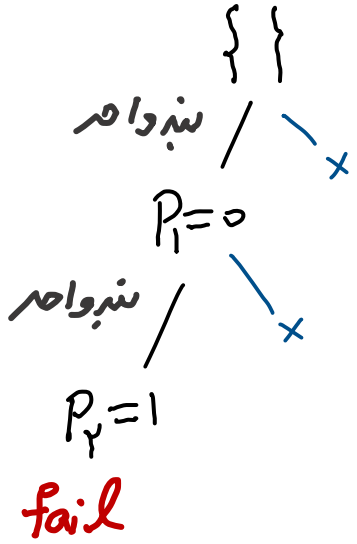


همین مثال را با الگوریتم DPLL و استفاده از هیوریستیک تک واحد انجام دهید.

نکته: وقتی مسئله را با ابزار تک واحد انجام می‌شود، سایر شاخه‌ها برای بررسی نیازی نیست چون این انتخاب، اجباری است. یعنی سایر شاخه‌ها حتماً شکست می‌خورند.

حل با الگوریتم DPLL و هیوریستیک‌های تک‌واحد و تک‌مقدار محض:

$$\{ P_2 \vee P_1 \;،\; P_2 \vee \neg P_3 \;،\; \neg P_2 \vee P_1 \;،\; P_1 \vee \neg P_2 \;،\; P_2 \vee P_3 \;،\; P_3 \vee \overset{\circ}{P_1} \;،\; P_2 \vee \neg P_1 \;،\; \neg P_1 \;،\; P_2 \vee P_1 \vee P_3 \}$$



تک‌واحد $\neg P_1$ را داریم پس $P_1 = 0$.

حالا تک‌واحد $P_1 \vee P_2$ را داریم پس $P_2 = 1$.

اما تک‌واحد $P_1 \vee \neg P_2$ را هم داریم. پس مجموعه ناسازگار و ارضاءنشدنی است.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

ادامه جلسه ۱۲

ابزارهای SAT/SMT برای کنترل زمان در مسائل بزرگ:

① تجزیه مولفه ای

② انتخاب ترتیب مناسب برای متغیرها و مقداردهی‌ها مانند مسائل CSP . مثلاً هیوریستیک درجه‌ای
(متغیری که در تعداد بیشتری از بندهای باقی‌مانده ظاهر شود.)

③ به‌جای عقب‌گرد ساده، از پرش به عقب هوشمند استفاده کنند. (توجه به محتوای برخوردها)

④ شروع مجدد تصادفی.

⑤ استفاده از شاخص‌های هوشمند.

1. **Component analysis**

نوعی تقسیم و غلبه ایجاد می‌کند. می‌توان مجموعه‌های مجزایی از جنبه‌هایی که متغیر مقداردهی نشده‌ی مشترک ندارند ایجاد کرد.

2. **Variable and value ordering**

می‌توان به جای ترتیب تصادفی و مقداردهی تصادفی {True, false}، برای ترتیب متغیرها از هیوریستیک درجه‌ای و برای مقداردهی از min-conflict با توجه به این که تعداد p_2 بیشتر است یا p_1 به استفاده کرد.

1. Component analysis

مثال از تحلیل مولفه ای :

$$K = \{ \underset{①}{\sim P_1 \vee P_4} \ , \ \underset{②}{P_2 \vee P_5 \vee P_3} \ , \ \underset{③}{P_4 \vee P_1} \ , \ \underset{④}{\sim P_5 \vee P_3} \ , \ \underset{⑤}{P_2 \vee \sim P_5} \}$$

می‌توان مولفه‌های

$$A = \{ ①, ③ \}$$

$$B = \{ ②, ④, ⑤ \}$$

را ایجاد کنیم. K ارضاءِ پذیر است $\Longleftrightarrow$ A و B ارضاءِ پذیر باشند.

2. Variable and value ordering

مثال: ترتیب مناسب متغیرها و مقادیر

$$\{P_2 \vee \neg P_3 \quad , \quad P_4 \vee \neg P_1 \quad , \quad P_3 \vee \neg P_1 \vee P_4 \quad , \quad \neg P_3 \vee P_2 \vee P_1\}$$

برای شروع: بهتر است از P_3 شروع کنیم که در بندهای بیشتری حضور دارد. (نوعی درجه‌ای)

برای مقداردهی: بهتر است از مقدار $P_3 = \text{False}$ شروع کنیم چون تعداد $\neg P_3$ بیشتر از P_3 است.

3. **Intelligent backtracking**

همان عقب گرد هوشمند در CSP ها.
وقتی در یک شاخه از مقداردهی به شکست می‌رسیم به جای یک مرحله عقب گرد، $P_i = ?$
به جایی برگردیم که ناسازگاری از آنجا تولید شده است.
مثلاً از بین متغیرهای مقداردهی شده‌ی قبل، به متغیری برگردیم که با P_i
احتمال در یک شبکه دارد. (آن که عمق بیشتری دارد).

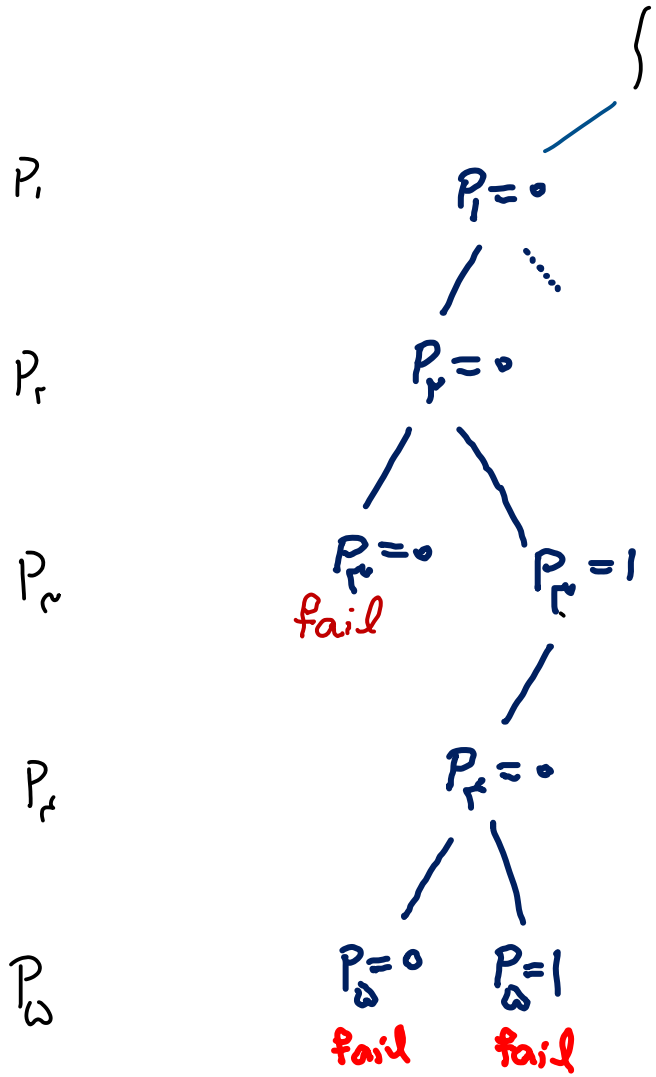
4. **Random restarts**

هر جا با شکست مواجه شدیم با انتخاب تصادفی از یک متغیر و مقدار دلخواه
کار را از نو آغاز کنیم.

5. **Clever indexing**

ذخیره و فهرست کردن اطلاعات مفید مانند اینکه همسایه‌هایی که P_i در آنها
حضور دارد را در یک جدول داشته باشیم.

$$\{ P_1 \vee P_5 \;,\; P_2 \vee P_3 \;,\; P_4 \vee \sim P_1 \;,\; P_1 \vee P_2 \vee \sim P_5 \}$$



3. Intelligent backtracking

مثال عقبگرد هوشمند:

وقتی در P_5 به بن بست می‌رسیم به جای بازگشت به P_4 به مجموعه برخورد (P_5 های تعارض‌دار) دقت کنیم. P_5 با P_1 و P_2 در برخی از بندها شرکت دارد و P_1 و P_2 مقداردهی شده‌اند پس این‌ها همگی مقداردهی شده P_5 هستند:

$$conflict(P_5) = \{ P_1 \cdot P_2 \}$$

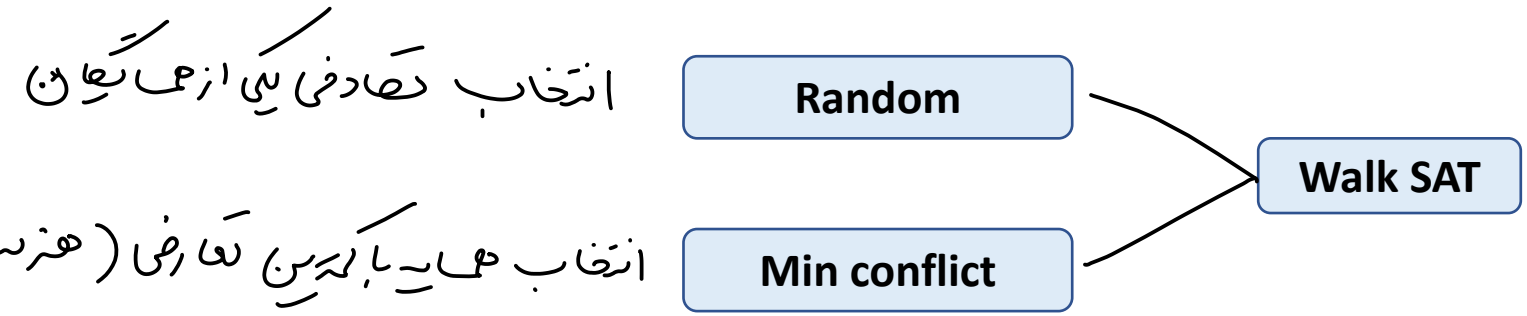
از بین آن‌ها به P_2 برمی‌گردیم که عمق بیشتری دارد. (برمی‌گردیم و $P_2=1$ قرار می‌دهیم)

حل SAT با جستجوی محلی:

به جای شروع با انتساب تهی، همه‌ی حالات (هر state) یک انتساب کامل هستند.

حالا مثلاً از انتساب $(0, 0, 0, \ldots, 0)$ شروع می‌کنیم و هر بار با تغییر دادن فقط یک متغیر، به یکی از همسایگان این حالت می‌رویم.

اگر n متغیر داشته باشیم، هر حالت دارای n همسایه است.



مثال: حل مسئله‌ی SAT برای این مجموعه از گزاره‌ها به روش WALK-SAT و min conflict:

$$\{P_2 \vee P_1 \;،\; P_2 \vee \sim P_3 \;،\; \sim P_2 \vee P_1 \;،\; P_1 \vee \sim P_2 \;،\; P_2 \vee P_3 \;،\; P_3 \vee P_1 \;،\; P_2 \vee \sim P_1 \;،\; \sim P_1 \;،\; P_2 \vee P_1 \vee P_3\}$$

تعداد اتم‌ها: $n=3$

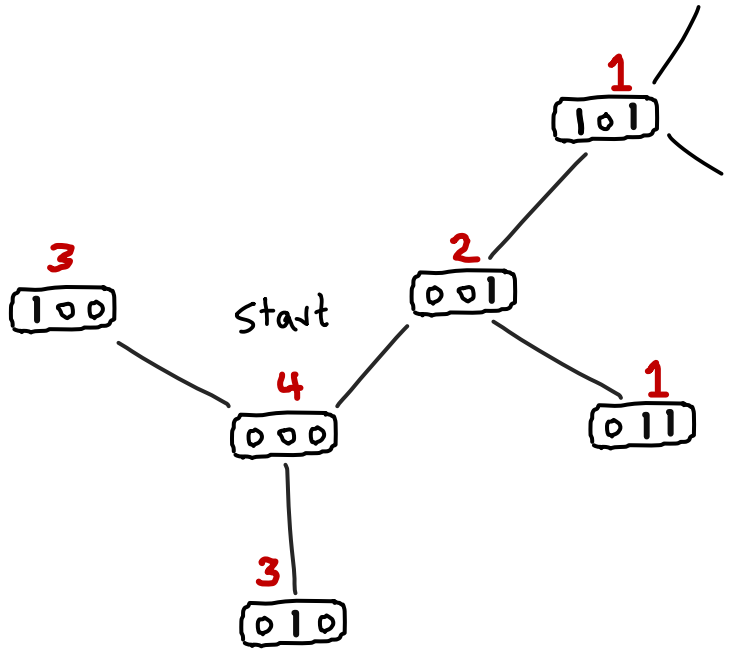
هر حالت: یک ۳تایی مرتب $\boxed{P_1 | P_2 | P_3}$

حالت شروع: مثلاً $\boxed{0 | 0 | 0}$

هزینه (تعداد تعارض‌ها = تعداد نام‌زگاری‌ها = تعداد بندهایی که ارزش false پیدا می‌کنند)

مجاورها: فقط در یک مولفه تفاوت دارند.

$P_2 \vee P_1$ ، $P_2 \vee \sim P_3$ ، $\sim P_2 \vee P_1$ ، $P_1 \vee \sim P_2$ ، $P_2 \vee P_3$ ، $P_3 \vee P_1$ ، $P_2 \vee \sim P_1$ ، $\sim P_1$ ، $P_2 \vee P_1 \vee P_3$



از 000 شروع میکنیم.

به 001 می‌آییم.

به یکی از حالات 101 یا 011 می‌آییم.

و به همین ترکیب ادامه می‌دهیم.

(بدون حرکت تصادفی احتمال متوقف شدن در بهینه یا محل وجود دارد)

توجه: نقطه ضعف حل SAT با جستجوی محلی و min_conflict آن است که ممکن است در min محلی گرفتار شود.

برای خارج شدن از min محلی می‌توان از قدم زدن تصادفی کمک گرفت و با احتمال کمی به یکی از همسایه‌ها رفت.

ساده‌ترین جستجوی محلی، قدم زدن تصادفی است. اما:

- اگر مجموعه‌ی K ارضاءپذیر باشد، متوقف خواهد شد و یک مدل برای K پیدا خواهد کرد.
- اگر مجموعه‌ی K ارضاءپذیر نباشد، ممکن است در loop گرفتار شود.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

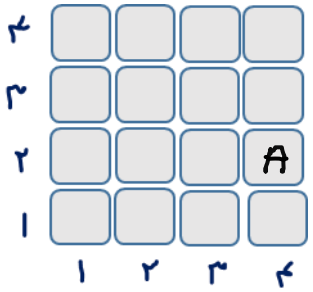
هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۱۳ فصل ۸

8 FIRST-ORDER LOGIC

Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	facts with degree of truth $\in [0, 1]$	known interval value



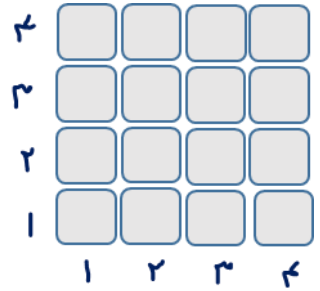
فقط یک عامل در $(۴,۲)$ قرار دارد

در منطق گزاره‌ای

$\sim A(۱,۴)$	$\sim A(۲,۴)$	$\sim A(۳,۴)$	$\sim A(۴,۴)$	
$\sim A(۱,۳)$	$\sim A(۲,۳)$	$\sim A(۳,۳)$	$\sim A(۴,۳)$	
$\sim A(۱,۲)$	$\sim A(۲,۲)$	$\sim A(۳,۲)$	$A(۴,۲)$	True
$\sim A(۱,۱)$	$\sim A(۲,۱)$	$\sim A(۳,۱)$	$\sim A(۴,۱)$	

در منطق مرتبه اول:

$\forall x, y \in \{۱,۲,۳,۴\}$ $\quad A(x,y) \longleftrightarrow (x=۴, y=۲)$



جمله‌ی: در سطر دوم گودال وجود دارد. (حداقل یک گودال وجود دارد)

منطق گزاره‌ای:

$$p(1,2) \vee p(2,2) \vee p(3,2) \vee p(4,2)$$

منطق مرتبه اول:

$$\exists x \in \{1,2,3,4\} \quad p(x,2)$$

$$
\begin{aligned}
\mathit{Sentence} &\rightarrow \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} &\rightarrow \mathit{Predicate} \mid \mathit{Predicate}(\mathit{Term}, \ldots) \mid \mathit{Term} = \mathit{Term} \\
\mathit{ComplexSentence} &\rightarrow (\ \mathit{Sentence}\ ) \mid [\ \mathit{Sentence}\ ] \\
&\mid \neg\ \mathit{Sentence} \\
&\mid \mathit{Sentence} \wedge \mathit{Sentence} \\
&\mid \mathit{Sentence} \vee \mathit{Sentence} \\
&\mid \mathit{Sentence} \Rightarrow \mathit{Sentence} \\
&\mid \mathit{Sentence} \Leftrightarrow \mathit{Sentence} \\
&\mid \mathit{Quantifier}\ \mathit{Variable}, \ldots\ \mathit{Sentence}
\end{aligned}
$$

- Objects:

Ahmad , Tehran , …

x , y , a, b, …

اشیاء: ثابت‌ها (با حروف بزرگ)، متغیرها (با حروف کوچک)

- Functions:

Father (x) , Leftleg(John) , …

Sum(3, 5) , …

Sum(a, b) , …

توابع: توانند یک یا چند آرگومان داشته باشند. مثلاً

پدر x = Father (x)

جمع a و b = Sum (a, b)

- Relations:

predicates:

روابط مسندها

، قیدها

King(John)

Black (Ball)

P(x)

Brother (Ali , Arash)

Likes(Javad , Ahmad)

x = y

روابط (نسبت‌های ربطی) و قیدها (صفات):

Brother(x, y) : x برادر y است

Black(x) : x سیاه‌رنگ است.

EQ(..., ...)

.. = ..

$$Term \rightarrow Function(Term, \ldots)$$
$$\mid Constant$$
$$\mid Variable$$

ترم ها
- ثابت ها: John , Ali , 5,...
- متغیرها: x , y , ...

توابعی از ترم ها: Father(Ali) , Father(x)

King(John) ← ترم نیست یک قید است

Brother (Ali , Arash) ← ترم نیست یک رابطه است.

Father (Ali) ← ترم است (تابع است)

ترم ثابت

ترم است (ثابت)

ترم نیستند نه ربط (رابطه) هستند.

- x = y
- Father (Ali) = Javad
- Likes(Javad , Ahmad)

ترم

Brother (Ali , John) ⇔ King(John)

(Father(Ali) = Javad) V Likes(Javad , Ahmad)

~ (x = y)

∃x (x=x) ⇒ ∀y ∃z (y=z)

∀x (P(x) V ~P(x))

∀x Smart(x) V (x=x)

فقط سورهای کلی و وجودی (جزئی)
یعنی $\forall$ و $\exists$ سورهای استاندارد
هستند. مثلاً سوری به نام $\nexists$
نداریم. به جای $\nexists x\, P(x)$
باید بگیریم $\forall x \sim P(x)$.

سورها

$$Quantifier \rightarrow \forall \mid \exists$$

ترم‌ها

$$Constant \rightarrow A \mid X_1 \mid John \mid \cdots$$

$$Variable \rightarrow a \mid x \mid s \mid \cdots$$

$$Predicate \rightarrow True \mid False \mid After \mid Loves \mid Raining \mid \cdots$$

$$Function \rightarrow Mother \mid LeftLeg \mid \cdots$$

روابط و قیدها
ترم محسوب نمی‌شوند
بلکه آن‌ها می‌توان ارزش
True یا False داد.

King(John)

Brother (Ali , Arash)

Father (Ali) = Javad

x = y

Likes(Javad , Ahmad)

Brother (Ali , John) $\Leftrightarrow$ King(John)

(Father(Ali) = Javad) V Likes(Javad , Ahmad)

$\sim (x = y)$

$\exists x\ (x=x) \Rightarrow \forall y\ \exists z\ (y=z)$

$\forall x\ (P(x)\ V \sim P(x))$

$\forall x$ Smart(x) V (x=x)

اولویت عملگرها

OPERATOR PRECEDENCE : $\neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow$

وقتی پرانتزگذاری نشده باشد، اولویت های بالا ترتیب عملگرها را مشخص می کنند.

مثلاً : $\sim P(x) \Rightarrow Q(y) \vee R(x)$

معادل است با : $(\sim P(x)) \Rightarrow (Q(y) \vee R(x))$

ground term : ترم زمینه‌ای : ترم بدون متغیر است.

ترم غیر زمینه‌ای:

x , y , …

Father (x) , Leftleg(y) , …

Sum(x, b) , …

ترم زمینه‌ای :

Ahmad , Tehran , …

Father (Ali) , Leftleg(John) , …

Sum(3, 5) , …

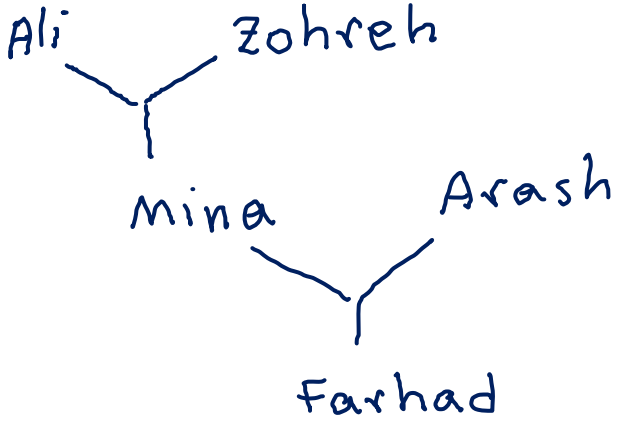
و این‌ها نمادهای رابطه و گزاره هستند ترم نیستند:

King(John)

Brother (Ali , Arash)

Father (Ali) = Javad

مثال: دنیای کوچکی با این اعضاء و روابط بین آنها را تصور کنید:



علی و زهره زوج هستند. مینا تنها فرزندآنهاست.

آرش و مینا زوج هستند. فرهاد تنها فرزندآنهاست.

اگر بخواهیم این دنیا را در منطق F05 بیان کنیم از این نمادها می‌توانیم استفاده کنیم:

ثابت‌ها: Ali, Arash, Farhad, Zohreh, Mina

توابع: Father(x) , Mother(x) , ... پدر x ، مادر x ، ...

قیدها (صفات) (مرد بودن، زن بودن، ...): Male(x) ; Female(x) ; ...

روابط دوتایی (مسندها) Marriage(x, y)

x و y همسر هم هستند

تفاوت توابع و مسندها:

هر نماد رابطه‌ای (جانشینِ ثابت به جای متغیر) یک گزاره منطقی است و می‌تواند True یا False باشد.

اما نماد تابعی چنین نیست. اگر F(x) نماد تابعی باشد، F(C) یک شیء است.

نمادهای تابعی قابل ترکیب هستند. F(F(x)) و .... خوش ساخت است.

در یک مدل با اعضای $A = \{ c_1 , c_2 , \ldots , c_m \}$ معنای جملات کلی و وجودی چنین است:

$$\forall x \; P(x) \equiv P(c_1) \wedge P(c_2) \wedge \ldots \wedge P(c_m)$$

$$\exists x \; P(x) \equiv P(c_1) \vee P(c_2) \vee \ldots \vee P(c_m)$$

* یک ویژگی مهم نگارشی برای سورهای وجودی در اعضای مجموعه A :

بیان حکم کلی، معمولاً به صورت گزاره شرطی قابل نوشتن است: جهانی $\forall x \in A \quad p(x)$ را به این صورت

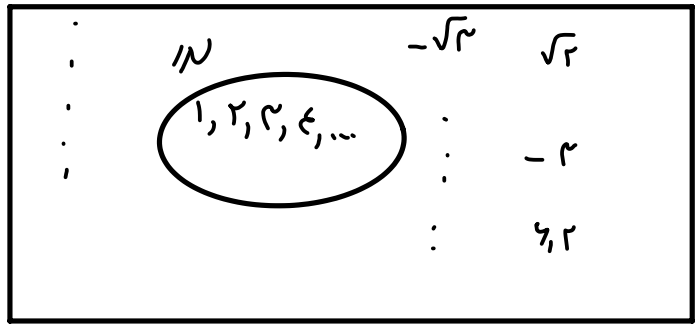
می‌توان نوشت: $\forall x \quad (x \in A \longrightarrow p(x))$

بیان حکم جزئی، معمولاً به صورت ترکیب and قابل نوشتن است: جهانی $\exists x \in A \quad p(x)$ را به این صورت

می‌توان نوشت: $\exists x \quad (x \in A \wedge p(x))$

مثال در دنیای اعداد، می‌خواهیم بگوییم همه اعداد طبیعی بزرگتر مساوی یک هستند:

دنیای اعداد



به صورت خلاصه: $\forall x \in \mathbb{N} \quad x \geq 1$

به صورت استاندارد: $\forall x \left(x \in \mathbb{N} \longrightarrow x \geq 1 \right)$

در دنیای انسان‌ها : می‌خواهیم بگوییم همه‌ی دانش‌آموزان ، علی را دوست دارند

(یکی محبوبه‌ی دانش‌آموزان باشه)

خلاصه‌نویسی: $\forall x \in S \quad likes(x, Ali)$

استاندارد: $\forall x \left(x \in S \Rightarrow likes(x, Ali) \right)$

یا به این صورت:

$\forall x \left(Student(x) \Rightarrow likes(x, Ali) \right)$

می خواهیم بگوییم برخی دانش‌آموزان، علی را دوست دارند

(یکی مجموعه‌ی دانش‌آموزان باشد)

خلاصه‌نویسی: $\exists x \in S \quad likes(x, Ali)$

استاندارد: $\exists x \ (x \in S \wedge likes(x, Ali))$

یا به این صورت:

$\exists x \ (Student(x) \wedge likes(x, Ali))$

همه‌ی دانش‌آموزان درس A را گذرانده‌اند

$$\forall x \quad \Big(student(x) \longrightarrow pass(x, A) \Big)$$

حداقل یکی از دانش‌آموزان درس A را گذرانده‌است.

$$\exists x \quad \Big(student(x) \wedge pass(x, A) \Big)$$

خدامی

$$[\forall x \in A \quad p(x)] \overset{\text{معادلند}}{\equiv} [\forall x \quad (x \in A \rightarrow p(x))]$$

$$[\exists x \in A \quad p(x)] \overset{\text{معادلند}}{\equiv} [\exists x \quad (x \in A \wedge p(x))]$$

جابه‌جایی سورها

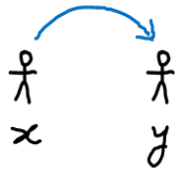
جابه جایی دو سور هم نوع معنای جمله را تغییر نمی‌دهد:

$$[\exists x \ \exists y \ f(x,y)] \overset{\text{معادلند}}{\equiv} [\exists y \ \exists x \ f(x,y)]$$

$$[\forall x \ \forall y \ f(x,y)] \overset{\text{معادلند}}{\equiv} [\forall y \ \forall x \ f(x,y)]$$

اما جملات $\forall x \exists y \ f(x,y)$ و $\exists y \forall x \ f(x,y)$ متفاوت هستند:

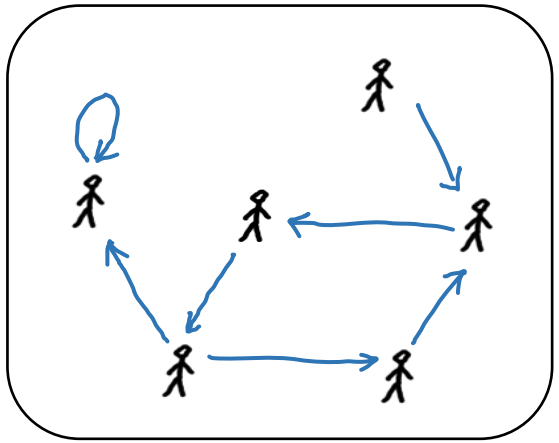
Loves(x, y)



$$A: \ \forall x \ \exists y \ Loves(x,y)$$

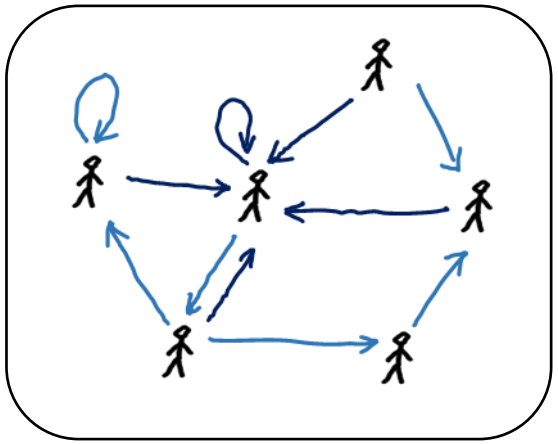
$$B: \ \exists y \ \forall x \ Loves(x,y)$$

(۱)



در دنیای ① هر x با حداقل یک y در رابطه است. با تغییر x آن y هم تغییر می‌کند. y وابسته به x است ($y = f_1(x)$) پس A ارزش True دارد. اما هیچ y ای وجود ندارد که همه x ها با آن در رابطه باشند. B ارزش False دارد.

(۲)



در دنیای ② یک y مشترک وجود دارد که همه x ها با آن در رابطه‌اند. این y وابسته به x ها نیست. B ارزش True دارد. پس A هم ارزش True دارد.

A: $\forall x \ \exists y \ Loves(x,y)$

B: $\exists y \ \forall x \ Loves(x,y)$

A با B هم ارز نیست.

$A \not\models B$

$B \models A$

(B از A قوی‌تر است.)

B ، A را نتیجه می‌دهد.

مثال

$\mathbb{N} = \{1, 2, 3, 4, 5, 6, 7, 8, \ldots, n, n+1, \ldots\}$

در مجموعه اعداد طبیعی:

A: $\forall x \; \exists y \quad x \leq y$ True

B: $\exists y \; \forall x \quad x \leq y$ False

چون $\mathbb{N}$ دارای max نیست.

در مجموعه‌ای که max داشته باشد:

$\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

A: True

B: True

در حذف سورِ وجود و نوشتن فرم اسکالم:

B: $\exists y \ \forall x \quad loves(x, y)$

A: $\forall x \ \exists y \quad loves(x, y)$

B: $loves(x, c_1)$ ← c_1 ثابت اسکالم است

A: $loves(x, F_1(x))$ ← $F_1(x)$ وابسته به x است.

تابع اسکالم $y = f_1(x)$.

استفاده مجدد از یک متغیر مقید:

وقتی یک متغیر برای بار دوم مقید به سوری شود، در واقع می‌توان از یک متغیر دیگر برای آن استفاده کرد.

$$\forall x\, P(x) \longrightarrow \forall x\, Q(x)$$

$$\forall x\, P(x) \longrightarrow \forall t\, Q(t)$$

$$\exists y\, P(y, A) \vee \forall y\, \exists x\, Q(x, y)$$

$$\exists y\, P(y, A) \vee \forall t\, \exists x\, Q(x, t)$$

یادآوری

$$\sim \sim P \equiv P$$

$$\sim (A \wedge B) \equiv \sim A \vee \sim B$$

$$\sim (A \vee B) \equiv \sim A \wedge \sim B$$

$$\sim (A \rightarrow B) \equiv A \wedge \sim B$$

توضیح:

$$(A \rightarrow B) \equiv \sim A \vee B$$

پس:

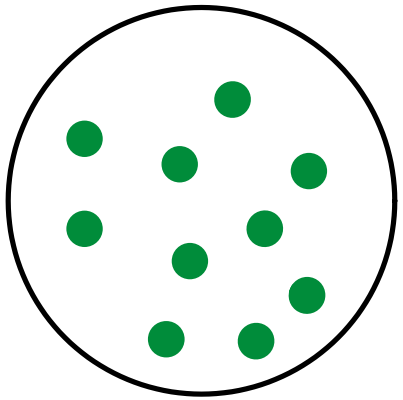
$$\sim (A \rightarrow B) \equiv A \wedge \sim B$$

توضیح مفهوم

$\forall x \quad F(x)$

نقیض

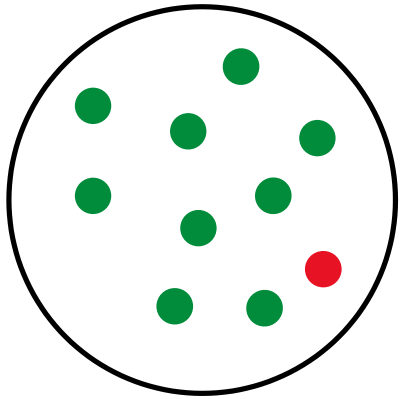
$\exists x \quad \sim F(x)$



برای رد این ادعا که: همه سبز هستند

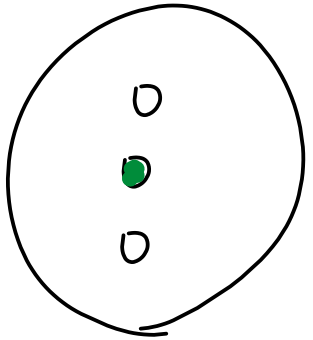
کافیت نشان دهیم:

یکی هست که سبز نیست.



توضیحات

برای ردِ این ادعا کافی: یک سبز وجود دارد



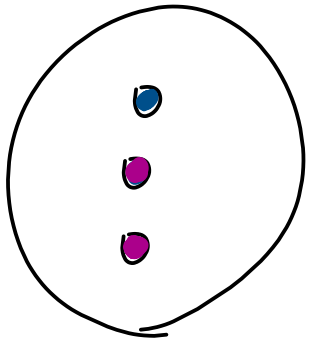
$\exists x \quad G(x)$

↓ نقیض

باید ثابت شود:

همه را بینی، سبز نیست.

$\forall x \quad \sim G(x)$



$$\neg\exists x\ P \equiv \forall x\ \neg P$$
$$\exists x\ \neg P \equiv \neg\forall x\ P$$
$$\neg\exists x\ \neg P \equiv \forall x\ P$$
$$\neg\forall x\ \neg P \equiv \exists x\ P$$

$$\sim \exists x\ \sim P(x,B) \equiv \forall x\ P(x,B)$$

$$\neg\exists x\ Likes(x,Parsnips) \equiv \forall x\ \neg Likes(x,Parsnips)$$

$$\neg\exists x\ \neg Likes(x,IceCream) \equiv \forall x\ Likes(x,IceCream)$$

مثال‌های بیشتر:

$$\forall x \; (P(x) \rightarrow Q)$$

نقیض $\sim$

$$\exists x \; (P(x) \wedge \sim Q)$$

$$\forall x \; P(x) \rightarrow Q$$

نقیض $\sim$

$$\forall x \; P(x) \wedge \sim Q$$

مثال

$$H: \quad \forall x \left[H(x) \wedge \exists y \left(\forall z\, P(z) \rightarrow Q(x,y) \right) \right]$$

$$\sim H: \quad \exists x \left[\sim H(x) \vee \forall y \left(\forall z\, P(z) \wedge \sim Q(x,y) \right) \right]$$

علامت سورها و نماد وی‌های چسبیده به سورها در نقیض، تغییری نمی‌کنند.

$$H: \forall x \in A \quad x^2 \geq 1 \quad \equiv \quad H: \forall x \ (x \in A \longrightarrow x^2 \geq 1)$$

نقیض ↓ نقیض ↓

$$\sim H: \exists x \in A \quad x^2 < 1 \quad \equiv \quad \sim H: \exists x \ (x \in A \wedge x^2 < 1)$$

در اسلایدهای بعدی

R را به جای Richard

$B(_,_)$ را به جای Brother$(_,_)$ استفاده کردیم.

$Brother(x, Richard)$ یعنی x برادر ریچارد است.

To say that: **Richard has at least one brothers**

$$\exists x \; B(x, R)$$

حداقل یک برادر

To say that: **Richard has at most one brothers**

$$\forall x \; \forall y \; \left(B(x,R) \wedge B(y,R) \longrightarrow x = y\right)$$

حداکثر یک برادر

To say that: **Richard has exactly one brother**

$B(x,R)$

دقیقاً یک برادر

$$\exists x \; B(x,R) \; \wedge \; \forall z \; \forall y \; \left(B(y,R) \wedge B(z,R) \longrightarrow z = y\right)$$

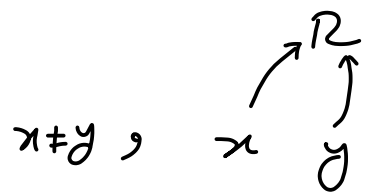
یا به این صورت:

$$\exists x \; B(x,R) \; \wedge \; \forall y \; \left(B(y,R) \longrightarrow y = x\right)$$

To say that: **Richard has at least two brothers**

$$\exists x \; \exists y \; \left(B(x, R) \wedge B(y, R) \wedge x \neq y \right)$$

حداقل ۲ برادر



To say that: **Richard has exactly two brothers.**

دقیقاً ۲ برادر:

$$\exists x \; \exists y \; \left(B(x, R) \wedge B(y, R) \wedge x \neq y \wedge \forall z \left(B(z, R) \longrightarrow z = x \vee z = y \right) \right)$$

حداقل ۲ برادر x و y دارد.

برادر دیگری وجود ندارد.

چند هم ارزی مهم:

① $\forall x \left(P(x) \wedge Q(x)\right) \equiv \forall x\, P(x) \wedge \forall x\, Q(x)$

② $\exists x \left(P(x) \vee Q(x)\right) \equiv \exists x\, P(x) \vee \exists x\, Q(x)$

③ $\exists x \left(P(x) \wedge Q(x)\right) \not\leftarrow \exists x\, P(x) \wedge \exists x\, Q(x)$

④ $\forall x \left(P(x) \vee Q(x)\right) \not\rightarrow \forall x\, P(x) \vee \forall x\, Q(x)$

این گزاره که تاتولوژی نیستند. این نتیجه‌گیری‌ها نادرستند. البته درحالت کلی نادرستند اما درحالات خاصی ممکن است معتبر باشند یعنی ارضاءپذیر هستند.

$$\exists x \; P(x) \wedge Q(x) \overset{?}{\leftarrow} \exists x \, P(x) \; \wedge \; \exists x \, Q(x)$$

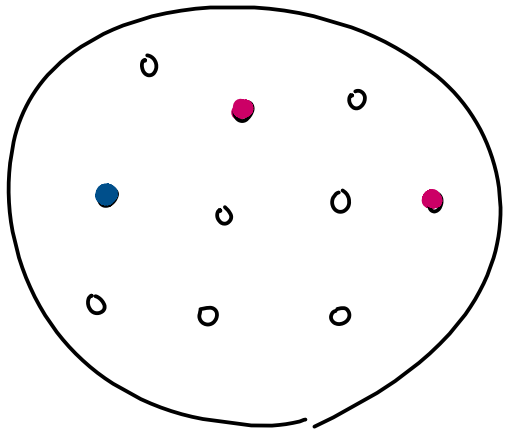
$$\forall x \; P(x) \vee Q(x) \overset{?}{\rightarrow} \forall x \, P(x) \; \vee \; \forall x \, Q(x)$$

مثال از نادرست بودن این نتیجه گیری:

$$\exists x \left(P(x) \wedge Q(x)\right) \overset{?}{\leftarrow} \left(\exists x\, P(x)\right) \wedge \left(\exists x\, Q(x)\right)$$

$$\exists x\, P(x) \wedge \exists x\, Q(x) \not\models \exists x \left(P(x) \wedge Q(x)\right)$$

$\exists x\, P(x)$: True — $\exists x\, Q(x)$: True — $\exists x \left(P(x) \wedge Q(x)\right)$: False



$P(x)$: آبی بودن x

$Q(x)$: قرمز بودن x

مثال از نادرست بودن این نتیجه گیری:

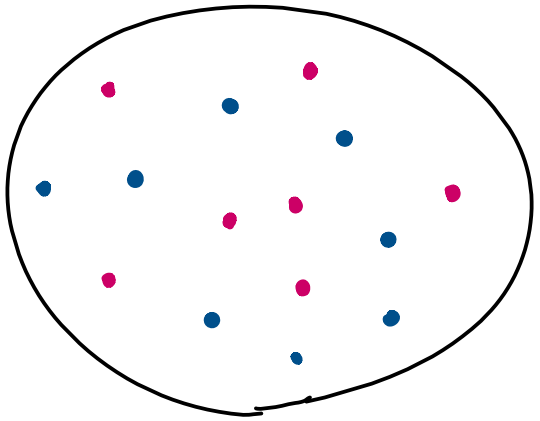
$$\forall x \; P(x) \vee Q(x) \xrightarrow{?} \forall x \; P(x) \vee \forall x \; Q(x)$$

$P(x)$ و $Q(x)$ هرکدام به معنای صوت و منکر بودن باشند.

$\forall x \; P(x) \vee Q(x)$ هوا نم، یا صوت است یا منکر.

اما چه‌بی $\forall x \; P(x)$ غلط است.

$\forall x \; Q(x)$ غلط است.



پس:

$$\forall x \; P(x) \vee Q(x) \models \forall x \; P(x) \vee \forall x \; Q(x)$$

True False False

۱۰ـ با این فرض که متغیر x در Q به صورت آزاد (*Free*) ظاهر نشده است، مقدار کدام یک از عبارات زیر در منطق مسندات (*Predicate Logic*) نادرست (*False*) است؟

۱) $(\exists x\,(p(x)\rightarrow Q))\rightarrow(\forall x\;\; p(x)\rightarrow Q)$

۲) $(\forall x\,p(x)\rightarrow Q)\rightarrow(\forall x\;\;(p(x)\rightarrow Q))$

۳) $(\exists x\,p(x)\rightarrow Q)\rightarrow(\forall x\;\;(p(x)\rightarrow Q))$

۴) $(\exists x\,p(x)\rightarrow Q)\rightarrow(\exists x\;\;(p(x)\rightarrow Q))$

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۱۴ – فصل ۸ و فصل ۹

۱۰ـ با این فرض که متغیر x در Q به صورت آزاد (*Free*) ظاهر نشده است، مقدار کدام یک از عبارات زیر در منطق مسندات (*Predicate Logic*) نادرست (*False*) است؟

۱) $(\exists x\,(p(x)\rightarrow Q))\rightarrow(\forall x\;\; p(x)\rightarrow Q)$

۲) $(\forall x\,p(x)\rightarrow Q)\rightarrow(\forall x\;\;(p(x)\rightarrow Q))$

۳) $(\exists x\,p(x)\rightarrow Q)\rightarrow(\forall x\;\;(p(x)\rightarrow Q))$

۴) $(\exists x\,p(x)\rightarrow Q)\rightarrow(\exists x\;\;(p(x)\rightarrow Q))$

۱۰ـ با این فرض که متغیر x در Q به صورت آزاد (*Free*) ظاهر نشده است، مقدار کدام یک از عبارات زیر در منطق مسندات (*Predicate Logic*) نادرست (*False*) است؟

C D

۱) $(\exists x\,(p(x) \rightarrow Q)) \rightarrow (\forall x\ p(x) \rightarrow Q)$

D A

۲) $(\forall x\, p(x) \rightarrow Q) \rightarrow (\forall x\ (p(x) \rightarrow Q))$ → غلط

B A

۳) $(\exists x\, p(x) \rightarrow Q) \rightarrow (\forall x\ (p(x) \rightarrow Q))$

B C

۴) $(\exists x\, p(x) \rightarrow Q) \rightarrow (\exists x\ (p(x) \rightarrow Q))$

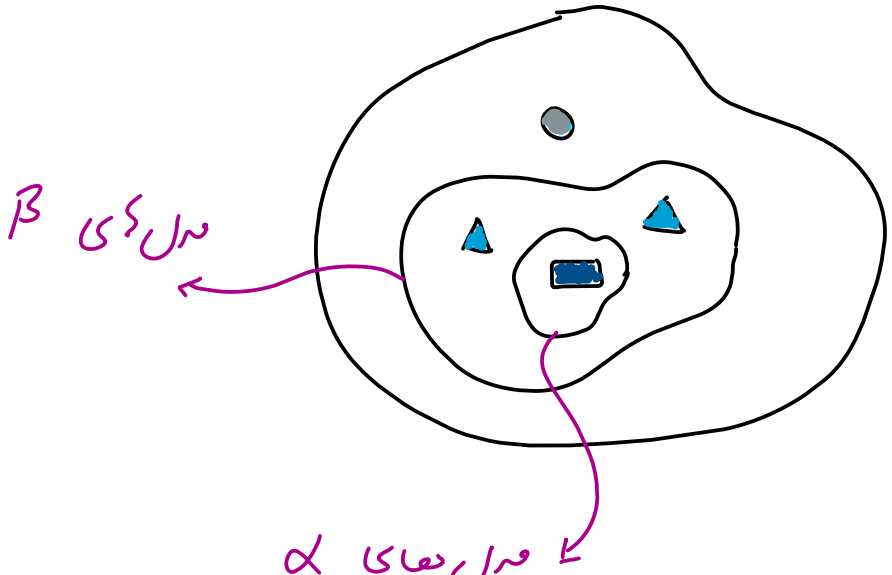
A: $\forall x\ (p(x) \longrightarrow Q)$

B: $\exists x\ p(x) \longrightarrow Q$

C: $\exists x\ (p(x) \longrightarrow Q)$

D: $\forall x\ p(x) \longrightarrow Q$

یادآوری: اگر مجموعه مدل‌های درستی α زیرمجموعه‌ی مدل‌های درستی β باشد آنگاه $\alpha \models \beta$



$$m(\alpha) \subseteq m(\beta)$$

$$\alpha \models \beta$$

A: $\forall x \ (P(x) \longrightarrow Q)$

B: $\exists x \ P(x) \longrightarrow Q$

C: $\exists x \ (P(x) \longrightarrow Q)$

D: $\forall x \ P(x) \longrightarrow Q$

نتیجه می‌دهم ارتباط منطقی چنین است:

A

(↑) } معادله A, B

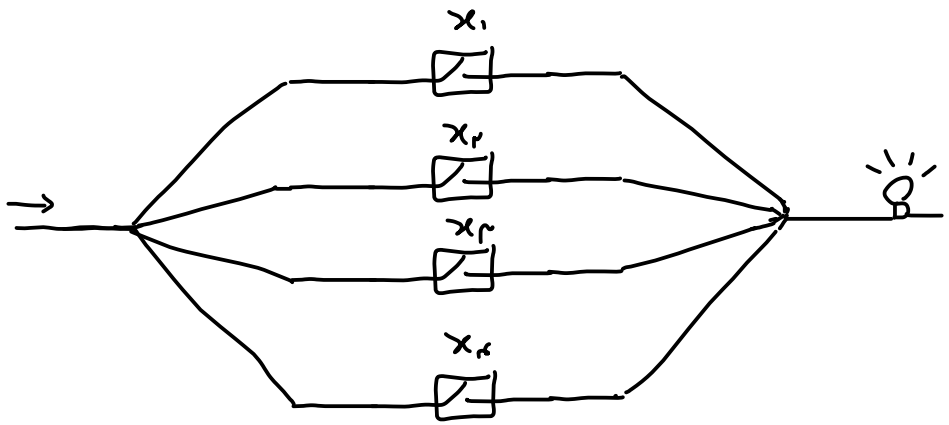
B

↓

C

↓

D



$P(x)$: اتصال طبیِ x

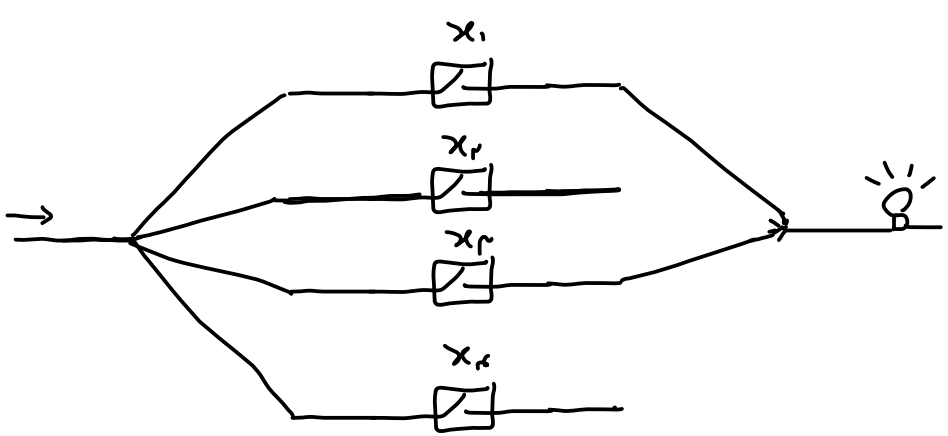
Q : روشن بودن لامپ

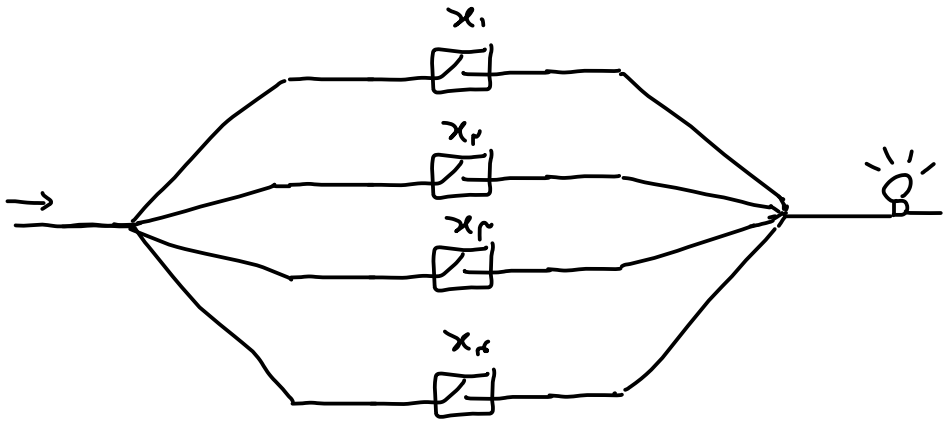
در این مدار :

A: $\forall x \quad (P(x) \longrightarrow Q)$

B: $\exists x \; P(x) \longrightarrow Q$

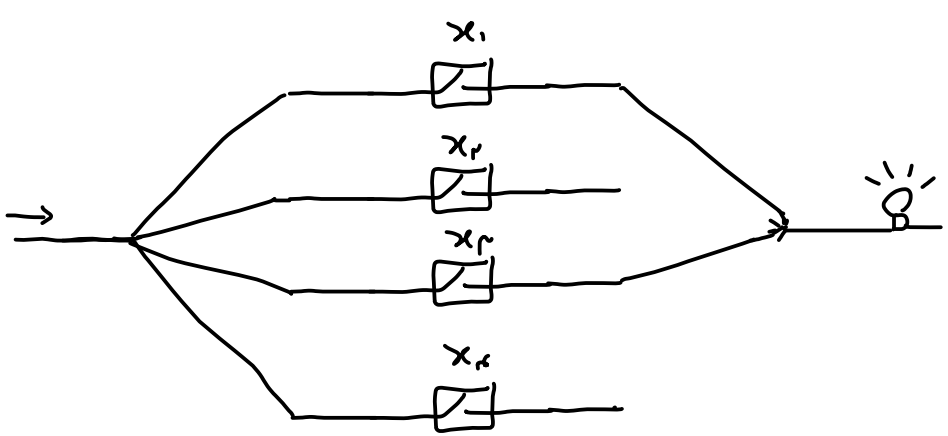
A و B هر دو معادل با هم هستند و این مدار را توصیف می‌کنند.

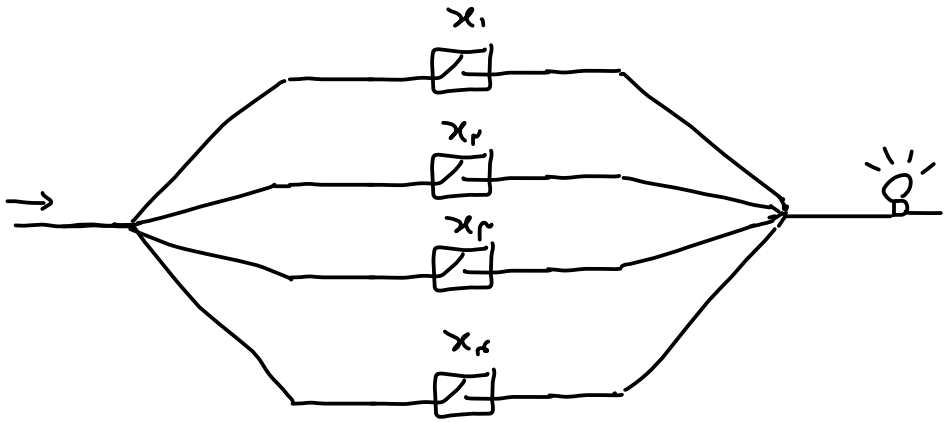




جمله‌ی C علاوه بر مدار قبل در این مدار هم درست است.

$$C: \quad \exists x \quad \left(P(x) \longrightarrow Q \right)$$







جمله‌ی D علاوه بر او و مدار قبل در این مدار هم درست است.

$$D: \quad \forall x \; p(x) \longrightarrow Q$$

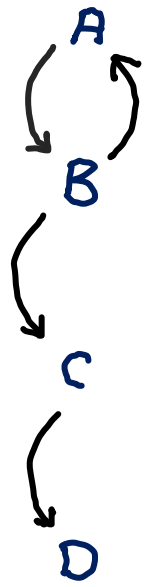
دیدیم که: $M(A)=M(B) \subseteq M(C) \subseteq M(D)$ پس:

A: $\forall x \ (P(x) \longrightarrow Q)$

B: $(\exists x \ P(x)) \longrightarrow Q$

C: $\exists x \ (P(x) \longrightarrow Q)$

D: $(\forall x \ P(x)) \longrightarrow Q$



KB
- Greedy(t),
- King(Richard),
- (King(x) ∧ Greedy(x) ⇒ Evil(x))

برای استفاده از عبارات در FOL به جانشینی نیاز داریم.

جانشینی‌های درست به این انواع هستند:

- x/Ali — به جای متغیر x یک ثابت قرار دهیم.
- $x/Father(Ali)$ — x یک تابع از ثابت قرار دهیم.
- x/z — x ، متغیری دیگر قرار دهیم
- به جای متغیر x، توابعی غیروابسته به x قرار دهیم.
  - $x/F(y)$
  - $x/F(y,z)$

مثال:

α: King(x) $\wedge$ Greedy(y) $\Rightarrow$ Evil(x))

لیست جانشانی: $\theta = \{ x/John , y/Father(John) \}$

حاصل جانشانی θ در α:

SUBST(θ, α) = King(John) $\wedge$ Greedy(Father(John)) $\Rightarrow$ Evil(John))

Var / Term

متغیری مانند x / یک ترم غیروابسته به x

چند نمونه جانشینی های قابل انجام:

x / A

x / F(A)

x / y

x / F(y)

x / F(y, A)

مثال کدام جانشینی در α مجاز است.

$\alpha: Q(F(x), H(y,A))$

$y/F(y)$	$F(x)/F(A)$	A/x
$y/H(y,A)$	$F(x)/B$	A/B
	$F(x)/y$	

جانشینی‌های نادرست (غیرمجاز)

x/B

$x/F(y)$

$x/H(B,A)$

y/x

$y/F(x)$

$y/F(B)$

جانشینی‌های درست (مجاز)

نمونه‌سازی وقتی جملات شامل سور $\forall x\, \alpha$ ، $\exists x\, \alpha$ در KB داریم برای استفاده از آنها و انجام استنتاج یک ابزار، نمونه‌سازی است یعنی استخراج جملات زمینه‌ای از آنها. روش کار برای $\forall x$ ، $\exists x$ متفاوت است.

نمونه‌سازی از سور کلی: جمله‌ی $\forall x\ \alpha$ را در KB داریم. جانشانی x/g را برای هر ترم زمینه‌ای g می‌توان انجام داد و $SUBST(\theta,\alpha)$ را استنتاج کرد. این کار قابل تکرار است.

$$\frac{\forall x\ \alpha}{SUBST(\theta,\alpha)}$$

$\theta = \{x/g\}$

ground term g

g یک ترم بدون متغیر است.

یعنی ترم زمینه‌ای است.

مثال: فرض کنید KB چنین باشد:

KB:
- $\forall x \quad King(x) \Rightarrow Evil(x)$
- $King(John)$
- $\sim Evil(Mary)$

با جانشانی $\theta = \{x/John\}$ یک نمونه سازی از سور کلی انجام می‌دهیم.

حالا داریم:

KB:
- $\forall x \quad King(x) \Rightarrow Evil(x)$
- $King(John)$
- $\sim Evil(Mary)$
- $King(John) \Rightarrow Evil(John)$

بازهم $\forall x \ \alpha$ را در KB داریم. نمونه سازی قابل تکرار است:

با جانشانی $\theta = \{x/Mary\}$ و یکسان‌سازی:

KB

$\forall x \quad King(x) \Rightarrow Evil(x)$

$King(John)$

$\sim Evil(Mary)$

$King(John) \Rightarrow Evil(John)$

$King(Mary) \Rightarrow Evil(Mary)$

با جانشانی $\theta = \{x/Father(Mary)\}$ و یکسان‌سازی:

KB

$\forall x \quad King(x) \Rightarrow Evil(x)$

$King(John)$

$\sim Evil(Mary)$

$King(John) \Rightarrow Evil(John)$

$King(Mary) \Rightarrow Evil(Mary)$

$King(Father(Mary)) \Rightarrow Evil(Father(Mary))$

نمونه‌سازی از سور وجودی:

skolemization

جمله‌ی $\exists x\ \alpha$ را در KB داریم. فقط یک بار حق نمونه‌سازی وجودی داریم. باید به جای x یک ثابت اسکالم یا تابع اسکالم قرار دهیم. اسکالم یعنی ثابت یا تابعی که تا آن مرحله در KB نامی از آن برده نشده باشد.

$$\frac{\exists x\ \alpha}{SUBST(\theta, \alpha)} \qquad \theta = \{x/skolem\}$$

معمولاً از اندیس برای اطمینان از اسکالم بودن استفاده می‌شود. مثلاً x/A_1

$x/F_1(A)$

⋮

مثال: فرض کنید KB چنین باشد:

KB
- $\exists x \quad King(x) \Rightarrow Evil(x)$
- $King(John)$
- $\sim Evil(Mary)$

برای عمومی سازی حق نداریم $x/John$ و $x/Mary$ انجام دهیم زیرا این ها ارکان نیستند

از جانشینی x/C_1 استفاده می‌کنیم و جمله‌ی $\exists x \, \alpha$ از KB حذف خواهد شد.

KB
- $King(John)$
- $\sim Evil(Mary)$
- $King(C_1) \Rightarrow Evil(C_1)$

یادآوری

اگر $\exists x\ \alpha$: x وابسته به دیگر متغیرها نیست؛ x/c_1

اگر $\forall x\ \exists y\ \alpha$ y وابسته به x است: $y/F_1(x)$

اگر $\forall x\ \forall y\ \exists z\ \alpha$ z وابسته به x و y است: $z/F_1(x,y)$

فرم کلازال و skolemization :

ابتدا گزاره‌های دوشرطی و شرطی را حذف می‌کنیم. نقیض را در صورت وجود پخش می‌کنیم (فرم CNF)
آنگاه به جای متغیرهای سور وجودی، عبارات اسکالم قرار می‌دهیم.

مثال: فرم اسکالم رانویسید:

$$\exists y \ \forall x \ \left(P(x,y) \vee \exists z \ Q(y,z) \right)$$

پاسخ: y/c_1 و $z/F_1(x)$:

$$P(x,c_1) \vee Q(c_1, F_1(x))$$

مثال: فرم اسکالم را بنویسید.

$$\forall x \ \forall y \left(P(x,y) \vee \exists z \ Q(y,z) \right)$$

پاسخ:

$$P(x,y) \vee Q(y, F_1(x,y))$$

۴۷ـ کدام مورد در خصوص انجام عمل *Skolemization* درست نیست؟ (مهندسی IT ـ دولتی ۹۸)

۱) $\forall x\,(\exists y\,(P(x)\wedge\neg Q(x,y))\vee\exists z\,(Q(z,x))$

$\forall x\,(P(x)\wedge\neg Q(x,F_1(x))\vee Q(F_2(x),x))$

۲) $\exists x\,(\forall y\,(P(x)\wedge\neg Q(x,y))\vee\exists z\,(Q(z,x))$

$\forall y\,(P(c_1)\wedge\neg Q(c_1,y))\vee Q(c_2,c_1)$

۳) $\exists x\,\forall y\,P(x,y)$

$\forall y\,P(F_1(y),y)$

۴) $\exists x\,(P(x)\wedge Q(x,A))$

$P(c_1)\wedge Q(c_1,A)$

۴۷ـ کدام مورد در خصوص انجام عمل *Skolemization* درست نیست؟ (مهندسی IT ـ دولتی ۹۸)

۱) $\forall x\,(\exists y\,(P(x)\wedge\neg Q(x,y))\vee\exists z\,(Q(z,x))$

$\forall x(P(x)\wedge\neg Q(x,F_1(x))\vee Q(F_2(x),x))$

۲) $\exists x\,(\forall y\,(P(x)\wedge\neg Q(x,y))\vee\exists z\,(Q(z,x))$

$\forall y\,(P(c_1)\wedge\neg Q(c_1,y))\vee Q(c_2,c_1)$

۳) $\exists x\,\forall y\,P(x,y)$

$\forall y\,P(F_1(y),y)$ ✓ غلط

باید x/c_1 قرار دهیم.

۴) $\exists x\,(P(x)\wedge Q(x,A))$

$P(c_1)\wedge Q(c_1,A)$

گزینه ۱: درست است زیرا $\exists y$ و $\exists z$ درون $\forall x$ قرار دارند. پس $z/F_2(x)$ ، $y/F_1(x)$ هم نباید گذاری باشند (انتظام باشند)

گزینه ۲: درست است. $\exists x$ و $\exists z$ هیچکدام درون ناحیه‌ی تاثیر $\forall y$ قرار ندارند.

گزینه ۳: هم درست است.

الگوریتم‌های استنتاج در منطق مرتبه اول:

اولین ایده، تبدیل از FOL به PL است یعنی پایگاه دانش گزاره‌های منطقی (زمینه‌ای) تبدیل کنیم. ـ جای سورهای کلی از عبارات انعکاس و ـ جای سور کلی $\forall x \dots$ تمام ثابت‌ها و ترم‌های زمینه‌ای دیگر را جایگزین کردن و ـ استنتاج در PL رو بیاوریم. (ساده‌سازی از FOL به PL)

به شرط آن که تابع نداشته باشیم تعداد گزاره‌های تولید شده متناهی خواهد بود. [تعداد ثابت‌ها و اعضای مدل را متناهی فرض کردیم]

مثال در یک دنیا با ۳ عضو $\{John, Bill, Mary\}$:

FoL – KB

$$\begin{cases} King(x) \longrightarrow Evil(x) \\ King(John) \end{cases}$$

$\longrightarrow$

PL – KB

$$\begin{cases} King(John) \longrightarrow Evil(John) \\ King(Bill) \longrightarrow Evil(Bill) \\ King(Mary) \longrightarrow Evil(Mary) \\ King(John) \end{cases}$$

سادسازی از FoL به PL در حضور نماد تابعی: نماد تابعی $F(\dots)$ و یک ثابت A را تصور کنید:

FoL_KB

$$\begin{cases} P(x) \rightarrow P(F(x)) \\ P(A) \end{cases}$$

PL_KB

$$\begin{cases} P(A) \\ P(A) \rightarrow P(F(A)) & \text{عمق 1} \\ P(F(A)) \rightarrow P(F(F(A))) & \text{عمق 2} \\ P(F(F(A))) \rightarrow P(F(F(F(A)))) & \text{عمق 3} \\ \vdots \end{cases}$$

اگر تعداد ثابت‌ها متناهی باشد و تعداد نمادهای رابط متناهی باشد و نماد تابعی حضور نداشته باشد
آنگاه تعداد جملاتی که در PL تولید می‌شود متناهی است. پس الگوریتم‌های (مانند رزولوشن)
استنتاج در PL قابل استفاده‌اند و این روش کامل است: اگر α قابل استنتاج باشد
استنتاج خواهد شد و اگر قابل استنتاج نباشد بازهم الگوریتم بالاخره متوقف خواهد شد.

اما در حضور نماد تابعی، تعداد جملات در PL نامتناهی است با این حال بازهم می‌توان
مرحله به مرحله عمق را افزایش داد و در هر عمق استنتاج $KB \models ? \alpha$ را بررسی کرد.

اگر α قابل استنتاج باشد ← الگوریتم با موفقیت متوقف می‌شود.
اگر α قابل استنتاج نباشد ← الگوریتم هرگز متوقف نمی‌شود. پس:

روش ساده سازی به منطق مرتبه اول:

کامل است، حتی وقتی ناسازگاری داشته باشیم.

اما اگر $KB \# \alpha$ آنگاه الگوریتم متوقف نمی‌شود.

> مسئله‌ی استنتاج در منطق مرتبه اول نیمه تصمیم پذیر است.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۱۵ فصل ۹

کاهش پایگاه دانش منطق مرتبه اولFOL به منطق گزاره‌ای LP

FOL

$$\forall x \;\; King(x) \wedge Greedy(x) \;\Rightarrow\; Evil(x)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)\ .$$

باعبارت‌های ثابت‌ها به جای x:

LP

$$King(John) \wedge Greedy(John) \;\Rightarrow\; Evil(John)$$
$$King(Richard) \wedge Greedy(Richard) \;\Rightarrow\; Evil(Richard)\ ,$$

$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)\ .$$

وجود نماد تابعی در پایگاه دانش منطق مرتبه اول باعث می‌شود تعداد لیترال‌ها و جملات در منطق گزاره ای نامتناهی شود.

$$Father(Father(Father(John)))$$

یادآوری:

مسئله ارضاپذیری و مسئله استنتاج در FOL تصمیم‌پذیر است.

اگر α قابل استنتاج از KB باشد : الگوریتم متوقف می‌شود.

اگر α قابل استنتاج از KB نباشد : الگوریتم ممکن است متوقف نشود.

یکسان‌سازی Unification

لیست جانشینی θ یک لیست یکسان‌ساز برای P و q است اگر و تنها اگر:

$$SUBST(\theta, p) = SUBST(\theta, q)$$

در این حالت می‌گوییم: $UNIFY(p, q) = \theta$

مثال: لیت یک ن ب ز این دو جمله را بیابید:

P: Likes(x , John)

Q: Likes(Mary , y)

مولفه های اول با هم مولفه های دوم با هم باید یک ن شوند

θ: x/Mary y/John

استانداردسازی: اگر دوگزاره‌ی P و Q دارای متغیرمشترک باشند با تغییرنام آن متغیراین اشتراک را حذف کنیم.
استانداردسازی پیش از بررسی یکان سازی، مجاز است.

Knows (John , x)
Knows (x , Elizabet)

استانداردسازی →

Knows (John , x)
Knows (z , Elizabet)

یکان سازی →

θ: x/Elizabet y/John

مثال: لیت یکسان‌سازی این دو جمله را بیابید:

1: Q(G(x,x) , y)

2: Q(y , G(A,B))

استانداردسازی →

1: Q(G(x,x) , y)

2: Q(z , G(A,B))

یکسان‌سازی →

θ: $z/G(x,x)$ $y/G(A,B)$

مولفه‌ای اول با هم مولفه‌های دوم با هم باید یکسان شوند

مثال: در مثال قبل مجموعه‌ترین لیست یک‌ن‌ساز را بنویسید:

1: Q(G(x,x) , y)

2: Q(y , G(A,B))

استانداردسازی ⟵

1: Q(G(x,x) , y)

2: Q(z , G(A,B))

جایگزینی‌های این یک‌ن‌ساز، الزامی هستند:

مجموعه‌ترین یک‌ن‌ساز ⟵

θ: $z/\mathrm{G(x,x)}$ $y/\mathrm{G(A,B)}$

البته می‌توانیم یک‌ن‌ساز حالت خاص هم بنویسیم. برای مثال به دلخواه می‌توان x/A قرار داد:

θ: x/A $\mathrm{z/G(A\,,A)}$ $y/\mathrm{G(A,B)}$

مثال: لیت یک ن‌ساز این دو جمله را بیابید:

1: Knows(x , x)

2: Knows(F(y) , y)

پاسخ: این جملات قابل یک‌ن‌سازی نیستند. زیرا اگر بخواهیم مولفه‌ها یکسان شوند باید داشته باشیم:

مولفه‌اول: $x / F(y)$ ← جایگزین

1: Knows(F(y) , F(y))

2: Knows(F(y) , y)

حال در مولفه دوم y و $F(y)$ با هم یکسان شوند اما جانشینی $y / F(y)$ غیرمجاز است.

نشانه‌های شکست در یکسان‌سازی Hints

① ناچار شویم به جای یک متغیر دو عبارت غیر یکسان شده قرار دهیم.

مثلاً ناچار شویم هم‌زمان x/c_1 و x/c_2 قرار دهیم.

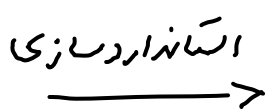
② ناچار شویم به جای x، تابعی برحسب x قرار دهیم مثلاً

$$x/F(x)$$

مثال: لیت کیک ن‌ساز این دو جمله را بیابیه. اگر استانداردسازی نکنیم چطور؟

1: x
2: F(x)

پاسخ: این جملات بدون استانداردسازی قابل کیک ن‌سازی نیستند. اما با استانداردسازی:



1: x
2: F(y)



کیک ن‌سازی با جانشان x / F(y)

مثال: لیست یکان ساز این دو جمله را بیابید.

1: $Q(G(w), H(w, I(x, u)))$

2: $Q(G(v), H(u, v))$

استانداردسازی ⟵

1: $Q(G(w), H(w, I(x, u)))$

2: $Q(G(v), H(z, v))$

لیست یکان ساز ⟵

θ: $v/I(x, u)$, $w/I(x, u)$, $z/I(x, u)$

در مولفه اول H باید w/z قرار دهیم اما پیش از آن $w/I(x,u)$

دس در واقع $z/I(x,u)$.

قانون قیاس استثنائی (Modus Ponens) در منطق گزاره‌ای

قانون ساده MP

$$\textbf{MP:}\quad \frac{p,\ p \to q}{\therefore\ q}$$

قانون کلی GMP

$$\textbf{GMP:}\quad \frac{p_1, p_2, \ldots, p_k,\ p_1 \wedge p_2 \wedge \ldots \wedge p_k \to q}{\therefore\ q}$$

قانون قیاس استثنائی در منطق مرتبه اول

اگر در پایگاه دانش p' و $p \rightarrow q$ را داشته باشیم و لیست جانشانی θ یکسان‌ساز p' و p باشد می‌توان با جانشانی θ در جملات، $SUBST(\theta, q)$ را نتیجه گرفت.

MP:

$$\frac{p',\ p \rightarrow q}{\therefore\ SUBST(\theta, q)} \qquad \theta = UNIFY(p', p)$$

شکل کلی‌تر MP نیز چنین است: لیست جانشانی θ باید یکسان‌ساز هر p'_i با p_i باشد.

GMP:

$$\frac{p'_1, p'_2, \ldots, p'_k,\ \ p_1 \wedge p_2 \wedge \ldots \wedge p_k \rightarrow q}{\therefore\ SUBST(\theta, q)}$$

:

$$\theta = UNIFY(p'_i, p_i)$$

$$SUBST(\theta, p_i') = SUBST(\theta, p_i)$$

مثال:

در FOL برای استفاده از MP باید به یکسان سازی فرض ها توجه کرد.

لیست $\theta = \{x/A\}$ باعث یکسان سازی $P(x)$ و $P(A)$می‌شود پس با جانشانی آن می‌توان از MP استفاده کرد و $Q(A)$ را نتیجه گرفت.

$$\{ P(A) , P(x) \rightarrow Q(x) \} \models Q(A)$$

$$\downarrow$$

$$\theta = \{x/A\}$$

مثال:

لیست $\theta = \{x/John\}$ باعث یکسان سازی $King(x)$ و $King(John)$ می‌شود

پس با جانشانی آن می‌توان از MP استفاده کرد و $Person(John)$ را نتیجه گرفت.

MP:

$$\frac{King(John) \quad , \quad King(x) \Rightarrow Person(x)}{Person(John)}$$

مثال:

لیست $\theta = \{x/John\}$ باعث یکسان‌سازی King(x) با King(John) همچنین Greedy(x) با Greedy(John) می‌شود پس با جانشانی آن می‌توان از GMP استفاده کرد و Evil(John) را نتیجه گرفت.

P_1 P_2 q

$\forall x \quad King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

P_1' $King(John)$

P_2' $Greedy(John)$

$Brother(Richard, John)$.

{x/John}

$\models$ Evil(John)

A simple forward-chaining algorithm

زنجیره‌سازی پیشرو (یک پیاده سازی ساده):

function FOL-FC-ASK(KB, α) **returns** a substitution or *false*
  inputs: KB, the knowledge base, a set of first-order definite clauses
    α, the query, an atomic sentence
  local variables: *new*, the new sentences inferred on each iteration

  repeat until *new* is empty
    new $\leftarrow \{\}$
    for each *rule* **in** KB **do**
      $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES(*rule*)
      for each θ such that SUBST($\theta, p_1 \wedge \ldots \wedge p_n$) = SUBST($\theta, p'_1 \wedge \ldots \wedge p'_n$)
          for some $p'_1, \ldots, p'_n$ in KB
        $q' \leftarrow$ SUBST(θ, q)
        if q' does not unify with some sentence already in KB or *new* **then**
          add q' to *new*
          $\phi \leftarrow$ UNIFY(q', α)
          if ϕ is not *fail* **then return** ϕ
    add *new* to KB
  return *false*

زنجیره‌سازی:

KB : پایه‌ی معین

ورودی: KB ، α

خروجی: یک لیست جایگزینی θ یا (false)

شروع: fact های KB.

انتها: یا رسیدن به معادل α یا رسیدن به Fixed point

کامل بودن: اگر KB پایه معین باشه، کامل است. (اگر α قابل استنتاج باشه، لیست θ را خواهد یافت)

مثال: آیا $KB \models \alpha$? با زنجیره سازی پیشرو بررسی کنید.

KB:

۱) $\forall x\ Shiny(x) \rightarrow Niceweather(x)$

۲) $\forall x\ \forall y\ Healthy(x) \wedge Niceweather(y) \rightarrow Gotoswim(x, y)$

۳) $\forall x\ Gotoswim(x, Friday) \rightarrow Healthy(x)$

۴) $Shiny(Saturday)$

۵) $Healthy(Amin)$

۶) $Gotoswim(Ali, Friday)$

α: $\models$?
Gotoswim(Amin , Friday)

جمله KB همگی فرم معین دارند و به شکل نرمال معین به الگوریتم داده می‌شوند:

گزاره‌های شرطی به فرم کلی (rule) قانون یا قواعد نام دارند

لیترال‌های مثبت، (fact) یا حقایق نام دارند.

rules

$$Shiny(x) \Rightarrow Niceweather(x)$$

$$Healthy(x) \wedge Niceweather(y) \Rightarrow Gotoswim(x, y)$$

$$Gotoswim(x, Friday) \Rightarrow Healthy(x)$$

facts

$$Shiny(Saturday)$$

$$Healthy(Amin)$$

$$Gotoswim(Ali, Friday)$$

$$\stackrel{?}{\models} Gotoswim(Amin, Friday)$$

① $Shiny(x) \Rightarrow Niceweather(x)$

② $Healthy(z) \wedge Niceweather(y) \Rightarrow Gotoswim(z, y)$

③ $Gotoswim(t, Friday) \Rightarrow Healthy(t)$

$\models$ $Gotoswim(Amin, Friday)$

fact {
- $Shiny(Saturday)$
- $Healthy(Amin)$
- $Gotoswim(Ali, Friday)$

یک مجموعه‌ی I داریم که مجموعه‌ی لیترال‌های اثبات شده (مقبر) در آن قرار دارد.

در شروع فقط fact ها در I قرار دارند.

I:
- ✓ Shiny(Saturday)
- ✓ Healthy(Amin)
- ✓ Gotoswim(Ali , Friday)

در هر تکرار همه rule ها بررسی میشوند هرکدام که با MP بتوان از آن نتیجه گرفت بالبت جانشی مناسب به نتایج حدیدی میرسند که به I اضافه میشوند:

① Shiny(x) ⇒ Niceweather(x)

مثلاً از قاعده‌ی ① با x/Saturdy داریم: Niceweather(Saturday) و این به I اضافه میشود:

I:
- ✓ Healthy(Amin)
- ✓ Shiny(Saturday)
- ✓ Gotoswim(Ali , Friday)
- ✓ Niceweather(Saturday)

حالا از ② با z/Amin , y/Saturday داریم:

② Healthy(z) ∧ Niceweather(y) ⇒ Gotoswim(z , y)

⊨ Gotoswim(Amin , Saturday)

به I اضافه می‌کنیم:

I:
- ✔ Healthy(Amin)
- ✔ Shiny(Saturday)
- ✔ Gotoswim(Ali , Friday)
- ✔ Niceweather(Saturday)
- ✔ Gotoswim(Amin , Saturday)

به همین ترتیب ادامه می‌دهیم. در پایان

اگر $\alpha \in I$ شود، خروجی الگوریتم شامل لیبت جانشانی θ است که استنتاج α را تایید می‌کند.

اگر در مرحله‌ای، هیچ لیترال جدیدی قابل استنتاج نباشد ($new = \{\}$) و α هنوز حاصل نشده باشد، (رسیدن به نقطه ثابت ـ fixed point) استنتاج شکست خورده یعنی α استنتاج نمی‌شود.

توجه: درصورت وجود تناقض، اگر α قابل استنتاج نباشد ممکن است در loop گرفتار شود و متوقف نشود.

مثال: آیا $KB \overset{?}{\models} \alpha$.

$$KB \begin{cases} \textcircled{1}\ P(x) \longrightarrow P(F(x)) \\ P(A) \\ P(B) \\ Q(B, C) \end{cases} \qquad \overset{?}{\models}\ \alpha : P(C)$$

در هر گام با جایگزاری x/A سپس $x/F(A)$ سپس $x/F(F(A))$ ..... نتایج $P(F(A))$ و $P(F(F(A)))$ و .... حاصل می‌شود هرگز نه به α می‌رسیم و نه new کمی می‌شود.

ابتدا: $I = \{ p(A) , p(B) , Q(B,C) \}$

حالا با x/A جمله‌ی $p(FIA))$ به دست می‌آید.

با $x/F(A)$ جمله‌ی $p(F(FIA)))$ به دست می‌آید

و .....

این کار تا بی‌نهایت ادامه دارد. اما هرگز به $\alpha : p(C)$ نمی‌رسد.

هرگز به قطعی ثابت هم نمی‌رسد.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

بخش دوم جلسه ۱۵ فصل ۹

بازگشت‌آوری بیشتر FC

① MRV : از گزاره‌هایی شروعی که متغیرهای کمتری دارند آغاز کنیم
از گزاره‌هایی که تعداد معدودات کمتری دارند آغاز کنیم.

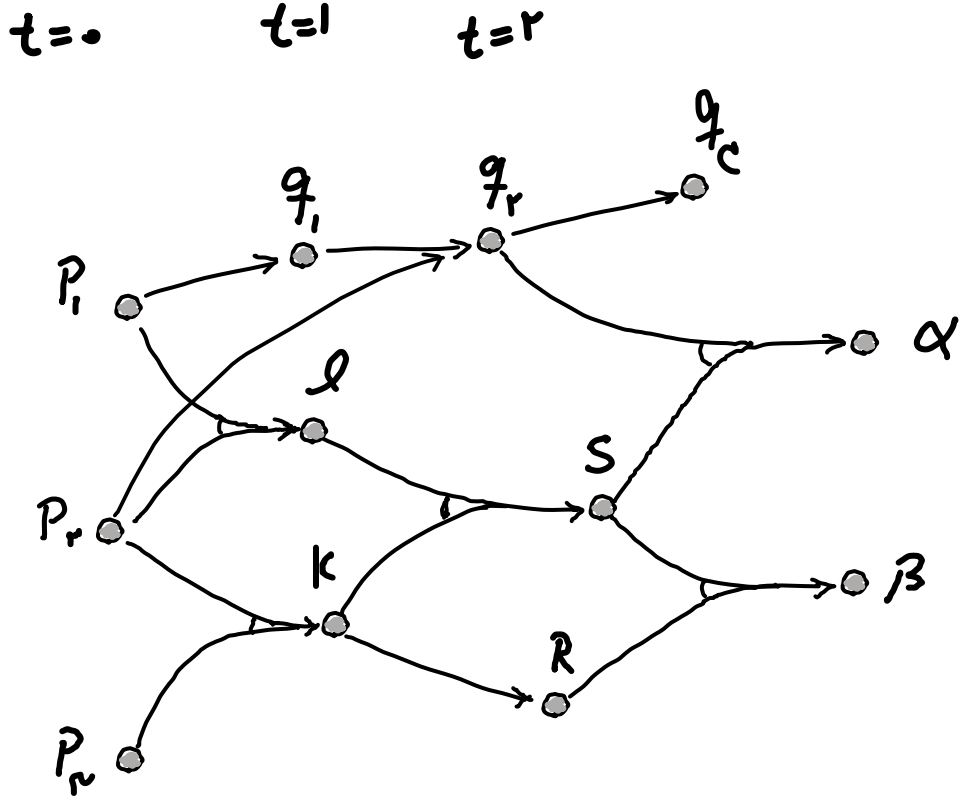
② ذخیره‌سازی افزایشی : در هر مرحله صرفاً دستکم از یکی از نتایج به دست آمده‌ی مرحله قبل استفاده کنیم

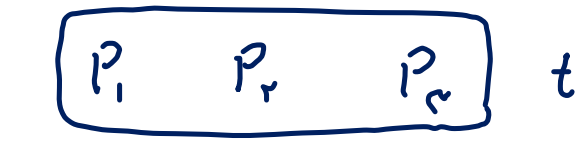
③ Magic set : پایین‌سیاگذاری روی جملاتی که در سیر به α تأثیرگذار هستند کار کنیم.

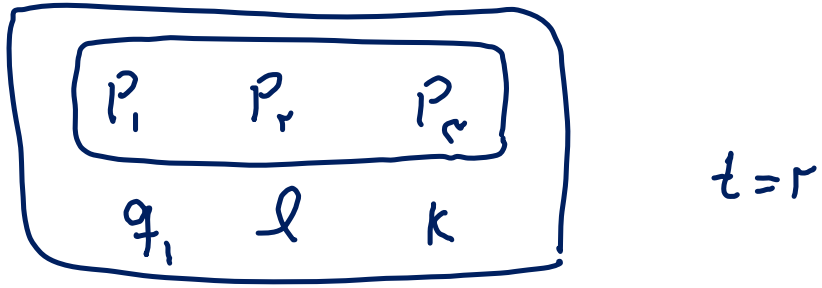
Incremental forward chaining

زنجیره سازی پیشرو افزایشی:

در هر بار فقط حملاتی را اکتیو می کند که حداقل یکی از مقدمات آن در مرحله ی $t-1$ به دست آمده باشد.







* درجه بعد، صحبت کامل کارآمدی FC را بررسی کرده است.

BACKWARD CHAINING

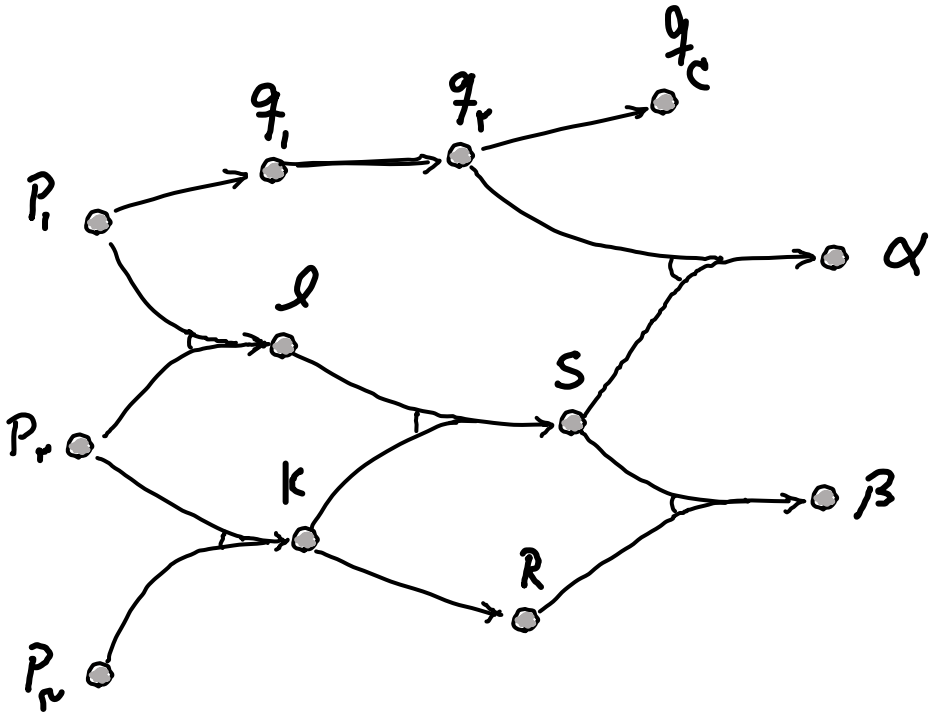
زنجیره سازی پسرو

ورودی: KB و α

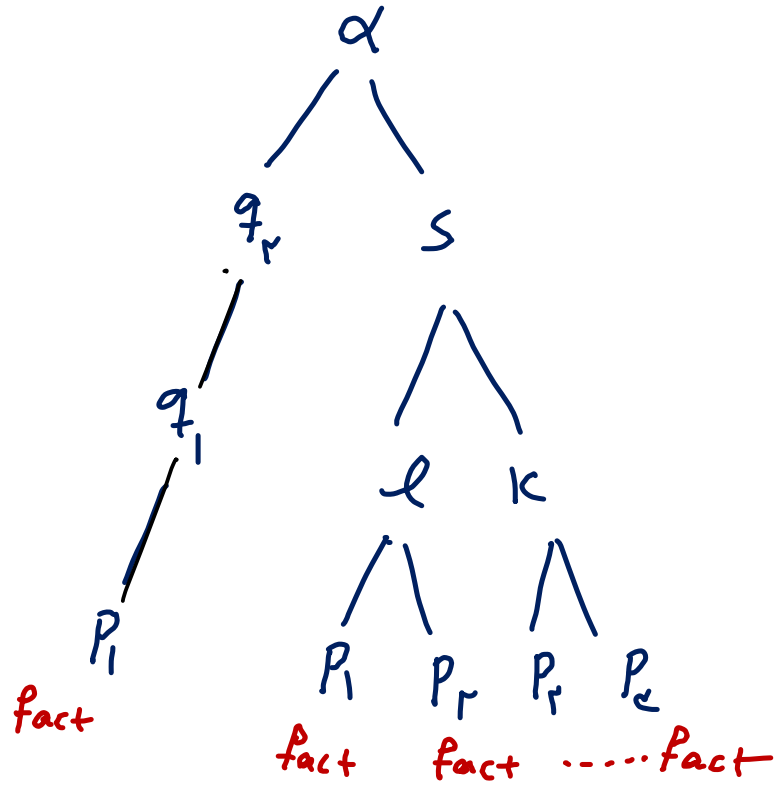
خروجی: مانند F.C

تیوه: مبتنی بر هدف (Goal Based)

[هر دورویش برای فرم معین، کابل هستند]



عمق‌گرد (goal Base):



			output
پیسرو	F.C	BFS	θ , fail
پسرو	B.C ,	DFS	θ , fail

۱۱۸- پایگاه دانش متشکل از عبارت‌های Horn زیر را در نظر بگیرید. فرض کنید روش زنجیره‌ای جلورو (Forward Chaining) با استفاده از جستجوی سطح اول(Breadth First Search) و روش زنجیره‌ای عقب‌رو (Backward Chaining) با استفاده از جستجوی عمق اول (Depth First Search) پیاده‌سازی شده‌است. در این‌صورت، کدام جمله نادرست است؟

$P(F(x)) \wedge Q(x) \Rightarrow Q(F(x))$

$P(x) \Rightarrow P(F(x))$

$P(A)$

$Q(A)$

۱) FC گزاره $Q(F(F(A)))$ را نتیجه می‌دهد. True

۲) FC قبل از گزاره $Q(F(A))$، گزاره $P(F(A))$ را نتیجه می‌دهد. True

۳) BC برای پرسمانی (query) به‌صورت $Q(F(A))$، مقدار درست (True) را برمی‌گرداند. True

۴) BC برای یک پرسمانی (query) به صورت $Q(F(F(A)))$ ، مقدار نادرست (False) را برمی‌گرداند. False

KB:

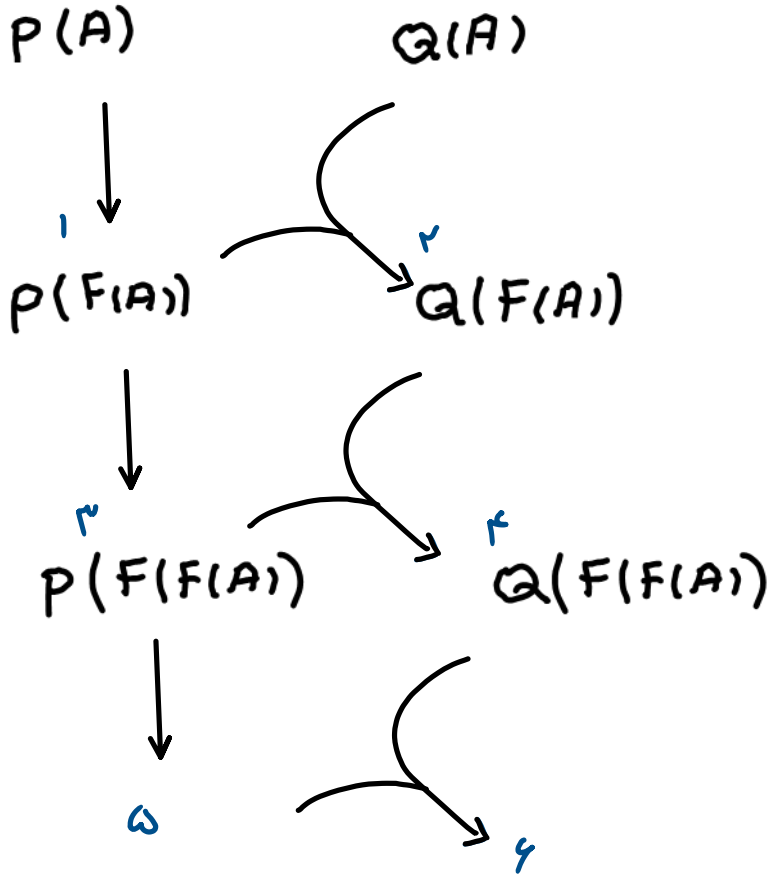
$P(x) \longrightarrow P(F(x))$

$P(F(x)) \wedge Q(x) \longrightarrow Q(F(x))$

$P(A)$

$Q(A)$

F.C:



RESOLUTION

رزولوشن در منطق مرتبه اول

برای رزولوشن در FOL ابتدا باید فرم کلازال جملات را بتوانیم بنویسیم. (همان CNF)

در FOL نیز هر جمله قابل نوشتن در فرم CNF است.

ترتیب اقدامات برای نوشتن فرم کلازال :

① حذف $\leftrightarrow$

② حذف $\rightarrow$

③ استاندارد سازی

④ توزیع $\sim$

⑤ استعلام

⑥ توزیع ۸ و ۷

$A \leftrightarrow B$

$(A \rightarrow B) \wedge (B \rightarrow A)$

$(\sim A \vee B) \wedge (\sim B \vee A)$

مثال:

$$\forall x \left(\forall y \ (A(y) \rightarrow L(x,y)) \rightarrow \exists y \ L(y,x) \right)$$

فرم کلازال را بنویسیم:

مرحله ۱

$$\forall x \left(\forall y \ (\sim A(y) \vee L(x,y)) \rightarrow \exists y \ L(y,x) \right)$$

مرحله ۲

$$\forall x \left(\exists y \ (A(y) \wedge \sim L(x,y)) \vee \exists y \ L(y,x) \right)$$

$$\forall x \left(\exists y \quad (A(y) \wedge \sim L(x,y)) \quad \vee \quad \exists z \; L(z,x) \right)$$

حذف سور‌ها :

$$(A(c_1(x)) \wedge \sim L(x, c_1(x))) \quad \vee \quad L(c_2(x), x)$$

توزیع $\vee$ روی $\wedge$:

$$\underset{①}{(A(c_1(x)) \vee L(c_2(x), x))} \quad \wedge \quad \underset{②}{(\sim L(x, c_1(x)) \vee L(c_2(x), x))}$$

دو بند به دست می‌آید:

$$\{ \underset{①}{A(c_1(x)) \vee L(c_2(x), x)} \quad , \quad \underset{②}{\sim L(x, c_1(x)) \vee L(c_2(x), x)} \}$$

Resolution در FoL :

ورودی : KB و α

* لزم : باید CNF باشد.
* کامل کاذب است.

① $\neg\alpha$ به KB اضافه می‌شود.

② رزولوشن روی جفت‌های پیاده می‌شود. (هر بار یک لیترال حذف می‌شود)

③ در پایان
- اگر تهی به دست آمد یعنی $KB \models \alpha$
- اگر نتوان جفت جدیدی ایجاد کرد و تهی به دست نیامد: $KB \not\models \alpha$

مقایسه‌ی نقاط ضعف و قوت رزولوشن و زنجیره‌سازی :

*۱ : هر گزاره را می‌توان به فرم CNF (کلازال) نوشت اما نمی‌توان هر گزاره را در فرم معین نوشت .

*۲ : اگر KB در فرم معین و α لیترال مثبت باشد، زنجیره‌سازی F.C از رزولوشن کارآمدتر است .

مثال: اجرای رزولوشن در منطق مرتبه اول FOL :

آیا در این مثال $KB \models \alpha$ ؟

KB:
۱) $\neg M(x,y) \vee W(x)$
۲) $\neg F(x,y) \vee \neg W(x)$
۳) $M(A,B)$

$\models \alpha$: $\sim F(A,C)$

پاسخ: ابتدا $\sim\alpha$ را به KB اضافه کنیم. KB و $\sim\alpha$ همه به فرم کلازال هستند:

KB:
۱) $\neg M(x,y) \vee W(x)$
۲) $\neg F(x,y) \vee \neg W(x)$
۳) $M(A,B)$

$\sim\alpha$
۴) $F(A,C)$

KB:
۱) $\neg M(x,y) \vee W(x)$
۲) $\neg F(x,y) \vee \neg W(x)$
۳) $M(A,B)$

$\sim \alpha$
۴) $F(A,C)$

استانداردسازی لازم است انجام شود.
اگر استانداردسازی نکنیم ممکن است برخی نتایج قابل استنتاج را از دست بدهیم.

KB:
۱) $\neg M(x,y) \vee W(x)$
۲) $\neg F(t,z) \vee \neg W(t)$
۳) $M(A,B)$

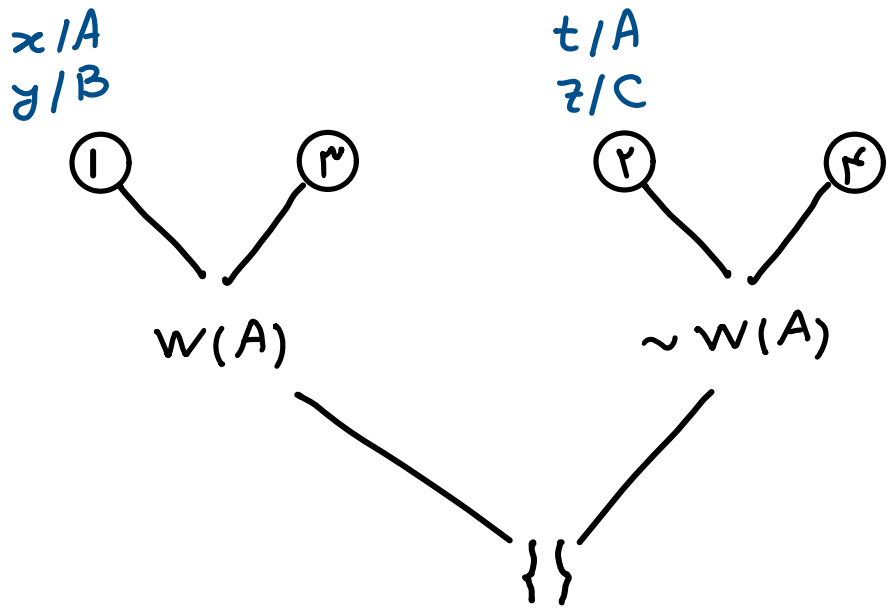
$\sim \alpha$
۴) $F(A,C)$

۱) $\neg M(x,y) \vee W(x)$

۲) $\neg F(t,z) \vee \neg W(t)$

۳) $M(A,B)$

۴) $F(A,C)$



پس α از KB استنتاج می‌شود.

درصورت عدم استفاده از استانداردسازی:

ممکن است برخی از α هایی که از KB نتیجه می‌شوند را اشتباهاً fail کنیم.

اما امکان ندارد α ای را نتیجه بگیریم که نباید از KB نتیجه می‌شد.

به زبان ساده: در FOL اگر در الگوریتم‌های استنتاج استانداردسازی انجام ندهیم

ممکن است برخی از نتایج را از دست بدهیم اما ممکن نیست نتیجه‌ی غلط

بگیریم.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۱۶ فصل ۹

مثال: زنجیره‌سازی پیشرو در FOL :

مثال: زنجیره‌سازی پیشرو در FOL :

$$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$$

$\exists x\ Owns(Nono, x) \wedge Missile(x)$ ← تک‌سازی وجودی

$Owns(Nono, M_1)$

$Missile(M_1)$

نوشتن فرضیات در منطق FOL:

"All of its missiles were sold to it by Colonel West":

$$Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$$

$Missile(x) \Rightarrow Weapon(x)$ موشک، سلاح است.

$Enemy(x, America) \Rightarrow Hostile(x)$ دشمن آمریکا، متخاصم محسوب می‌شود.

"West, who is American ...": ⟶ $American(West)$

"The country Nono, an enemy of America ...": ⟶ $Enemy(Nono, America)$

پس این عبارات در KB قرار دارند:

rule:

1 $American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$

2 $Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

3 $Missile(x) \Rightarrow Weapon(x)$

4 $Enemy(x, America) \Rightarrow Hostile(x)$

fact:

5 $American(West)$

6 $Enemy(Nono, America)$

7 $Owns(Nono, M_1)$

8 $Missile(M_1)$

سوال: $KB \overset{?}{\models} Criminal(West)$

- On the first iteration,

تکرار اول FC :

Rule 2 is satisfied with $\{x/M_1\}$, and $Sells(West, M_1, Nono)$ is added.
Rule 3 is satisfied with $\{x/M_1\}$, and $Weapon(M_1)$ is added.
Rule 4 is satisfied with $\{x/Nono\}$, and $Hostile(Nono)$ is added.

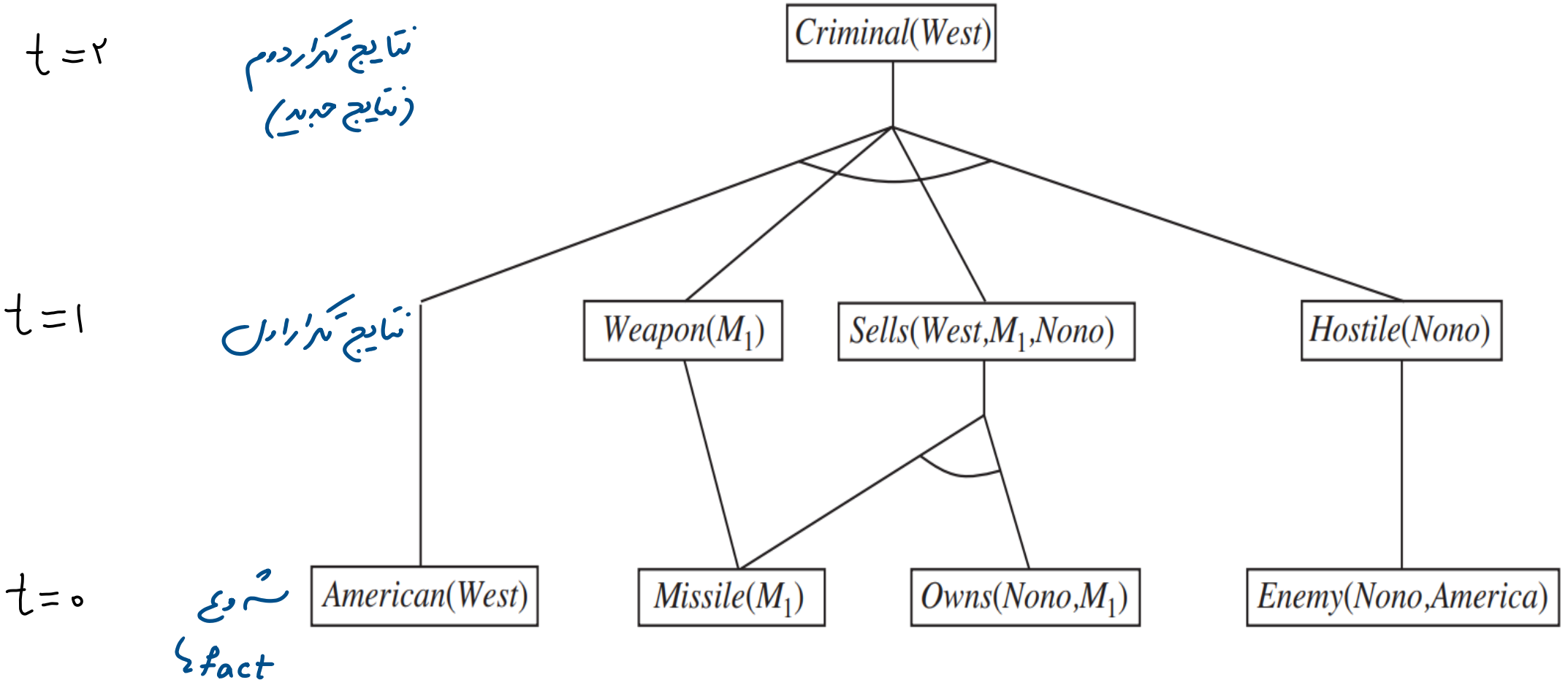
- On the second iteration,

تکرار دوم :

Rule 1 is satisfied with $\{x/West, y/M_1, z/Nono\}$, and

$Criminal(West)$ is added.

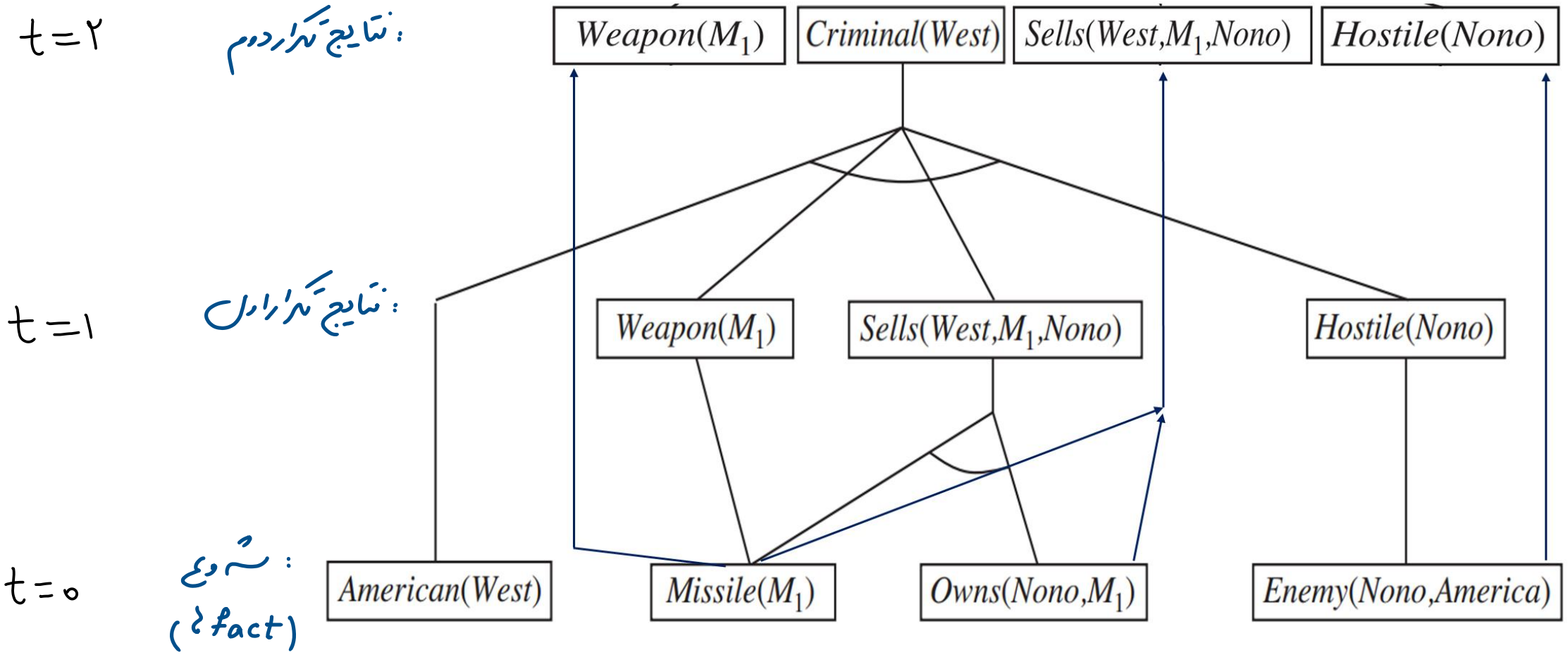
درخت زنجیره‌سازی FC مثال حل (از برگ‌ها به ریشه) (از پایین به بالا)



توجه:

در نسخه‌ی عادی F.C تمام لیترال‌های تولید شده در تکرار اول، دوباره در تکرار دوم نیز تولید می‌شوند و این باعث ناکارآمدی است.

یعنی در حقیقت F.C در عمل به این صورت است: (از پایین به بالا) . برای جلوگیری از این مشکل روش زنجیره‌سازی افزایشی را معرفی می‌کنیم.



زنجیره‌سازی افزایشی:

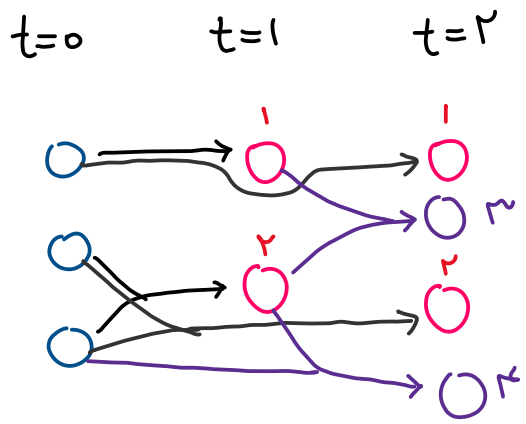
برای جلوگیری از تولید مجدد نتایج گذر اول، از زنجیره‌سازی افزایشی استفاده می‌کنیم.

در هر گذر فقط استدلال‌هایی تولید می‌شوند که حداقل یک مقدم آن در گذر قبل تولید شده باشد.

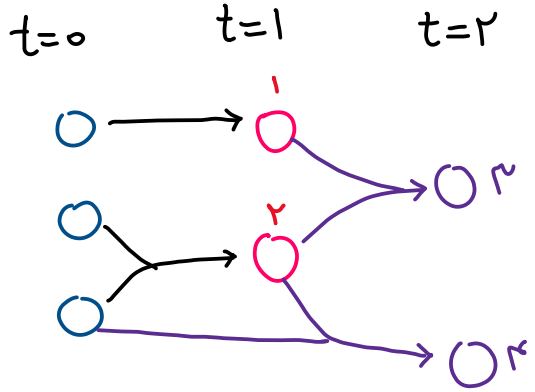
Incremental forward chaining

Every new fact inferred on iteration t must be derived from at least one new fact inferred on iteration $t - 1$.

نحوه‌ی معمول تولید نتایج FC:



جهش‌یابی از تولید مجدد نتایج در زنجیره‌سازی افزایشی:



منابع ناکارآمدی forward chaining:

pattern matching → حلقه‌ی داخلی، در هر تکرار همه‌ی یکان‌سازهای همه‌ی fact ها با همه مقدمات را پیدا می‌کند که هزینه زمانی زیادی دارد.

rechecks every rule → در هر تکرار دوباره تمام rule ها بررسی می‌شوند حتی اگر فقط یک fact اضافه شده باشد.

generate irrelevant → ترکیب جملات غیر مرتبط با goal

conjunct ordering problem:

l_1 l_2

$$Missile(x) \wedge Owns(Nono, x) \longrightarrow Sells(West, x, Nono)$$

Missile(A)

Missile(B)

Owns(Nono, D)

Owns(Nono, E)

Owns(Nono, G)

Owns(Nono, A)

Owns(Nono, L)

Owns(Nono, P)

از آنجایی که تعداد مسنه‌های Owns بیشتر از Missle است ترتیب بهتر آن است که ابتدا l_1 را با fact ها یکی می‌کنیم. x/A و x/B بدست می‌آید حال برای آنها مسنه l_2 را بررسی می‌کنیم.

It turns out that finding the optimal ordering is NP-hard, but good heuristics are available.

ترتیب مناسب :

۱) یافتن بهترین ترتیب np-hard است.

۲) چند هیوریستیک (ابزارهای تخمینی) برای یافتن ترتیب‌های تا حدی مناسب وجود دارد.

minimum-remaining-values (MRV)

ابتدا قواعد شرطی با تعداد مقدمات کمتر را بررسی کنیم. [مثلاً ابتدا از 3 استفاده کنیم]
در بین جمله‌ها، ابتدا آنهایی که تعداد آرگومان‌های متغیر کمتری
دارند شروع کنیم. [مثلاً در 1، جمله sells را در انتها بررسی کنیم]

1 $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

2 $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

3 $Missile(x) \Rightarrow Weapon(x)$

4 $Enemy(x, America) \Rightarrow Hostile(x)$

data complexity

متناهی بودن تعداد جهات زمینه‌ای

وقتی تعداد مسندها و ثابت‌ها متناهی

باشه و نماد تابع نداشته باشیم:

n = تعداد ثابت

p = تعداد مسندها

m = حداکثر آرگومان‌ها

ثابت‌ها $\{c_1, c_2, \ldots, c_n\}$

همه‌ی مسندهایی که در قواعد شرطی و fact های KB استفاده شده‌اند

$H_1(x_1, x_2)$

$H_2(x_1, x_2, x_3)$

$H_p(x_1, x_2, \ldots, x_m)$

بدون نماد تابع: تعداد لیترال‌های زمینه‌ای $\leq p\, m^n$

$H_1(\cdots)$

$\vdots$

$H_p(x_1, \ldots, x_m)$

x_1: m حالت زمینه‌ای دارد

x_m: m حالت

x_1/c_1

x_1/c_2

$\vdots$

x_1/c_n

بدون وجود تابع.

حداکثر تعداد سندرم و نمادهای بطی زمینه‌ای:

$$p \times \underbrace{m \times \cdots \times m}_{\text{تا } n} = p^{m^n}$$

با حضور نماد تابع ، تعداد جملات زمینه‌ای نامتناهی است اما در صورتی که α قابل استنتاج باشد

بازهم پیچیدگی زمانی FC ، چند جمله‌ای است: (برحسب تعداد اعضای KB)

$O(N^c)$ $\quad$ $N = |KB|$

(اگر نماد تابع داشته باشیم و $KB \not\models \alpha$ الگوریتم F.C ممکن است متوقف نشود)

magic set hybrid between forward inference and backward preprocessing.

* یک نقطه ضعف FC تولید نتایج غیر مرتبط است. برای جلوگیری از آن می‌توان راهکاری به نام مجموعه جادویی به کار برد که به نوعی زنجیره‌سازی پسرو BC را در FC شبیه‌سازی می‌کند.

در این مثال:

rule:

1. $American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$
2. $Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
3. $Missile(x) \Rightarrow Weapon(x)$
4. $Enemy(x, America) \Rightarrow Hostile(x)$

5. $American(West)$ ، American(Ali) ، American(Nader) ، ...

fact:

6. $Enemy(Nono, America)$
7. $Owns(Nono, M_1)$
8. $Missile(M_1)$

$$KB \overset{?}{\models} Criminal(West)$$

چون حکم درباره‌ی West است پس اطلاعاتی که درباره‌ی West داریم می‌کاهند

در رسیدن به هدف، موثر باشد. پس گزاره شرطی ① و Fact ها را جزئی تغییر دهیم:

1 $Magic(x) \wedge American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$.

9 $Magic(West)$

fact

شرط Magic(x) را به مقدمات این rule اضافه می‌کنیم.

پس فقط برای West ، یک fact به صورت Magic(West) به KB اضافه می‌کنیم

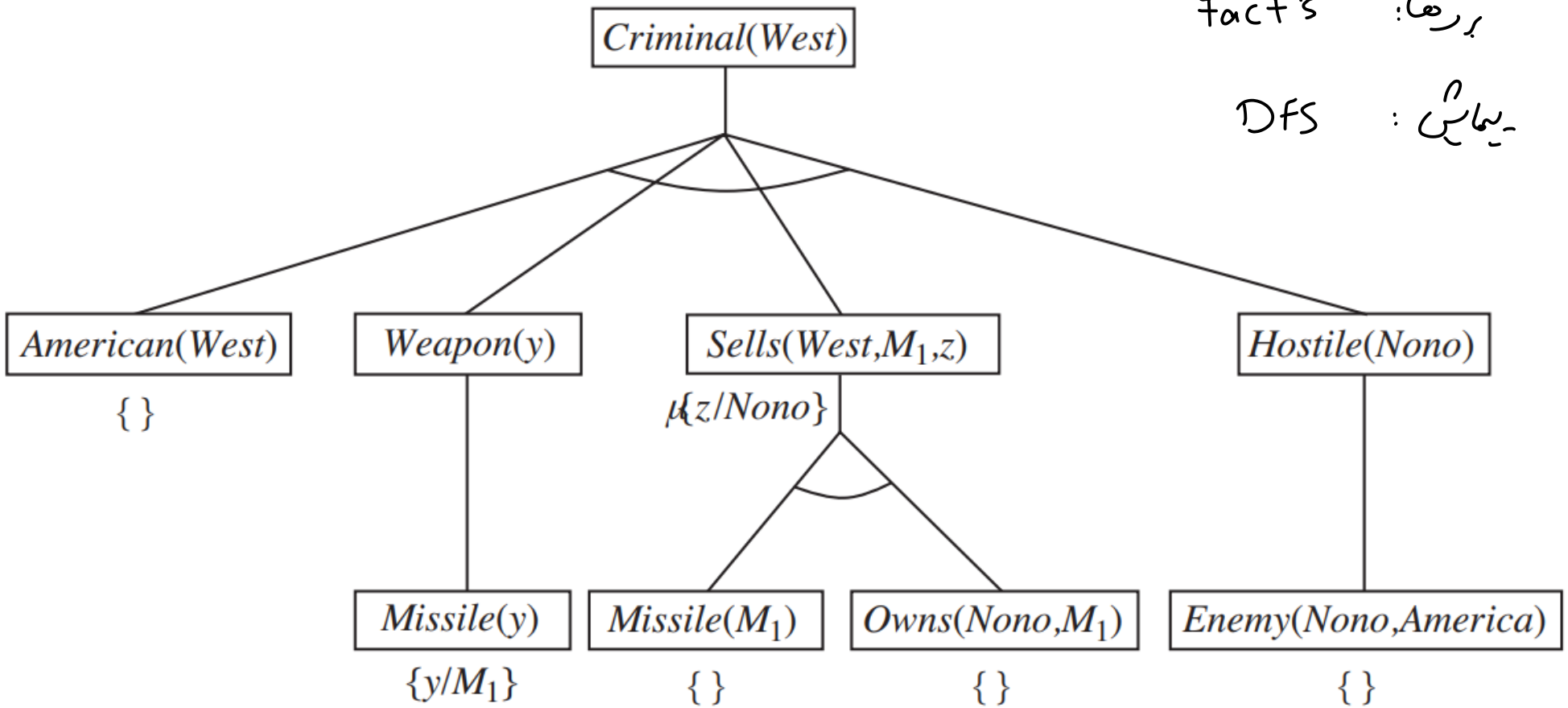
backward-chaining

درخت BC : (از ریشه به سمت برگ‌ها)

ریشه : goal

برگ‌ها : fact's

پیمایش : DFS



Storage and retrieval

Tell: STORE(s) ذخیره‌ی دانش جدید s در KB

Ask: FETCH(q) بررسی امکان یکسان بودن q با جملات موجود

یک روش عملی اما غیر کارآ برای پیاده سازی FETCH : قرار دادن جملات fact در یک لیست.

Knows(A,B)	*Brother*(A,E)	*Brother*(B,E)	*Knows*(E,D)	*Brother*(D,P)	*Brother*(D,E)	*Knows*(E,B)	*Knows*(D,E)	*Brother*(E,D)

اگر q : $Knows(x, B)$

آنگاه Fetch، یک یک جملات لیست را با q مقایسه می کند برای یافتن یکان ساز.

predicate indexing

نکته بیاری از یکسان سازی که واضح است و می‌توان از اتلاف وقت در مورد آنها جلوگیری کرد برای مثال واضح است که سندهای متمایز قابل یکسان سازی نیستند.

پس می‌توان حقایق درباره‌ی هر سند، در یک جدول مجزا ذخیره کرد. (استفاده از نام سند به عنوان اندیس)

Knows:

Knows(A,B)	Knows(B,A)	Knows(E,D)	Knows(D,E)	Knows(F,E)	...

Brother:

Brother(A,E)	Brother(B,E)	Brother(D,P)	Brother(E,A)	Brother(E,B)	...

حالا اگر $q = Knows(x,B)$ باشد فقط نیاز با اندیس knows جستجو خواهد شد.

اگر تعداد سند‌ها نسبتاً زیاد و ولی تعداد سند‌های هر کدام نسبتاً کم باشد، ایندکس گذاری سند‌ها کمک زیادی به سرعت اجرای Fetch می‌کند.

Knows:		شناختن
Brother:		برادر بودن
Friend:		دوست بودن
Sister:		خواهر بودن
Employes:		استخدام کردن
Help:		کمک کردن

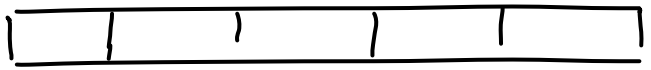
این روشی (فهرست کردن با برچسب نام سند) هنگامی که تعداد سندها کم اما تعداد بندهای هرکدام زیاد باشد کمک چندانی به کاهش زمان جستجو نمی‌کند. باید از موارد دیگر مانند آرگومان‌ها برای اندیس گذاری استفاده کنیم.

Employes(x, y)

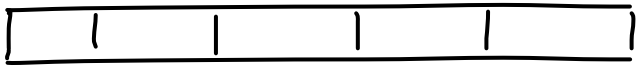
IBM

NAVY

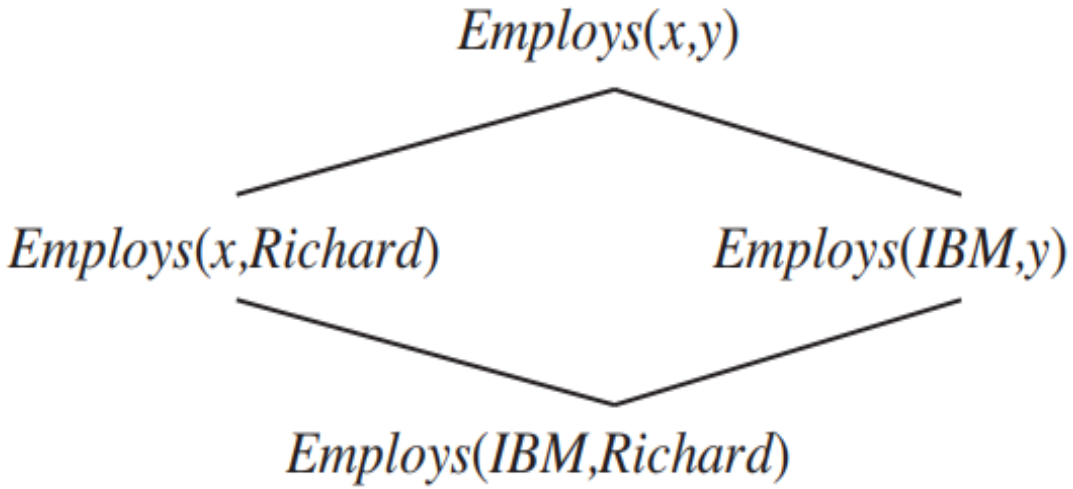
Employes(IBM, ...)



Employes(NAVY, ...)

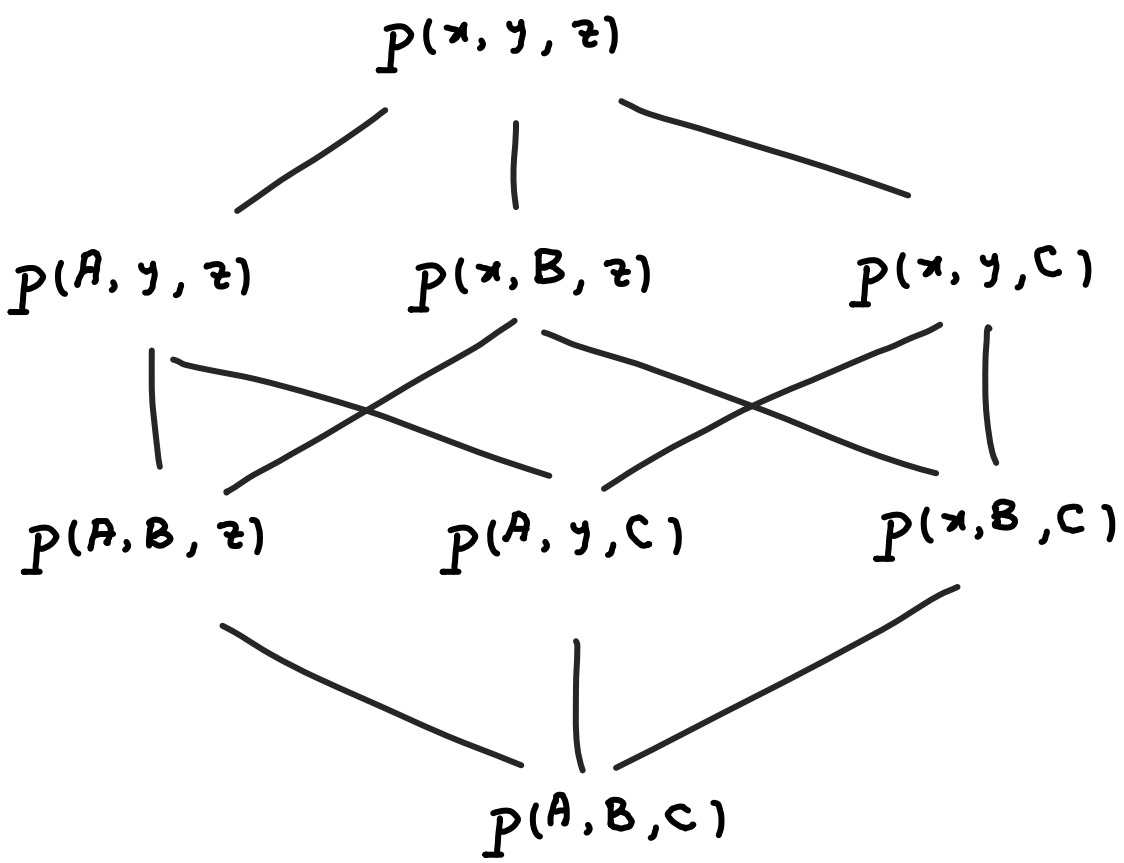


شبکه‌ی شمول:



تعداد گره‌ها $= O(2^n)$ $\Longleftarrow$ تعداد گره‌های برگ $= n$

لیست کامل سازه‌ها به صورت زیر است.



گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۱۷ فصل ۹

در معناشناسی FoL ، (مانند آنچه در دنیای واقعی داریم):

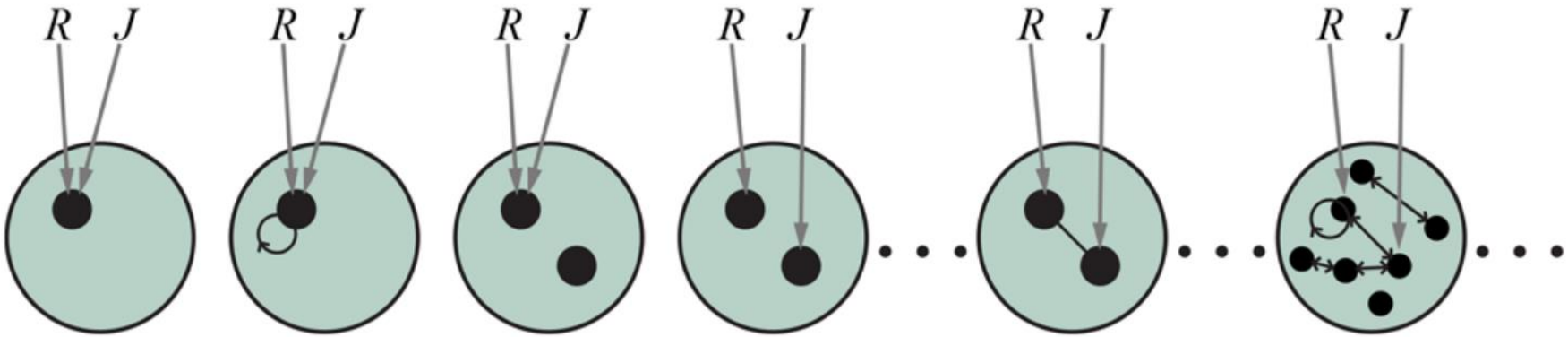
۱) ممکن است یک شیء واحد دارای چندین نام باشد. مثلاً ثابت‌های A و B ممکن است هر دو، نام یک شیء واحد باشند

۲) ممکن است برخی اشیاء نامی نداشته باشند یا در پایگاه دانش از آن‌ها نامی نیامده باشد.

۳) هر جمله‌ای که در KB آمده درست فرض می‌شود. ارزش True دارد اما جمله‌ای که در KB نیامده‌اند لزوماً نادرست نیستند بلکه ارزش آن‌ها نامشخص است. (unknown)

مثال بامعناشناسی FOL :

با دو ثابت R و J و فقط یک نماد رابطه‌ای باینری $P(x, y)$ بی‌نهایت مدل مختلف وجود دارد:



ممکن است جهان، یک، دو، سه، ... عضوی باشد.
ممکن است اعضای آن رابطه‌ی P داشته باشند یا نداشته باشند.

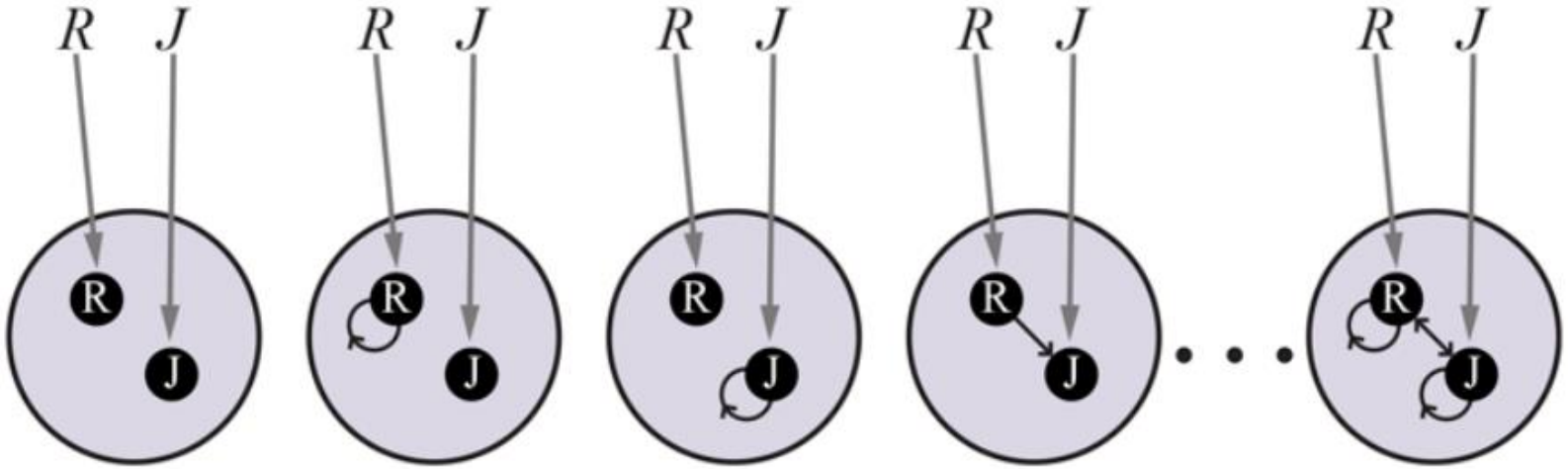
معناشناسی پایگاه داده با معناشناسی FOL تفاوت دارد:

database semantics

- **unique-names** assumption — یکتایی نام: هر شیء یک نام منحصربه‌فرد دارد.
- **closed-world** assumption — جهان بسته: هر جمله‌ای که در KB نیامده ارزش False دارد.
  مثلاً اگر ۳ شیء A,B,C داریم و $P(A) \wedge P(B)$
  در KB آمده است یعنی $P(C)$ ارزش False دارد.
- **domain closure** — دامنه‌ی بسته‌شده: یعنی به‌جز اسم‌های ثابت که نام آورده‌ایم، هیچ شیء دیگری وجود ندارد.

تحت فرض‌های پایگاه داده:

با ۲ نام ثابت R, J و مسند باینری $P(x, y)$



جهان دقیقاً ۲ عضوی است. فقط با توجه به وجود یا عدم وجود رابطه‌ی $P(x,y)$ بین R و J ممکن است ۱۶ وضعیت داشته باشیم [فقط با یک مسند $P(x,y)$ و داشتن n ثابت تعداد حالات ممکن 2^{n^2} است.]

توجه: با داشتن n عضو در جامعه، و با وجود یک مسند $p(x,y)$:

تعداد دنیاهای ممکن 2^{n^2} است. (در کلاس اشتباهاً 2^{2^n} نوشتیم)

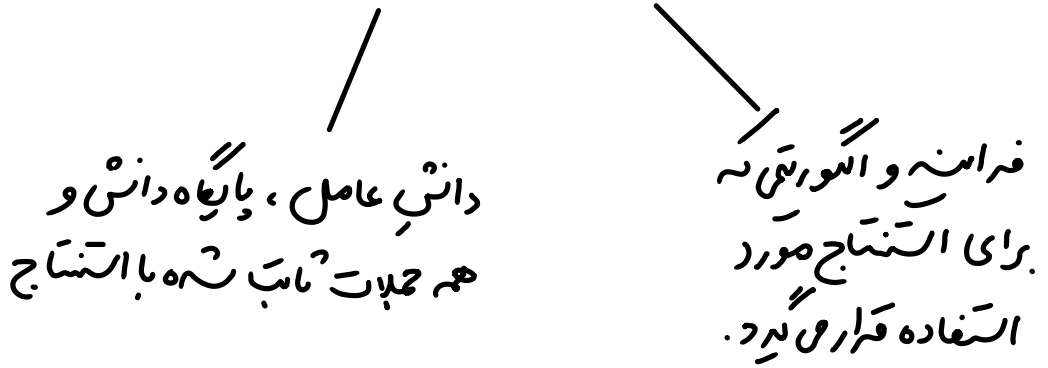
تعداد حفت های (x,y) : $n \times n$ است.

برای هر حفت می‌توانند رابطه‌ی $p(x,y)$ ارزش True یا False داشته باشد.

پس: 2^{n^2}

Logic programming

Robert Kowalski's equation: **Algorithm = Logic + Control .**



آشنایی با زبان منطقی Prolog:

متغیرها: با حروف بزرگ آغاز می‌شوند:

X, Y, Z

Time, ...

ثابت‌ها و مسندها: با حروف کوچک آغاز می‌شوند:

```
like(ali, mina).
need(X, food).
```

انتهای هر جمله، نقطه باشد.

likes(X, ali). همه علی را دوست دارند

Tell

hasname(Y) هر کس دارای نام است

Brother(arash, mina) آرش برادر مینا است.

④ نوشتن and، به صورت کاما:

$$p(X) \wedge q(a) \wedge q(b)$$

در یک سطر:

$$p(X),\ q(a),\ q(b).$$

یا چند سطر:

$$p(X),$$
$$q(a),$$
$$q(b).$$

Fail : ارزشی آن شرط را False می‌کند.

مثلاً: $\sim likes(a, b)$

در Prolog :

likes(a,b) , Fail .

۵) rules (گزاره‌های شرطی در فرم صحیح)

در Prolog:

b :- a. $a \rightarrow b$

r :- p, q. $p \wedge q \rightarrow r$

⑥ گزاره $p \vee q \rightarrow r$ در prolog:

توضیح: این گزاره معادل است با $(p \rightarrow r) \wedge (q \rightarrow r)$ پس:

```
r :- p.
r :- q.
```

مثال:

$$p(a) \vee (q(x) \wedge r(x)) \longrightarrow s(b)$$

در prolog:

```
s(b) :- p(a) ,
s(b) :- q(X), r(X).
```

⑦ پرسش و خروجی گرفتن در prolog :

? — عبارت .

اگر عبارت یک جمله‌ی بدون متغیر باشد، منظور ما از این سئوال برقراری یا عدم برقراری آن است.

مثلاً می‌نویسیم: ? — like(ali, iran).

یعنی پرسیده‌ایم: آیا علی، ایران را دوست دارد؟

خروجی این سئوال، یعنی پاسخ آن yes یا no است.

ب) اگر عبارت شامل متغیری مانند X باشد، برنامه سئوال شما را با سور وجودی ∃ می خواند:

```
?- like(X, iran).
```

در واقع معنای سئوال این است: آیا X ای وجود دارد که ایران را دوست داشته باشد؟

خروجی برنامه دو حالت دارد:

- یا لیست جانشانی $\theta = \{X/\ldots\}$ را می دهد یعنی این X ها در عبارت بالا صدق می کنند. [اثبات]
- یا پاسخ no می دهد یعنی چنین X ای وجود ندارد.

TELL:

likes(ali, mina).

likes(zahra, ahmad).

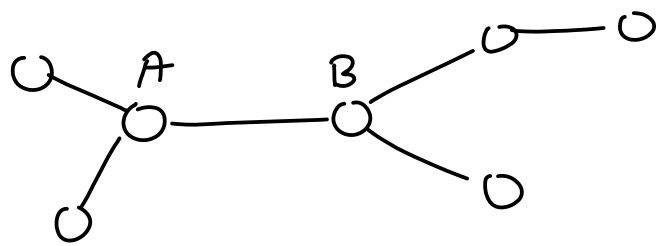
ASK:

?- likes(X, mina).

---

output X=ali.

تعریف مسنده‌ای جدید برحسب قبلی‌ها:



فرض کنید link(A,B) به معنای وجود یال از A به B باشد.

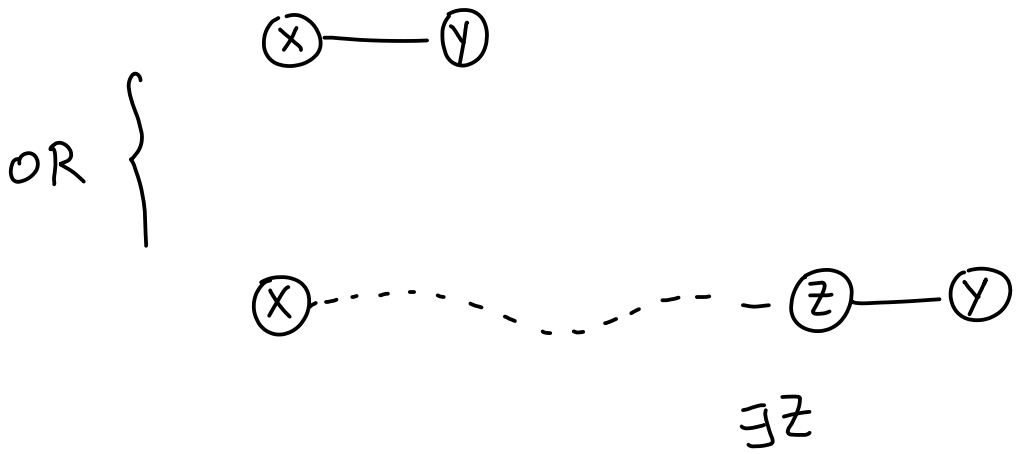
مسند (مسیر) path(X,Y)

(وجود مسیر از X به Y) را می‌خواهیم معرفی کنیم.

قرارداد: فقط وقتی ارزش آن True باشد که مسیری از X به Y وجود دارد.

می دانیم که :

$$link(x,y) \vee [path(x,z) \wedge link(z,y)] \rightarrow path(x,y)$$

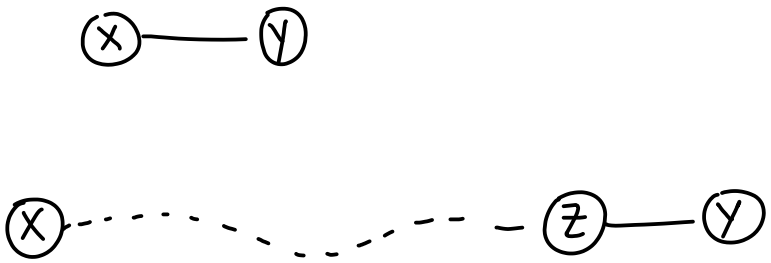


موفی نادربط (مُستند) وجود مسیر از x به z در Prolog:

Tell

```
path(X,Z) :- link(X,Z).

path(X,Z) :- path(X,Y), link(Y,Z).
```

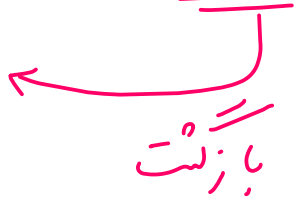


اغلب در تعریف مسئله‌ها یک سطر اولیه (غیربازگشتی) و یک سطر بازگشتی داریم.

در ادامه توضیح می‌دهیم چرا سطر بازگشتی باید در سطر دوم بیاید.

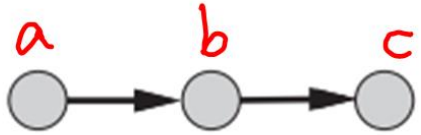
```
path(X,Z) :- link(X,Z).

path(X,Z) :- path(X,Y), link(Y,Z).
```



مثال: در این مثال ما سه میز را در ذهن تعریف کرده‌ایم.

و می‌خواهیم در جواب سوال ?- path(a,c) پاسخ مثبت بدهد.



Tell

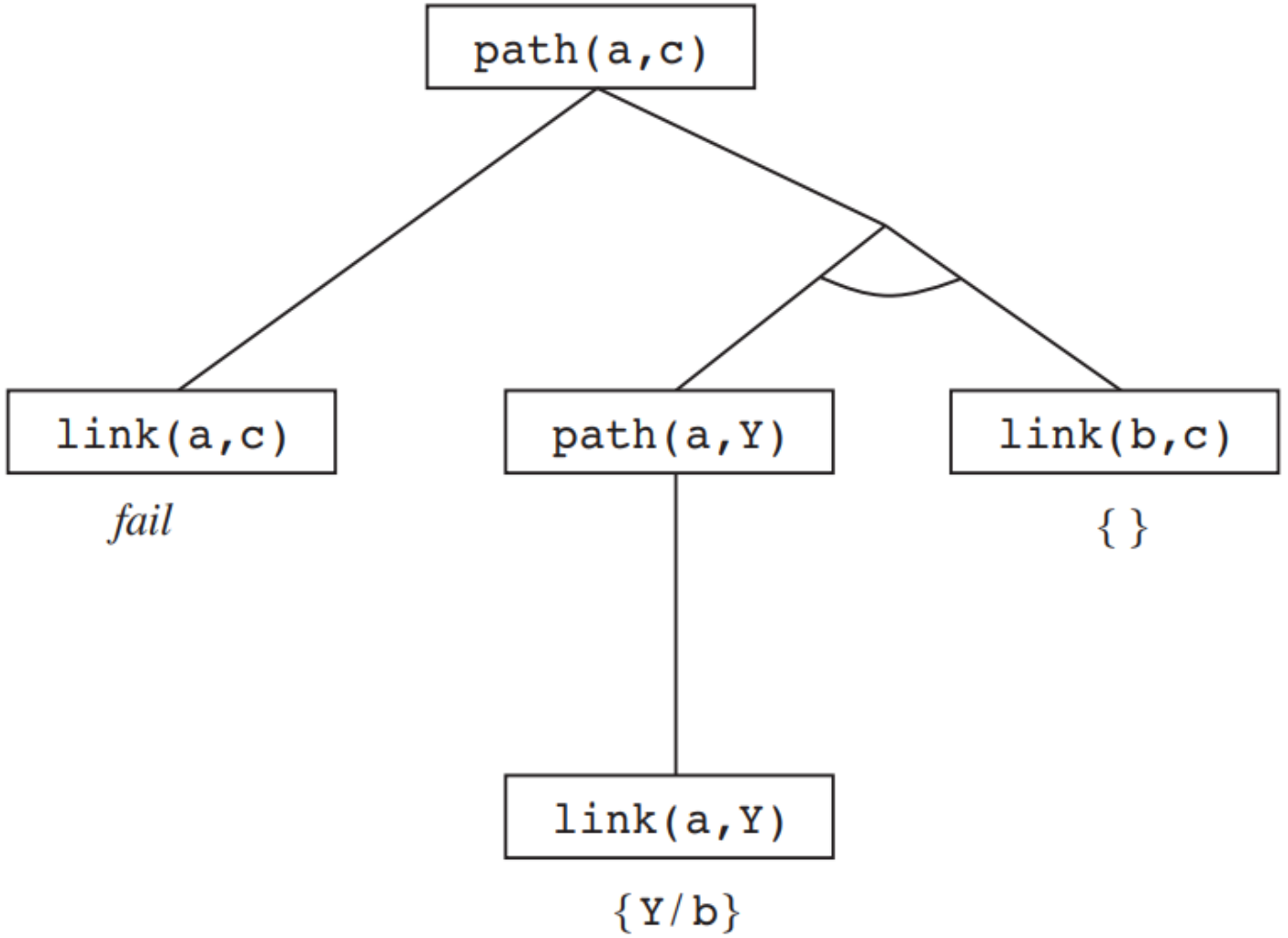
```
link(a,b).
link(b,c).
path(X,Z) :- link(X,Z).
path(X,Z) :- path(X,Y), link(Y,Z).
```

Ask

```
?- path(a,c)
```

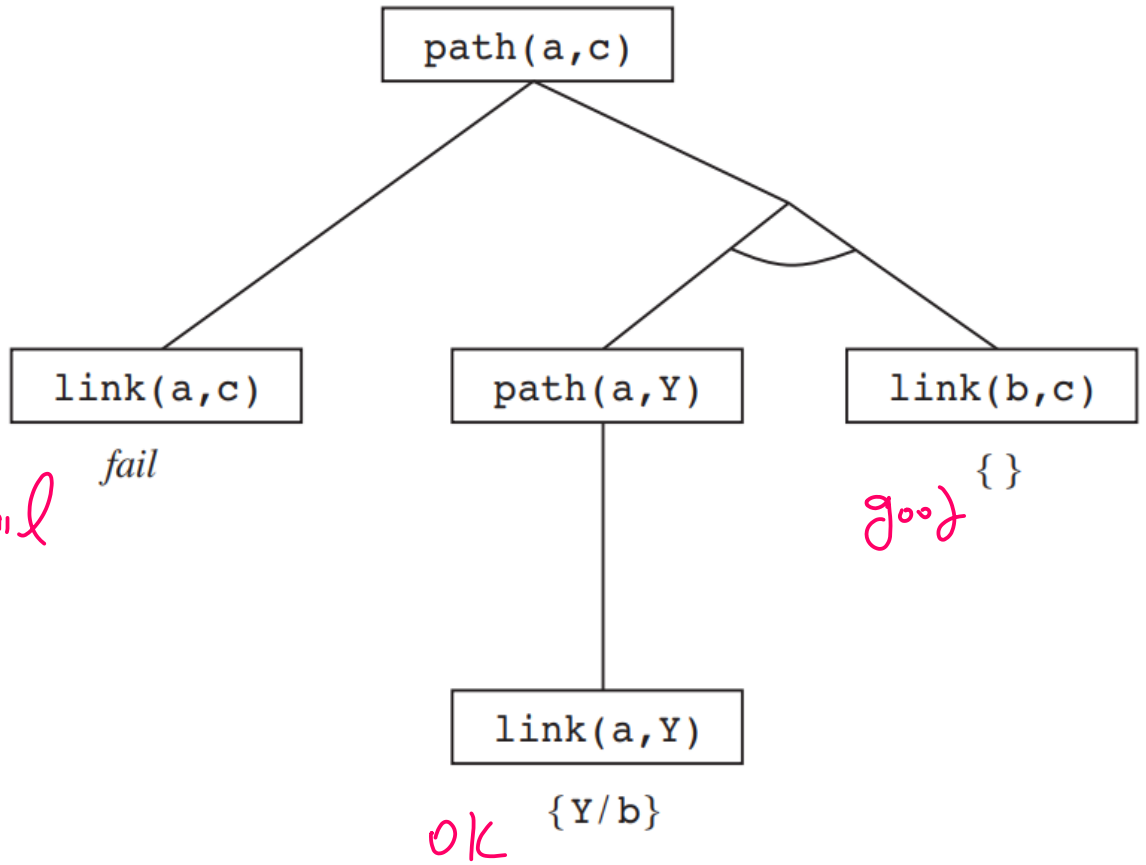
درخت جستجوی پاسخ:

پس از یک مدت زمان متناهی
مقدار True را برمی‌گرداند.



① path(X,Z) :- link(X,Z).

② path(X,Z) :- path(X,Y), link(Y,Z).



fail

good

ok

توضیح: ابتدا طبق سطر اول

به ازای X/a Z/c

نیاز به link(a,c) دارد اما

این گزاره false است. (fail)

به سراغ سطر دوم می‌رود. باید

path(a,Y) ∧ link(Y,c)

برای یک Y برقرار باشد.

دوباره برای یافتن پاسخ path(a,Y)

به سطر اول برمی‌گردد به ازای Y/b برقرار است. (ok)

برای این Y=b، عبارت link(b,c) نیز برقرار است.

پس جواب True است.

Redundant inference and infinite loops

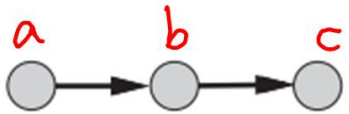
فرم نادرست تعریف: اگر استنتاج جمله‌ی بازگشتی را بنویسیم، prolog در پاسخ‌دهی به سئوال در loop

گرفتار می‌شود.

فرم نادرست .

```
path(X,Z) :- path(X,Y), link(Y,Z).

path(X,Z) :- link(X,Z).
```



Tell

```
link(a,b).
link(b,c).
```

① `path(X,Z) :- path(X,Y), link(Y,Z).`

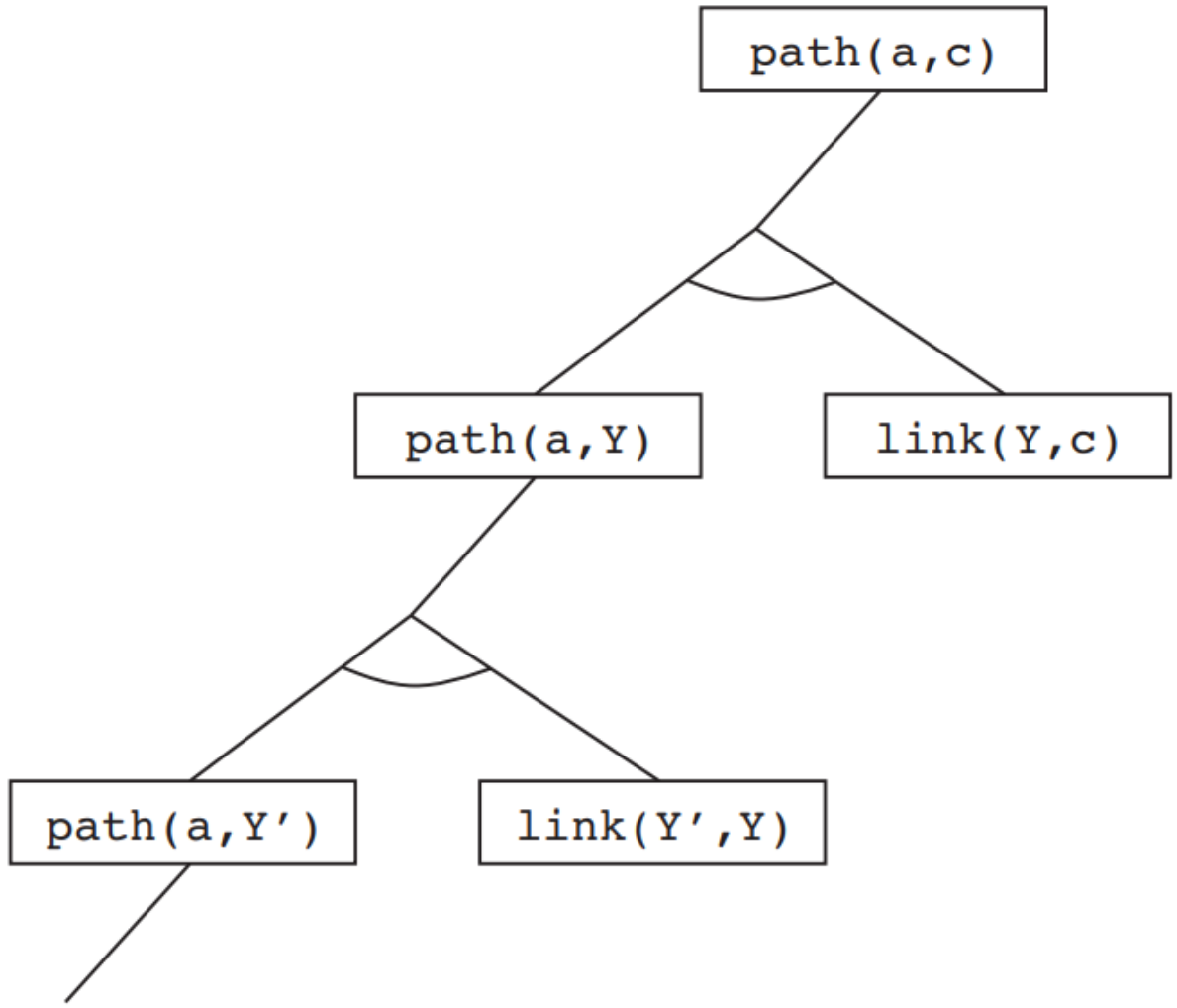
② `path(X,Z) :- link(X,Z).`

Ask

```
?- path(a,c)
```

این بار برای پاسخ به سوال درباره‌ی path(...,...) ابتدا باید به سوال درباره‌ی path(...,...) پاسخ دهد و این loop هرگز به پایان نمی‌رسد.

loop:



نماد (=) :

در prolog تساوی به معنای امکان یکسان سازی دو عبارت است نه تساوی آنها به لحاظ مقدار عددی :

$$?- book(title(X), author(Y)) = book(Z, author(john)).$$

$$Z = title(X)$$

$$Y = john$$

$$?- parent(X, ali) = parent(reza, Y).$$

$$X = reza$$

$$Y = ali$$

?- ۲+۲ = X .

fail

Tell:

king (john).

king (richard).

evil (jack).

evil (robert).

evil (bob).

Ask:

?- king (X), evil (Y).

و خروجی

output:

X = john	*Y = jack;*
X = john	*Y = robert;*
X = john	*Y = bob;*
X = richard	*Y = jack;*
X = richard	*Y = robert;*
X = richard	*Y = bob;*

مثال

KB شامل حقایق و قوانین زیر باشد :



parent (*hossein* , *mehdi*).

parent (*hossein* , *ahmad*).

parent (*behzad* , *hossein*).

grandparent (*X* , *Z*) :- *parent*(*X* , *Y*) , *parent*(*Y* , *Z*).

?-*grandparent* (*behzad* , *mehdi*).

yes

?-*grandparent*(*A* , *B*).

A=*behzad* *B*=*mehdi*;

A=*behzad* *B*=*ahmad*;

?-*grandparent*(*behzad* , *A*).

A=*mehdi*;

A=*ahmad*;

مثال

?- X+Y=۲+۵

X=۲ Y=۵

?- ۲+۲=۵

no

?- ۲*X=Y+۲

no

دستور (is) :

اگر بخواهیم عبارت سمت راست محاسبه شود و سپس با عبارت سمت چپ یکسان سازی شود

از is استفاده می‌کنیم.

```
?- X=5, Y is X+3
```

هم باید عبارت های X و 5 یکسان شوند

و هم حاصل ریاضی X+3 و Y یکسان شوند.

خروجی: X/5 Y/8

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

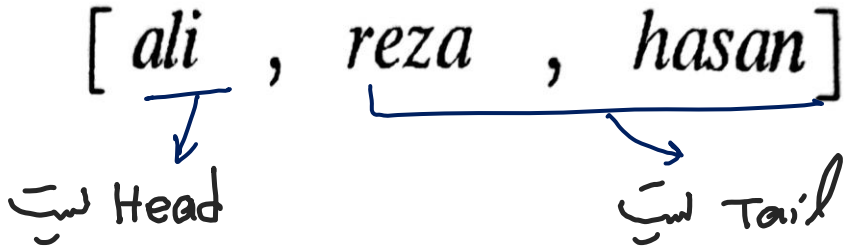
A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

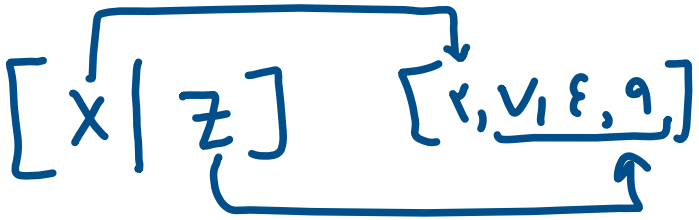
لیست‌ها در prolog :

[ ] لیست تهی است. برای هر لیست غیرتهی اولین مولفه را Head و نامیم. Head نمی‌تواند تهی باشد. اما Tail می‌تواند.



?- [X | Z] = [۲, ۶, ۷, ۹]

پاسخ: X=۲
Z = [۶, ۷, ۹]



در لیست‌های غیرتهی می‌توان Head را با علامت زیر جدا کرد:

[H | ....]

H ← Head غیرتهی

?- [x|y] = [۲]

X=۲

Y=[ ]

?- [x|y] = [ ]

fail

x نمی‌تواند چیزی باشد

چون head است.

می‌توان چند مولفه‌ی اول را نیز مشخص کرد:

(دومین چیز ، سومین چیز، ...)

```
?- [X,Y|Z] = [A,B,C,D,E].
```

output:

```
X = A
Y = B
Z = [C,D,E]
```

تک‌ن سازی لیست‌ها مانند مسندهای دیگر است:

مولفه‌های
don't care

$?-[_,_,X]=[ali\ ,\ reza\ ,\ hasan].$

$X=hasan$

چند سؤال درباره‌ی لیست‌ها:

به پاسخ (خروجی) دقت کنید:

```
?- [X | Y] = [a,b,c,d,e].
X = a
Y = [b, c, d, e]
```

```
?- [X | Y] = [a].
X = a
Y = []
```

```
?- [X | Y] = [].
no
```

```
? - [X,Y | Z]=[a,b,c,d,e].

    X = a

    Y = b

    Z = [c, d, e]
```

مسئله عضویت:

۱) $member(X, [X \mid R])$.

۲) $member(X, [Y \mid R]) :- member(X, R)$.

① head همیشه عضو لیست است. مثلاً member(۲, [۲,۵,۶]) ارزش True دارد.

② اگر X عضو لیست R باشد آنگاه X عضو لیست [Y|R] نیز هست.

۱) $member(X, [X \mid R])$.

۲) $member(X, [Y \mid R]) :- member(X, R)$.

?- member(۵, [۲, ۵, ۶])

① ۵ با head کی‌ان نیت.

② با حذف head داریم: member(۵, [۵, ۶]) ? که ارزش True دارد.

```
? - member(X, [ali, reza, hasan]).

X = ali;

X = reza;

X = hasan;
```

```
? - member([۳, Y], [[۱, a], [۳, m], [۲, z], [۳, p]]).

Y = m;

Y = p;
```

? - *member* (X , [٢٣, ٦٧, ٩, ٢٢, ٦]) , Y *is* X * X , Y < ١٠٠.

$X = ٩$ $Y = ٨١$;

$X = ٦$ $Y = ٣٦$;

مسئله الحاق (الحاق لیست‌ها)

وقتی می‌گوییم append(X, Y, Z) یعنی Z حاصل الحاق X و Y است.

مثلاً این ادعا درست است: append([۲,۳], [۴,۵,۶], [۲,۳,۴,۵,۶])

تعریف مسئله الحاق:

```
append([],Y,Y).
append([A|X],Y,[A|Z]) :- append(X,Y,Z).
```

```
append([],Y,Y).
append([A|X],Y,[A|Z]) :- append(X,Y,Z).
```

Database semantics of Prolog

یادآوری:

در Prolog برخلاف FOL از معناشناسی پایگاه داده استفاده می‌کنیم.

* نمادهای ثابت، دقیقاً به تعداد ثابت‌ها شیء وجود دارد. هر نام ثابت دقیقاً به یک شیء اشاره می‌کند و بالعکس.

* جهان بسته: تمام جمله‌ای که از KB قابل استنتاج نیستند، نادرست فرض می‌شوند.

مثال: فرض کنید در KB این جملات را داریم:

$$Course(CS, 101),\ Course(CS, 102),\ Course(CS, 106),\ Course(EE, 101).$$

(۱) یک دوره درسی EE در اتاق ۱۰۱ برگزار می‌شود.

(۲) یک دوره درسی CS در اتاق ۱۰۱ برگزار می‌شود.

(۳) یک دوره درسی CS در اتاق ۱۰۲ برگزار می‌شود.

(۴) یک دوره درسی CS در اتاق ۱۰۶ برگزار می‌شود.

با معناشناسی FOL: نتیجه می‌گیریم حداقل یک اتاق وجود دارد. (ممکن است اتاق ۱۰۱ همان ۱۰۲ و همان ۱۰۶ باشد)
حداقل یک رشته درسی وجود دارد. (ممکن است CS همان EE باشد)
حداکثر: ∞ اتاق و ∞ رشته درسی ممکن است وجود داشته باشد.

با معناشناسی پایگاه داده: نتیجه می‌گیریم دقیقاً ۳ اتاق وجود دارد. (۱۰۱ و ۱۰۲ و ۱۰۶ متمایز هستند و به جز آن فعلاً هیچ اتاقی نداریم)
دقیقاً ۲ رشته درسی وجود دارد. (CS و EE) دقیقاً ۴ کلاس برگزار می‌شود.

عملگر cut (!)

پس از یافتن اولین جواب، cut می‌شود و سایر جواب‌ها را بررسی نمی‌کند.

البته کاربردهای وسیعی با همین ویژگی ساده پیدا می‌کند. (قطع عملیات جستجو با یافتن اولین خروجی)

p,! به محض True شدن P برنامه را متوقف می‌کند

مثال تأثیر cut در خروجی :

```
? - member(X, [1,2,3,4]).
```

به ازای این مقدار True برمی‌گردد:

X=1
X=2
X=3
X=4

```
? - member(X, [1,2,3,4])!.
```

خروجی: X=1 ← cut

به محض یافتن اولین جواب
از رشته member(1,[1,2,3,4])
cut است پس True
برمی‌گردد.

مثال: می‌خواهیم در prolog کدی بنویسیم که بتواند ماکزیمم دو عدد را تعیین کند.

مسند: max (X , Y , Z) یعنی Z ، ماکزیمم X و Y است.

اگر این مسند را به این شکل معرفی کنیم:

```
max (A,B,A) :- A > B .
max (A,B,B).
```

شرط

ایرادش آن است که در هر صورت سطر دوم اجرا می‌شود.

غلط

```
max (A,B,A) :- A>B.
max (A,B,B).
```

طبق این کد، جمله‌ی max(۵,۲,۲) نیز

ارزش True پیدا می‌کند که اشتباه است. زیرا ۵ = max(۵,۲) است.

باید از cut استفاده کنیم تا اگر A>B ارزش True داشت دیگر سطر دوم اجرا نشود.

درست

```
max (A,B,A) :- A>B,!.
max (A,B,B).
```

```
max(A,B,A):-A>B,!.

max(A,B,B).
```

دس لطر دوم فقط وقتی اجرا می‌شود که A بزرگتر از B نباشد.

یعنی معادل است با دستورات زیر:

if A>B then max(A,B) = A

else

max(A,B) = B

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۱۸ فصل ۱۰

اشاره ای به فصل ۹

اشاره به چند موضوع باقی مانده از فصل‌های ۷، ۸، ۹

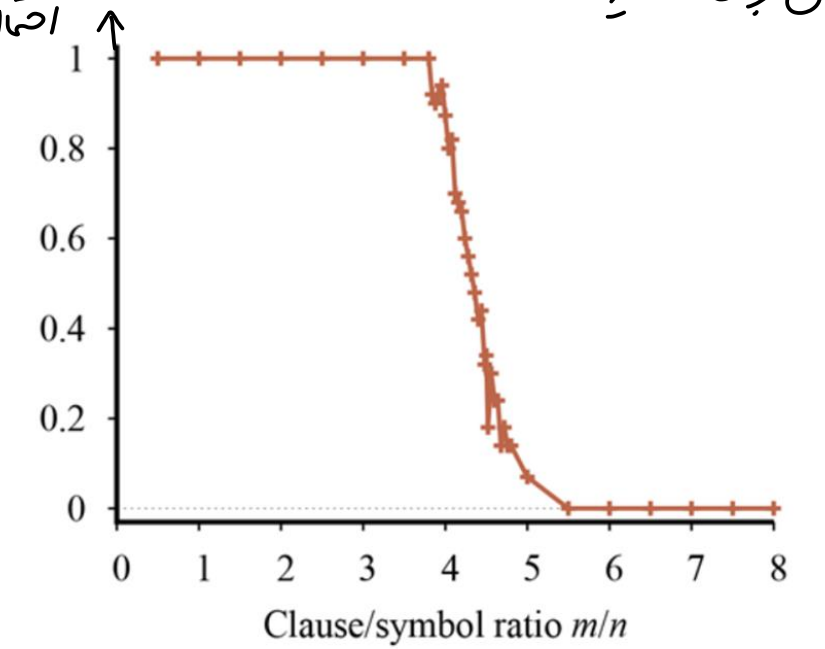
تحلیل و مقایسه زمان اجرای الگوریتم‌های SAT :

Dpll و WALK-SAT ، مجموعه‌ای از گزاره‌ها به‌عنوان ورودی گرفته و بررسی می‌کنند که ارضاپذیر است یا خیر. اگر ارضاپذیر باشند، مدل را می‌یابد. اگر ارضاپذیر نباشد، false برمی‌گرداند.

این نمودارها برای ۳-CNF با $n=50$ متغیر به دست آمده‌اند.

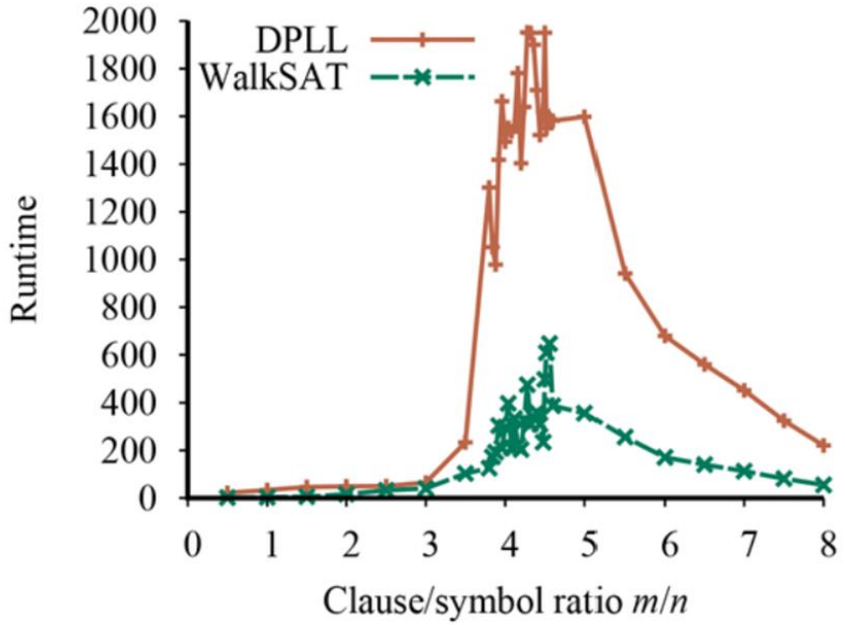
محور عمودی

احتمال ارضاپذیری ۳-CNF



(a)

محور عمودی: زمان اجرای الگوریتم‌ها



(b)

محور افقی نسبت $\frac{m}{n}$ است.

$\frac{\text{تعداد بندها}}{\text{تعداد متغیرها}} = \frac{m}{n}$

Figure 7.19 (a) Graph showing the probability that a random 3-CNF sentence with $n = 50$ symbols is satisfiable, as a function of the clause/symbol ratio m/n. (b) Graph of the median run time (measured in number of recursive calls to DPLL, a good proxy) on random 3-CNF sentences. The most difficult problems have a clause/symbol ratio of about 4.3.

تحلیل زمان اجرای الگوریتم‌های بررسی ارضاپذیری (WALKSAT و DPLL) برای k-CNF ها:

فرض کنیم: تعداد بندها $= m$

تعداد متغیرها $= n$

$\frac{m}{n}$ کوچک باشد ← احتمال ارضاپذیری $p=1$ و زمان یافتن مدل بسیار کم است.

$\frac{m}{n}$ کوچک باشد ← احتمال ارضاپذیری $p \simeq \frac{1}{2}$ و زمان اجرای الگوریتم‌ها زیاد است.

$\frac{m}{n}$ بزرگ باشد ← احتمال ارضاپذیری $p \simeq 0$ و زمان تشخیص ارضاناپذیری نسبتاً کم. (اندک)

مثلاً: 3-CNF با $m=2$ بند و $n=5$ متغیر.

$$(P_1 \vee \sim P_2 \vee P_4) \wedge (P_5 \vee P_2 \vee \sim P_4)$$

① ②

$\frac{m}{n}$ بسیار کوچک است، مدل به سرعت یافت می‌شود. کافیست $P_1=\text{True}$ و $P_2=\text{True}$ انتخاب شود و مقداری متغیرها مهم نیست.

در این مثال: $CNF=3$ با $m=12$ بند $n=4$ متغیر

داریم. $\frac{m}{n}=3$.

احتمال ارضاءپذیری نزدیک $\frac{1}{2}$ است .

با بررسی نسبتاً وقت‌گیر ارضاءپذیر بودن یا نبودن

مشخص می‌شود.

ابتدا تعداد P_1 از P_1 و $\sim P_1$ بیشتر است پس $P_1=True$. (۱،۲،۳،۴،۵،۶،۷ معتبر می‌شوند)

در سایر بندها: تعداد P_2 از P_2 و $\sim P_2$ بیشتر است پس $P_2=True$ (۱،۲،۳،۹،۱۱،۱۲ معتبر می‌شوند)

حال بندهای ۸ و ۱۰ باقی می‌مانند، در یکی P_4 و در دیگری $\sim P_4$ داریم.

پس این مسیر به شکست می‌رسد. باید با عقب‌گرد، سایر شاخه‌ها بررسی شوند

[با این مثال متوجه شدیم وقتی احتمال ارضاءپذیری $\frac{1}{2}$ است زمان اجرا زیاد است]

۱ $(P_1 \vee \sim P_2 \vee \sim P_3)$

۲ $(P_1 \vee P_2 \vee \sim P_3)$

۳ $(P_1 \vee \sim P_2 \vee P_3)$

۴ $(P_1 \vee P_2 \vee P_3)$

۵ $(P_1 \vee \sim P_2 \vee P_3)$

۶ $(P_1 \vee P_2 \vee \sim P_3)$

۷ $(P_1 \vee \sim P_2 \vee \sim P_3)$

۸ $(\sim P_1 \vee \sim P_2 \vee \sim P_4)$

۹ $(\sim P_1 \vee P_2 \vee \sim P_3)$

۱۰ $(\sim P_1 \vee \sim P_2 \vee P_4)$

۱۱ $(\sim P_1 \vee P_2 \vee P_3)$

۱۲ $(\sim P_1 \vee P_2 \vee P_4)$

وقتی $\frac{m}{n}$ به مقدار کافی زیاد باشد احتمال ارضاپذیری نزدیک به صفر

و زمان بررسی نسبتاً کم است. (حتی زود عدم وجود مدل مشخص میشود)

البته این مثال خیلی نمونه خوبی برای نشان دادن این حالت نیست

(یک مثال مناسب، 3-CNF دارای $n=50$ متغیر

و $m=200$ میباشد)

۱ $P_1 \vee P_2$

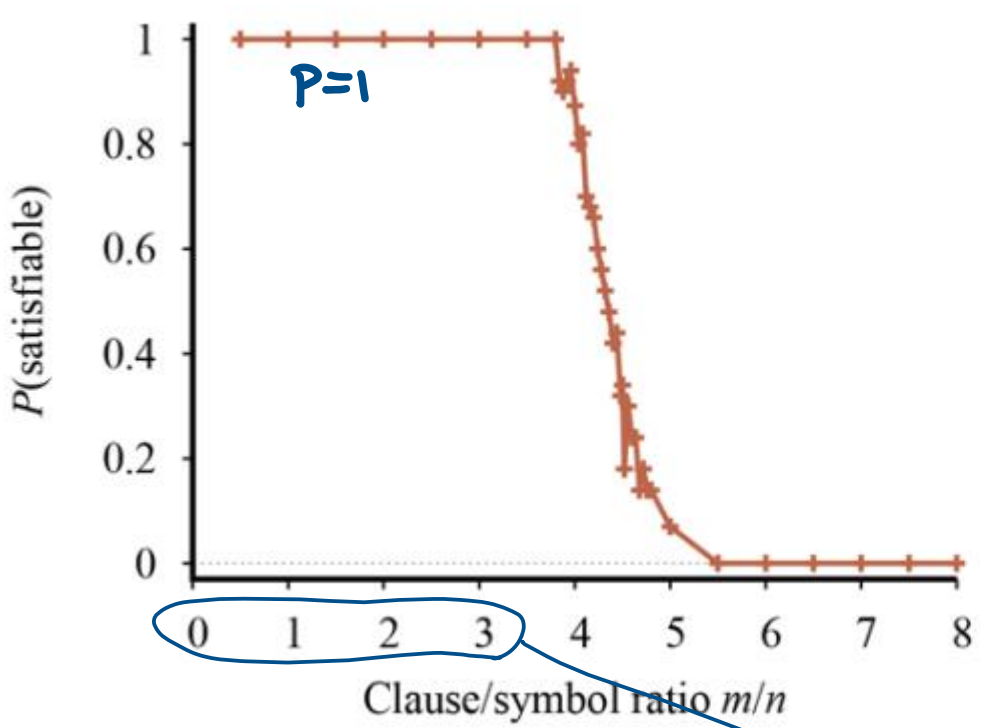
۲ $\sim P_1 \vee P_2$

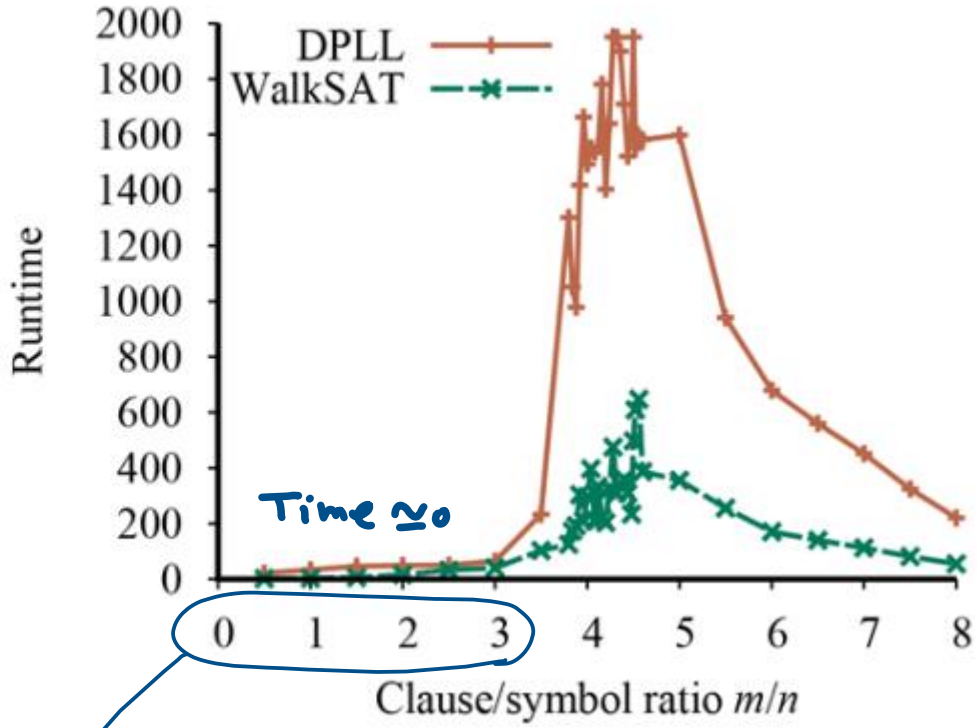
۳ $P_1 \vee \sim P_2$

۴ $\sim P_1 \vee \sim P_2$

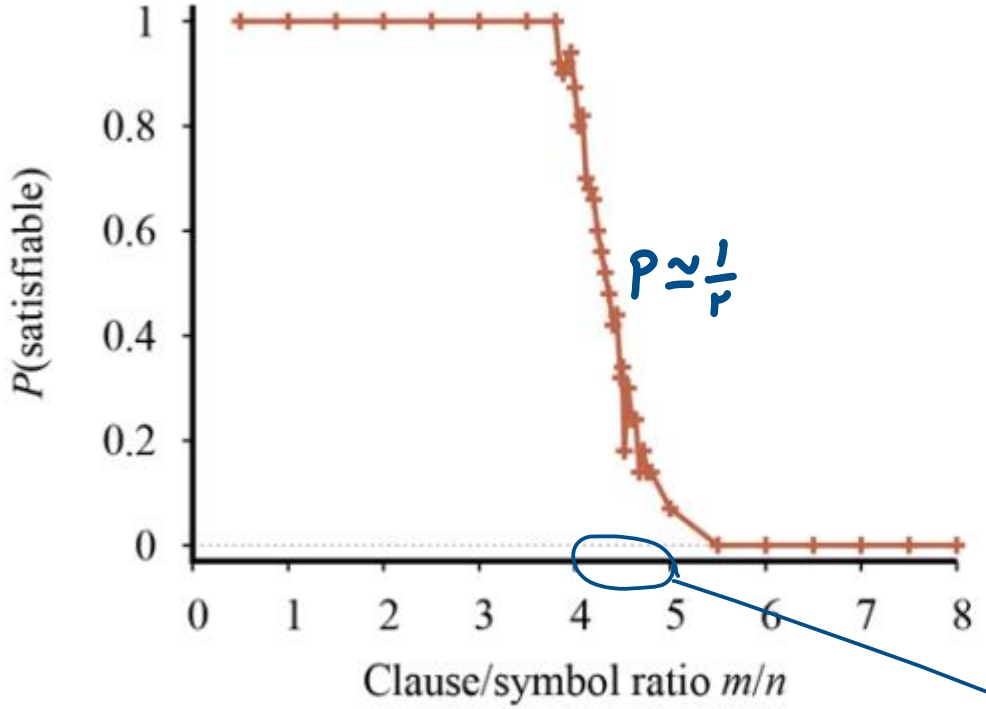
خلاصه آنچه گفتیم در ۳ اسلاید تکمیلی زیر بیان شده است.

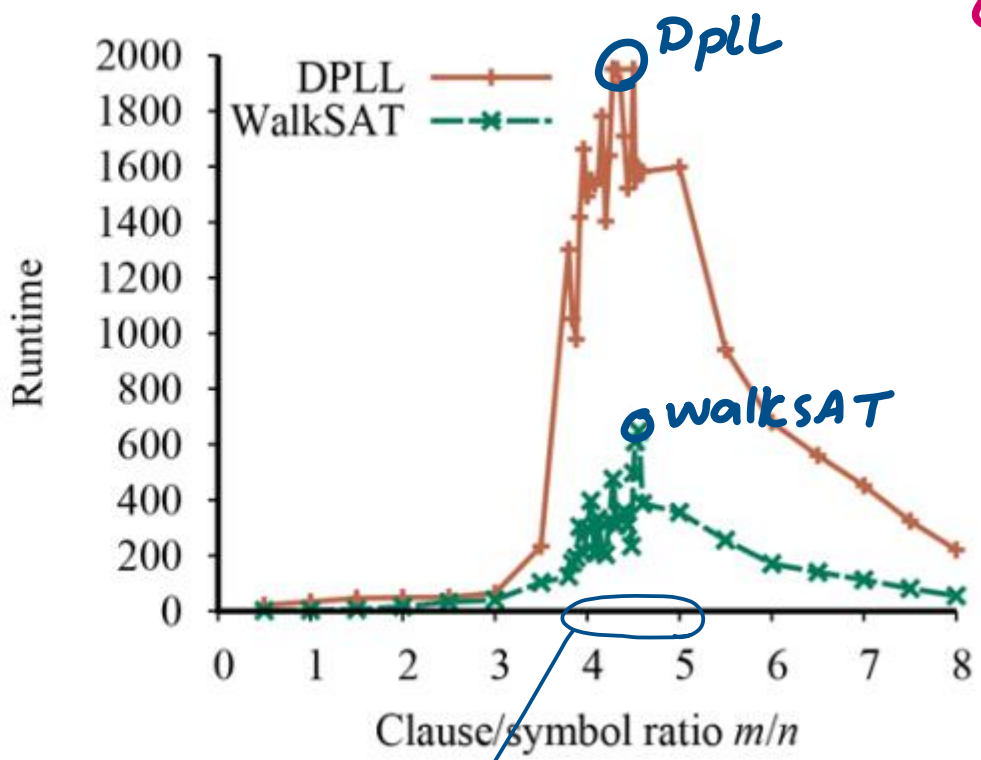
اسلایدِ تکمیلی





$\frac{m}{n} \leq 3$ ← احتمال ارضاپذیری $P=1$ است و الگوریتم‌ها خیلی زود مدل را پیدا می‌کنند.

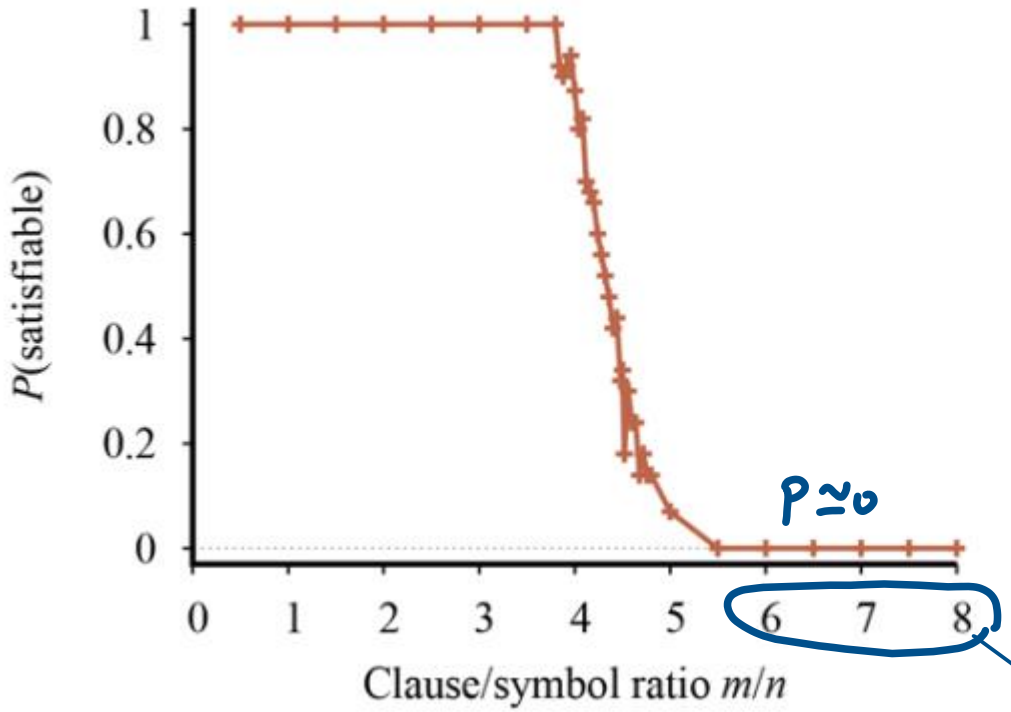


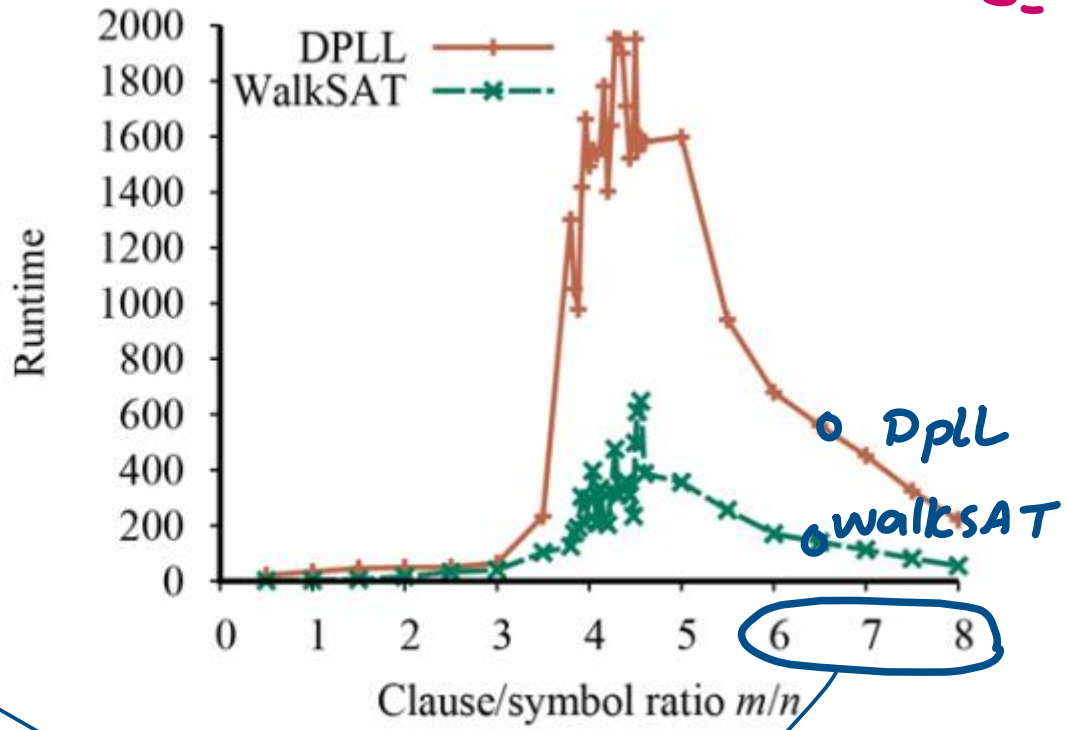


$\frac{m}{n} \simeq 4.5$ ← احتمال ارضاپذیری $P \simeq \frac{1}{2}$ است و بهترین زمان اجرا به دست می‌آید.

(طبق شکل برای 3-CNF ها walk-SAT سریعتر از DPLL است)

اسلایدِ تکمیلی





$\frac{m}{n} \geq 6$ ← احتمال ارضاپذیری $P \simeq 0$ است و زمان اجرای نسبتاً کاهش می‌یابد اما نسبت به وقتی که $\frac{m}{n} < 4$ بود، زمان بیشتری لازم است.

تاوی رابطه‌ی تاوی (یعنی این-همانی) در زندگی واقعی به این معناست که آن دو ترم، یکی هستند

مثلاً پدرِ پدرِ علی همان پدربزرگ پدری علی است.

اگر بخواهیم عامل منطقی این رابطه را درک کنیم و در استنتاج‌ها مورد استفاده قرار دهیم چگونه باید آن را به دانش عامل اضافه کنیم؟

Equality

درباره‌ی مسند تساوی؛ (این همانی) . الگوریتم‌های استنتاج FOL رابطه‌ی تساوی را چگونه می‌توانند اعمال کنند؟

در این جزوه به جای $Equal(x, y)$ از همان علامت $x=y$ استفاده خواهیم کرد.

راه اول اضافه کردن این فرض ها (اصول موضوعی تاوی) به پایگاه دانش است.

① $\forall x \;\; x = x$

② $\forall x,y \;\; x = y \Rightarrow y = x$

③ $\forall x,y,z \;\; x = y \wedge y = z \Rightarrow x = z$

برای هر مسند P

$\forall x,y \;\; x = y \Rightarrow (P_1(x) \Leftrightarrow P_1(y))$

$\forall x,y \;\; x = y \Rightarrow (P_2(x) \Leftrightarrow P_2(y))$

$\vdots$

برای هر تابع F

$\forall w,x,y,z \;\; w = y \wedge x = z \Rightarrow (F_1(w,x) = F_1(y,z))$

$\forall w,x,y,z \;\; w = y \wedge x = z \Rightarrow (F_2(w,x) = F_2(y,z))$

$\vdots$

سه تای اول اصول موضوع اصلی هستند.

بقیه اصول، بیان می‌کنند که وقتی $x = y$ است، و F یک تابع دلخواه $F(x) = F(y)$ است. به همین ترتیب برای همه‌ی مسندها:

demodulation, روش دوم استفاده از قاعده‌ی دمدولاسیون و پارامدولاسیون است:

تا وی

$$KB = \{ \quad x = y \quad , \quad l_1 \vee l_2 \vee \dots \vee l_k \}$$

α شامل ترم z است. جانشانی θ وجود دارد که z و x را یکسان می‌کند.

می‌توان جانشانی θ را انجام داد. $SUBST(\theta, \alpha)$ را احتجاج کرد

و در آن به جای x، y را قرار داد.

example, given

$$Father(Father(x)) = PaternalGrandfather(x)$$
$$Birthdate(Father(Father(Bella)), 1926)$$

we can conclude by demodulation

$$Birthdate(PaternalGrandfather(Bella), 1926).$$

- **PARAMODULATION:** For any terms x, y, and z, where z appears somewhere in literal m_i, and where $\textsc{Unify}(x,z) = \theta$,

$$\frac{\ell_1 \vee \cdots \vee \ell_k \vee x = y, \qquad m_1 \vee \cdots \vee m_n}{\textsc{Sub}(\textsc{Subst}(\theta,x), \textsc{Subst}(\theta,y), \textsc{Subst}(\theta,\ell_1 \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_n)}.$$

For example, from

$$P(F(x,B),x) \vee Q(x) \qquad \text{and} \qquad F(A,y) = y \vee R(y)$$

we have $\theta = \textsc{Unify}(F(A,y),F(x,B)) = \{x/A,y/B\}$, and we can conclude by paramodulation the sentence

$$P(B,A) \vee Q(A) \vee R(B).$$

Paramodulation yields a complete inference procedure for first-order logic with equality.

For example,

$$KB = \{\underbrace{H(x, F(Ali, x)) \vee Q(x) \vee P(t)}_{\text{بند ①}}, \underbrace{F(y, Arash) = Mina \vee S(y)}_{\text{بند ②}}\}$$

در بند ② تک وی $F(y, Arash) = Mina$ داریم. در بند ① ترم $F(Ali, x)$ را داریم که با $F(y, Arash)$ یکسان می‌شود: $\theta = \{x/Arash, y/Ali\}$ این جانشانی را در تمام جملات اعمال کنیم سپس به جای $F(Ali, Arash)$ از Mina استفاده کنیم. حالا کاری شبیه به رزولوشن انجام دهیم:

نتیجه:

$$H(Arash, Mina) \vee Q(Arash) \vee P(t) \vee S(Ali)$$

10 CLASSICAL PLANNING

Classical planning is defined as the task of finding a sequence of actions to accomplish a goal in a discrete, deterministic, static, fully observable environment.

در فصل ۷ (منطق گزاره‌ای)

برای n^2 محل و از هر محل ۴ حرکت،

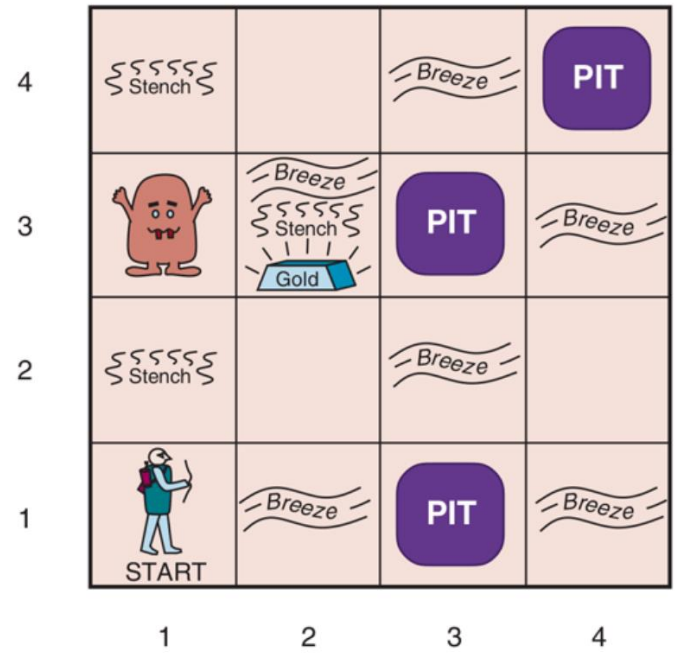
که در زمان‌های $t = 1, 2, \ldots, T$ باید تکرار شود،

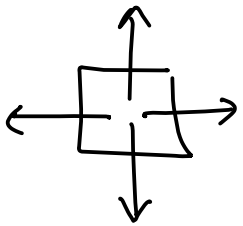
به تعداد $4n^2T$ جمله نیاز داریم.

اما ما خواهیم همه این اقدامات را در قالب یک

طرح واره اقدام (با استفاده از متغیرها)

بیان کنیم.





مثلاً اقدام حرکت را در وامپوس
در چنین تعریف می‌کنیم:

نام اقدام

Action: move((i,j) , (k,l))

پیش‌شرط‌ها:

PRECOND: $AT(i,j) \wedge (|i-k|+|j-l|=1) \wedge (\sim B(i,j)) \wedge \dots$

$AT(i,j)$ ← اکنون در (i,j) باشیم.

$|i-k|+|j-l|=1$ ← یعنی دو خانه‌ی مجاور باشند.

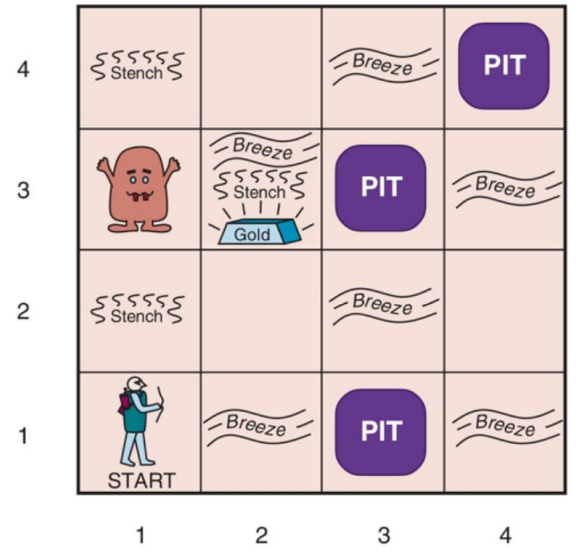
$\sim B(i,j)$ ← نیم در (i,j) احتمالی نبود.

اثرات:

EFEC: $\sim AT(i,j) \wedge AT(k,l) \wedge \dots$

$\sim AT(i,j)$ ← دیگر در (i,j) نیستم

$AT(k,l)$ ← در (k,l) هستم



PDDL, the Planning Domain Definition Language,

Database semantics

زبان PDDL را به عنوان نمونه‌ای از زبان‌های برنامه‌ریزی توضیح خواهیم داد.

این زبان از معناشناسی پایگاه داده استفاده می‌کند:

- کیتای نام‌ها
- فرض جهان بسته
- بدون تابع

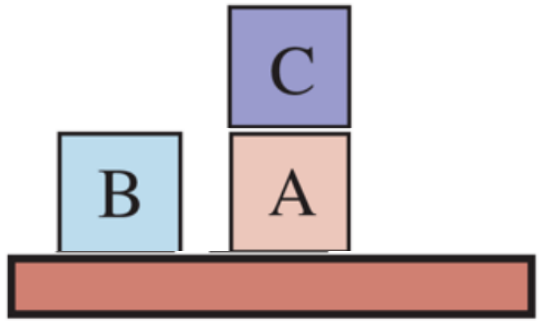
Each **state** is represented as a conjunction of fluents that are ground, functionless atoms. For example, $Poor \wedge Unknown$ might represent the state of a hapless agent, and a state in a package delivery problem might be $At(Truck_1, Melbourne) \wedge At(Truck_2, Sydney)$. **Database semantics** is used: the closed-world assumption means that any fluents that are not mentioned are false, and the unique names assumption means that $Truck_1$ and $Truck_2$ are distinct. The following fluents are *not* allowed in a state: $At(x, y)$ (because it is non-ground), $\neg Poor$ (because it is a negation), and $At(Father(Fred), Sydney)$ (because it uses a function symbol). The representation of states is carefully designed so that a state can be treated either as a conjunction of fluents, which can be manipulated by logical inference, or as a *set* of fluents, which can be manipulated with set operations. The **set semantics** is sometimes easier to deal with.

State: ترکیب عطفی از لیترال‌های مثبت زمینه‌ای بدون تابع

Fluents: روان‌ها، گزاره‌هایی که ارزش آنها ممکن است تغییر کند.

پس هر state را می‌توان یک مجموعه از لیترال‌های مثبت زمینه‌ای بدون تابع، فرض کرد.

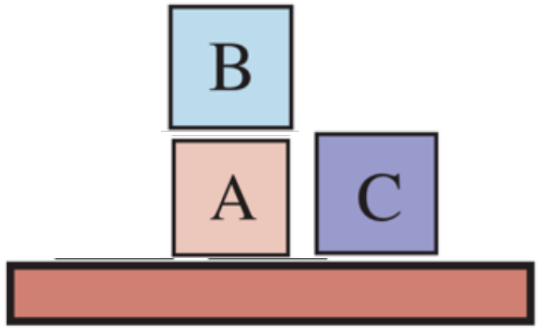
مثال ۱: این state در دنیای مکعب‌ها چنین نوشته می‌شود:



هم‌ترال‌ها بدون متغیر (زمینه‌ای) هستند.
مثبت هستند، نمادتابع ندارند.

$$On(A, Table) \wedge On(B, Table) \wedge On(C, A) \wedge Block(A) \wedge Block(B) \wedge Block(C) \wedge Clear(B) \wedge Clear(C))$$

مثال ۲:



$$On(A, Table) \wedge On(C, Table) \wedge On(B, A) \wedge Block(A) \wedge Block(B) \wedge Block(C) \wedge Clear(B) \wedge Clear(C))$$

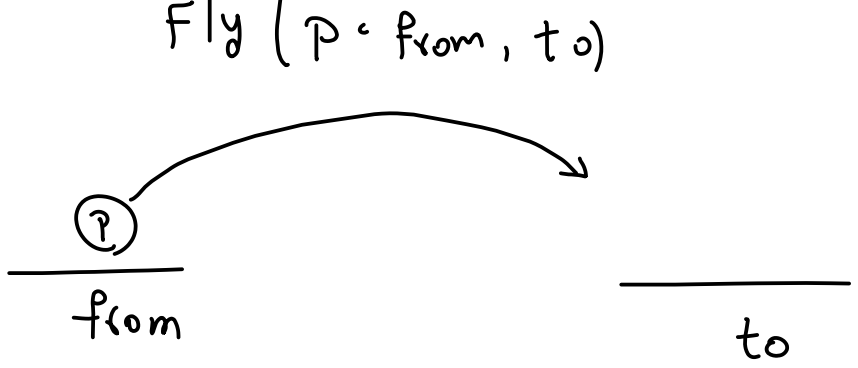
طرح واره‌ی اقدام:

- نام اقدام
- پیش‌شرط‌ها
- اثرات
  - ADD(a): اثرات مثبت
  - DEL(a): اثرات منفی

مثال:

$Action(Fly(p, from, to),$ ← نام
PRECOND: $At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$ ← پیش‌شرط‌ها
EFFECT: $\neg At(p, from) \wedge At(p, to))$ ← اثرات

p: یک مسافرت (هواپیمایی که می‌تواند باشد)
from: هر فرودگاه
to: هر فرودگاه



اصل موضوع پیش شرط

اقدام a در s قابل اجراست اگر و تنها اگر پیش شرط های آن در s نتیجه شوند (مقید شوند)

$$(a \in \text{ACTIONS}(s)) \Leftrightarrow s \models \text{PRECOND}(a)$$

نتیجه‌ی اجرای a در حالت s :

$$\text{RESULT}(s, a) = (s - \text{DEL}(a)) \cup \text{ADD}(a)$$

از لیترال‌های کم، اثرات منفی حذف و اثرات مثبت اضافه می‌شوند.

$$s^+ = (S - \text{DEL}(a)) \cup \text{ADD}(a)$$

مثال (کمک آموزشی)

اکنون در یک هستیم:

$S = P_1 \wedge P_2 \wedge P_3 = \{P_1, P_2, P_3\}$

اقدام a چنین است:

Action: a

PREC: $P_1 \wedge P_2$

EFEC: $\sim P_1 \wedge P_4 \wedge P_3$

در یک شرایط a برقرار است، و با انجام آن به این state می‌رسیم

$$\text{RESULT}(s, a) = (s - \text{DEL}(a)) \cup \text{ADD}(a)$$

$$= (\{P_1, P_2, P_3\} - \{P_1\}) \cup \{P_3, P_4\}$$

$$= \{P_2, P_3, P_4\}$$

propositionalize a PDDL problem-

هر طرح واره اقدام با جانشینی ثابت ها به جای متغیرها به یک اقدام زمینه ای (در منطق گزاره ای) تبدیل می شود.

مثال:

، P/P_1 from/SFO to/JFK .

پرواز P_1 از سانفرانسیسکو به جان اف کندی .

$$Action(Fly(P_1, SFO, JFK),$$
$$\text{PRECOND:} At(P_1, SFO) \wedge Plane(P_1) \wedge Airport(SFO) \wedge Airport(JFK)$$
$$\text{EFFECT:} \neg At(P_1, SFO) \wedge At(P_1, JFK))$$

In PDDL the times and states are implicit in the action schemas: the precondition always refers to time t and the effect to time $t + 1$.

در PDDL : پیش شرط ها در زمان t
اثرات مربوط به زمان $t+1$ هستند.

F^t = روان های معتبر در لحظه t

F^{t+1} = روان های معتبر در $t+1$.

$$F^{t+1} \Leftrightarrow ActionCausesF^t \vee (F^t \wedge \neg ActionCausesNotF^t)$$

The **initial** **state** is a conjunction of ground atoms.

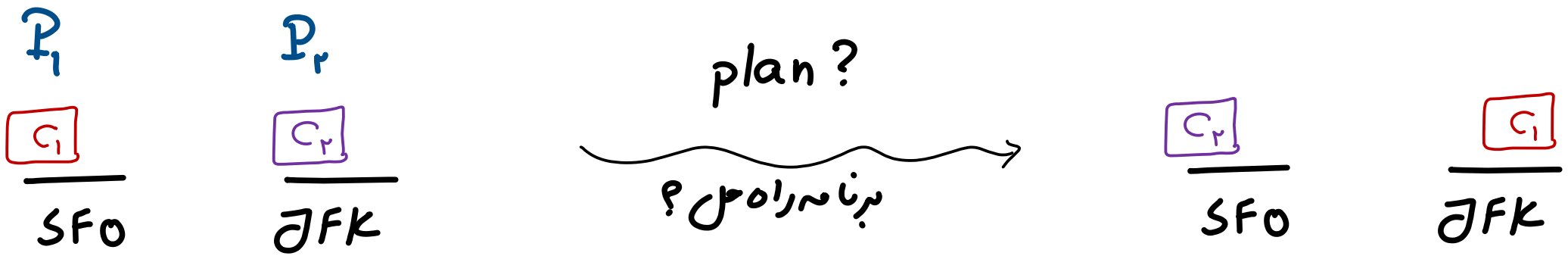
The **goal** is just like a precondition: a conjunction of literals (positive or negative) that may contain variables, such as $At(p, SFO) \wedge Plane(p)$.

در تعریف goal می‌توان از لیترال‌های مثبت یا منفی و متغیر نیز استفاده کرد.

Figure 10.1 A PDDL description of an air cargo transportation planning problem.

مثال حمل ونقل فرودگاهی

$$
\begin{aligned}
&Init(At(C_1, SFO) \land At(C_2, JFK) \land At(P_1, SFO) \land At(P_2, JFK) \\
&\quad \land Cargo(C_1) \land Cargo(C_2) \land Plane(P_1) \land Plane(P_2) \\
&\quad \land Airport(JFK) \land Airport(SFO)) \\
&Goal(At(C_1, JFK) \land At(C_2, SFO)) \\
&Action(Load(c, p, a), \\
&\quad \text{PRECOND: } At(c, a) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a) \\
&\quad \text{EFFECT: } \neg At(c, a) \land In(c, p)) \\
&Action(Unload(c, p, a), \\
&\quad \text{PRECOND: } In(c, p) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a) \\
&\quad \text{EFFECT: } At(c, a) \land \neg In(c, p)) \\
&Action(Fly(p, from, to), \\
&\quad \text{PRECOND: } At(p, from) \land Plane(p) \land Airport(from) \land Airport(to) \\
&\quad \text{EFFECT: } \neg At(p, from) \land At(p, to))
\end{aligned}
$$



(وضعیت اولیه هواپیماها و محموله‌ها)

لیترال‌های هدف.

مکان هواپیماها مهم نیست پس

در واقع ۴ وضعیت هدف داریم.

The following plan is a solution to the problem:

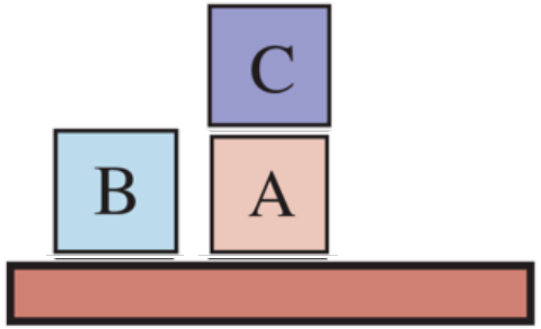
یک راه‌حل مسئله‌ی حمل‌ونقل:

$$[Load(C_1, P_1, SFO), Fly(P_1, SFO, JFK), Unload(C_1, P_1, JFK),$$
$$Load(C_2, P_2, JFK), Fly(P_2, JFK, SFO), Unload(C_2, P_2, SFO)] .$$

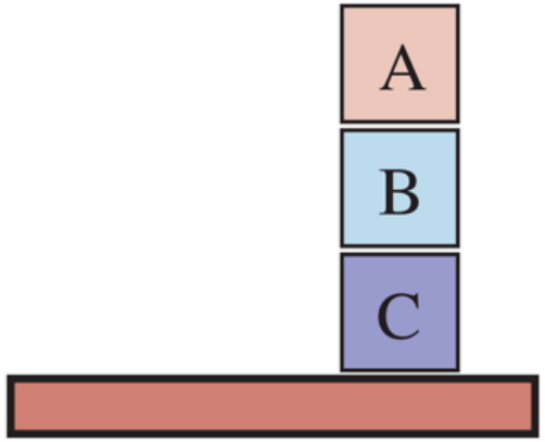
مدل‌سازی مکعب‌ها:

$$
\begin{aligned}
&Init(On(A, Table) \land On(B, Table) \land On(C, A) \\
&\quad \land Block(A) \land Block(B) \land Block(C) \land Clear(B) \land Clear(C)) \\
&Goal(On(A, B) \land On(B, C)) \\
&Action(Move(b, x, y), \\
&\quad \text{PRECOND: } On(b, x) \land Clear(b) \land Clear(y) \land Block(b) \land Block(y) \land \\
&\qquad (b \neq x) \land (b \neq y) \land (x \neq y), \\
&\quad \text{EFFECT: } On(b, y) \land Clear(x) \land \neg On(b, x) \land \neg Clear(y)) \\
&Action(MoveToTable(b, x), \\
&\quad \text{PRECOND: } On(b, x) \land Clear(b) \land Block(b) \land (b \neq x), \\
&\quad \text{EFFECT: } On(b, Table) \land Clear(x) \land \neg On(b, x))
\end{aligned}
$$

Figure 10.3 A planning problem in the blocks world: building a three-block tower. One solution is the sequence $[MoveToTable(C, A), Move(B, Table, C), Move(A, Table, B)]$.



حالت اولیه



Goal State

هدف

الگوریتم‌های حل مسائل برنامه‌ریزی کلاسیک:

1. جستجو در فضای حالت
   - Forward
   - Backward
2. الگوریتم graphplan
3. جستجو در فضای برنامه‌ها (مانند P.O.P)
4. ...

ALGORITHMS FOR PLANNING AS STATE-SPACE SEARCH

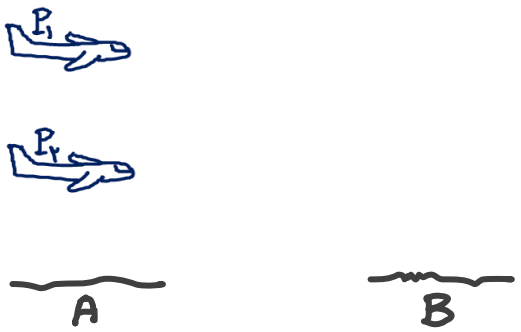
حل مسائل برنامه ریزی به عنوان جستجو در فضای حالات

forward. جستجو رو به جلو

Backward. جستجو رو به عقب

یک مثال ساده: فقط اقدام پرواز را داریم

Fly(P, x, y) .

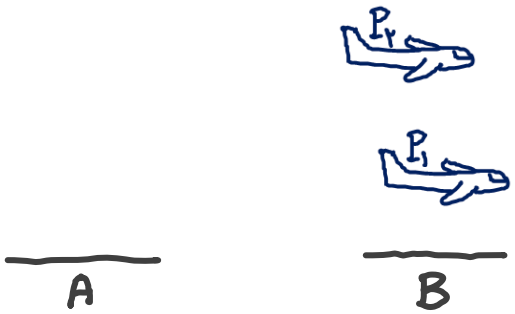


وضعیت اولیه:

A و B فرودگاه هستند.

P_1 و P_2 هواپیما هستند.

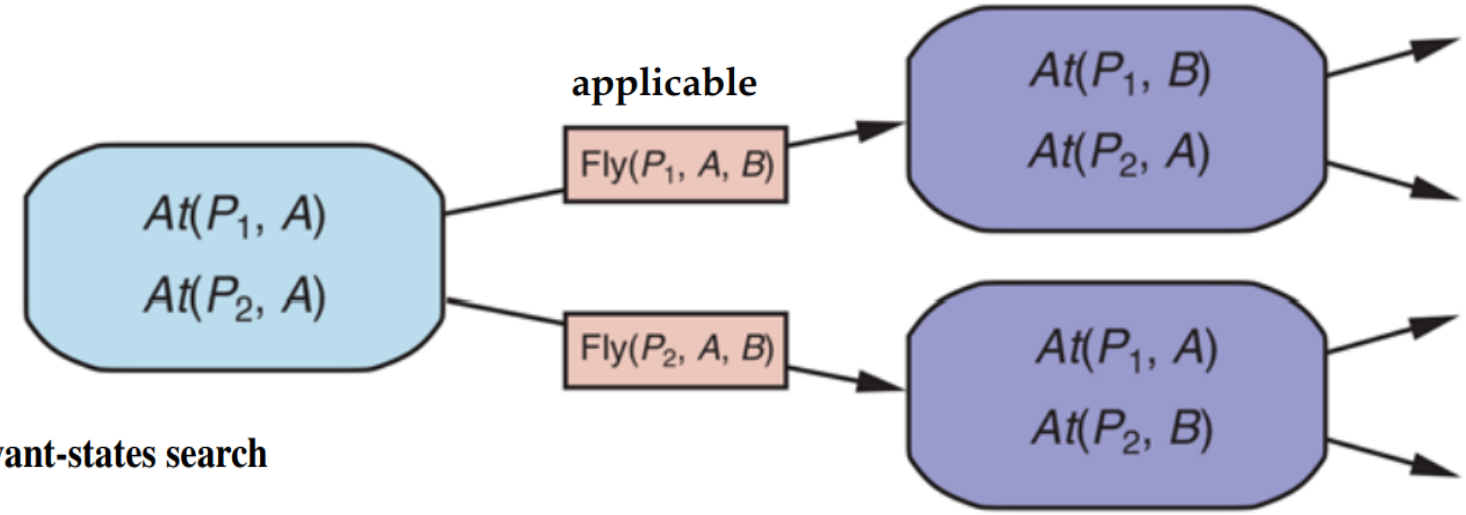
هر دو در A قرار دارند.



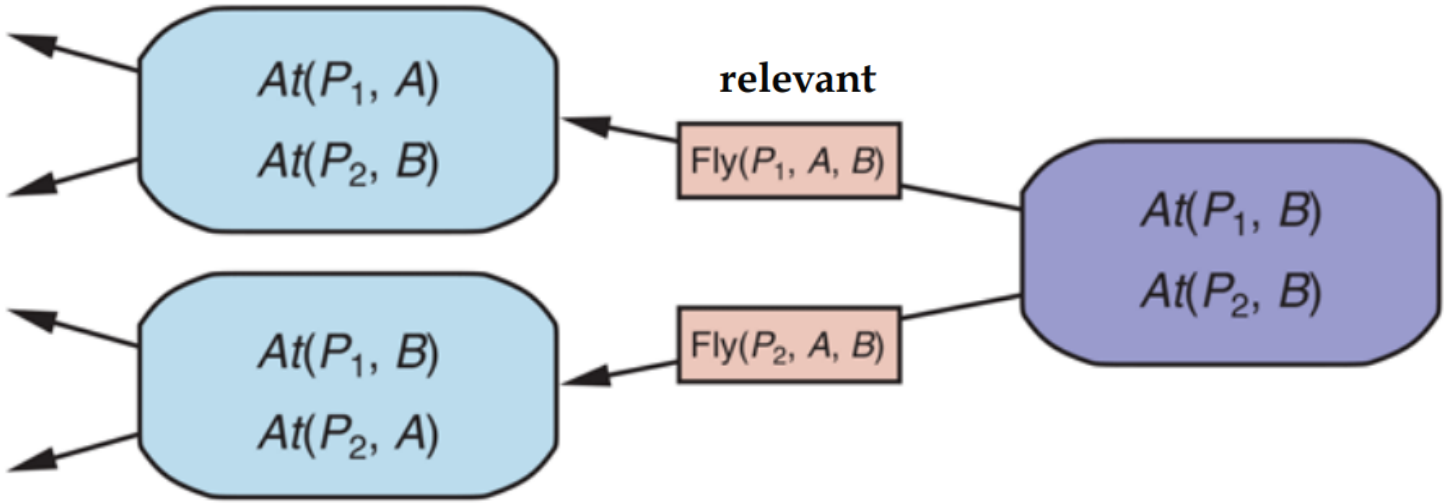
هدف:

هر دو هواپیما در B باشند

Forward (progression) state-space search



Backward (regression) relevant-states search



Forward (progression) state-space search

Forward(S, goal)

S در آزمون هدف صدق می‌کند؟ ← yes

↓ No

Actions(S) = S مجموعه‌ی اقدامات قابل اجرای

آیا Actions(S) = ∅ است ← yes fail

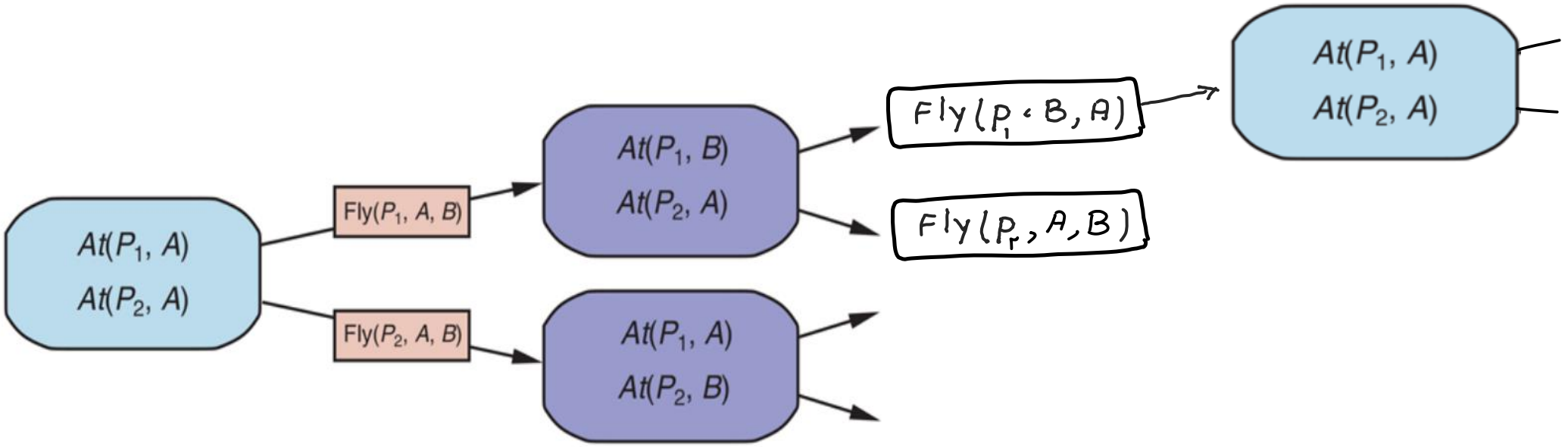
↓ No

استخراج $a \in Actions(S)$:

$S^+ = Result(S, a) = (S - DEll(a)) \cup Add(a)$

Forward(S^+, goal)

Forward (progression) state-space search



Forward : ممکن است در loop قرار بگیرد. اما با کنترل حالات تکراری یا با محدودیت عمق تکراری،
یک جستجوی کامل است. (عاد تابعی نداریم تعداد حالات متناهی است)

Backward (regression) relevant-states search

چالش های جستجو در فضای حالات به صورت Backward:

① کدام اقدام، آخرین اقدام می تواند باشد؟ relevant

② حالت ما قبل چگونه تعیین شود؟

پاسخ سوال ① :

A relevant action is one with an effect that **unifies** with one of the goal literals, but with no effect that negates any part of the goal.

اقدامی میتواند relevant باشد که دو شرط را داشته باشد:

برای اینکه به هدف موثر باشد: یعنی حداقل یکی از لیترال های هدف، در اثرات مثبت آن اقدام موجود باشد (قابل یکسان سازی باشد)

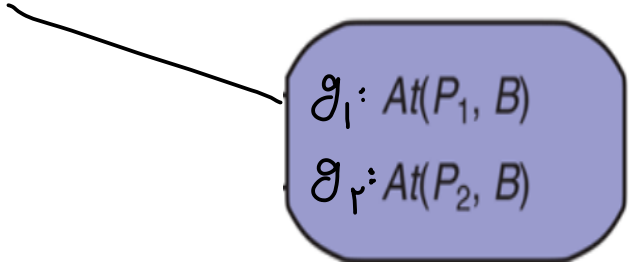
با هدف سازگار باشد: هیچکدام از لیترال های هدف در لیست اثرات منفی آن (لیست حذف) قرار نداشته باشد.

مثال: کدام اقدام می‌تواند relevant باشد؟

a_1: Fly(P_2, A, B)

a_2: Fly(P_1, B, A)

a: Fly(P_1, A, B)



goal = $g_1 \wedge g_2$

اقدام a_1: موثر نیست. هیچکدام از g_1 و g_2 را تولید نمی‌کند

اقدام a_2: سازگار نیست: باعث حذف g_1 می‌شود.

اقدام a: هم موثر است ($g_1 \in ADD(a)$) و هم سازگار است (g_1 و g_2 در لیست حذف a قرار ندارند)

حداقل شرایط لازم برای اقدام Relevant

a با g سازگار باشد:

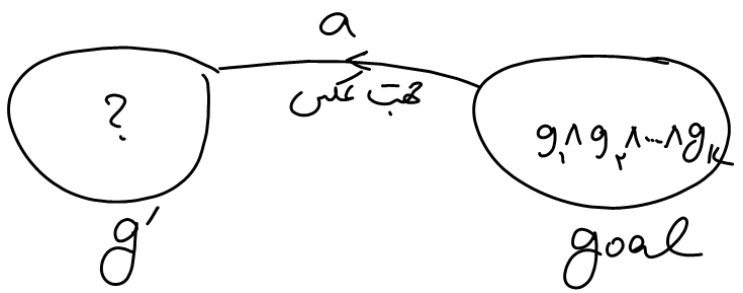
$$DELL(a) \cap g = \phi$$

a برای رسیدن به g موثر باشد:

$$ADD(a) \cap g \neq \phi$$

پاسخ چالش ②

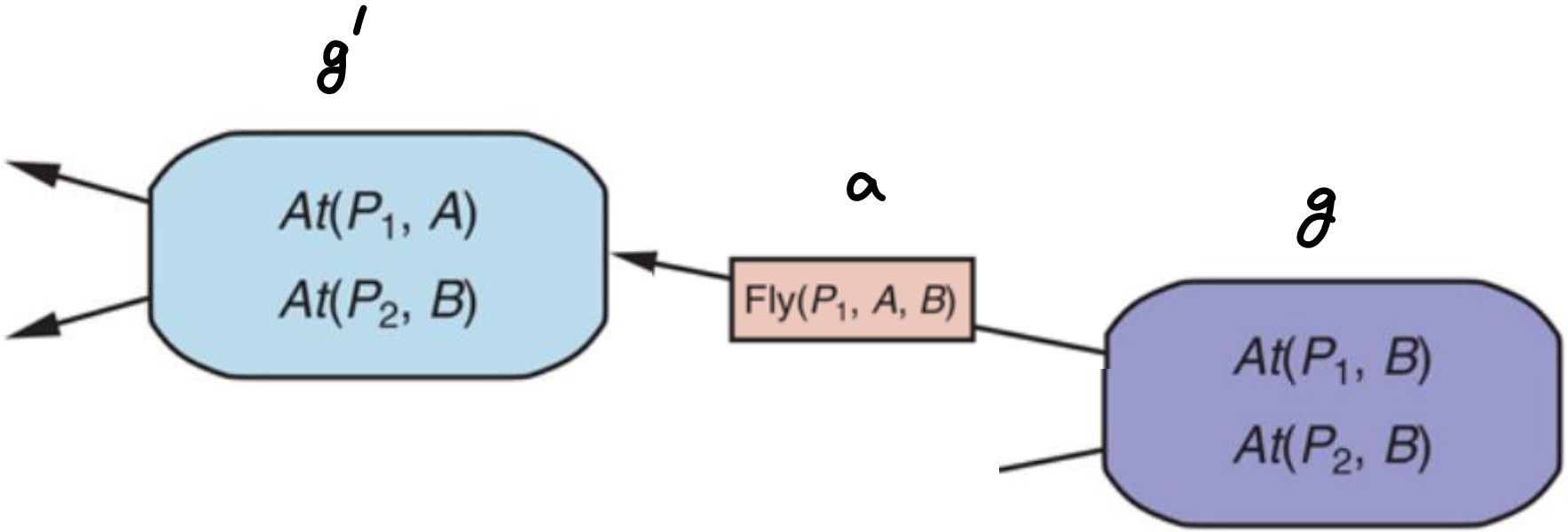
چگونه از g و a به g' می‌رسیم؟



$$g' = (g - \text{ADD}(a)) \cup Precond(a) \ .$$

یا به تفکیک لیترال‌های مثبت و منفی:

$$\text{POS}(g') = (\text{POS}(g) - \text{ADD}(a)) \cup \text{POS}(Precond(a))$$
$$\text{NEG}(g') = (\text{NEG}(g) - \text{DEL}(a)) \cup \text{NEG}(Precond(a)) \ .$$



$$g' = \left(\{ At(P_1, B) , At(P_2, B) \} - \{ At(P_1, B) \} \right) \cup \left(At(P_1, A) \right)$$

Goal

↓ اثرات مثبت a

↓ پیش شرط‌های a

(البته در این مثال به شرط‌هایی مانند فرودگاه بودن A و B و هواپیما بودن P_1, P_2 اشاره نکرده‌ایم)

در برنامه ریزی کلاسیک { Forward (progression) state-space search
Backward (regression) relevant-states search

هیچکدام بهینه نیستند

با استفاده از هیوریستیک های مناسب (حالات تکراری) کامل هستند.

در برنامه ریزی کلاسیک تعداد حالات متناهی است.

هردو، جستجو در فضای حالات هستند.

B.S در عمل سریعتر از F.S است.

زمان اجرای آنها در حالت کلی چندجمله ای نیست (نمایی است)

جلسه بعد: هیوریستیک های برنامه ریزی چگونه به دست می آیند.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

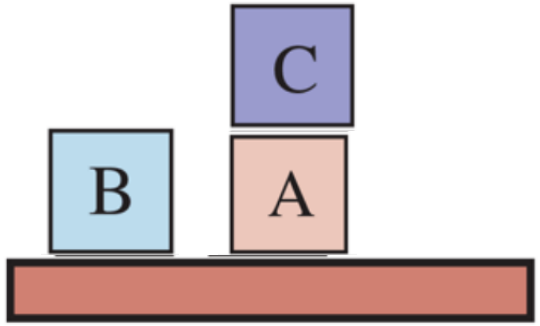
Artificial Intelligence

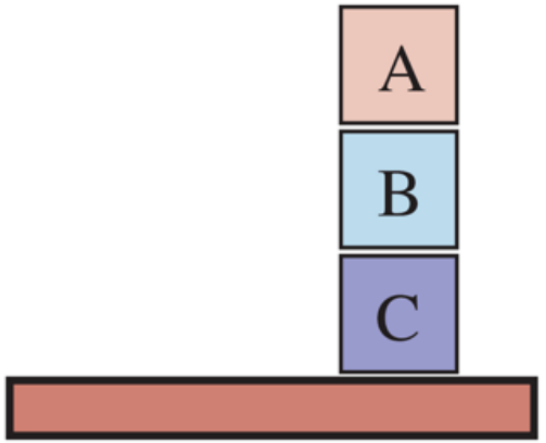
A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۱۹ فصل ۱۰





Goal State

یک مثال ساده برای معرفی الگوریتم‌ها



goal:

At(P1, B)

At(P2, B)

اقدام: Fly(p, from, to)



حالت اولیه

Forward (progression) state-space search

Backward (regression) relevant-states search

} هردو جستجو در فضای حالات هستند.

اما برخی الگوریتم‌ها مانند POP جستجو در فضای برنامه‌ها هستند.

Forward (progression) state-space search

Forward(S, goal)

S در آزمون هدف صدق می‌کند؟ ← Yes

↓ No

Actions(S) = S مجموعه‌ی اقدامات قابل اجرای

آیا Actions(S) = ∅ است ← Yes fail

↓ No

استخراج: $a \in Actions(S)$

$S^+ = Result(S, a) = (S - DEL(a)) \cup ADD(a)$

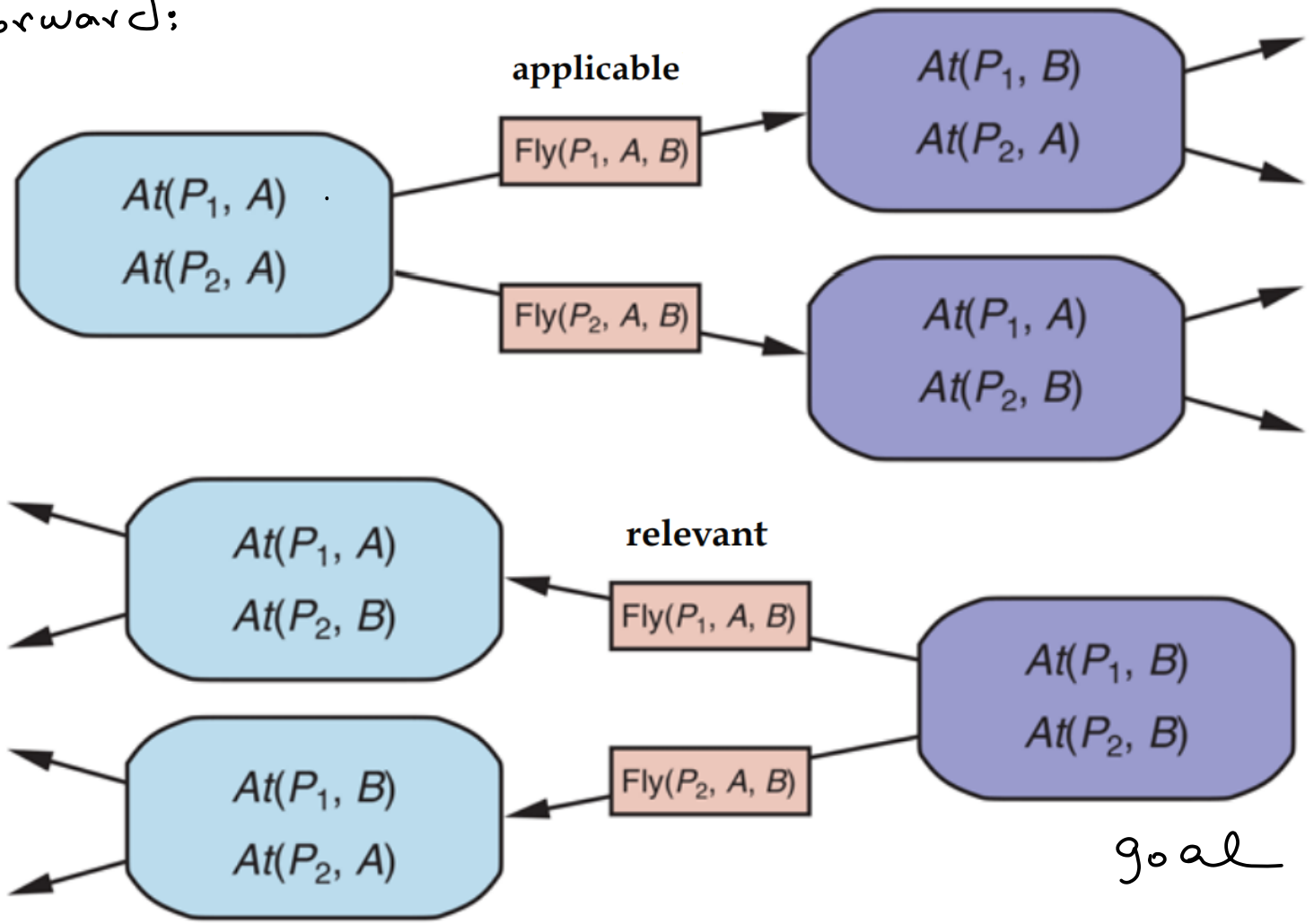
Forward(S^+, goal)

هر بار یکی از اقدامات قابل اجرا را انتخاب و اجرا می‌کنیم. اگر مجموعه اقدامات قابل اجرا تهی شود عقب‌گرد می‌کنیم

اگر ایجاد حالات تکراری جلوگیری کنیم روش کامل است

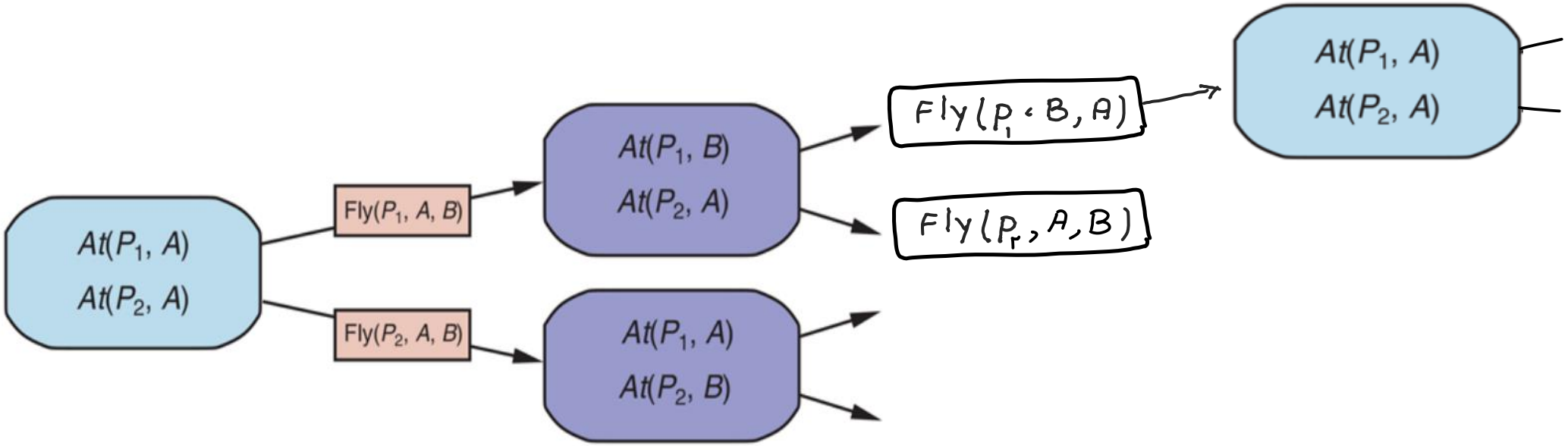
البته بهینه نیست.

forward:



Backward

Forward (progression) state-space search



Forward : ممکن است در loop قرار بگیرد. اما با کنترل حالات تکراری یا با محدودیت عمق تکراری،
یک جستجوی کامل است. (عادتا بلی نداریم تعداد حالات متناهی است)

Backward (regression) relevant-states search

جستجوی عقب‌گرد در فضای حالات به صورت Backward:

① کدام اقدام، آخرین اقدام می‌تواند باشد؟ (relevant) $(a=?)$

② حالت ما قبل چگونه تعیین شود؟ $g'=?$

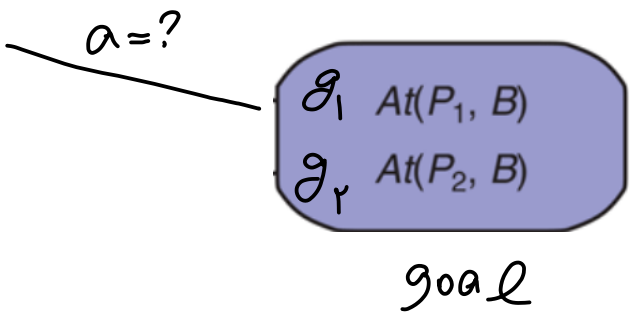
جواب چالش ① :

A relevant action is one with an effect that **unifies** with one of the goal literals, but with no effect that negates any part of the goal.

اقدام a باید موثر باشد، سازگار باشد.

موثر: حداقل یکی از لیترال‌های هدف با اثرات مثبت a قابل یکان‌سازی باشد.

سازگار: هیچکدام از لیترال‌های هدف در اثرات منفی a حضور نداشته باشد.



a_1: Fly(P_2, A, B) موثر نیست (g_1 و g_2 هیچکدام تولید نمی‌شوند)

a_2: Fly(P_1, B, A) سازگار نیست (g_1 حذف می‌شود)

a_3: Fly(P_1, A, B) } موثر و سازگار است ✓ OK

relevant:

$DELL(a) \cap g = \phi$: a و g بی‌ارتباط

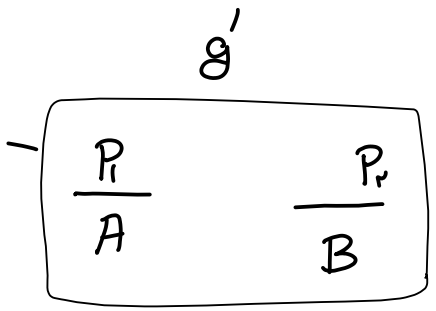
$ADD(a) \cap g \neq \phi$: a برای رسیدن به g موثر باشد

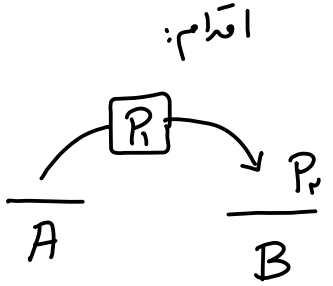
پاسخ چالش ② : یافتن حالتِ قبل از g :

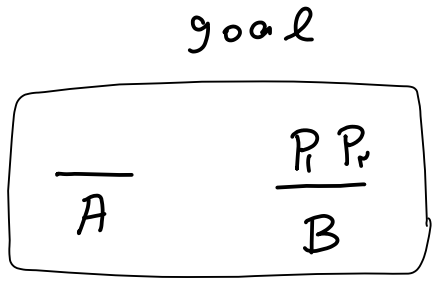
$$g' = (g - \text{ADD}(a)) \cup Precond(a) \ .$$

به تفکیک لیترال‌های مثبت و منفی :

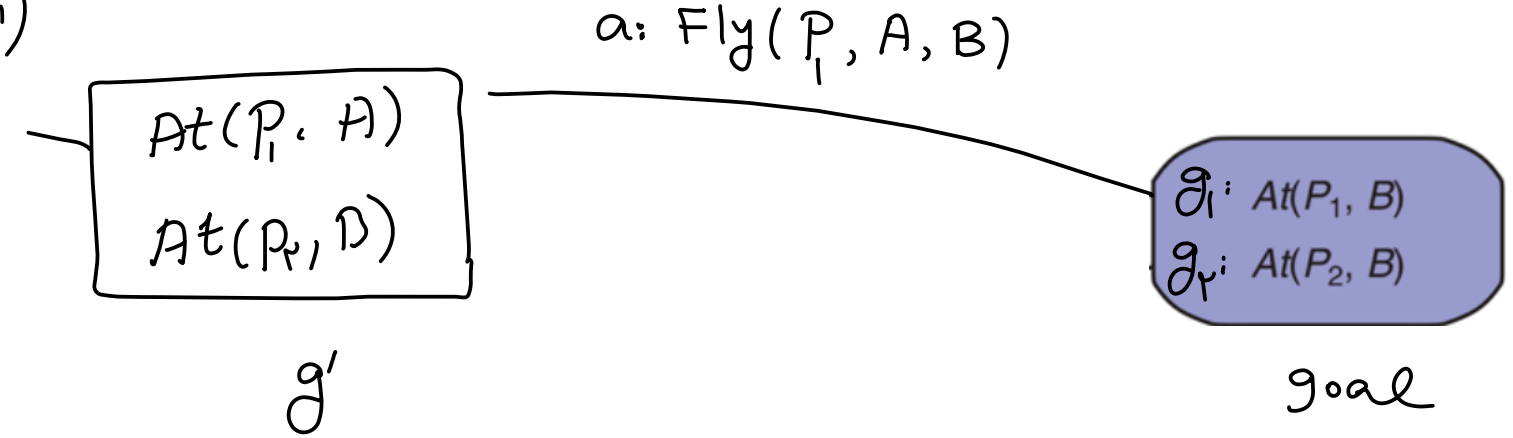
$$\begin{aligned}
\text{POS}\,(g') &= (\text{POS}\,(g) - \text{ADD}\,(a)) \cup \text{POS}\,(Precond\,(a)) \\
\text{NEG}\,(g') &= (\text{NEG}\,(g) - \text{DEL}\,(a)) \cup \text{NEG}\,(Precond\,(a))\,.
\end{aligned}$$







(Backward ادامه‌ی اول)



در برنامه ریزی کلاسیک

Forward (progression) state-space search

Backward (regression) relevant-states search

- هیچکدام بهینه نیستند
- با استفاده از هیوریستیک های مناسب (حالات تکراری) کامل هستند.
- در برنامه ریزی کلاسیک تعداد حالات متناهی است.
- هردو، جستجو در فضای حالات هستند.
- B.S در عمل سریعتر از F.S است.
- زمان اجرای آنها در حالت کلی هیچ حدی نیست (نمایی است)

Neither forward nor backward search is efficient without a good heuristic function.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

بخش دوم جلسه ۱۹ فصل ۱۰

Heuristics for planning

تفاوت هیوریستیک های عمومی مسائل برنامه‌ریزی با هیوریستیک های فصل سوم :

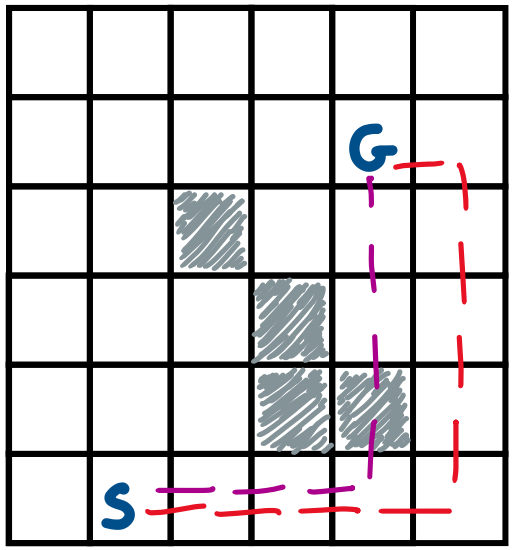
در مسائل برنامه‌ریزی لازم نیست برای هر مساله یک هیوریستیک متفاوت پیدا کنیم.

هیوریستیک ها ، عمومیت دارند.

در فصل سوم برای هر مساله یک هیوریستیک قابل قبول و ژره‌ی همان مساله داشتیم

مثلاً فاصله‌ی منهتن برای حل این مساله قابل قبول بود

$h \leq h^*$ اما در سایر مسائل (مثلاً کوتاه‌ترین مسیر نقشه) هیوریستیک قابل قبول نبود (آنجا فاصله‌ی مستقیم خوب بود).



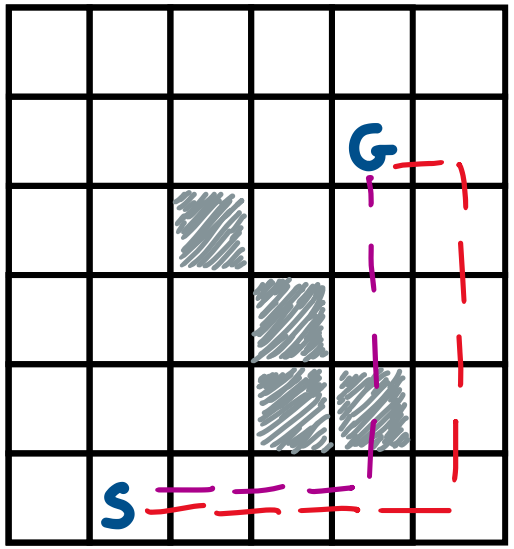
فاصله‌ی منهتن h ——

مسیر بهینه h^* ——

اما در مسائل برنامه‌ریزی، هیوریستیک‌های قابل قبول، برای همه مسائل برنامه‌ریزی قابل قبول هستند.

یادآوری: اگر مسئله‌ای را با برداشتن محدودیت‌ها یا افزایش اقدامات مجاز، ساده‌سازی کنیم

$(A \subseteq A')$



جواب مسئله‌ی ساده‌شده،

هیوریستیک قابل قبول برای

مسئله‌ی اصلی است.

ignore preconditions heuristic

① تعداد اهداف ارضاء نشده

اگر ساده سازی این باشه که: از پیش شرط ها چشم پوشی کنیم (همه اقدامات همیشه قابل اجرا باشند)

به این هیوریستیک میرسیم:

h(n) = the number of unsatisfied goals = تعداد اهداف ارضاء نشده

اگر goal = $g_1 \wedge \dots \wedge g_k$ پس دستکم k اقدام لازم است.

این هیوریستیک قابل قبول نیست زیرا:

(1) some action may achieve multiple goals

(2) some actions may undo the effects of others.

مثال:

هدف

$$p_1 \wedge p_2 \wedge p_3$$

اولیه: S_0

$$q_1 \wedge q_2$$

$$q_1 \wedge q_2 \xrightarrow{a_1} \sim q_2 \wedge p_1 \wedge p_3$$

$$p_1 \wedge p_3 \xrightarrow{a_2} p_2$$

$$p_2 \wedge q_2 \xrightarrow{a_3} p_1 \wedge \sim q_1$$

اقدامات ساده اصلی:

اثرات: پیش شرط:

ساده سازی

حذف پیش شرط ها

$$\xrightarrow{a_1} \sim q_2 \wedge p_1 \wedge p_3$$

$$\xrightarrow{a_1} p_2$$

$$\xrightarrow{a_1} p_1 \wedge \sim q_1$$

با این ساده‌سازی، تعداد اهداف ارضاءنشده در هر گره برابر ۳ تا است. پس

$h(s_0) = 3$ هیوریستیک پیشنهادی

حال ببینیم $h^*(s_0)$ در مسئله اصلی کدام است؟



در مساله اصلی با دو اقدام، اهداف برقرارمی شوند

$h^*(s_0) = 2$

پیش شرط:

اثرات:

$p_1 \wedge p_2 \xrightarrow{a_1} \sim p_2 \wedge g_1 \wedge g_2$

$g_1 \wedge g_2 \xrightarrow{a_2} g_2$

$g_2 \wedge p_2 \xrightarrow{a_3} g_1 \wedge \sim p_1$

پس $h(s)$ قابل قبول نیست.

چرا تعداد اهداف ارضاء نشده خوب نیست؟

(1) some action may achieve multiple goals

(2) some actions may undo the effects of others.

① ممکن است یک اقدام چند هدف را تولید کند

② برخی اقدامات اثرات اقدامات قبل را حذف می‌کنند.

در سیاست‌های دوم فکر می‌کنیم:

First, we relax the actions by removing all preconditions and all effects except those that are literals in the goal. Then, we count the minimum number of actions required such that the union of those actions' effects satisfies the goal. This is an instance of the set-cover problem

set-cover (۲)
مجموعه‌ی پوشش

با چشم‌پوشی از پیش شرط‌ها و اثرات به جز لیترال‌های هدف g_i، پیدا کردن کمترین تعداد اقدامات که همه‌ی اهداف را تولید می‌کنند.

$N = \min |\text{set cover}|$

قابل قبول است.

یافتن set cover کمینه NP-hard است.

روشی حریصانه نیز دارد اما پاسخ آن کمینه نیست

یک $O(\log N \cdot N)$ است و قابل قبول نیست.

مثال

تعرف

$$g_1 \wedge g_2 \wedge g_3$$

حالت: n

$$\ell_1 \wedge \ell_2$$

$$a_1 \longrightarrow g_1$$

$$a_2 \longrightarrow g_2 \wedge g_4$$

$$g_1 \xrightarrow{a_3} g_2$$

$$g_2 \xrightarrow{a_4} g_3$$

ساده سازی

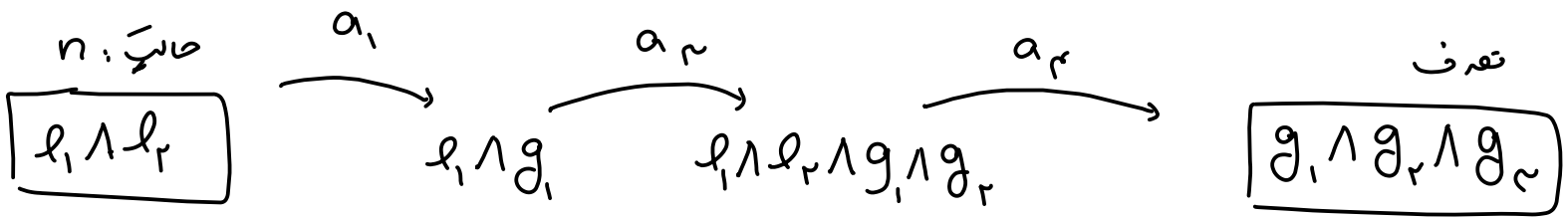
$$\ell_1 \wedge \ell_2 \xrightarrow{a_1} g_1 \wedge \sim \ell_2$$

$$\ell_1 \wedge \ell_2 \xrightarrow{a_2} g_2 \wedge g_4 \wedge \sim \ell_1$$

$$g_1 \xrightarrow{a_3} \ell_2 \wedge g_2$$

$$\ell_2 \wedge g_2 \xrightarrow{a_4} g_3 \wedge \ell_1$$

جواب بهینه مسئله اصلی: $h^*(n) = ۳$



$l_1 \wedge l_2 \xrightarrow{a_1} g_1 \wedge \sim l_2$

$l_1 \wedge l_2 \xrightarrow{a_2} g_2 \wedge g_3 \wedge \sim l_1$

$g_1 \xrightarrow{a_3} l_2 \wedge g_2$

$l_2 \wedge g_2 \xrightarrow{a_4} g_3 \wedge l_1$

اما set cover کمینه‌ش ۲ اقلام است:

$h(n) = 2$

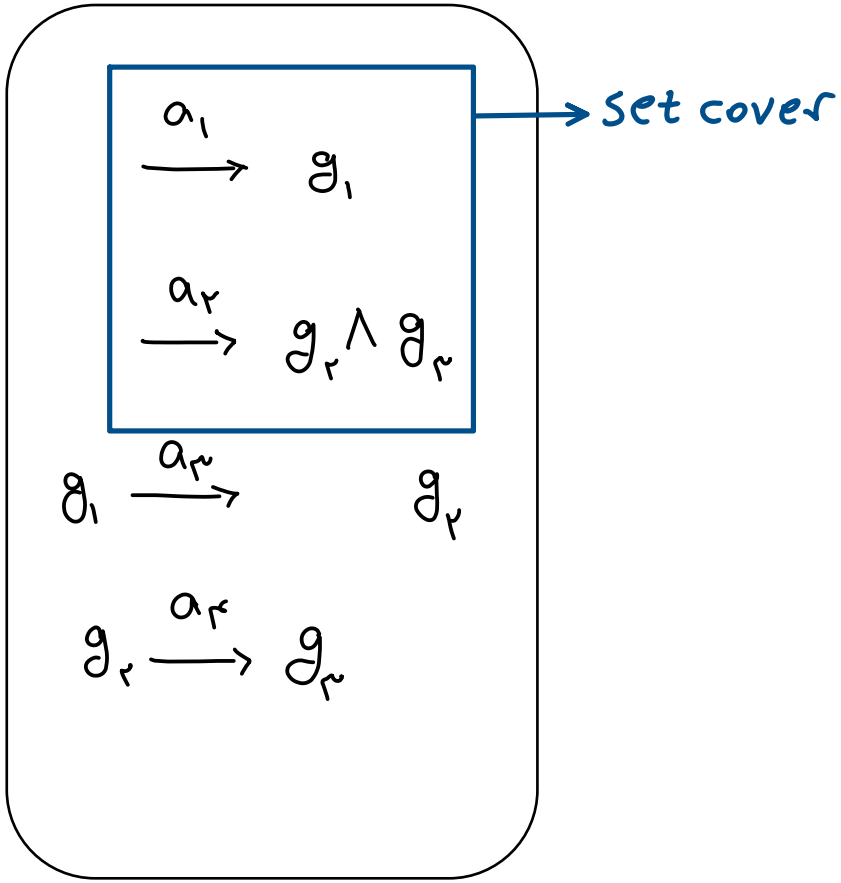
حالت هرف:

$g_1 \wedge g_2 \wedge g_3$

حالت n:

$l_1 \wedge l_2$

ساده‌سازی



پس در مثال قبل دیدیم که $h(n) \leq h^*(n)$.

در حالت کلی هم set-cover کمتر قابل قبول است

اما در حالت کلی یافتن آن NP-hard است .

اگر set-cover کمتر N باشد راه حل حریصانه

هم داریم که پاسخ آن کمتر نیست بلکم یک set-cover

با اندازه $O(N \log(N))$ است.

سادہ سازی سوم :

Assume for a moment that :
all goals and preconditions contain only positive literals

ignore delete lists ③

روشی لیست حذف ها

removing the delete lists from all actions

اگر اهداف و پیش شرط ها همگی مثبت باشند ، با چشم پوشی از اثرات منفی اقدامات مساله را ساده کنیم .

این هیوریستیک قابل قبول است

اما محاسبه آن در حالت کلی NP-hard است .

در زمان چند جمله ای با روشی تکه نوردی قابل تخمین است .

subgoal independence

تجزیه ی اهداف:

اگر اهداف به شکل مستقل قابل تفکیک باشند آنگاه:

G_i and G_j are **independent**

If the effects of P_i leave all the preconditions and goals of P_j unchanged,

$\text{COST}(P_i) + \text{COST}(P_j)$ is admissible.

مجموع هزینه ها یک هیوریستیک قابل قبول است.

PLANNING GRAPHS

تعداد ثابت‌ها، روابط، اعمال، متناهی باشد. (تا در تابع نداریم)

گزاره‌ها هم زمینه‌ای نوشته شوند.

اقدام تداوم :



persistence action

$Init(Have(Cake))$
$Goal(Have(Cake) \land Eaten(Cake))$
$Action(Eat(Cake)$
PRECOND: $Have(Cake)$
EFFECT: $\neg Have(Cake) \land Eaten(Cake))$
$Action(Bake(Cake)$
PRECOND: $\neg Have(Cake)$
EFFECT: $Have(Cake))$

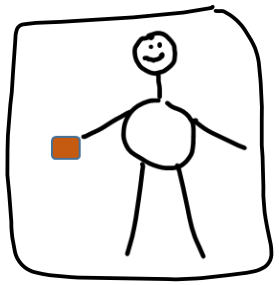
مسأله : داشتن و خوردن کیک

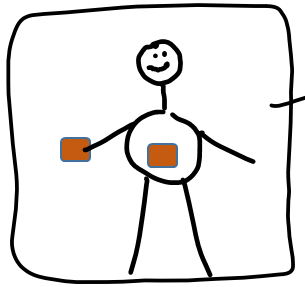
ابتدا : کیک داریم

هدف: کیک خورده باشیم و کیک داشته باشیم

عمل: خوردن کیک

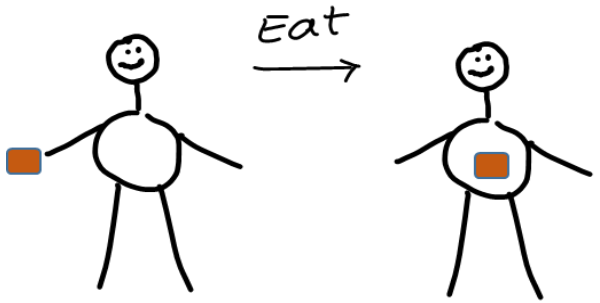
عمل: پختن کیک

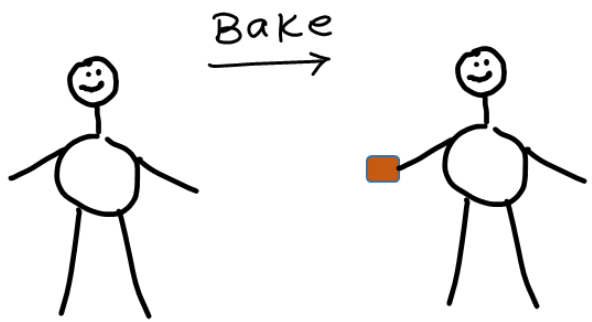




$$Init(Have(Cake))$$
$$Goal(Have(Cake) \land Eaten(Cake))$$
$$Action(Eat(Cake)$$
$$\quad \text{PRECOND: } Have(Cake)$$
$$\quad \text{EFFECT: } \neg\, Have(Cake) \land Eaten(Cake))$$
$$Action(Bake(Cake)$$
$$\quad \text{PRECOND: } \neg\, Have(Cake)$$
$$\quad \text{EFFECT: } Have(Cake))$$

افعال اصلی:





افعال کمکی:





رسم گراف برنامه‌ریزی:

S_0 : لیترال‌های مثبت و منفی که در شروع از آنها اطمینان داریم.

S_0

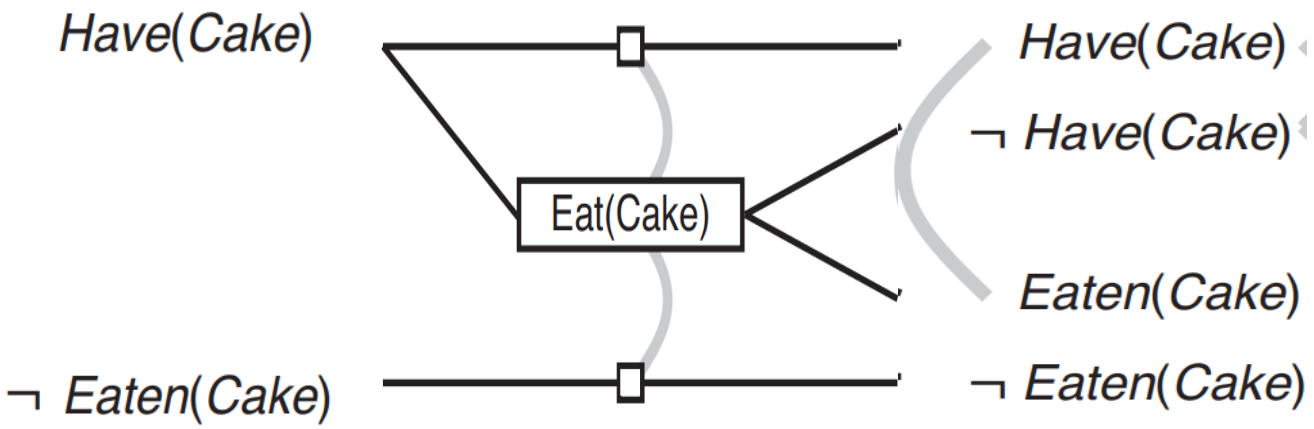
Have(*Cake*)

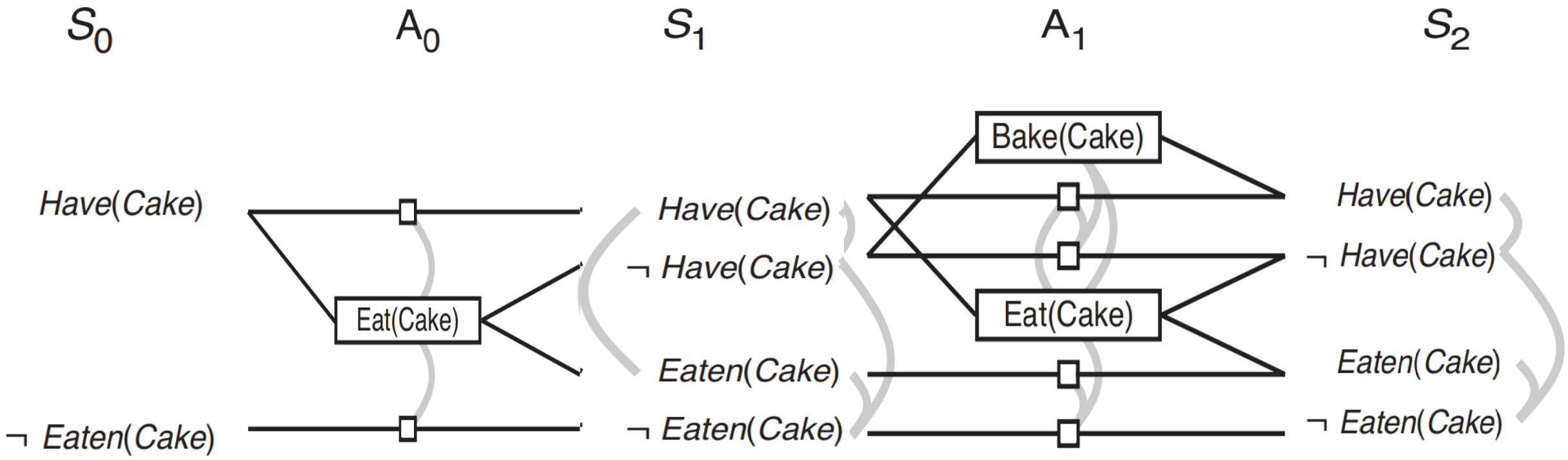
¬ *Eaten*(*Cake*)

A_0 : اقداماتی که پیش شرط‌های آن در S_0 موجود است. به‌جز اقدامات تداوم

S_1 : همه نتایج واژات اقدامات A_0

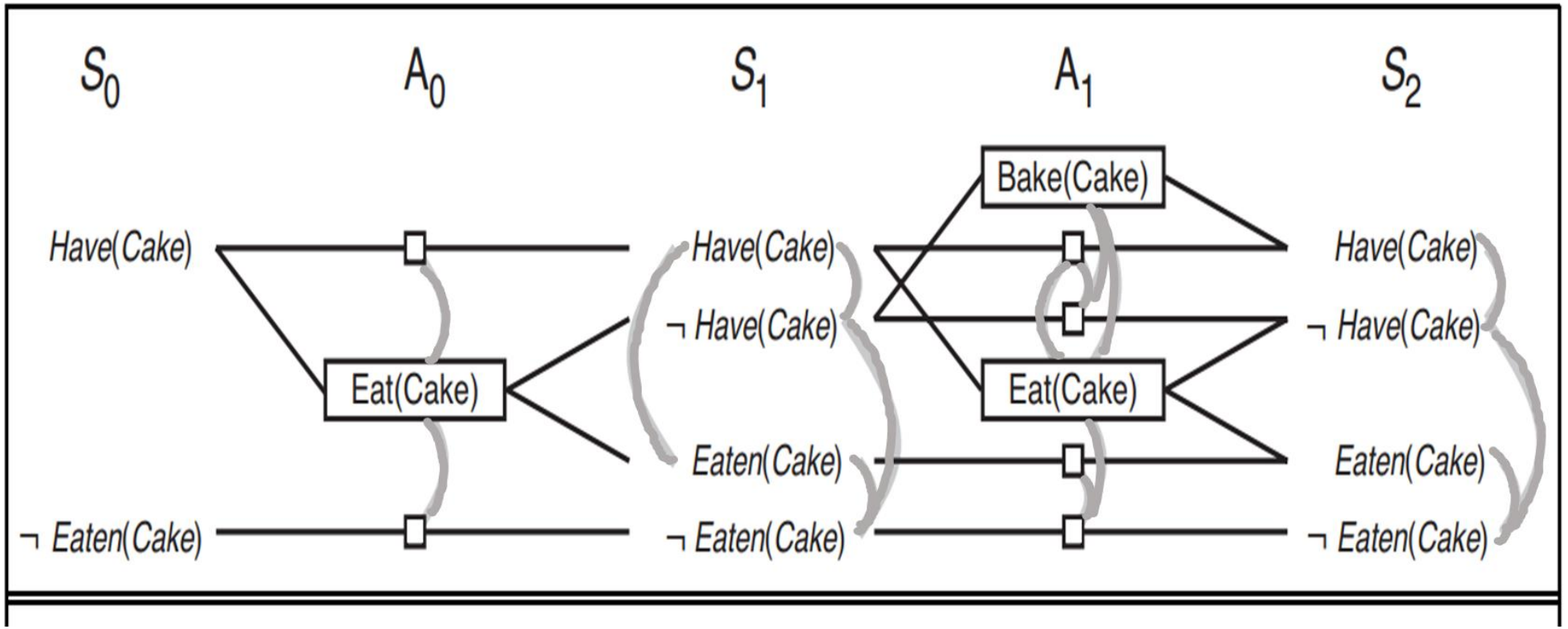
S_0 A_0 S_1





به دلیل متناهی بودن حالات و لیترال‌ها و اقدامات، گراف حتماً متوقف می‌شود.

(A_1 همان A_2 می‌شود
S_1 همان S_2 می‌شود)



به دنبال وجود اتفاقات متوالی داریم:

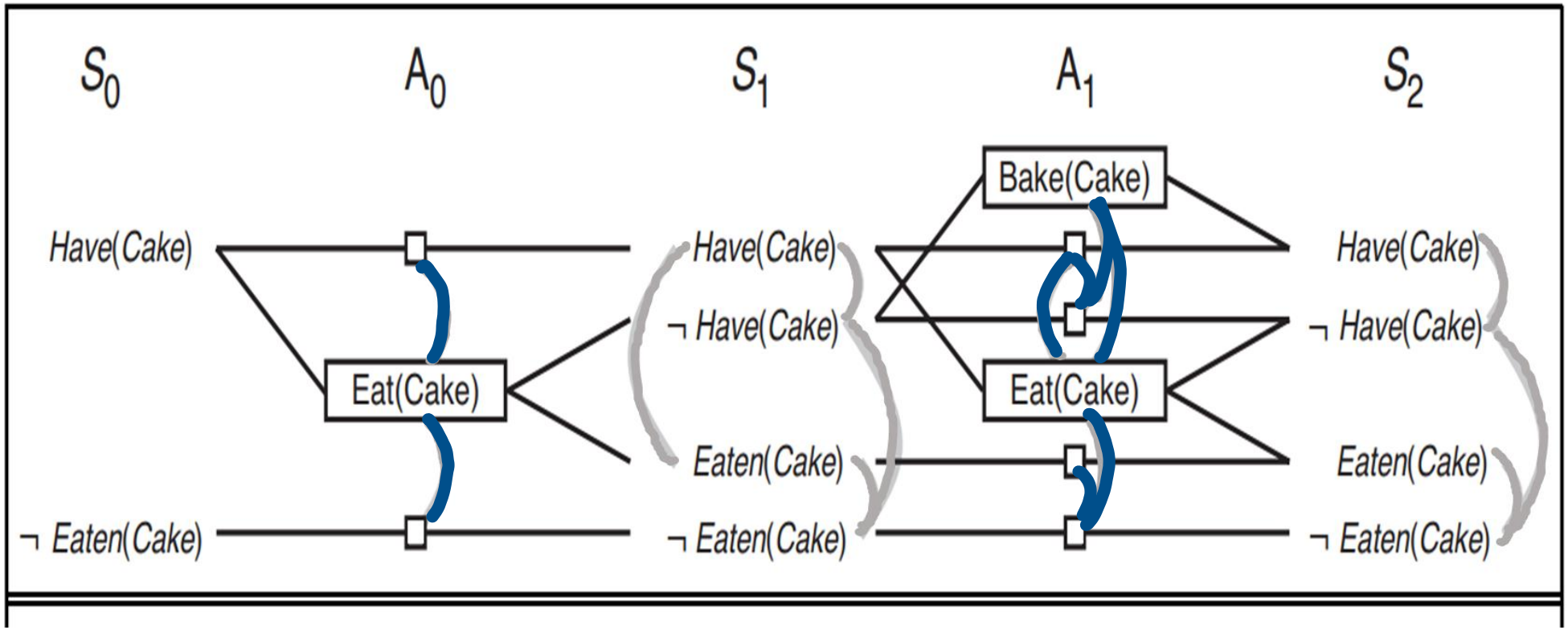
$$\begin{cases} S_0 \subseteq S_1 \subseteq S_2 \subseteq \dots \subseteq S_n \\ A_0 \subseteq A_1 \subseteq A_2 \subseteq \dots \subseteq A_n \end{cases}$$

S_0 : لیترال‌های مثبت و منفی در حالت اولیه. (برخلاف PDDL) (در نوشتن PDDL حالت اولیه فقط لیترال مثبت دارد)

A_0 : تمام اقداماتی که پیش‌شرط آن‌ها در S_0 قرار دارد و اجرایی هستند (به همراه تداوم‌ها)

S_1 : حالت با ور تمام لیترال‌های مثبت و منفی که ممکن است با اقدامات A_0 به دست آیند.

دس از چند مرحله خواهیم داشت : $S_i = S_{i+1}$.

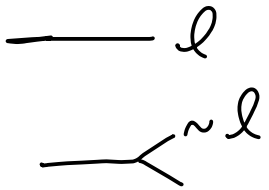


در هر مجموعه‌ی A_i بین اقدامات رابطه‌ی طرد متقابل دارند

mutex links

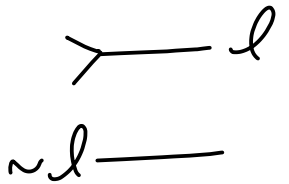
طرد متقابل بین اقدامات

① **Inconsistent effects:** ناسازگاری اثرات



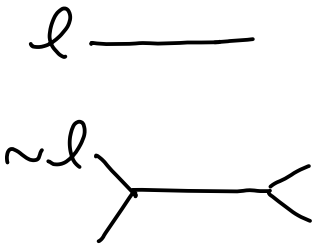
اثرات نقیض هم

② **Interference:** تداخل

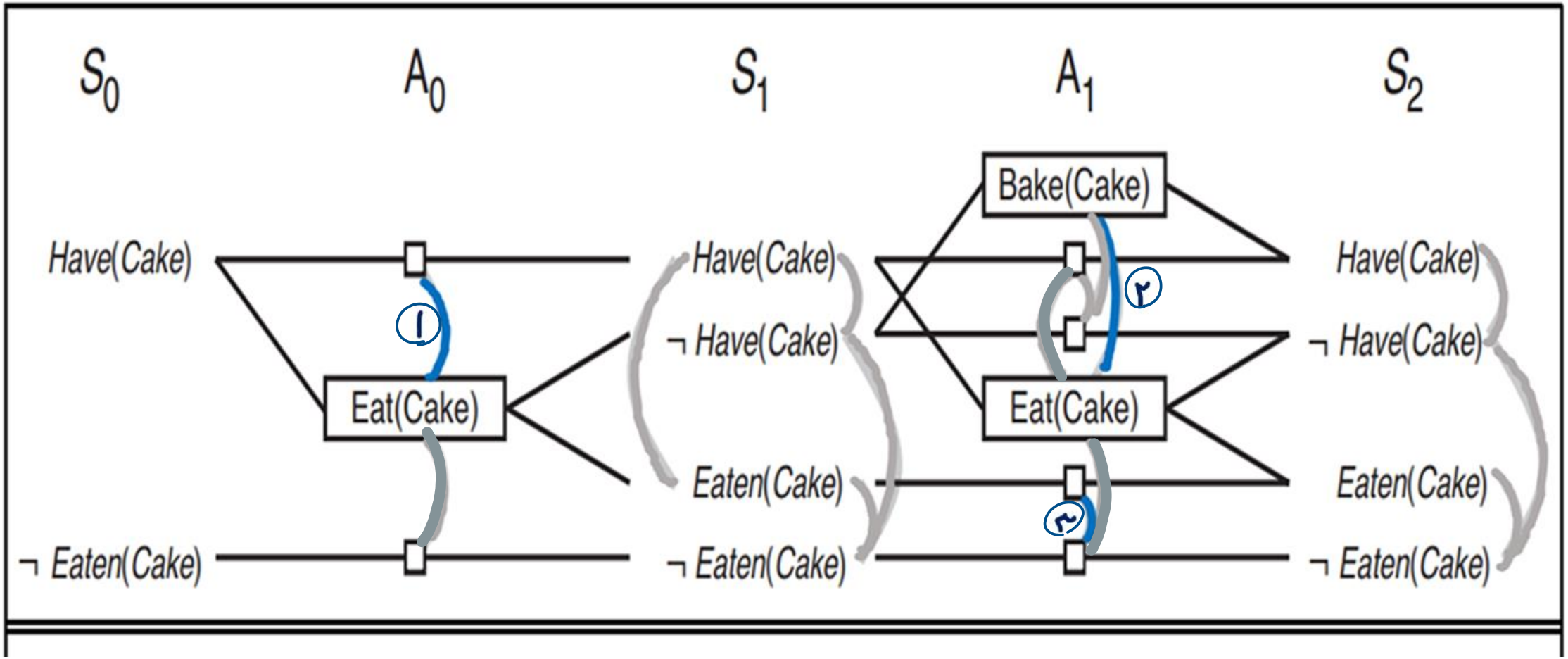


اثر من و پیش نیاز دیگری

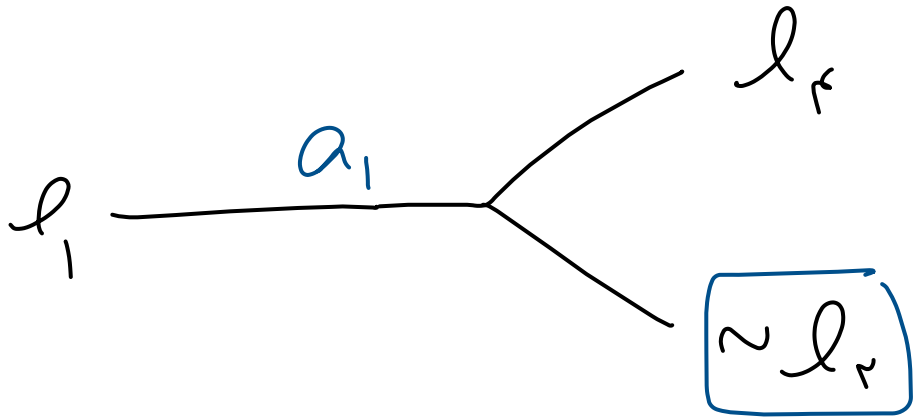
③ **Competing needs:** رقابت در نیازها

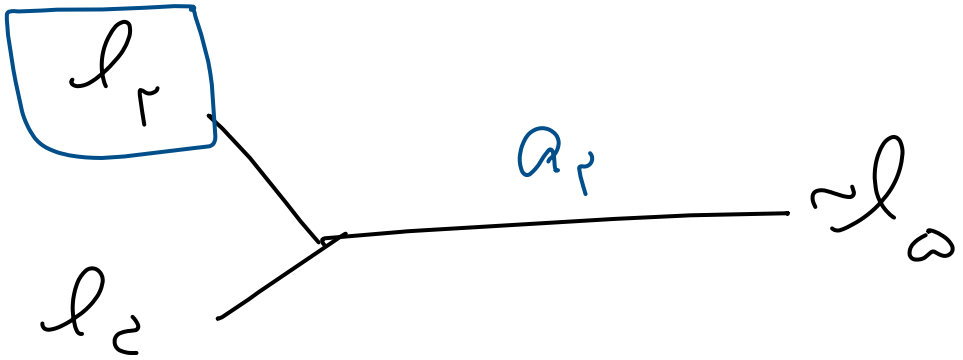


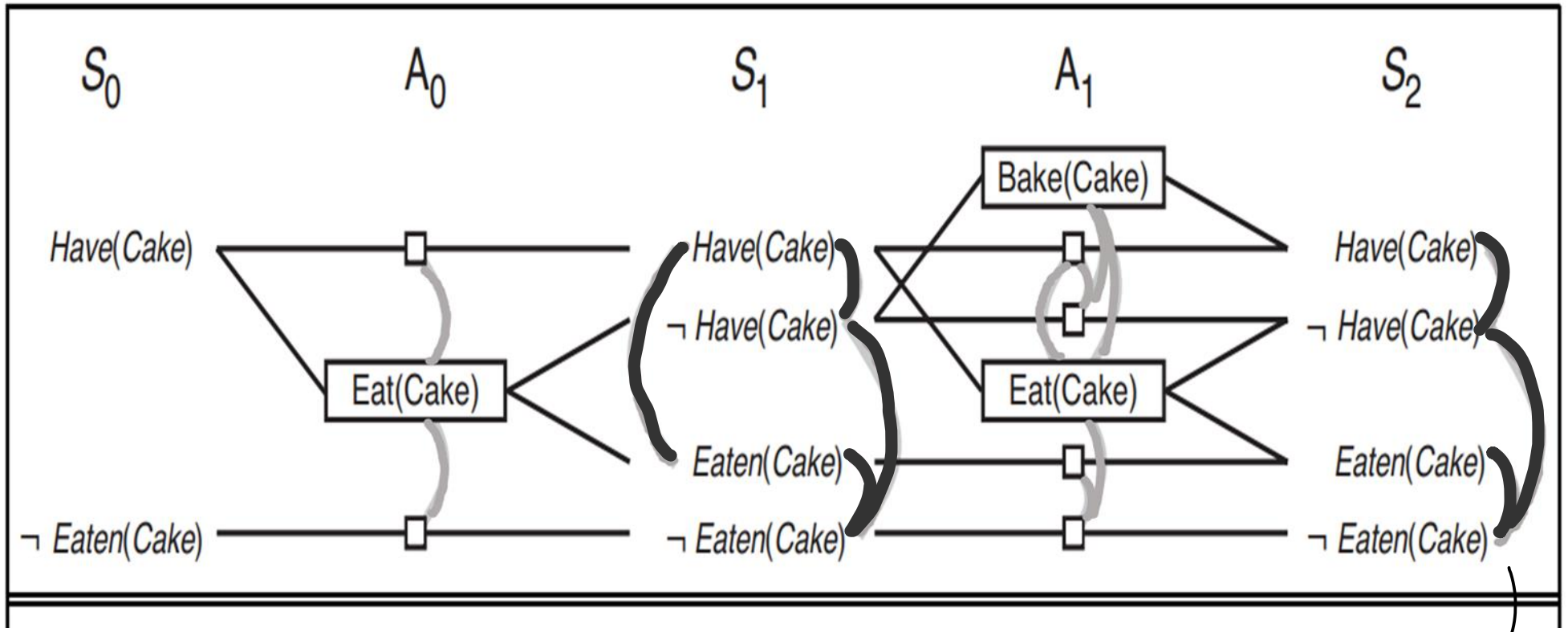
پیش نیازی نقیض هم


S_0
A_0
S_1
A_1
S_2
Bake(Cake)
Have(Cake)
¬ Have(Cake)
Eat(Cake)
Eaten(Cake)
¬ Eaten(Cake)
Have(Cake)
¬ Have(Cake)
Eaten(Cake)
¬ Eaten(Cake)
Have(Cake)
¬ Have(Cake)
Eaten(Cake)
¬ Eaten(Cake)
١
٢
٣

تداخل a_1 و a_2:







درهر S_i لیتال نیز ممکن است طرد متقابل داشته باشند :

طرد متقابل بین لیترال های هم سطح

mutex relation between two **literals**

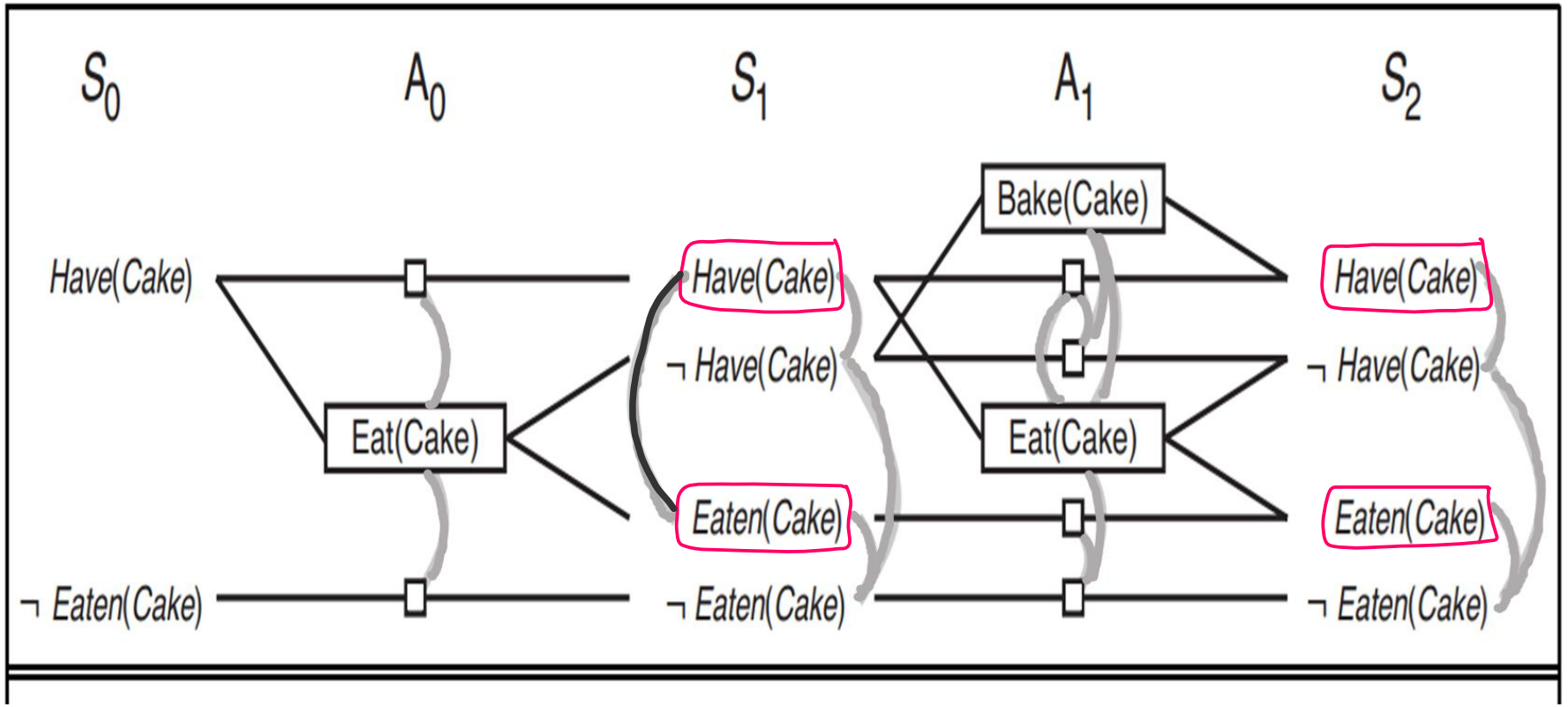
negation تقیض بودن

Have(Cake)

~Have(Cake)

inconsistent support پشتیبانی ناسازگار

دو حقیقت از اقدامات که A_1 و A_2 را تولید می‌کنند در طرد متقابل باشند. ممکن است A_1 و A_2 در سطح S_i ناسازگار باشند اما در سطح S_{i+1} سازگار شوند.



مثال:

Have(Cake) و Eaten(Cake) در S_1 ناسازگارند زیرا اقداماتی که آنها را تولید می‌کنند رابطه طرد دارند. (تضمینی ناسازگار)

اما Have(Cake) Eaten(Cake) در S_2 سازگار هستند زیرا اقدامات Bake(cake) و تداوم Eaten(cake) می‌توانند آنها را تولید کنند و این حفظ اقدام، رابطه طرد ندارند.

* اندازه‌ی گراف، از مرتبه‌ی خیلی جمله‌ای است:

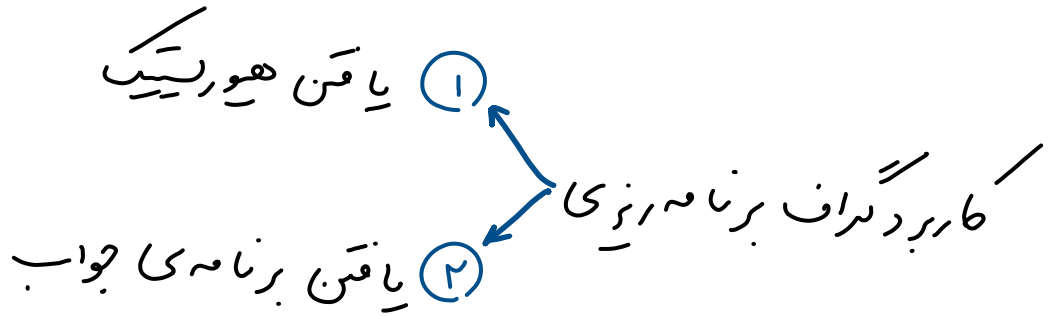
l = تعداد لترال‌ها

a = تعداد اقامات

$|S_i| \leq l$: حداکثر l^2 تعداد اتصالات از S_i به S_i

$|A_i| \leq l+a$: حداکثر $(l+a)^2$ تعداد اتصالات از A_i به A_i

اگر کلاً n سطح داشته باشیم، اندازه‌ی گراف برنامه‌ریزی $= O(n(l+a)^2)$


کاربرد گرا
برنامه ریزی
۱
یافتن هیوریستیک
۲
یافتن برنامه ی جواب

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۲۰ فصل ۱۰

کاربرد گراف برنامه‌ریزی

① یافتن هیوریستیک ← (چه توابع هیوریستیکی؟)

② یافتن برنامه‌ی جواب

Planning graphs for heuristic estimation

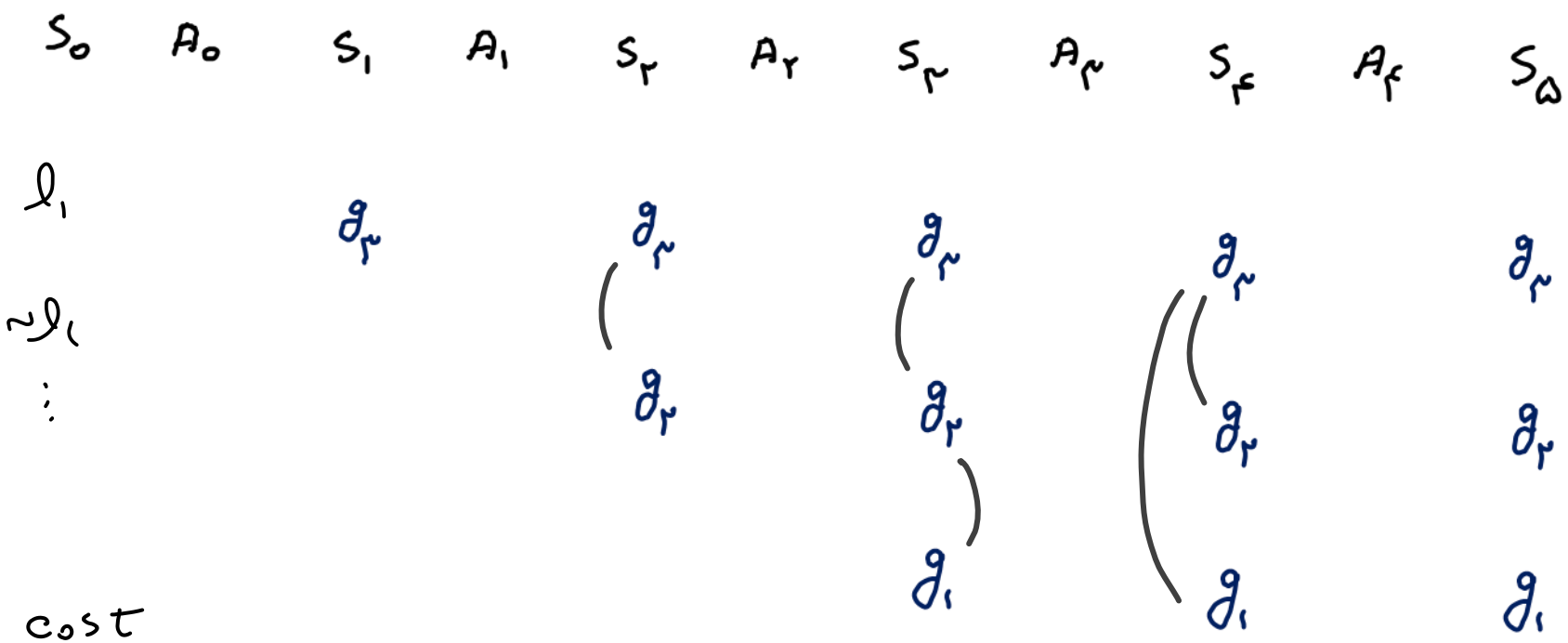
هدف، ترکیب عطفی از لیترال‌های مثبت g_i است. هر کدام را یک لیترال هدف می‌نامیم:

$$g = g_1 \wedge g_2 \wedge \dots \wedge g_k$$

شماره اولین سطحی که g_i در آن تولید می‌شود. هزینه‌ی سطحی g_i نام دارد.

$$\text{levelcost}(g_i)$$

goal: $g_1 \wedge g_2 \wedge g_3$



level	cost
g_1	۳
g_2	۲
g_3	۱

هیوریستیک‌های پیشنهادی برای planning

- max-level — هیوریستیک سطح بیشینه (بزرگترین level_cost را در نظر بگیریم)
- level sum — هیوریستیک مجموع سطوح (مجموع همه‌ی level_cost ها را در نظر بگیریم)
- set-level — هیوریستیک سطح مجموعه (شماره‌ی اولین سطحی که همه‌ی اهداف در آن ظاهر شده باشند و بین هیچ جفت از آن‌ها رابطه‌ی متقابل وجود نداشته باشد)

max-level هیوریستیک سطح بیشینه قابل قبول اما دارای خطای زیاد

level sum هیوریستیک مجموع سطوح خیر قابل قبول

set-level هیوریستیک سطح مجموعه قابل قبول است، از سطح بیشینه دقیق کراست.

* هیوریستیک‌های فوق و روابط طرد متقابل بین جفت‌های (g_i, g_j)، ناسازگاری‌های ۳گانه حیّ‌گانه را تشخیص نمی‌دهند.

مثال در سه مکعب‌ها

$$goal\ (\ on(A,B) \wedge on(B,C) \wedge on(C,A)\ )$$

$$g_1 \qquad g_2 \qquad g_3$$

هرکدام از اهداف g_i در سطحی از گراف تولید خواهند شد.

هر جفت از آن‌ها، پس از چند سطح، باهم سازگار می‌شوند

اما عدم وجود رابطه‌ی طرد بین هر جفت از اهداف،

به معنای تضمین وجود یک راه‌حل (plan) نیست.

هدف $g_1 g_2 g_3$ غیرقابل دستیابی است.

کاربرد گراف برنامه‌ریزی

۱) یافتن هیوریستیک

۲) یافتن برنامه‌ی جواب ← (حال این را مطالعه کنیم)

The GRAPHPLAN algorithm

در برنامه ریزی کلاسیک ، با استفاده از گراف برنامه ریزی، راه حل را در صورت وجود می یابد.

عدم وجود راه حل را نیز می تواند تشخیص دهد.

این الگوریتم، بهینه نیست اما کامل است .

```
function GRAPHPLAN(problem) returns solution or failure

  graph ← INITIAL-PLANNING-GRAPH(problem)
  goals ← CONJUNCTS(problem.GOAL)
  nogoods ← an empty hash table
  for tl = 0 to ∞ do
      if goals all non-mutex in S_t of graph then
          solution ← EXTRACT-SOLUTION(graph, goals, NUMLEVELS(graph), nogoods)
          if solution ≠ failure then return solution
      if graph and nogoods have both leveled off then return failure
      graph ← EXPAND-GRAPH(graph, problem)
```

EXPAND-GRAPH گسترش گراف (تولید سطح بعدی)

EXTRACT-SOLUTION استخراج راه حل

$[a_1, a_2, \ldots, a_k]$

نحوه‌ی اجرای graphplan :

تا هنگامی که همه‌ی اهداف $g_1, g_2, \ldots, g_m$ بدون روابط طرد در S_t تولید نشده باشند نیازی به صدا زدن EXTRACT-SOLUTION نداریم چون قطعاً راه حلی تا آن سطح وجود ندارد. EXPAND-GRAPH ادامه می‌یابد.

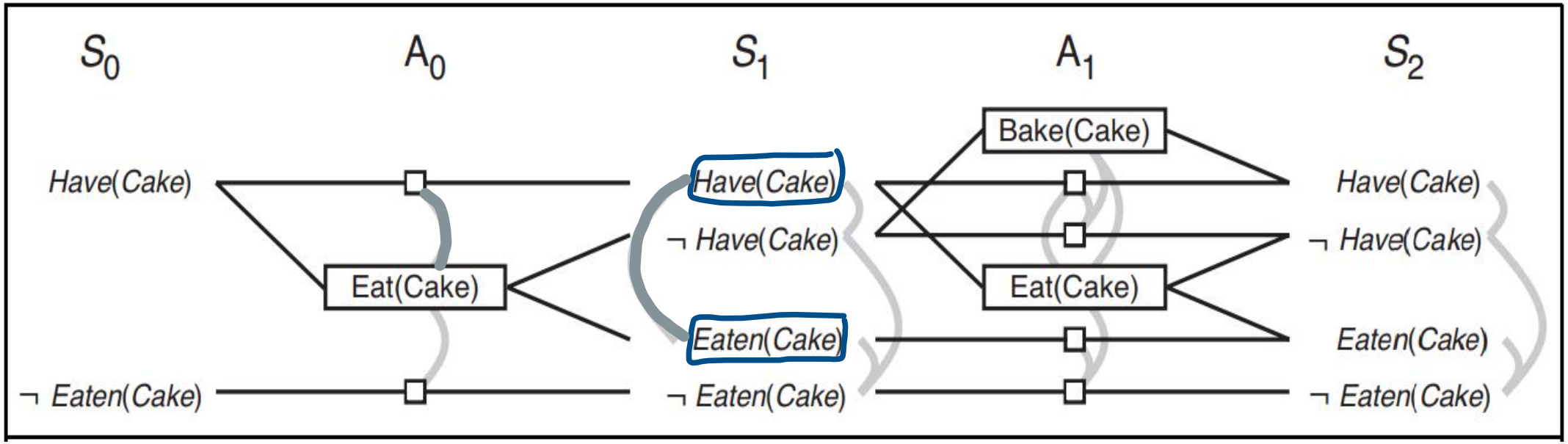
وقتی همه اهداف بدون روابط طرد متقابل در S_t ظاهر شوند، EXTRACT-SOLUTION را صدا می‌زنیم. اگر برنامه‌ای یافت کار تمام است. در غیر این صورت EXPAND-GRAPH را صدا می‌زنیم تا سطح بعدی را تولید کند.
البته وقتی راه حل یافت نشده باشد یعنی برخی اهداف قابل تولید به شکل سازگار نبوده‌اند (no-good ها)
اگر در دو تکرار متوالی هم سطوح S_i و S_{i+1} و هم (no-good) ها ثابت بمانند الگوریتم با شکست متوقف می‌شود.

EXTRACT-SOLUTION : می‌تواند به صورت Backward پیاده‌سازی شود. (یا CSP)

توضیح اجرای graphplan در مسئله‌ی کیک:

در S_0 و P_2 تولید نشده. در S_1، هردو هدف تولید شده‌اند

در S_1، g_1 و g_2 تولید شده‌اند اما رابطه‌ی طرد دارند (پشتیبانی ناسازگار) پس بازهم Extract را اجرا نمی‌زنند.

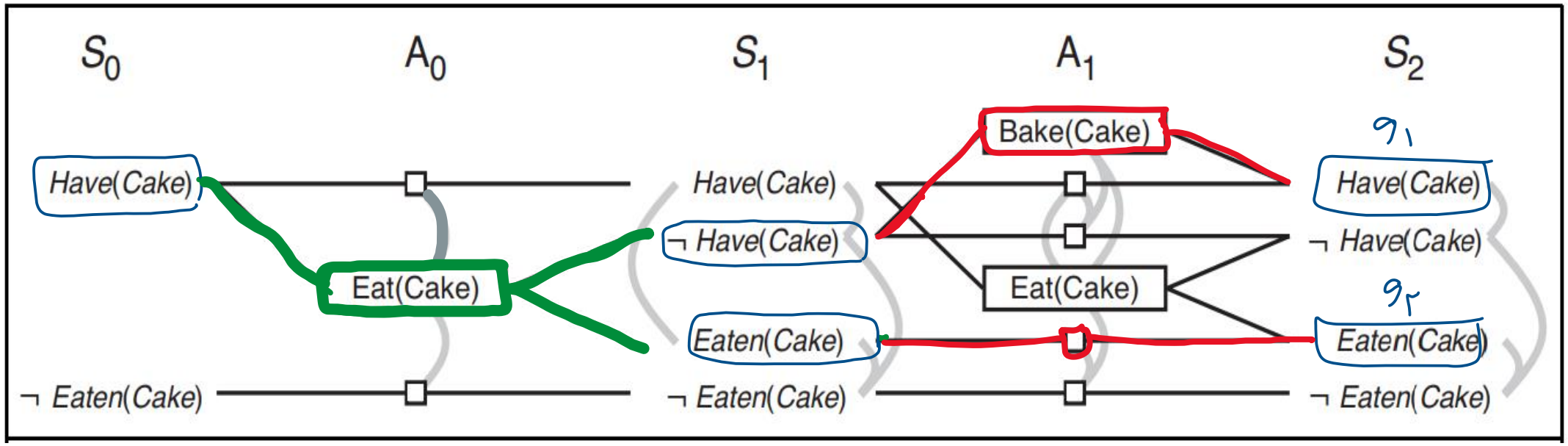


$$Goal(Have(Cake) \wedge Eaten(Cake))$$

g_1 g_2

در S_2 برای اولین بار g_1 و g_2 بدون طرد متقابل تولید می‌شوند پس احتمال وجود راه‌حل داده می‌شود.

Extract را صدا زده و به صورت Backward اجرا می‌کنیم:



$$Goal(Have(Cake) \wedge Eaten(Cake))$$

g_1 g_2

```
function GRAPHPLAN(problem) returns solution or failure

  graph ← INITIAL-PLANNING-GRAPH(problem)
  goals ← CONJUNCTS(problem.GOAL)
  nogoods ← an empty hash table
  for tl = 0 to ∞ do
      if goals all non-mutex in S_t of graph then
          solution ← EXTRACT-SOLUTION(graph, goals, NUMLEVELS(graph), nogoods)
          if solution ≠ failure then return solution
      if graph and nogoods have both leveled off then return failure
      graph ← EXPAND-GRAPH(graph, problem)
```

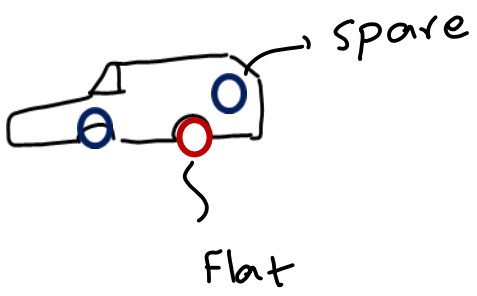
اگر دو لایه‌ی گراف یعنی S_n و S_{n+1} کاملاً یکی‌ن همچنین no-good ها کاملاً یکی‌ن شدند تشخیص میدهد که راه‌حل وجود ندارد. (توقف با شکست)

مثال: تعویض تایر و قراردادن تایر یدکی به جای آن.

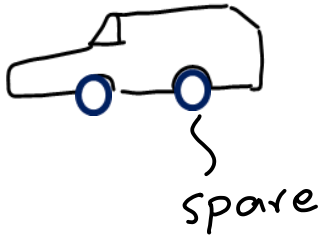
$Init(Tire(Flat) \land Tire(Spare) \land At(Flat, Axle) \land At(Spare, Trunk))$
$Goal(At(Spare, Axle))$
$Action(Remove(obj, loc),$
PRECOND: $At(obj, loc)$
EFFECT: $\neg At(obj, loc) \land At(obj, Ground))$
$Action(PutOn(t, Axle),$
PRECOND: $Tire(t) \land At(t, Ground) \land \neg At(Flat, Axle)$
EFFECT: $\neg At(t, Ground) \land At(t, Axle))$
$Action(LeaveOvernight,$
PRECOND:
EFFECT: $\neg At(Spare, Ground) \land \neg At(Spare, Axle) \land \neg At(Spare, Trunk)$
$\land \neg At(Flat, Ground) \land \neg At(Flat, Axle) \land \neg At(Flat, Trunk))$

Figure 10.2 The simple spare tire problem.

init:

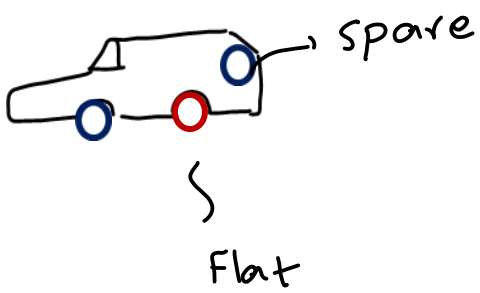


goal:



dont care

Flat



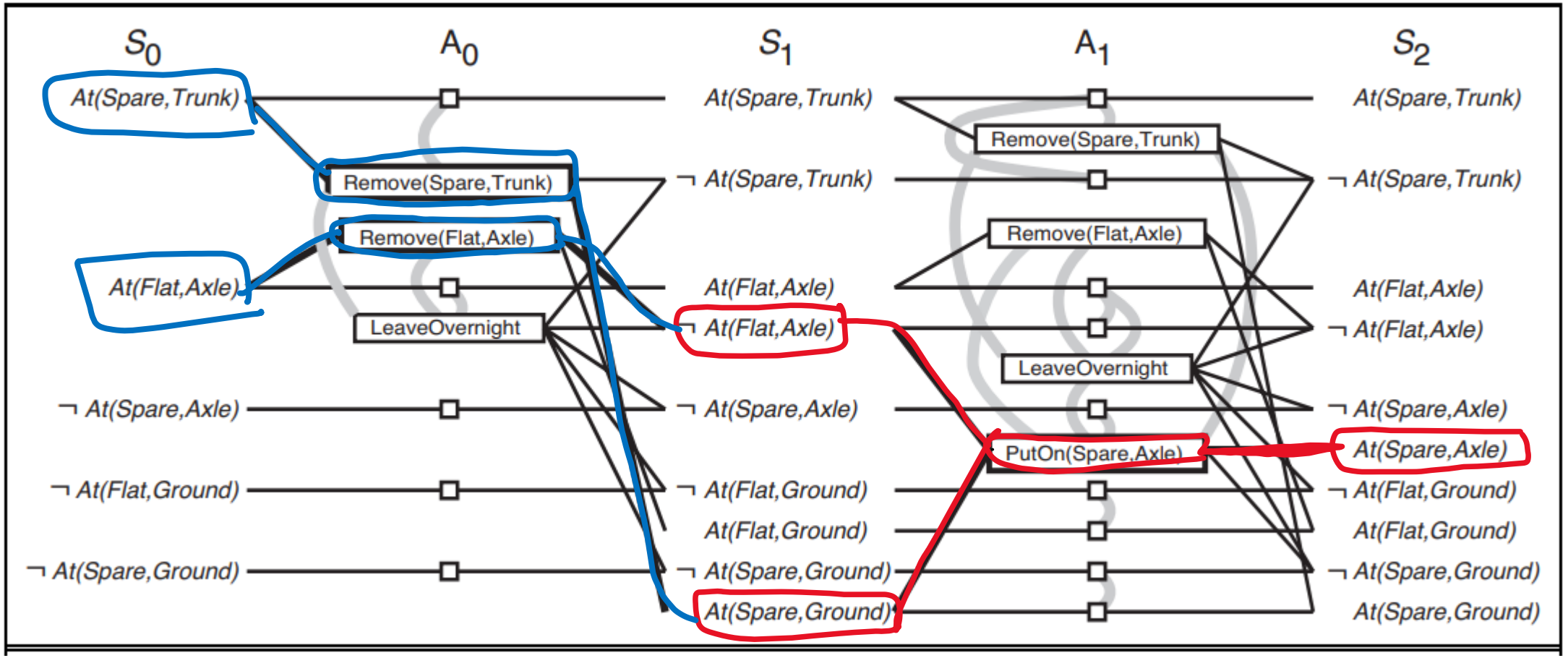
اقدامات:

$$Action(Remove(obj, loc))$$

$$Action(PutOn(t, Axle))$$

$$Action(LeaveOvernight$$

اجرای graphplan :



g_1

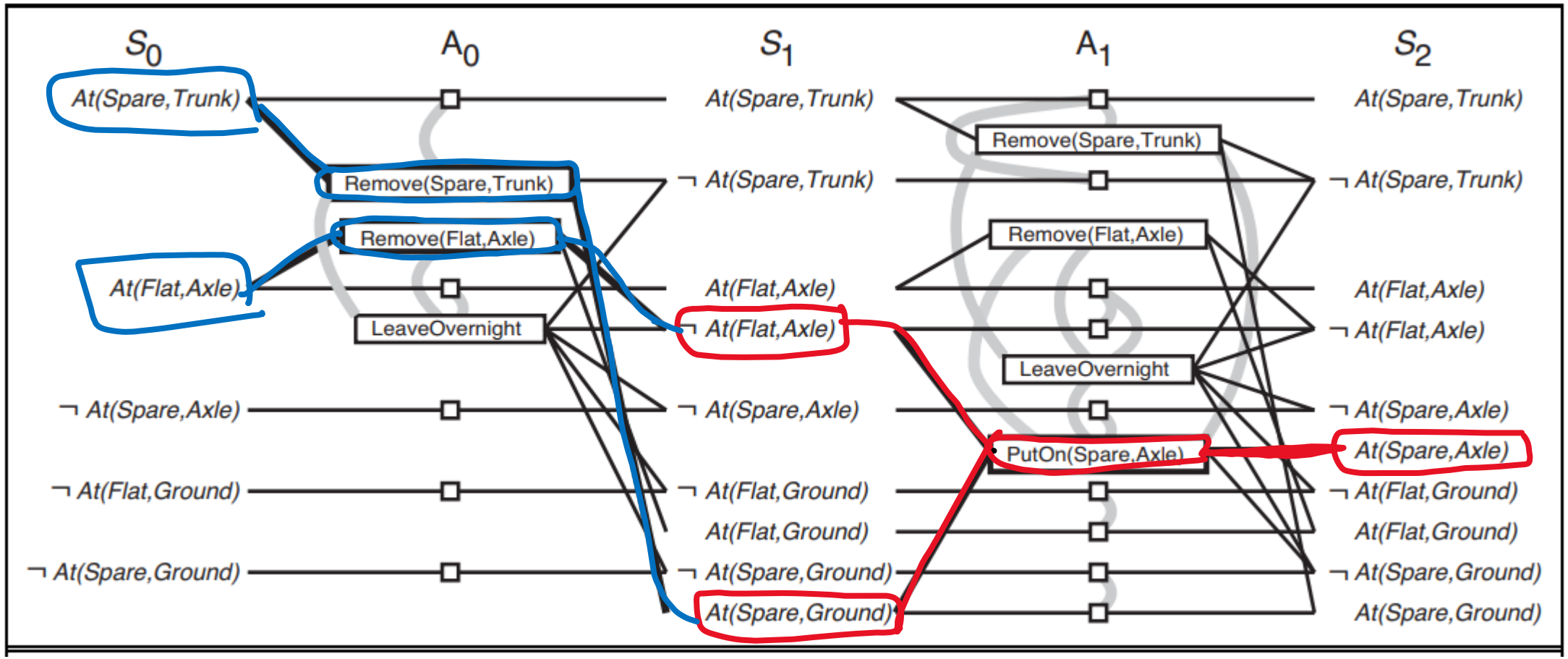
$$Goal(At(Spare, Axle))$$

g_1

راه حل یافت شده:

۲ اقدام از لایه‌ی A_0 : (خارج کردن یدکی از صندوق) و (خارج کردن تایر خراب از روی محور چرخ)

1 اقدام از لایه‌ی A_1 : (نصب یدکی روی محور چرخ)



g_1

Termination of GRAPHPLAN

شرط توقف با پیروزی : یافتن برنامه‌ای که تمام اهداف را تولید کند. $[a_1, a_2, \ldots, a_k]$

شرط توقف و اعلام شکست :

برای سطح S_t ، برنامه‌ای یافت نشده باشد و پس از تولید S_{t+1} مشاهده شود که

این دو سطح از هر لحاظ (حتی از نظر روابط طرد متقابل) یکی هستند و همچنین از نظر

no-good ها نیز یکی شوند. [مجموعه‌ی اهداف به دست نیامده، تغییر نکنند].

EXPAND-GRAPH	$O(n(l+a)^2)$	مرتبه زمانی ساخت گراف حیله جمله‌ای است.
EXTRACT-SOLUTION	$O((l+a)^n)$	مرتبه زمانی جستجوی جواب نمایی است.

planning graphs are Monotonically

لایه به لایه که جلو می‌رویم این شرایط را خواهیم دید:

Literals increase monotonically: $S_0 \subseteq S_1 \subseteq S_2 \subseteq \dots$

ویژگی‌ی افزایشی

Actions increase monotonically: $A_0 \subseteq A_1 \subseteq A_2 \subseteq \dots$

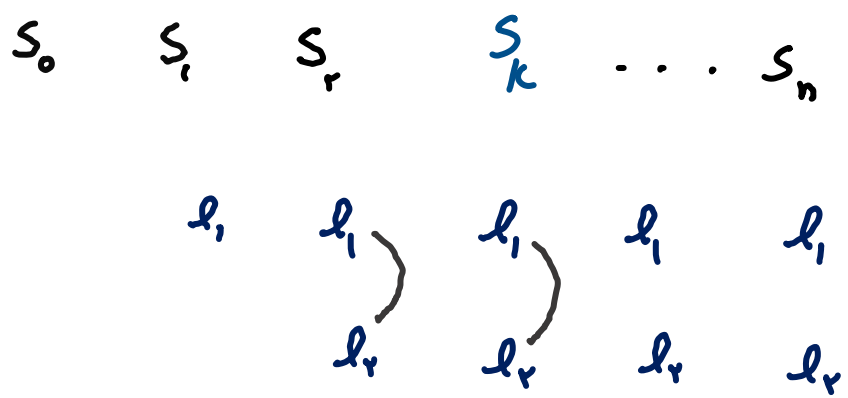
Mutexes decrease monotonically:

ویژگی‌ی کاهشی:

روابط طرد متقابل به مرور ممکن است از بین برود
اما افزایش نخواهد یافت.

No-goods decrease monotonically:

مجموعه اهداف به دست نیامده (no-good ها)
ممکن است کاهش یابد یا ثابت بماند اما افزایش نخواهد
یافت.



اگر h_1 و h_2 در S_k دارای طرد متقابل باشند

در سطوح قبل نیز نمی‌توانند به شکل سازگار

تولید شوند (بادنبال‌های از حداکثر k اقدام نزدیک

h_1 و h_2 را تولید کرد)

اگر h_1 و h_2 در S_k موجود و سازگار باشند

در سطوح بعدی هم سازگار هستند.

کامل است.

بهینه نیست.

غیرقابل حل بودن را تشخیص می دهد و متوقف می شود.

graph plan

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

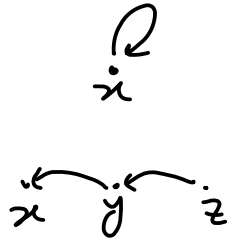
ابوالفضل گیلک
گروه بابان

یادآوری هایی از درس گسسته

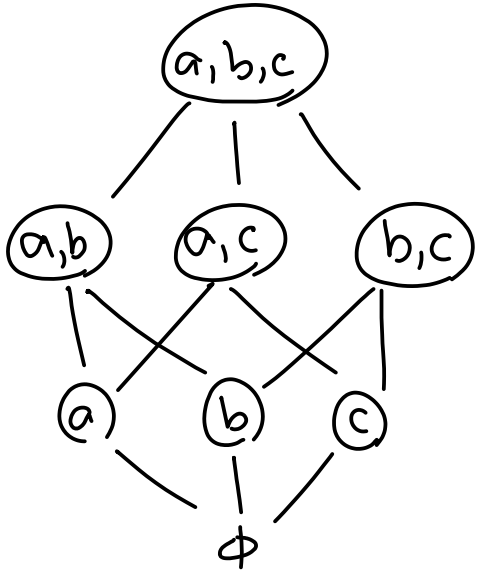
انعکاسی $x \leq x$

متعدی $x \leq y \wedge y \leq z \Rightarrow x \leq z$

پادمتقارن $x \leq y \wedge y \leq x \Rightarrow x = y$



$$(R \subseteq R^{-1}) \Rightarrow R = R^{-1}$$



مثال رابطه‌ی $\subseteq$

برای زیرمجموعه‌های $X=\{a,b,c\}$

یک ترتیب جزئی است:

$$(P(X), \subseteq)$$



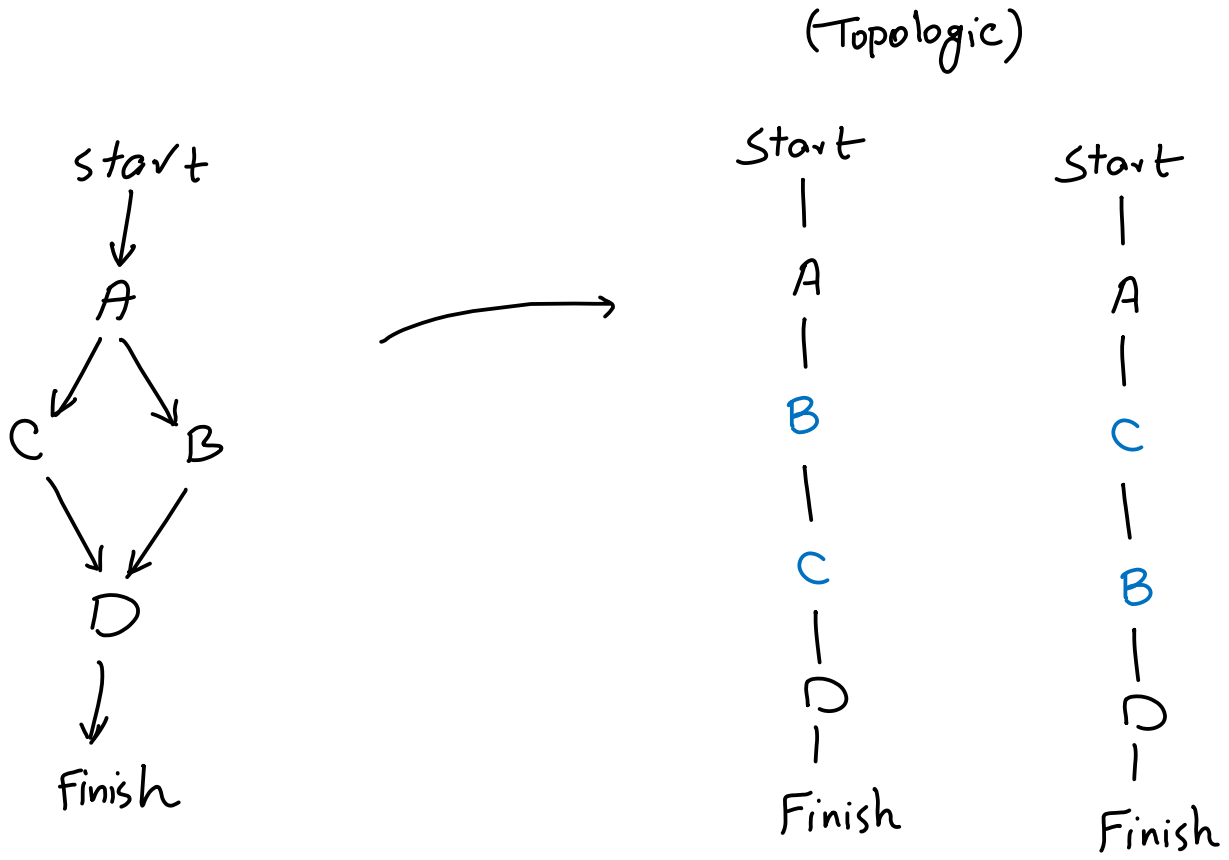
مثال: ترتیب معمولی اعداد رابطه‌ی ترتیب خطی (کلی) است چون در یک صف قرار می‌گیرند.

برخی از روابط ترتیب، ترتیب کلی هم هستند. یعنی همه ی اشیاء با هم قابل مقایسه هستند و در یک خط (صف) قرار می گیرند.

ترتیب توپولوژیک : دو ترتیب جزئی، دارای حداقل یک ترتیب کلی است که رابطه ی ترتیب جزئی را هم حفظ کرده باشد. آن را ترتیب توپولوژیک گوییم.

مثال: شکل لیست چپ یک برنامه با ترتیب جزئی است. در این برنامه، ترتیب B و C نسبت به هم مشخص است اما برای اقدامات دارای ترتیب هستند (ترتیب جزئی) ما می‌توانیم به دو صورت این برنامه را با حفظ روابط ترتیب جزئی خواسته شده، به یک برنامه‌ی کلاگرب تبدیل کنیم. (دو ترتیب توپولوژیک داریم)

(Topologic)

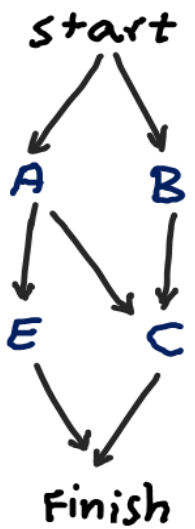


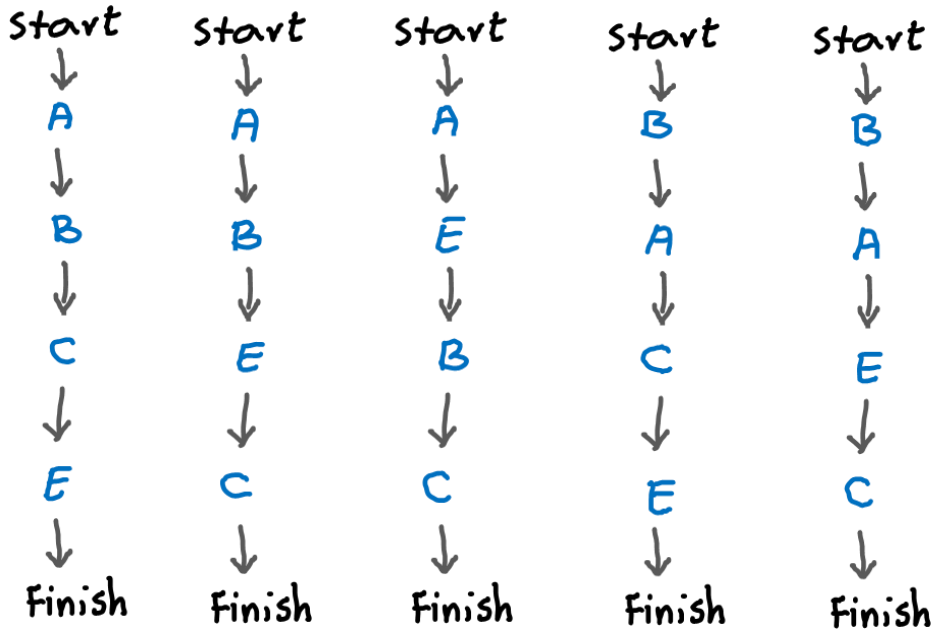
یادآوری: در ساختمان داده یادگرفتیم که ترتیب حدی بین گره‌های گراف را می‌توان در زمان $O(E+n)$ با استفاده از DFS به ترتیب Topologic تبدیل کنیم. برای درخت‌ها زمان اجرا $O(n)$ است.

P.O.P : خروجی الگوریتم P.O.P (در صورت وجود راه حل) یک plan با ترتیب جزئی و سازگار است

تبدیل جزئی برنامه‌ای به برنامه‌ی مرتب کامل، کار ساده ای است. (ترتیب Topologic)

(خروجی P.O.P به عنوان مثال)





Partially Ordered Plans

POP:

جستجو در فضای برنامه‌ها است.

نوعی جستجوی عقب‌گرد است.

کامل است. (درصورت وجود راه‌حل آن را می‌یابد)

بهینه نیست.

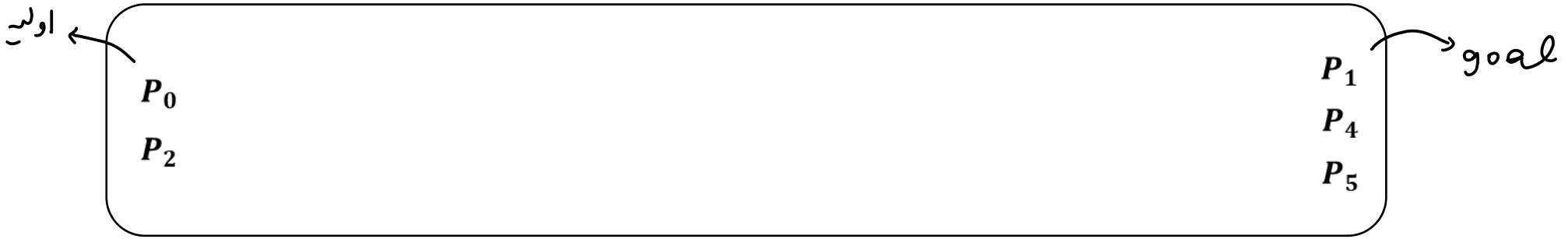
خروجی آن درصورت وجود جواب: یک برنامه‌ی جزئی مرتب سازگار است که لیست باز تهی داشته باشد.

(لیست باز، در هر مرحله، اهدافی است که هنوز ارضاء نشده‌اند)

درصورت عدم وجود جواب، در loop گرفتار شده متوقف نمی‌شود.

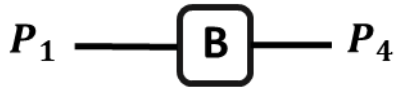
عدم وجود جواب را تشخیص نمی‌دهد.

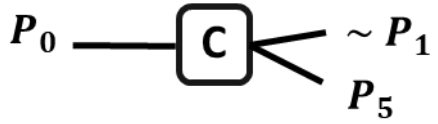
از این مثال برای معرفی ابزار ترفیع، تنزل در الگوریتم P.O.P استفاده کردیم.



پیش‌شرط‌ها اثرات







تنزل و ترفیع:

تصور کنید B یک اقدام است که به پیش‌شرط P_1 نیاز دارد و A اقدامی است که P_1 در اثرات مثبت آن قرار دارد. حال اگر اقدام C می‌خواهد به برنامه اضافه شود و P_1 در اثرات منفی آن قرار دارد،

C نباید بین A و B قرار داشته باشد، یا باید ترفیع یابد (C بعد از B باشد)

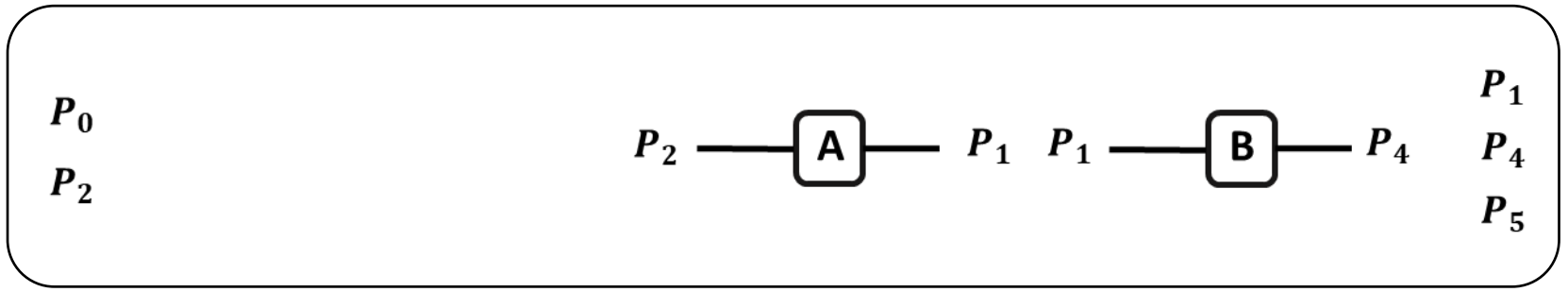
Before(**B**,**C**)

یا باید تنزل یابد (C قبل از A باشد).

Before(**C**,**A**)

توضیح آنچه در کلاس گفتم:

initial:



goal = $P_1 \wedge P_4 \wedge P_5$

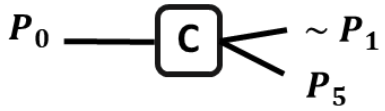
مثلاً: برای تولید P_4 از B استفاده کرده‌ایم

B به پیش‌شرط P_1 نیاز دارد.

A را به plan اضافه کرده‌ایم تا P_1 را تولید کند.

تا اینجا اقدامات  به برنامه اضافه شده‌اند و A باید قبل از B باشد.

حالا برای تولید P_5 می‌خواهیم از اقدام C استفاده کنیم:



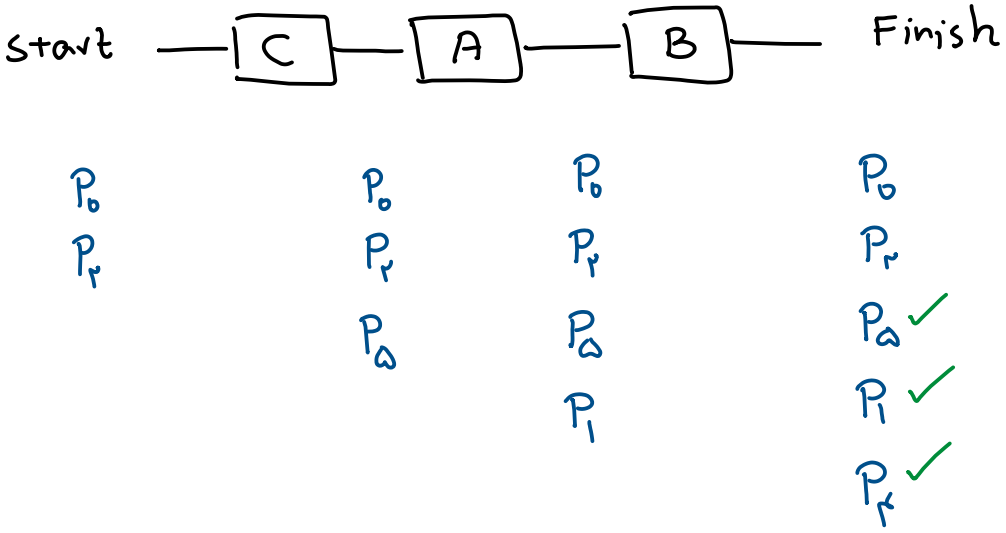
از آنجا که اثر منفی آن $\sim P_1$ است و B به پیش‌شرط P_1 نیاز دارد پس C باید بین A و B باشد

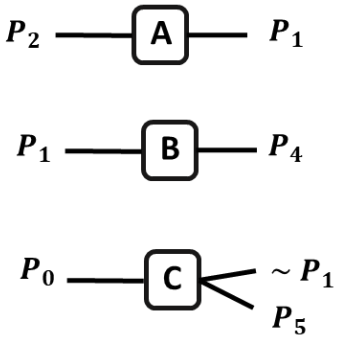
باید به قبل از A یا به بعد از B منتقل شود. البته این بررسی‌ها ادامه خواهد یافت.

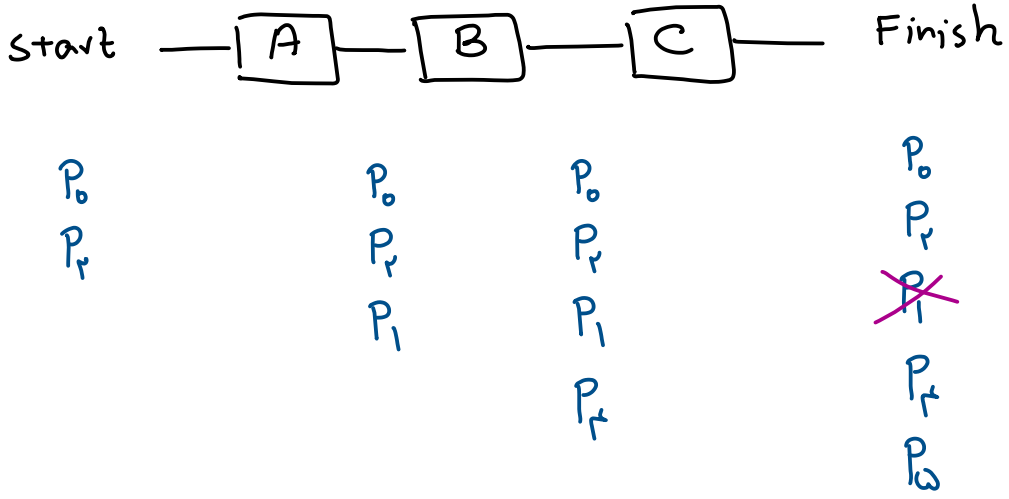
البته در این مثال اگر به مطالعه و بررسی ادامه دهیم از بین دو پیشنهاد ترفیع و تنزل ،

پیشنهاد تنزل یک راه حل برای مسائلست ؛

مثلاً بررسی میزان کدی از
اجزای هواپیما نشان میدهد
برنامه‌ی C — A — B
یک برنامه سازگار است و
همه اهداف را تولید می‌کند.
سازگار: یعنی نوبت هواپیما که
شود، پیش شرط آن برقرار است.







اما برنامه‌ی

start – A – B – C – Finish

راه حل نیست چون در پایان، P_1 را

از دست می‌دهد در حالی که P_1 حیز

اهداف بود.

Start — P_0

P_6 — Finish

مثال: مساله‌ای را که دارای حالات ممکن $P_0, P_1, \ldots, P_7$ است در نظر بگیرید.

فرض کنید در ابتدا P_0 برقرار است. (فرض دنیای بسته را در نظر بگیرید)

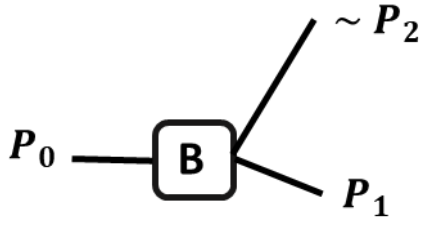
هدف هنگامی برقرار است که P_6 معتبر شود.

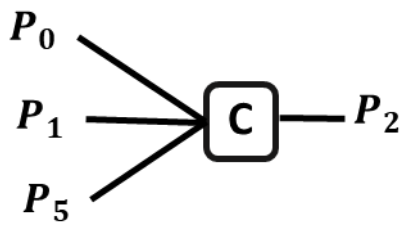
اقدامات، پیش‌شرط‌ها و اثرات آنها به این صورت‌اند:

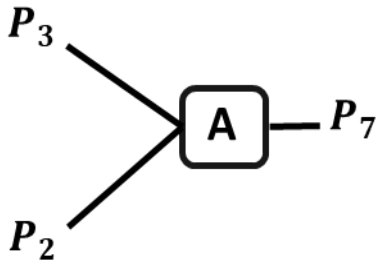
اقدامات:

پیش‌شرط‌ها

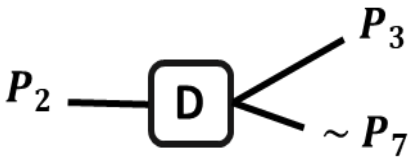
اثرات:

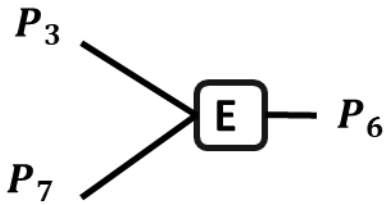




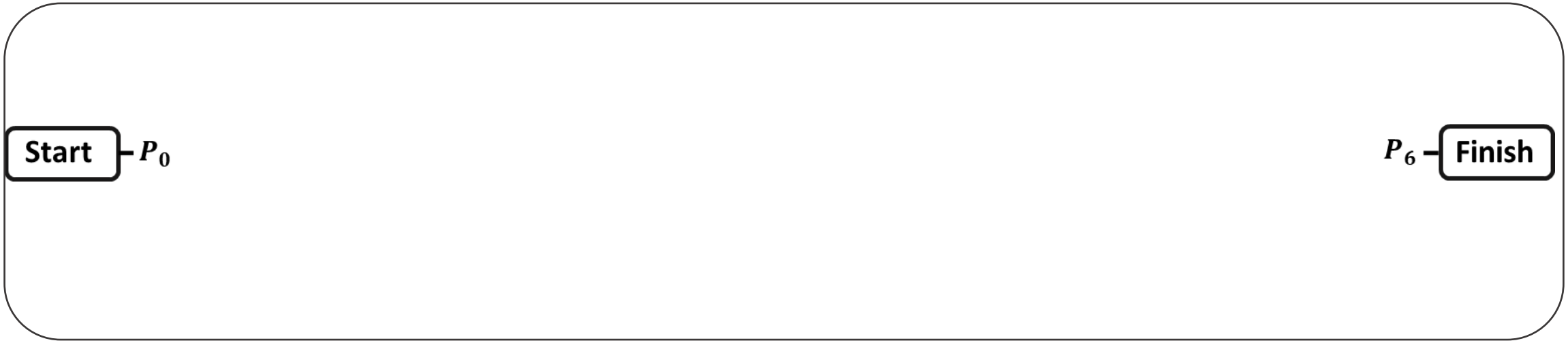






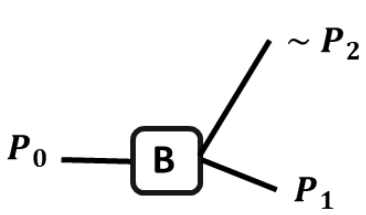


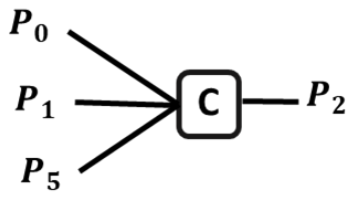
شروع: در ابتدا یک plan تهی داریم:

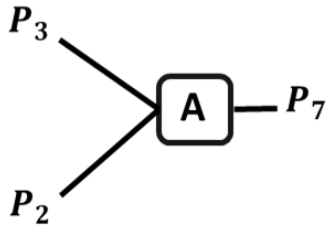


Open list: P_6

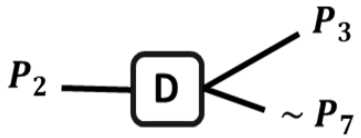
در ابتدا لیست باز شامل لیترال‌های هدف است.

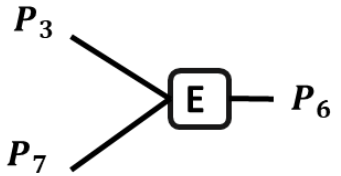


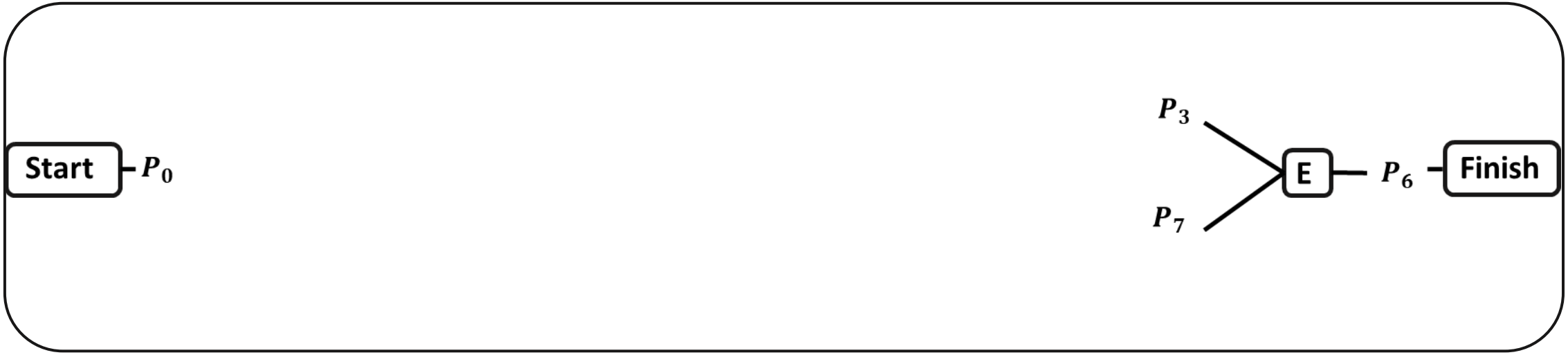




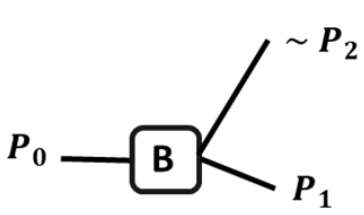


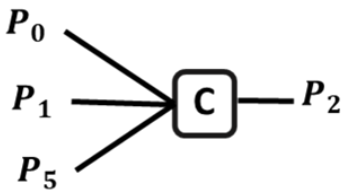


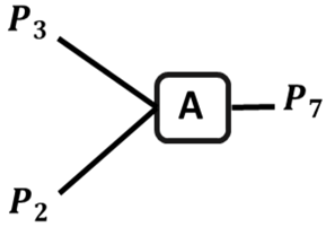




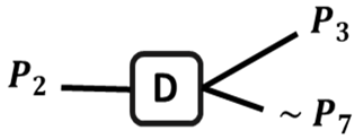
Open list: P_2, P_V

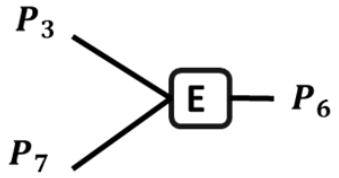


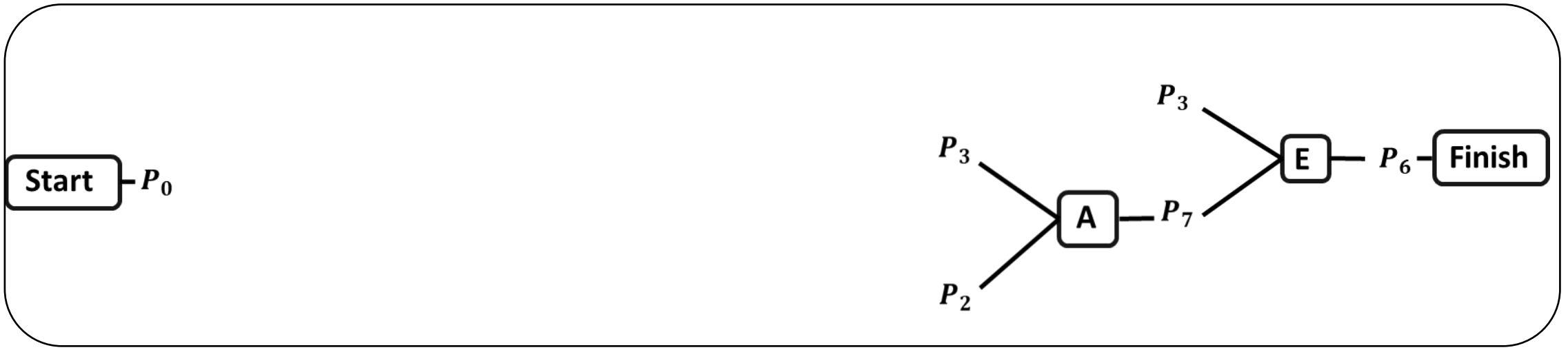




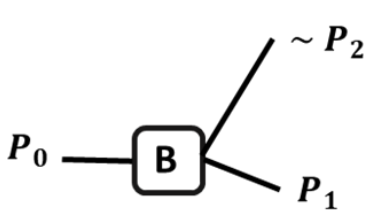


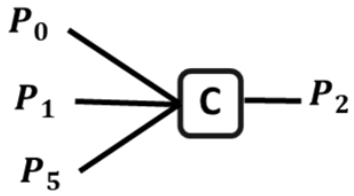


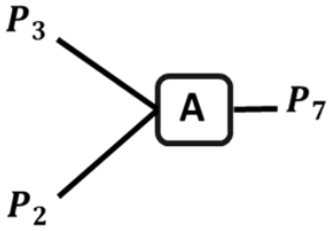




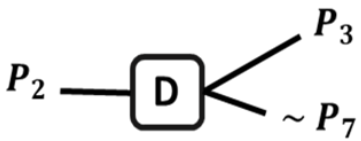
Open list: P_3 , P_2

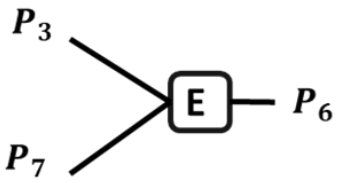


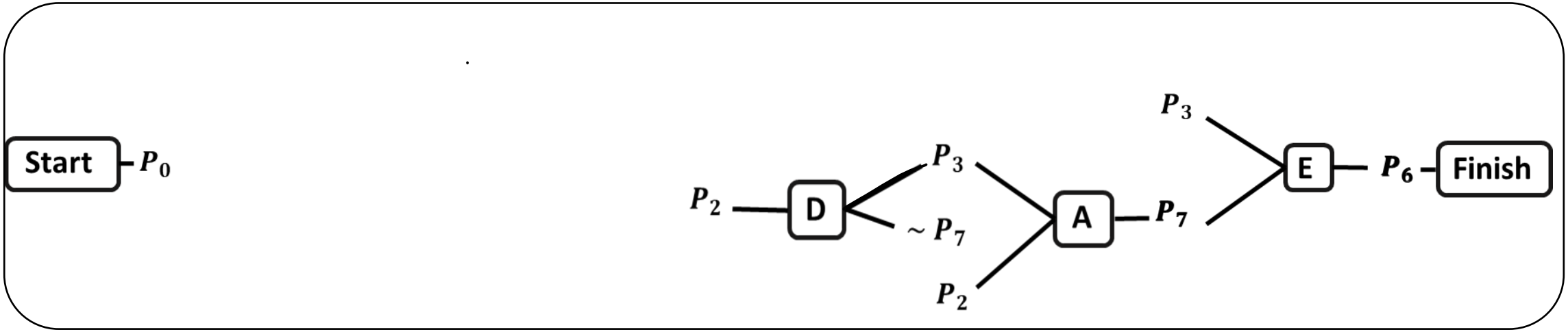




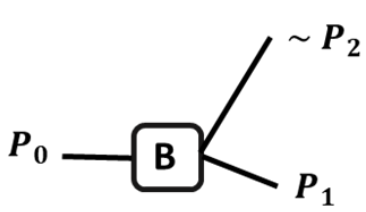


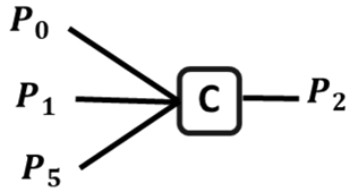


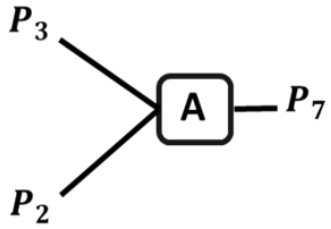



Start
P_0
P_2
D
P_3
$\sim P_7$
P_2
A
P_7
P_3
E
P_6
Finish

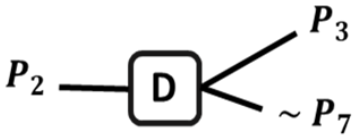
Open list: P_7

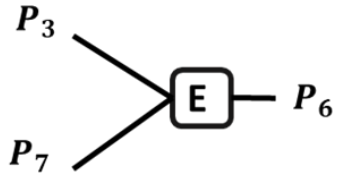

P_0
B
$\sim P_2$
P_1

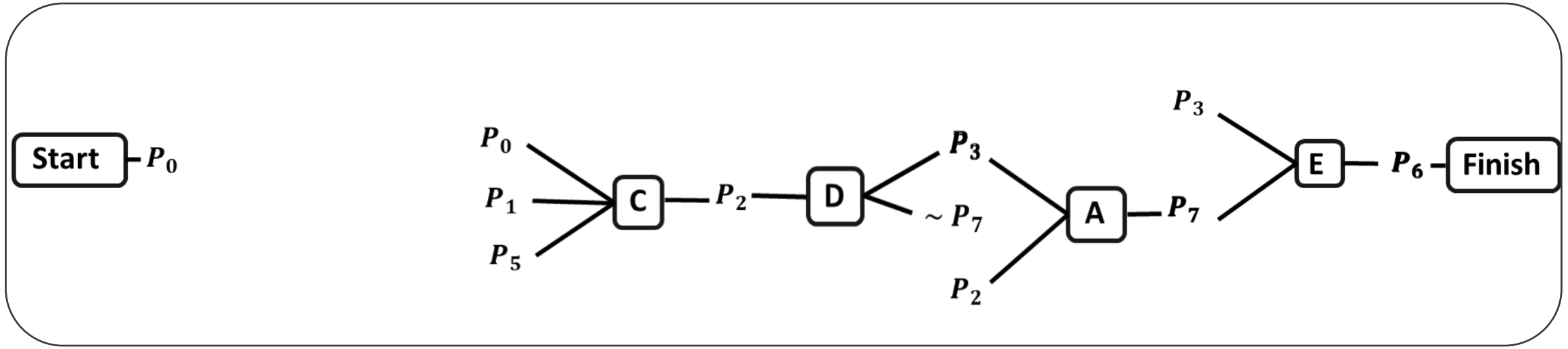

P_0
P_1
P_5
C
P_2


P_3
P_2
A
P_7

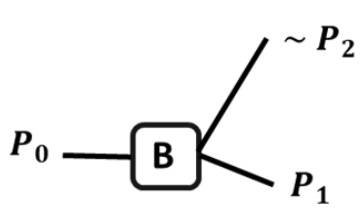

P_0
F
P_5

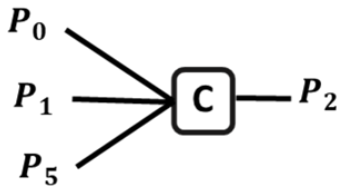

P_2
D
P_3
$\sim P_7$

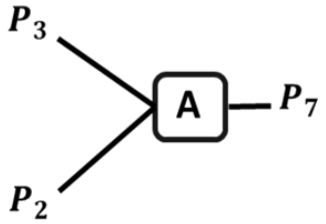

P_3
P_7
E
P_6



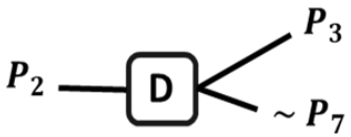
Open list: P_1 , P_2

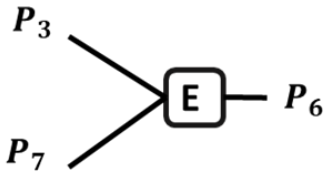


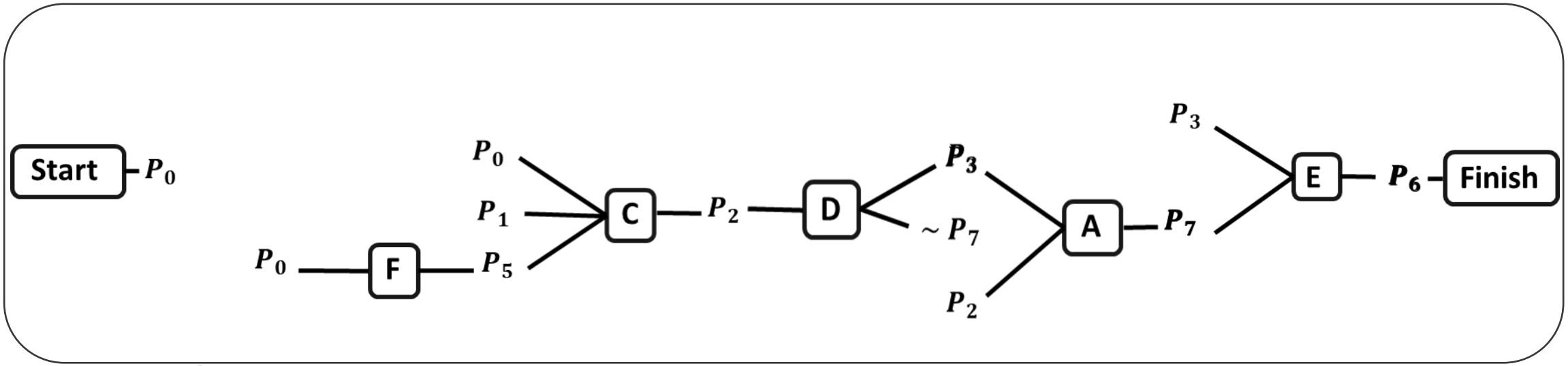




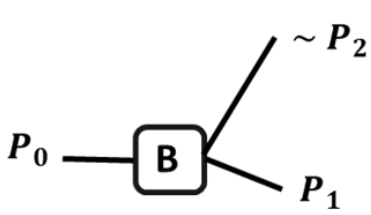


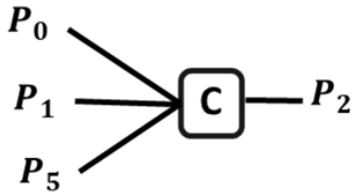


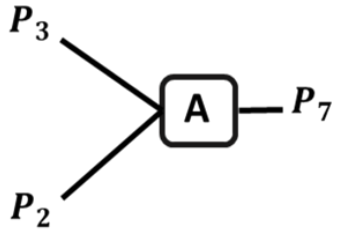



Start
P_0
P_0
P_1
P_0
F
P_5
C
P_2
D
P_3
$\sim P_7$
P_2
A
P_3
P_7
E
P_6
Finish

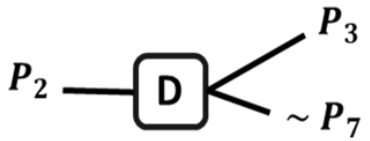
Open list: P_1

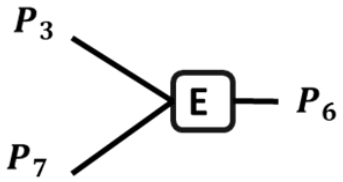

P_0
B
$\sim P_2$
P_1

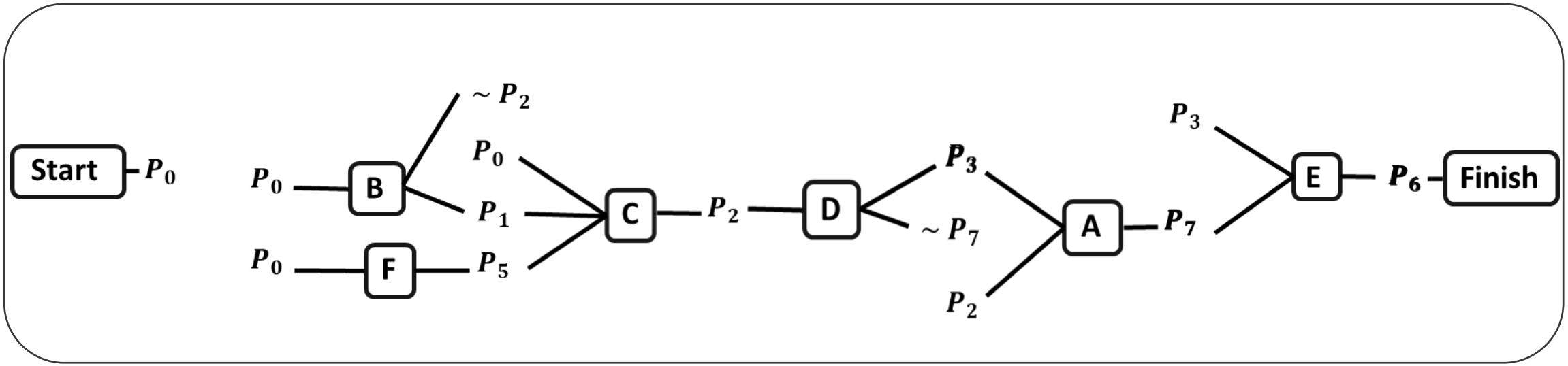

P_0
P_1
P_5
C
P_2


P_3
P_2
A
P_7


P_0
F
P_5

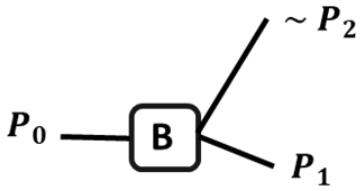

P_2
D
P_3
$\sim P_7$

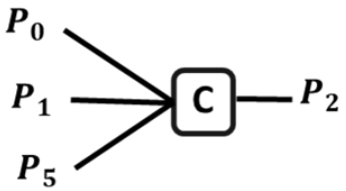

P_3
P_7
E
P_6

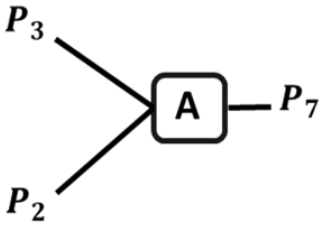


Open list: { }

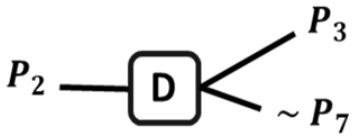
در انتهای لیست باز تهی می شود.

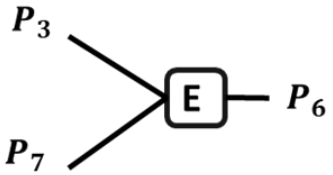




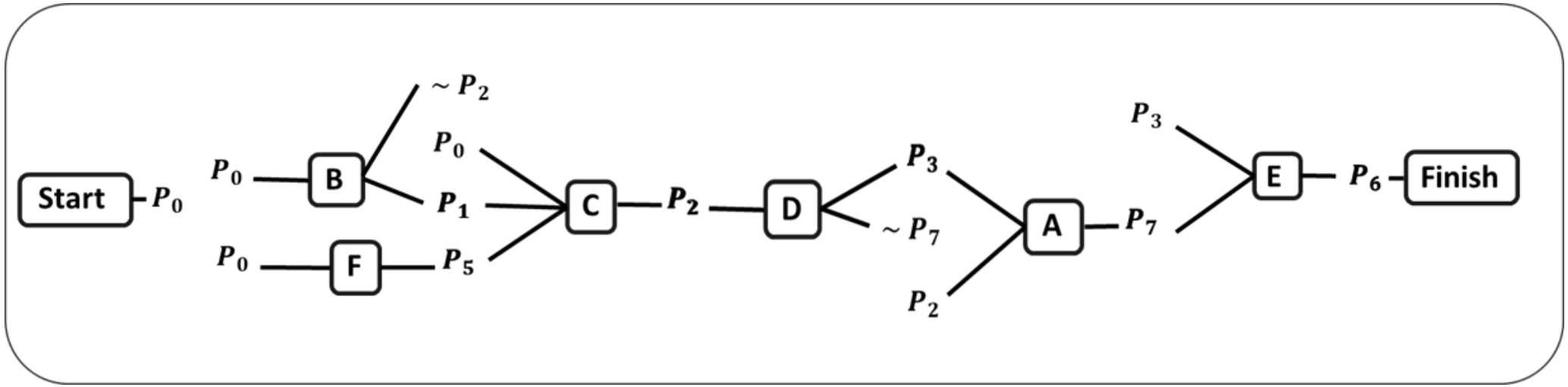


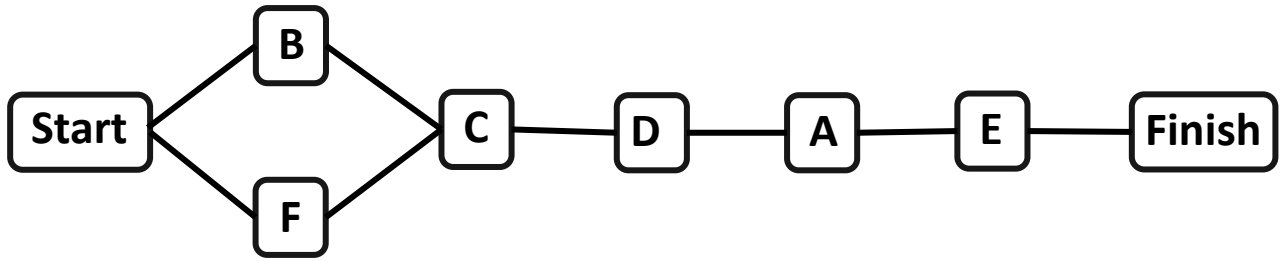






خروجی P.O.P
یک برنامه
چند تا مرتب
سازگارانه با
لیت باز تهی.





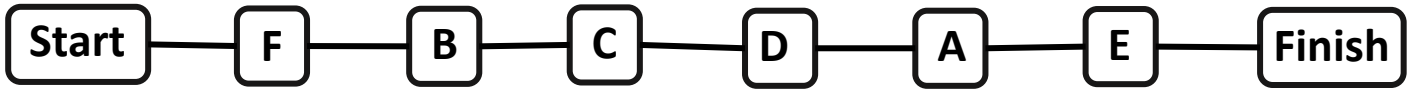
خروجی pop:

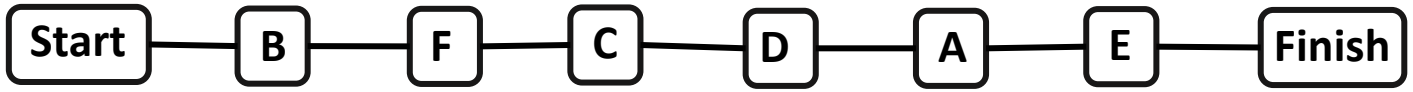
یک برنامه کامل، سازگار، دارای ترتیب جزئی:

↳ openlist = {}

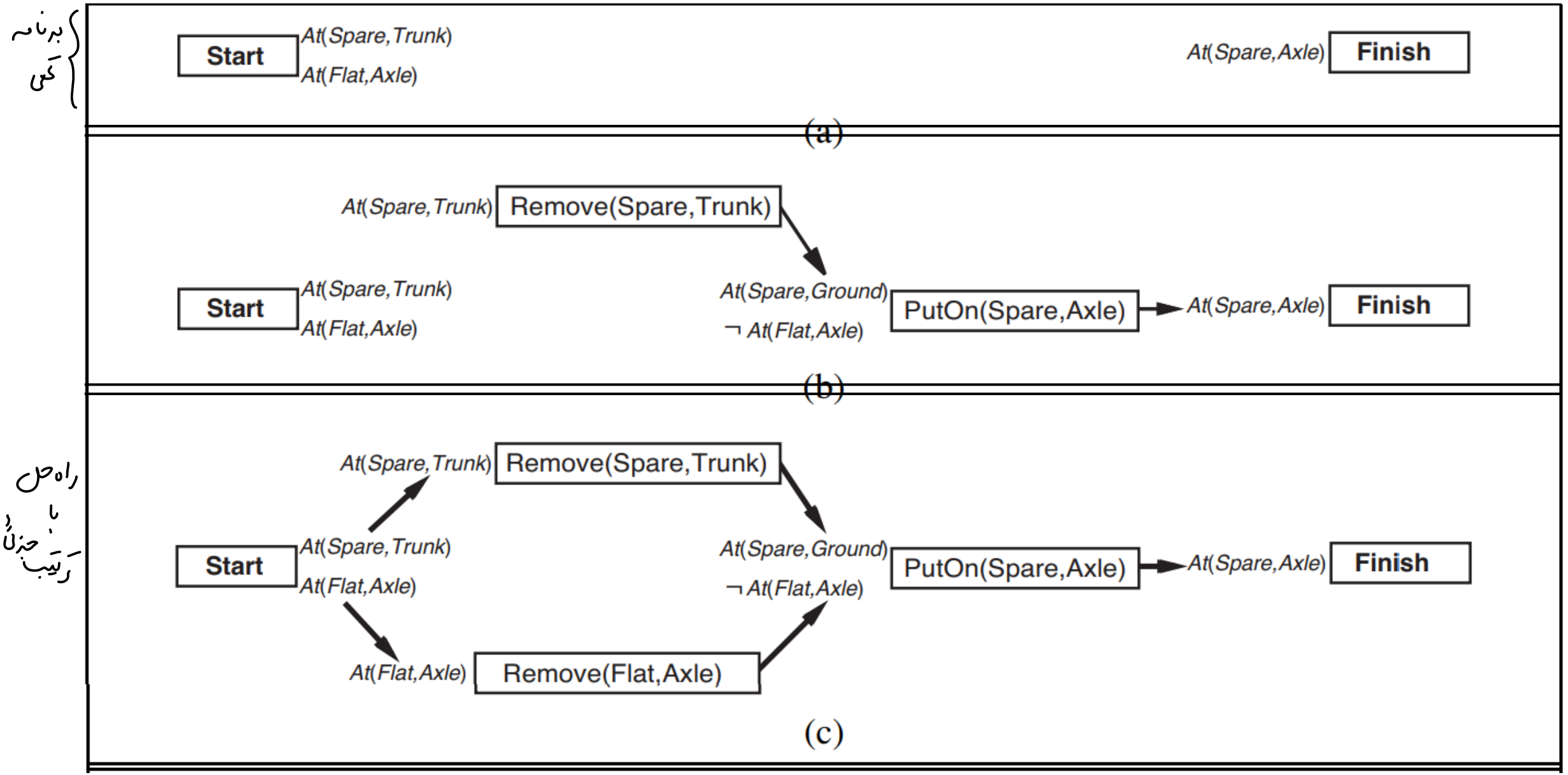
↳ اقدامات قابل اجرا

دو ترتیب Topologic یعنی دو plan با ترتیب خطی از این خروجی به دست می‌آید که هردو راه‌حل‌های مرتب کلی هستند





مثال تعویض تایر با روش P.O.P:



برخلاف graphplan ، الگوریتم POP نمی‌تواند عدم وجود جواب را تشخیص دهد.

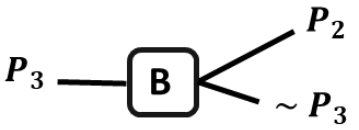
اگر جواب وجود نداشته باشد، POP در loop قرار می‌گیرد و متوقف نمی‌شود.

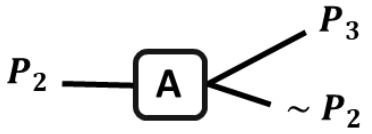
به مثال زیر توجه کنید:

مثال: حالت اولیه: P_0

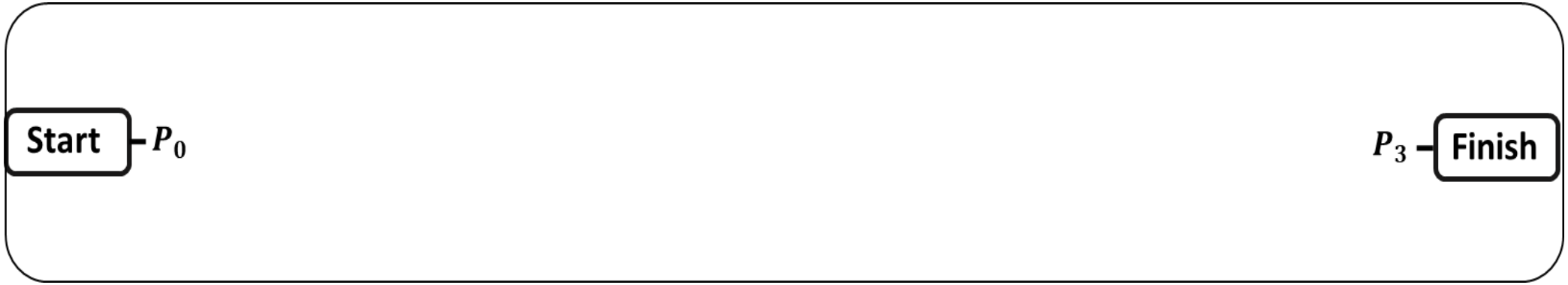
هدف: P_3

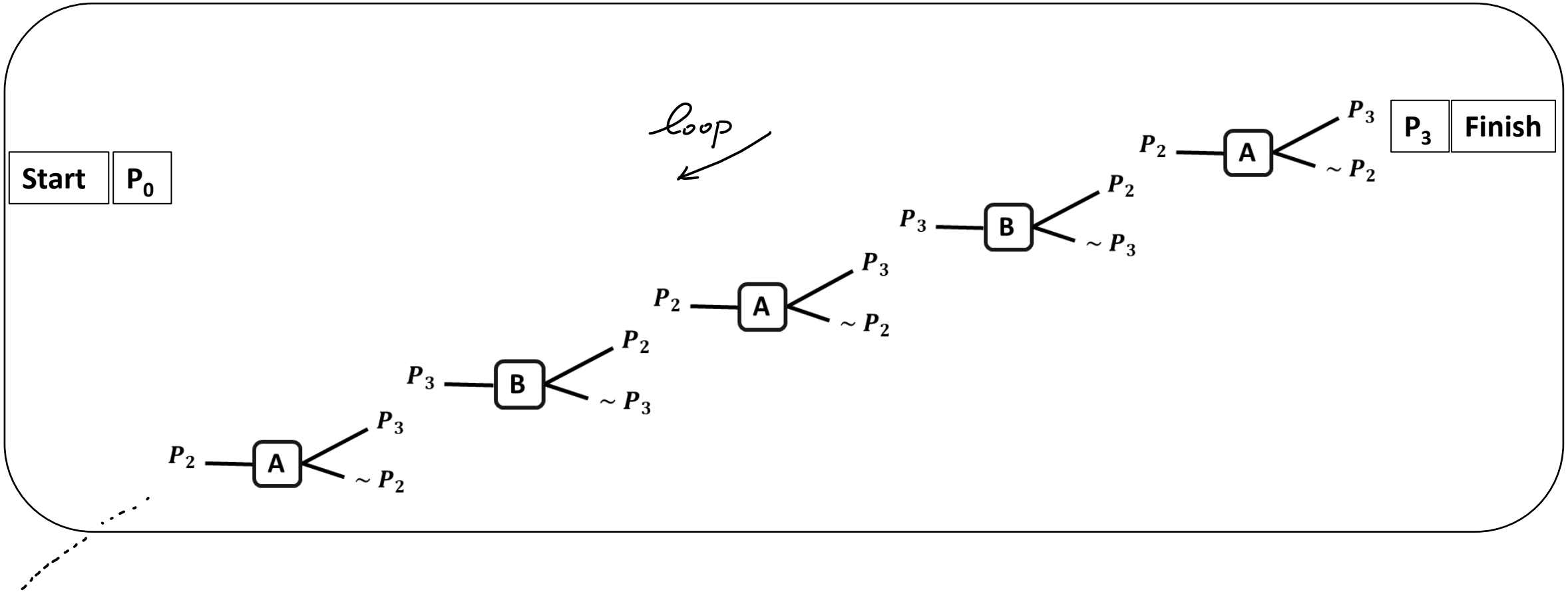
اقدامات:





در این مثال در P.O.P حلقه loop گرفتار شده و هرگز متوقف نمی‌شود. (تشخیص میدهد که مسأله راه‌حل ندارد)




Start
P_0
loop
P_2
A
P_3
$\sim P_2$
P_3
B
P_2
$\sim P_3$
P_2
A
P_3
$\sim P_2$
P_3
B
P_2
$\sim P_3$
P_2
A
P_3
$\sim P_2$
P_3
Finish

اشاره‌ای به سایر الگوریتم‌های برنامه‌ریزی کلاسیک:

Planning as constraint satisfaction (CSP)

Classical planning as Boolean satisfiability (SAT)

Planning as first-order logical deduction: Situation calculus (FOL) (حساب موقعیت)

Classical planning as Boolean satisfiability

در حل مسائل برنامه‌ریزی با روش ارضای پذیری بولی :

① نمایش وضعیت اولیه به صورت مجموعه‌ای از گزاره‌های زمینه‌ای مثبت در شروع مسأله :

F° : گزاره‌هایی که در شروع True هستند

$\sim F$: گزاره‌هایی که در شروع false هستند

② اقدامات : باید به صورت زمینه‌ای معرفی شوند (بدون متغیر)

③ هدف : به جای ترکیب عطفی از لیترال‌های دارای متغیر ، باید ترکیب OR از همه حالات زمینه‌ای هدف را نوشت :

(۲۰) اقدامات باید زمینه‌ای باشند:

P_1 P_2

$\overline{SFO}$ $\overline{JFK}$

مثلاً در مثال پرواز هواپیما (در PDDL به طرح واره‌ی Fly(p, from, to) را داشتیم. اما در حل با ارضاپذیری بولی (استفاده از منطق گزاره‌ای) باید ۴ اقدام زیر را جداگانه معرفی کنیم:

Fly(P_1, SFO, JFK)

Fly(P_1, JFK, SFO)

Fly(P_1, JFK, JFK)

Fly(P_1, SFO, SFO)

(۴) هدف که در PDDL به صورت ترکیب and از لیترال‌های دارای متغیر بود در روش (SAT) به صورت (DNF زمینه‌ای) نوشته می‌شود. یعنی ترکیب OR از and ها و بدون متغیر.

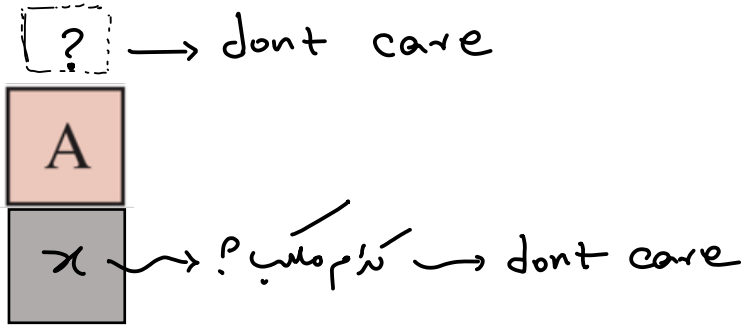
goal : $On(A, x) \wedge Block(x)$

مثال:

در دنیای مکعب‌ها اگر هدف آن باشد که A روی یک مکعب قرار داشته باشد:

[در حالت هدف، برایمان وضعیت مبدا، این که A در طبقه‌ی دوم باشد یا سوم، بالای A خالی باشد یا نه؟ مهم نیست.

فقط می‌خواهیم A روی یک مکعب باشد.



goal : $On(A, x) \wedge Block(x)$

در روش ارضاءپذیری SAT این هدف چنین بیان می‌شود: (بدون متغیر و فرم DNF)

goal: $(On(A, A) \wedge Block(A)) \vee (On(A, B) \wedge Block(B)) \vee (On(A, C) \wedge Block(C))$

④ اصل موضوع حالت مابعد (حالتِ وِسین)

- Add successor-state axioms: For each fluent F, add an axiom of the form

$$F^{t+1} \Leftrightarrow ActionCausesF^t \vee (F^t \wedge \neg ActionCausesNotF^t) ,$$

یک اِوان در F^{t+1} معتبر است اگر و تنها اگر: نتیجه ی مثبت یک عمل در مرحله قبل باشه
یا آن که در F^t باشه و عمل انجام شده آن را
در لیست حذف نداشته باشه .

④ اصل موضوع پیش شرط: هر اقدامی که در A^t قرار داشته باشد (اجرایی باشد) حتماً پیش شرط هایش برقرار است.

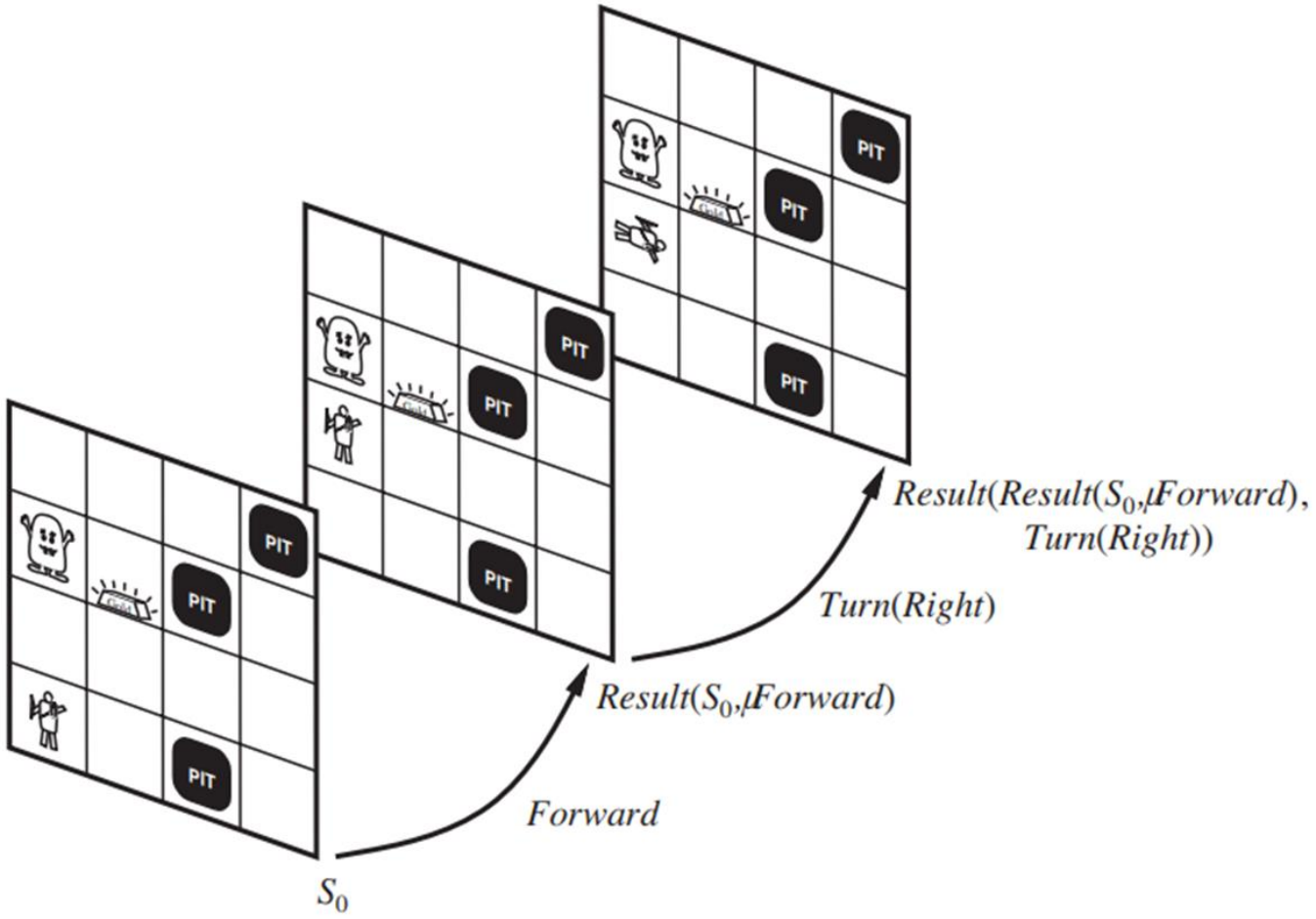
④ • Add precondition axioms: For each ground action A, add the axiom $A^t \Rightarrow \text{PRE}(A)^t$, that is, if an action is taken at time t, then the preconditions must have been true.

⑤ • Add action exclusion axioms: say that every action is distinct from every other action.

⑤ اصل موضوع تمایز: هیچ دو اقدامی، یکسان نیستند. (دقت کنید که اقدامات زمینه‌ای هستند)

Planning as first-order logical deduction: Situation calculus

برنامه‌ریزی با حساب موقعیت (در FOL)



A function or relation that can vary from one situation to the next is a **fluent**.

روان : منه‌ها و توابعی که مقدار یا ارزش آنها از یک موقعیت به موقعیت دیگر تغییر کند.

مسأله‌ی قاب: نحوه‌ی نمایش و ذخیره‌ی همه روابط و توابعی که تغییر نمی‌کنند چگونه است؟

Frame problem

situation:

الف) وضعیت اولیه یک موقعیت است،

ب) اگر s یک موقعیت باشد و a یک عمل، آنگاه RESULT(s, a) نیز یک موقعیت است.

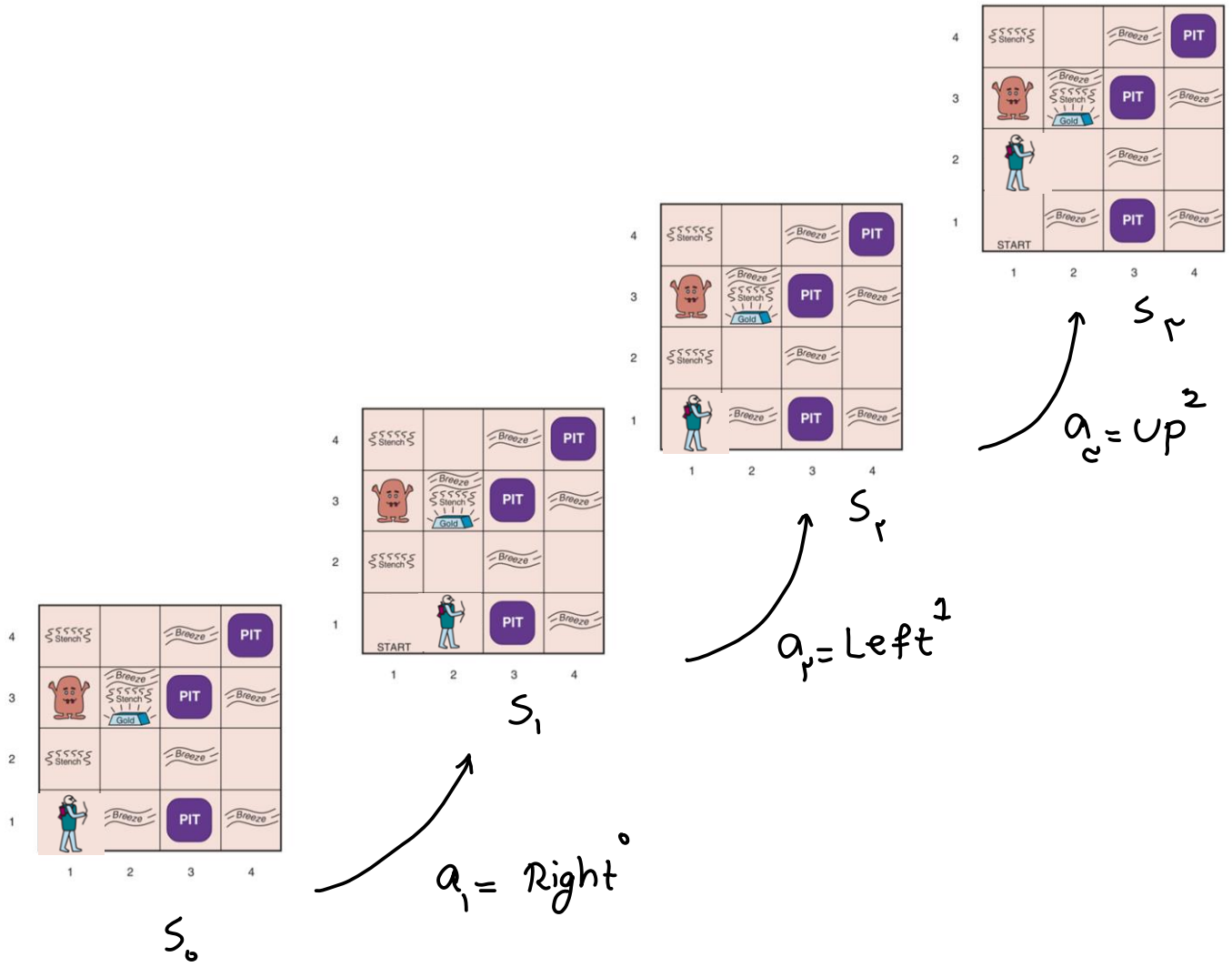
ج) موقعیت دیگری به جز آنهایی که با قانون ب ساخته می‌شوند وجود ندارد.

د) یکتا بودن نحوه‌ی تولید موقعیت، در جهان موقعیت، هر موقعیت فقط و فقط از یک طریق با شروع از S_0 به دست می‌آید.

$$(\text{RESULT}(s,a) = \text{RESULT}(s',a')) \Leftrightarrow (s = s' \wedge a = a').$$



توضیح بیشتر درمورد یکپارچگی توابع موقعیت و یکپارچگی اقدامات:



در دیدگاه حسابان موقعیت،
با یک زنجیره رو به جلو از
موقعیت ها سروکار داریم.
حتی اگر اقدام a را انجام
دهیم و سپس آن اقدام را برگردیم
نمی‌رویم به S_0 برگشتیم زیرا
زمان (t) تغییر کرده است.

با این دیدگاه هر موقعیت ، معادل است با دنباله‌ای یکتا از اقدامات با شروع از s_0

unique action axioms

$$A_i(x_1, \ldots, x_n) = A_i(y_1, \ldots, y_n) \Leftrightarrow x_1 = y_1 \wedge \ldots \wedge x_n = y_n .$$

کتابی اکشنات

موفقیت هدف:

- A solution is a situation (and hence a sequence of actions) that satisfies the goal.

موقعیتی است که هم اکشن‌های هدف در آن مجزا شوند.

ویژگی مسئله‌ی برنامه‌ریزی:

مسئله‌ی ① **PlanSAT** آیا برنامه‌ای برای رسیدن به اهداف وجود دارد؟

مسئله‌ی ② **Bounded PlanSAT** آیا برنامه‌ای برای رسیدن به اهداف با حداکثر k اقدام وجود دارد؟

الف: در برنامه‌ریزی کلاسیک (تعداد اقدامات و متغیرها متناهی) هر دو مسئله تصمیم‌پذیر هستند.

ب: اگر نماد تابعی هم داشته باشیم، تعداد متغیرها نامتناهی می‌شود در این حالت مسئله‌ی ① نیمه‌تصمیم‌پذیر است.
مسئله‌ی ② هنوز تصمیم‌پذیر است.

در حالت کلی: هودومساله، به لحاظ زمانی NP-hard هستند.

به لحاظ فضای حافظه مورد نیاز PSPACE هستند (فضای مورد نیاز خیلی زیاد)

هودومساله توسط یک ماشین تورینگ قطعی حل می‌شوند.

بدون اثرات منفی اقدامات ← هودومساله NP-hard باقی می‌ماند.

بدون اثرات و پیش شرط‌های منفی ← مساله ① ازردی P است.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۲۱ فصل ۱۰ الی ۱۳

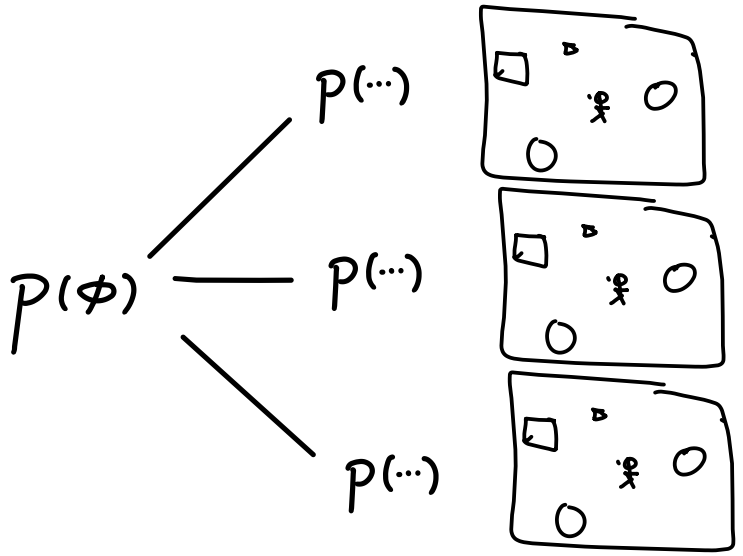
$$\textit{Decision theory} = \textit{probability theory} + \textit{utility theory} \ .$$

اصول نظریه احتمال:

$0 \leq P(\omega) \leq 1$ for every ω and $\sum_{\omega \in \Omega} P(\omega) = 1$.

$$\sum_{\omega \in \Omega} P(\omega) = 1$$

For any proposition ϕ, $P(\phi) = \sum_{\omega \in \phi} P(\omega)$.



inclusion–exclusion principle:

$$P(a \vee b) = P(a) + P(b) - P(a \wedge b)$$

مثال

1	2	3
.30	.50	.20

$P(Y=1) = 0.50$

Y: رده سنی
Y = 1 کودک
Y = 2 میان‌سال
Y = 3 سالمند

D: ابتلا

~D	D=0
D	D=1

	1	2	3
~D	.20	.30	.5
D	.10	.20	.15

$P(D \mid Y=2) = 0.4$

۴۰ درصد از ۵۰ درصد $= (0.4)(0.5) = 0.2$

$$P(D) = \sum_{y=1}^{3} P(D \mid Y=y)$$

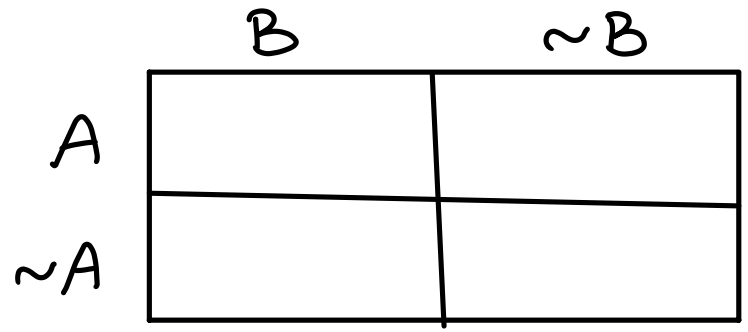
$$= .10 + .20 + .15 = 0.45$$

	1	2	3
~D	.20	.30	.5
D	.10	.20	.15

$$P(D, Y=1) = 0.10$$

$$P(D \mid y=1) = \frac{0.10}{0.10 + 0.20} = \frac{1}{3}$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

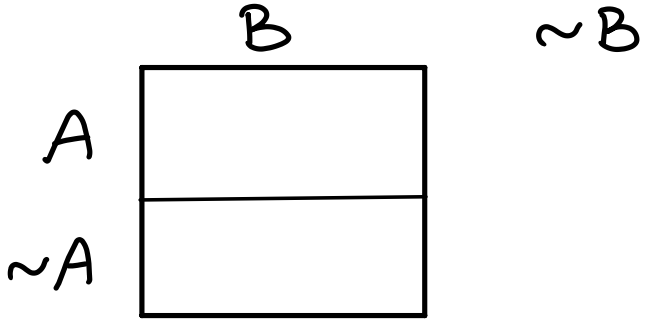


احتمال به شرط B یعنی فرض کن در دنیای B

قرار داریم. حالا هم $A \cap B$

از کل B را حساب کن

$P(A|B)$



اصل توام $p(A,B) = p(B)\,p(A|B)$

$= p(A)\,p(B|A)$

برای احتمال توام

$$p(A,B,C) = p(A)\,p(B|A)\,p(C|A,B)$$

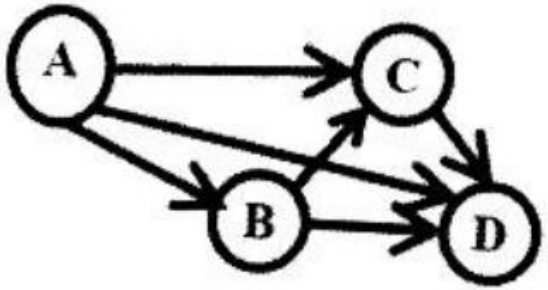
$$= p(C)\,p(B|C)\,p(A|B,C)$$

$$= p(B)\,p(A|B)\,p(C|B,A)$$

$$= \cdots$$

از نظر قواعد احتمال،
همه این‌ها درست است اما
با توجه به شکل شبکه‌ی بیز
یکی را باید استفاده کنیم.

۹۳- برای شبکه باور بیزی زیر، احتمال شرطی معادل با تابع توزیع توأم متغیرها کدام است؟



۱) $P(A).P(B|A).P(C|A).\ P(D|A)$

۲) $P(A).P(B|C).P(C|D).|P(D|A)$

۳) $P(B|A).P(C|A,B).P(D|A,B,C).P(A)$

۴) $P(A|B,C,D).P(B\ C,D).P(C|D).P(D)$

مهندسی کامپیوتر ۱۳۹۹

معنای استقلال به زبان ساده

گاهی اوقات $P(A,B)=P(A)P(B)$ می‌شود این را استقلال گوییم مثلاً:

$P(A)=\frac{20}{100}=\frac{1}{5}$

$P(A|B)=\frac{5}{25}=\frac{1}{5}$

$P(A|\sim B)=\frac{15}{75}=\frac{1}{5}$

شرط B روی A اثری ندارد.

$P(A,B)=P(B)P(A|B)$

$=P(B)P(A)$

A و B مستقل‌اند.

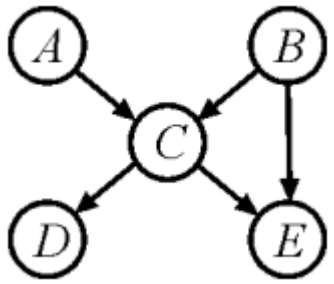
درس در این مثال:

	A	$\sim A$
B	5	20
$\sim B$	15	60

به شرط B

A	$\sim A$
5	20

۱۲ـ برای شبکه بیز ین (*bayesian network*) روبه‌رو، احتمال $P(A,B,C,D,E)$ برابر کدام است؟ (مهندسی IT ـ دولتی ۹۲)



۱) $P(C|A,B)P(E|B,C)P(D|C)$

۲) $P(C|D,E)P(B|C,E)P(A|C)$

۳) $P(A)P(B)P(C|A,B)P(D|C)P(E|C,B)$ ✓

۴) $P(D)P(E)P(C|D,E)P(B|C,E)P(A|C)$

B پیشی شرطی ندارد مانند A.

حاصلضرب = توام

$$p(A,B) = p(A)\,p(B)$$

مفهوم استقلال:

مفهوم استقلال به شرط C : (بعداً درمورد استقلال مشروط صحبت می‌کنیم)

$$p(A,B|C) = p(A|C)\,p(B|C)$$

تمرین

توجه: A و B مستقل هستند

$$P(A|B) = P(A) \iff$$

$$\iff P(B|A) = P(B)$$

$$\iff P(A,B) = P(A)\,P(B)$$

حاصلضرب = توام

$$P(A,B) = P(A)\,P(B)$$

وقتی A و B مستقل باشند.

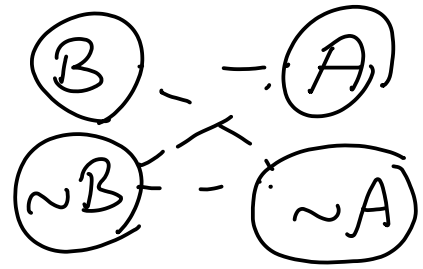
$$P(A|B) = P(A)$$

$$P(B|A) = P(B)$$

$$P(\sim A|B) = P(\sim A)$$

$\sim A$ و $\sim B$ هم نسبت به هم مستقل اند.

مستقل



تمرین

توجه:

$A \perp B$ متغیر B و A مستقل هستند

$A \perp \sim B$ متغیر $\sim B$ و A مستقل هستند

$\sim A \perp B$ متغیر B و $\sim A$ مستقل هستند

$\sim A \perp \sim B$ متغیر $\sim B$ و $\sim A$ مستقل هستند

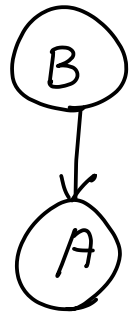
قانون مودفر:

$$p(A) + p(\sim A) = 1$$

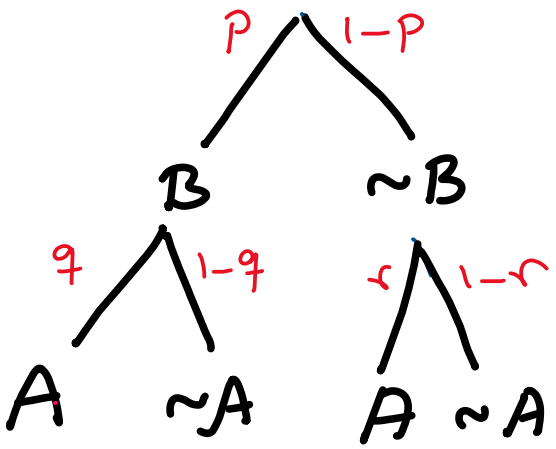
با شرط B هم برقرار است:

$$p(A|B) + p(\sim A|B) = 1$$

شبکه:



درخت احتمال شرطی



$p(A|B) = q \Rightarrow p(\sim A|B) = 1-q$

همان طور که

$$P(A) + P(\sim A) = 1$$

با شرط X نیز داریم:

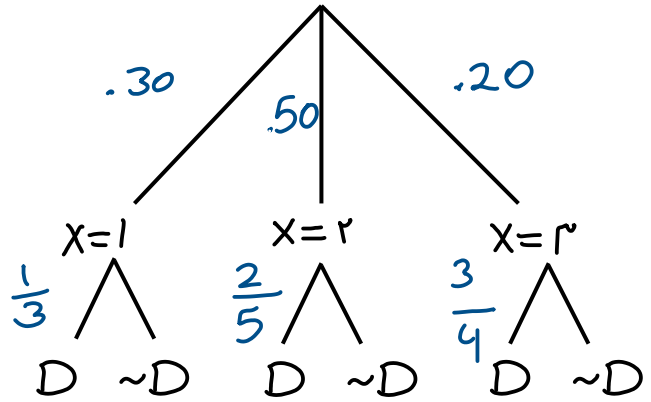
$$P(A|X) + P(\sim A|X) = 1$$

X: ردیف سنی

D: ابتلا به بیماری

X:	1	2	3
~D	.20	.30	.5
D	.10	.20	.15

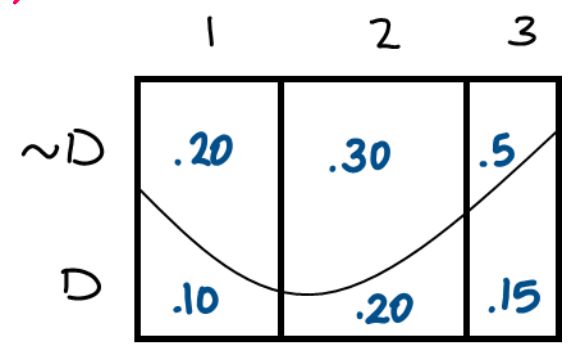
درخت احتمال شرطی در این مثال:



$$p(D|X=1) = \frac{10}{30} = \frac{1}{3}$$

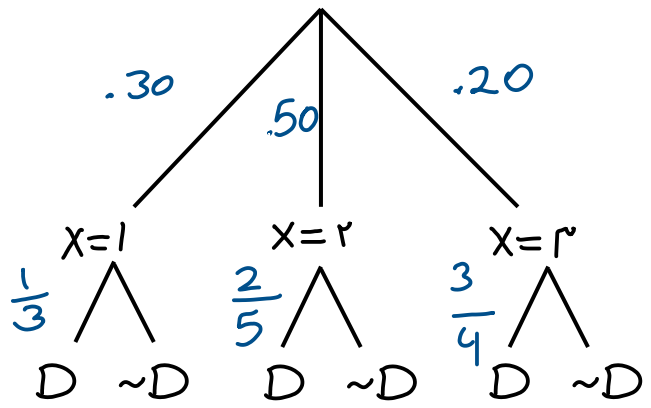
$$p(D|X=2) = \frac{20}{50} = \frac{2}{5}$$

$$p(D|X=3) = \frac{15}{20} = \frac{3}{4}$$



$$p(D) = \sum_{y=1}^{r} p(D, Y=y)$$

قانون جمع احتمالات: $P(D)$ چگونه است؟

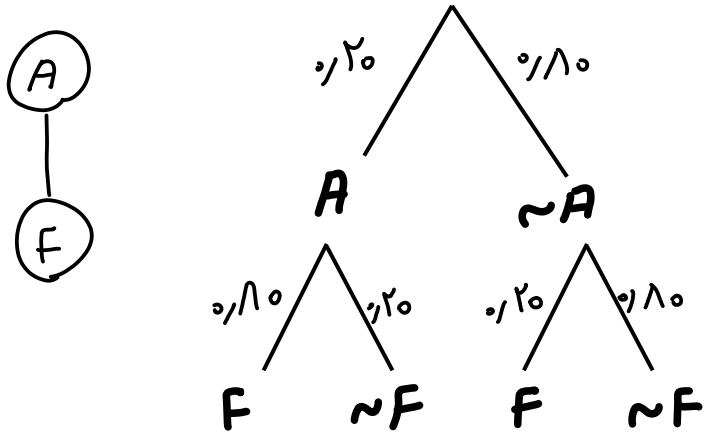


$$P(D) = \sum_{x=1}^{3} P(X=x, D)$$

$$= \sum_{x=1}^{3} P(X=x)\, P(D \mid X=x)$$

$$= (.30)\left(\frac{1}{3}\right) + (.50)\left(\frac{2}{5}\right) + (.20)\left(\frac{3}{4}\right)$$

قانون بیز: رسیدن به احتمال پیش (رسیدن از معلول به علت)



A : آنفلوانزا

F : تب

پزشک می‌داند احتمال آنفلوانزا بودن بیمار ۲۰٪ است (احتمال پیشین Prior)

$P(A) = 20\%$

پزشک می‌داند که: آنفلوانزا با احتمال ۸۰٪ و سایر بیماری‌ها با احتمال ۱۰٪ تب ایجاد می‌کنند.

$P(F|A) = 80\%$

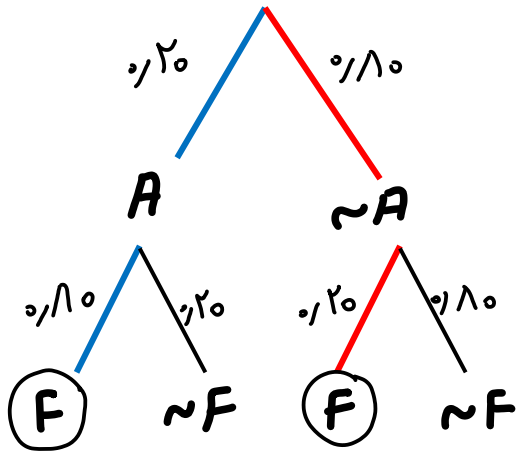
$P(F|\sim A) = 10\%$

اکنون مشاهده می‌شود که بیمار تب دارد (F رخ داده است) احتمال آنفلوآنزا اکنون چقدر است؟ (احتمال پسین Posterior)

$P(A|F) = ?$

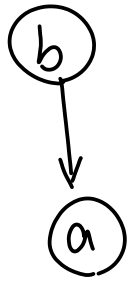
فرمول بیز

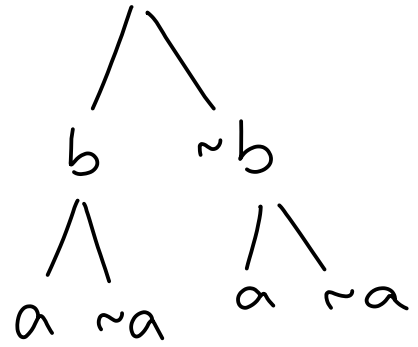
$$P(A|F) = \frac{P(A,F)}{P(F)} = \frac{P(A)\,P(F|A)}{P(A)\,P(F|A) + P(\sim A)\,P(F|\sim A)}$$



$$= \frac{(۰٫۲۵)\,(۰٫۸۰)}{(۰٫۲۵)\,(۰٫۸۰) + (۰٫۸)(۰٫۲۵)}$$

Bayes' rule $P(b \mid a) = \dfrac{P(a \mid b)P(b)}{P(a)} = \dfrac{P(b)\,P(a \mid b)}{P(b)\,P(a \mid b) + P(\sim b)\,P(a \mid \sim b)}$





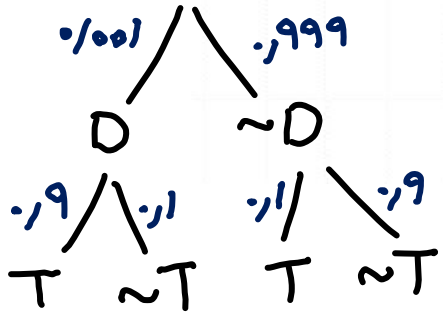
۱۳۰- اگر بدانیم $P(Test|Disease) = 0.9$، $P(\neg Test|\neg Disease) = 0.9$ و $P(Disease) = 0.001$ است، کدام گزینه مقدار تقریبی $P(Disease|Test)$ را نشان می‌دهد؟

۱) ۰/۰۰۴

۲) ۰/۰۴

۳) ۰/۸

۴) ۰/۰۸ → ۰/۰۰۸



$p(T|D) = ۰/۹$

$p(\sim T|\sim D) = ۰/۹$

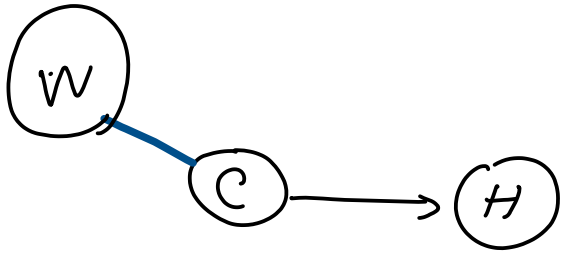
$p(D) = ۰/۰۰۱$

$p(D|T) = ?$

$$p(D|T) = \frac{۰/۰۰۱ \quad ۰/۹}{۰/۰۰۱ \quad ۰/۹ + ۰/۹۹۹ \quad ۰/۱} = \frac{۹}{۹+۹۹۹} = \frac{۱}{۱+۱۱۱} = \frac{۱}{۱۱۲} = ۰/۰۰۸$$

۱۵ـ در صورتی که بدانیم احتمال سردرد یک فرد سالم برابر ۰/۱ است و احتمال سرماخوردگی در زمستان ۰/۴ است و احتمال سردرد در صورت سرماخوردگی ۰/۸ می‌باشد. با فرض اینکه در زمستان هستیم و علی سردرد دارد احتمال سرماخوردگی علی چقدر است؟ (مهندسی IT ـ دولتی ۹۴)

۱) ۰/۸۴ ۲) ۰/۳۲ ۳) ۰/۸ ۴) ۰/۴

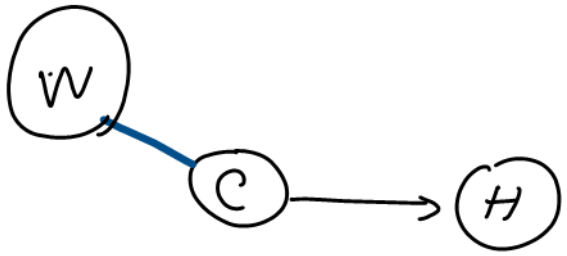


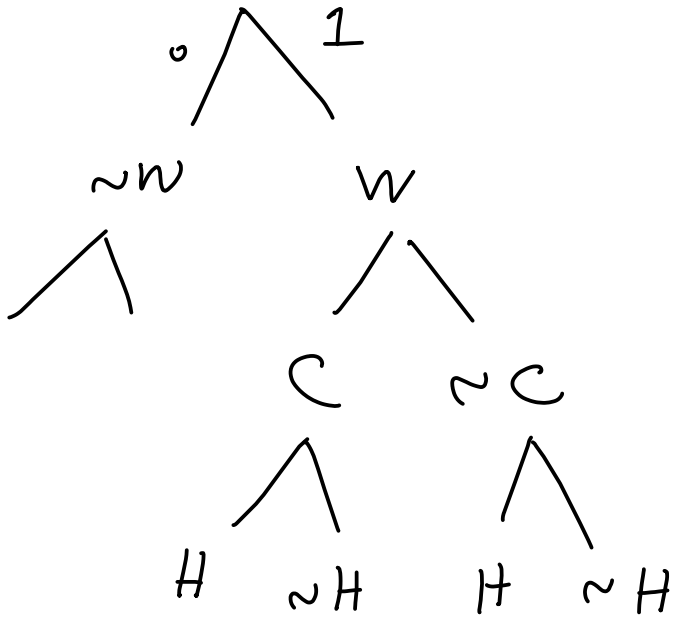
$$P(H \mid \sim C) = ./1$$

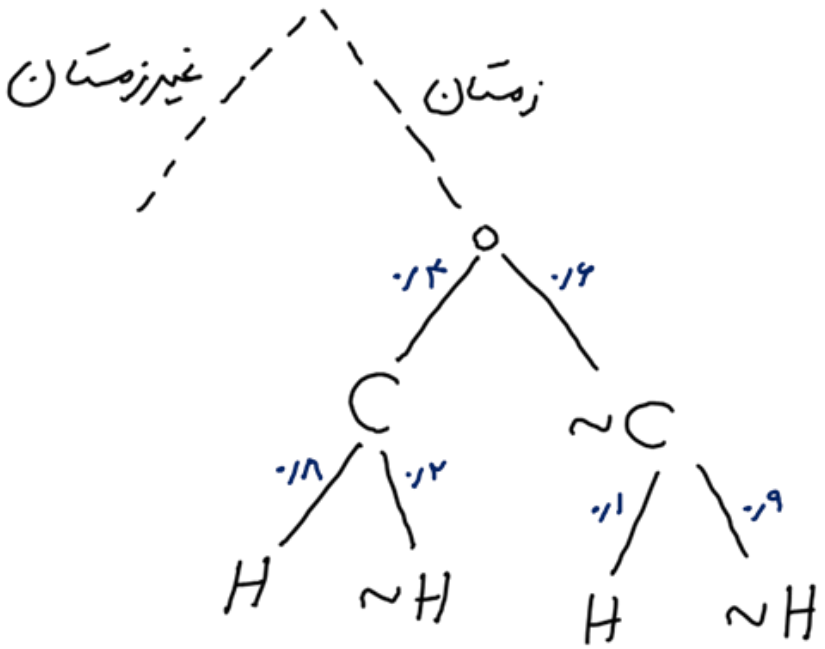
$$P(H \mid C) = ./\wedge$$

$$P(C \mid W) = ./۴$$

شرط در زمستان بودن $W=1$ با ثبت می‌شود که مربوط به C و H باشد







$$p(C \mid H) = \frac{(۰/۸)(۰/۴)}{(۰/۸)(۰/۴) + (۰/۶)(۰/۱)}$$

۵ـ در شهری از هر ۲۵ نفر، یک نفر سیگاری است و $\frac{۱}{۵}$ این افراد سیگاری نوجوان هستند. اگر نوجوانان ۱۰٪ افراد این شهر را تشکیل دهند، احتمال سیگاری شدن یک نوجوان در این شهر چقدر است؟ (مهندسی کامپیوتر ـ دولتی ۸۵)

$P(S|N)=?$

۱) $\frac{۱}{۱۲۵}$

۲) $\frac{۸}{۱۰۰}$

۳) $\frac{۱۰}{۱۰۰}$

۴) $\frac{۲۰}{۱۲۵}$

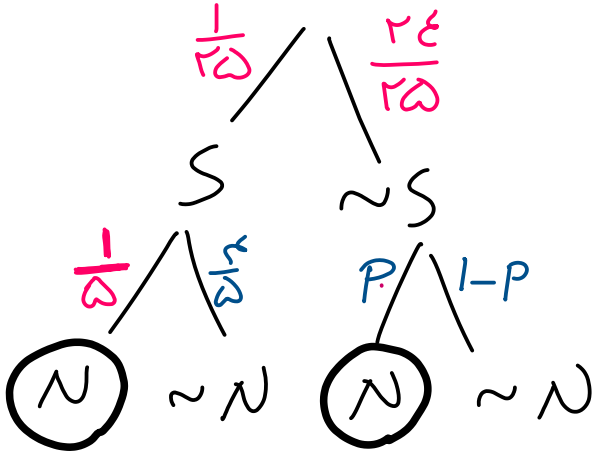


$P(N)=\frac{۱۰}{۱۰۰}$

$P(S)=\frac{۱}{۲۵}$

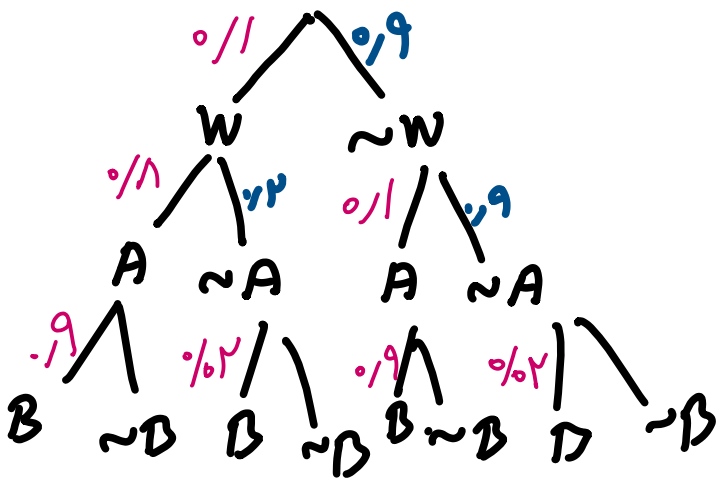
$P(N|S)=\frac{۱}{۵}$ ← ... (پس N وابسته به S نبود)

$P(S|N) = ?$



$$P(S|N) = \frac{P(S,N)}{P(N)} = \frac{\frac{1}{20}\frac{1}{5}}{0/10} = \frac{10}{125} = \frac{2}{25} = \frac{8}{100}$$

داره ~ 0



۱۱ـ فرض کنید رباتی دارید که برای حمل اشیاء سنگین استفاده می‌شود. اگر شی‌ای سنگین‌تر از آستانه تحمل ربات روی آن قرار دهید، به احتمال ۸۰٪ چراغ خطرآن روشن می‌شود. در غیر این صورت (اگر شیء سنگین‌تر از آستانه نباشد) فقط ۱۰٪ احتمال روشن شدن چراغ خطر هست. روشن شدن چراغ خطر در ۹۰٪ موارد منجر به خاموش شدن موتور ربات می‌شود. در ۲٪ موارد ممکن است بدون روشن شدن چراغ، موتور ربات خاموش شود. اگر فقط ۱۰٪ اشیاء قابل حمل سنگین‌تر از آستانه تحمل ربات باشند.احتمال اینکه بعد از گذاشتن شیء روی ربات، چراغ روشن نشود ولی موتور خاموش شود، چقدر است؟ (مهندسی کامپیوتر ـ دولتی ۹۱)

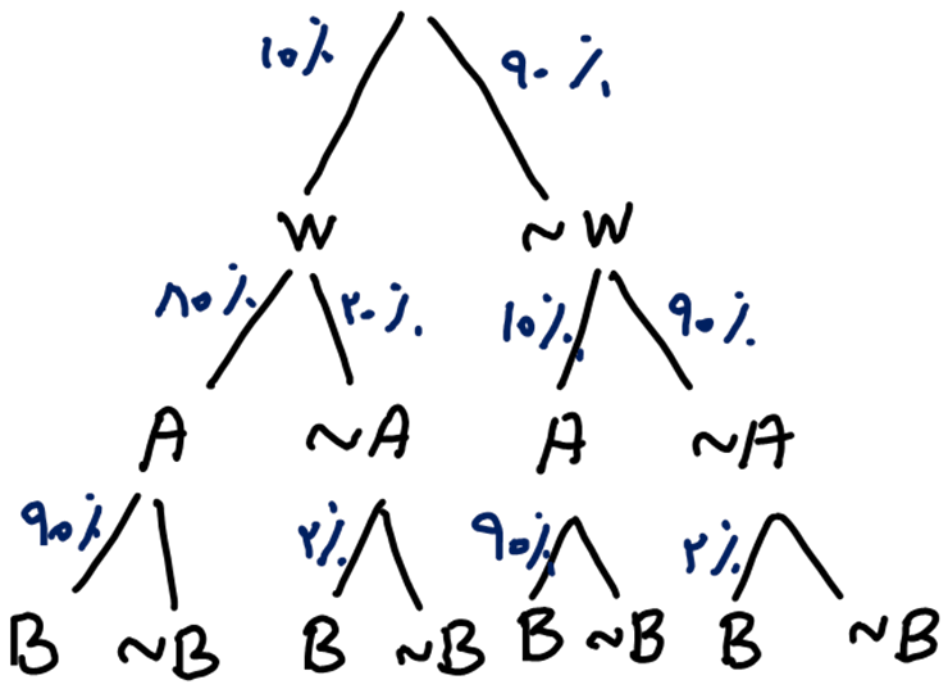
۱) ٪۰/۰۴ ۲) ٪۱/۶۶ ۳) ٪۲ ۴) ٪۰/۸۱

$$p(\sim A \wedge B) = \quad .1 \quad .2 \quad 0.02 \quad + \quad .9 \quad .9 \quad 0.02 \quad = \% \, 1.66$$

W : شیء سنگین‌تر از حد نقل است

A : چراغ روشن شود.

B : موتور خاموش شود.



$$P(Weather = sunny) = 0.6$$
$$P(Weather = rain) = 0.1$$
$$P(Weather = cloudy) = 0.29$$
$$P(Weather = snow) = 0.01\ ,$$

بردار احتمال:

$$\mathbf{P}(Weather) = \langle 0.6, 0.1, 0.29, 0.01 \rangle$$

برای یک متغیر بانری:

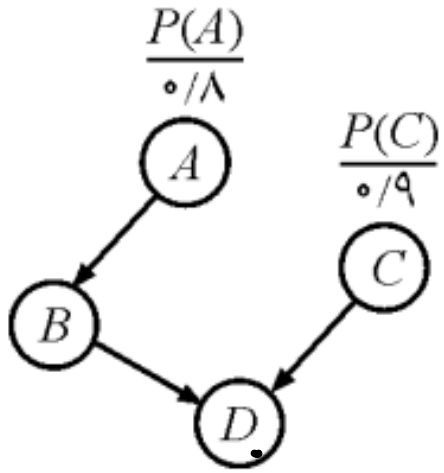
$$\big(P(A)\ ,\ P(\sim A) \big)$$

مثلاً

$$(\%۷\ ,\ \%۳)$$

۱۷ـ در شبکه بیزین (*Bayesian Network*) زیر احتمال $P(A, \neg B, D)$ چقدر است؟

(مکاترونیک و مهندسی IT ـ دولتی ۸۸)



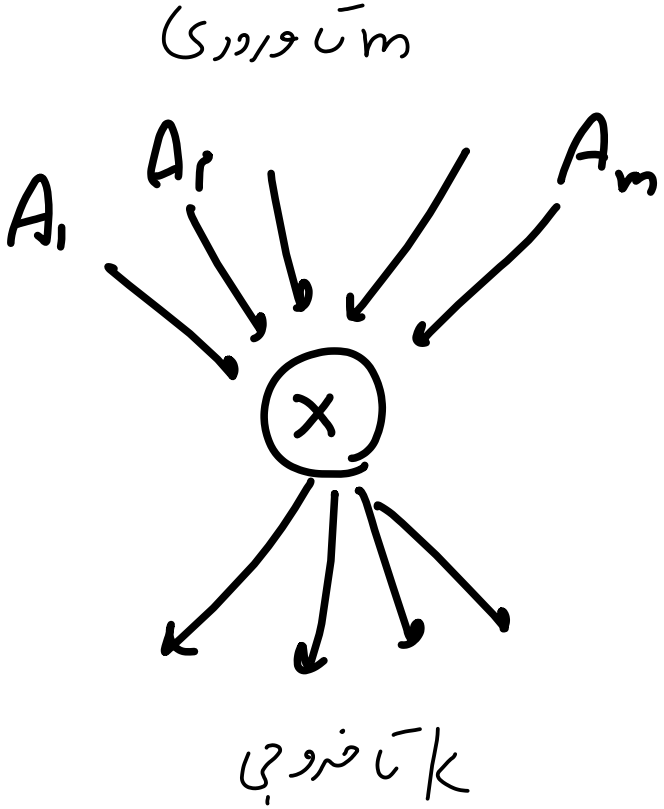
A	$P(B)$
T	۰/۸
F	۰/۲

B	C	$P(D)$
T	T	۰/۳
T	F	۰/۳
F	T	۰/۱
F	F	۰/۳

۱) ۰/۰۱۹۲ ۲) ۰/۰۶۴ ۳) ۰/۰۹۶ ۴) ۰/۴

ابتدا یک توضیح درباره جداول CPT (احتمال شرطی)

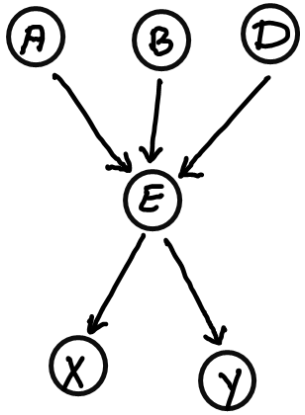
توضیح درباره جداول CPT (احتمال شرطی)



A_1 ..... A_m	$P(X \mid A_1 \dots A_m)$	$P(\neg X \mid \dots)$
t t t t	۰.۸	۰.۲
t f f t	۰.۴	۰.۶
⋮	⋮	⋮

2^m سطر لازم دارد. در هر سطر یک احتمال.

اگر هم $P(\neg X \mid \dots)$ را اضافه کنیم 2×2^m عدد دارد.



برای شبکه بیز باینری، هر متغیر ۲ مقدار ممکن دارد. True ، False .

اگر E دارای m ورودی و K خروجی باشد، جدول احتمالات شرطی آن دارای 2^m مقدار است. (برای حالات مختلف ورودی‌ها)

البته اگر احتمالات E به راهم با همین شرط‌ها به جدول اضافه کنیم $2 \times 2^m = 2^{m+1}$ عدد لازم می‌شود. اما این کار لازم نیست.

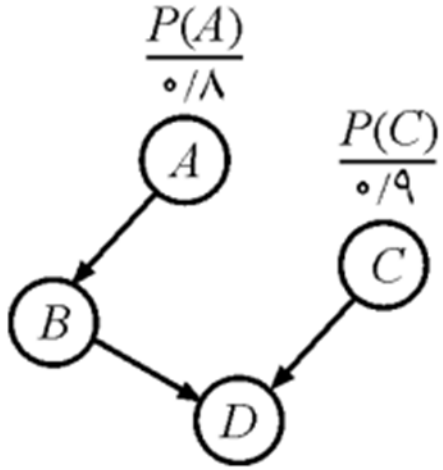
A	B	D	P(E|A,B,D)
T	T	T	۰/۱
T	T	F	۰/۳
T	F	T	۰/۲
	⋮		۰/۸
			۰/۹
T	T	T	⋮

2^m سطر

۱۷ـ در شبکه بیزین (*Bayesian Network*) زیر احتمال $P(A, \neg B, D)$ چقدر است؟

(مکاترونیک و مهندسی IT ـ دولتی ۸۸)

A	$P(B)$
T	۰/۸
F	۰/۲



B	C	$P(D)$
T	T	۰/۳
T	F	۰/۳
F	T	۰/۱
F	F	۰/۳

۱) ۰/۰۱۹۲ ✓ ۲) ۰/۰۶۴ ۳) ۰/۰۹۶ ۴) ۰/۴

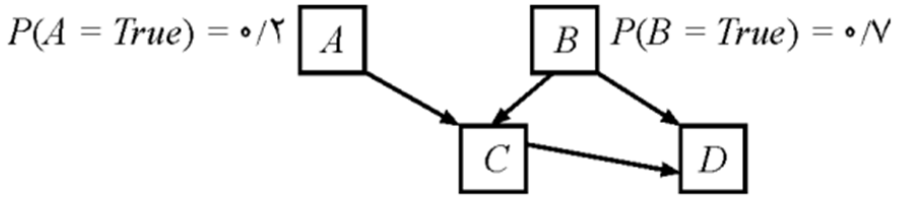
C دارد نه ۰ از قانون جمع استفاده کنیم:

$p(A, \sim B, D, C) = ۰/۸ \quad ۰/۹ \quad ۰/۲ \quad ۰/۱ = \frac{۸ \times ۱۸}{۱۰۰۰۰}$

$p(A, \sim B, D, \sim C) = ۰/۸ \quad ۰/۱ \quad ۰/۲ \quad ۰/۳ = \frac{۸ \times ۶}{۱۰۰۰۰}$

} جمع $\rightarrow \frac{۸ \times ۲۴}{۱۰۰۰۰} = \frac{۱۹۲}{۱۰۰۰۰}$

۱۰ـ شبکه بیزین زیر و جدول‌های پیوست را که در آن متغیرهای A تا D بولی هستند را در نظر بگیرید. احتمال اینکه C و D هر دو درست و A نادرست باشد، چقدر است؟ (کامپیوتر ـ دولتی ۸۹)



A	B	P(C=True | A,B)
False	False	۰/۱
False	True	۰/۵
True	False	۰/۴
True	True	۰/۹

B	C	P(D=True | B,C)
False	False	۰/۸
False	True	۰/۶
True	False	۰/۳
True	True	۰/۱

۱) ۰/۰۴۲۴ ۲) ۰/۰۲۸۰ ۳) ۰/۳۳۶ ۴) ۰/۱۴

احتمال توام $P(\sim A, D, C)$ را می‌خواهیم. روی B و $\sim B$ از قانون جمع استفاده می‌کنیم:

$P(\sim A, D, C, B) = ۰/۸ \times ۰/۷ \times ۰/۵ \times ۰/۱ = \frac{۸ \times ۴۵}{۱۰۰۰۰}$

$P(\sim A, D, C, \sim B) = ۰/۸ \times ۰/۳ \times ۰/۱ \times ۰/۶ = \frac{۸ \times ۱۸}{۱۰۰۰۰}$

$\frac{۸ \times ۴۵}{۱۰۰۰۰} + \frac{۸ \times ۵۴}{۱۰۰۰۰} = \frac{۴۲۴}{۱۰۰۰۰}$

توزیع حاشیه اگر جدول احتمال توام W, A را داشته باشیم:

A \ W	(۱) sunny	(۲) rain	(۳) cloudy	(۴) snow
(۱) true	۰/۰۴	۰/۰۲	۰/۱۲	۰/۲۲
(۰) false	۰/۱۸	۰/۰۸	۰/۲۴	۰/۱۰

می‌توان توزیع احتمال هرکدام را محاسبه کرد:

$$p(W)=\sum_{a} p(W,A=a)$$

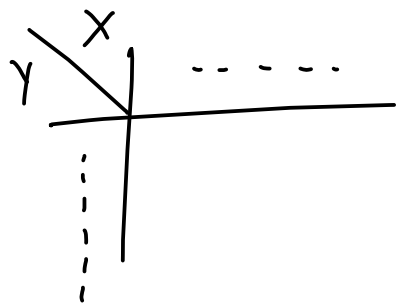
$$p(A)=\sum_{w} p(W=w,a)$$

توزیع W:

W	(۱) sunny	(۲) rain	(۳) cloudy	(۴) snow
$p(W=w)$	۰/۲۲	۰/۱۰	۰/۳۶	۰/۳۲

توزیع A:

a	$p(A=a)$
(۰) false	۰/۴۰
(۱) true	۰/۶۰



$$p(X=x) = \sum_{y=1}^{m} p(X=x, Y=y)$$

	A		$\sim A$	
	B	$\sim B$	B	$\sim B$
D	۰/۰۸	۰/۰۷	۰/۴۱	۰/۰۴
$\sim D$	۰/۰۹	۰/۰۱	۰/۲۷	۰/۰۳

تمرین

$$P(D \mid A \vee B) = ?$$

$$P(D \mid A \vee B) = \frac{\text{مجموع احتمالات توامی که } D \wedge (A \vee B) \text{ رخ داده}}{\text{مجموع همه احتمالات توام که در آنها } A \text{ یا } B \text{ رخ داده}} = \frac{۰/۰۸ + ۰/۰۷ + ۰/۴۱}{۰/۰۸ + ۰/۰۷ + ۰/۴۱ + ۰/۰۱ + ۰/۰۹ + ۰/۲۷}$$

$D \wedge (A \vee B)$

	A		$\sim A$	
	B	$\sim B$	B	$\sim B$
D	۰/۰۸	۰/۰۷	۰/۴۱	۰/۰۴
$\sim D$	۰/۰۹	۰/۰۱	۰/۲۷	۰/۰۳

$A \vee B$

$P(\sim D)$ $P(D)$

4 3

توضیح نرمال سازی

$$(P(D), P(\sim D)) \sim \left(\frac{3}{7}, \frac{4}{7}\right)$$

مثال دیگر:

$$(P(A), P(\sim A)) \propto (\%6, \%8) \xrightarrow{\text{نرمال}} \left(\frac{\%6}{1,4}, \frac{\%8}{1,4}\right)$$

نرمالسازی توزیع شرطی

	A		$\sim A$	
	B	$\sim B$	B	$\sim B$
D	۰/۰۸	۰/۰۷	۰/۴۱	۰/۰۴
$\sim D$	۰/۰۹	۰/۰۱	۰/۲۷	۰/۰۳

می‌خواهیم توزیع احتمال $D|A$ را تعیین کنیم.

$$(p(D|A), p(\sim D|A)) = \left(\frac{p(D,A)}{p(A)}, \frac{p(\sim D,A)}{p(A)}\right)$$

می‌توان ضریب ثابت $\frac{۱}{p(A)}$ را محاسبه نکرد در عوض در پایان نرمال‌سازی انجام داد:

$$p(D|A) = p(D,A,B) + p(D,A,\sim B) = ۰/۰۸ + ۰/۰۷ = ۰/۱۵$$
$$p(\sim D,A) = p(\sim D,A,B) + p(\sim D,A,\sim B) = ۰/۰۹ + ۰/۰۱ = ۰/۱۰$$

نرمال‌سازی

$$(p(D|A), p(\sim D|A)) = \left(\frac{۰/۱۵}{۰/۲۵}, \frac{۰/۱۰}{۰/۲۵}\right) = (۰/۶, ۰/۴)$$

در نتیجه:

$$\left(P(D|A),\ P(\sim D|A)\right) \propto \left(P(D,A),\ P(\sim D,A)\right)$$

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

جلسه ۲۲ فصل ۱۳ و ۱۴

وقتی x و y مستقل باشند:

$$x \perp y \longrightarrow p(x,y) = p(x)\,p(y)$$

$$p(x|y) = p(x)$$

استقلال مشروط

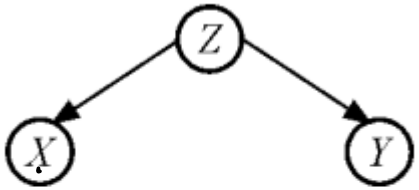
الف: زنجیر ساده



$(X \sim Y), (X \perp Y \mid Z)$

وابسته
به شرط Z مستقل می‌شوند.

ب: والد مشترک

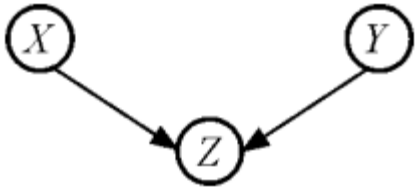


$(X \sim Y)$ وابسته‌اند

$(X \perp Y \mid Z)$

به شرط Z مستقل می‌شوند

ج: فرزند مشترک



مستقل‌اند: $(X \perp Y)$

$(X \sim Y \mid Z)$

به شرط Z،
وابسته می‌شوند

تمیز

① زنجیره ای :



* A و C وابسته اند

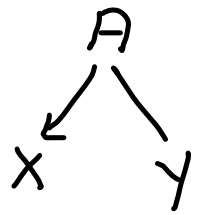
* با دانستن شواهد درباره‌ی B، A و C مستقل می‌شوند

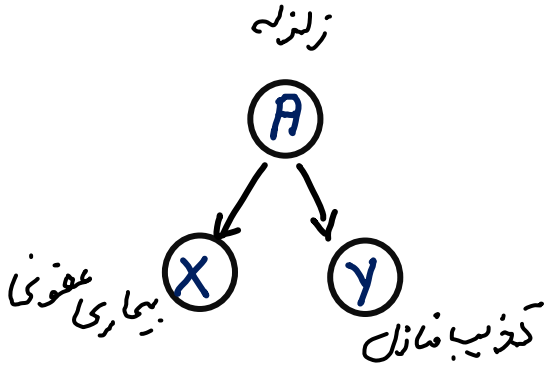
$$A \perp C \mid B$$

* دس دانستن شواهد، (مقدار B) رابطه‌ی C و A را مسدود می‌کند و باعث استقلال آنها از هم می‌شود.

تمیی

② والد مشترک:





* X و Y وابسته اند

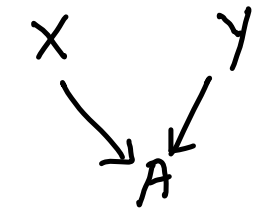
* با دانستن شواهد درباره‌ی A ، X و Y مستقل می‌شوند

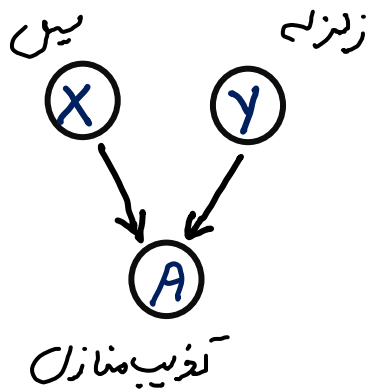
$$X \perp Y \mid A$$

* دس دانستن شواهد، (مقدار A) رابطه‌ی X و Y را مسدود می‌کند و باعث استقلال آنها می‌شود.

تمرین

③ فرزند مشترک:





* X و Y مستقل هستند. $X \perp Y$

* با دانستن شواهد درباره‌ی A، X و Y وابسته می‌شوند

* بی دانستن شواهد، (مقدار A) رابطه‌ی X و Y را برقرار می‌کند.

مثال



در شبکه‌ی مقابل:

$$p(x,y) \neq p(x)\,p(y)$$

زیرا x و y وابسته‌اند.

اما به شرط دانستن z مستقل می‌شوند:

$$x \perp y \mid z$$

پس به شرط z داریم:

$$p(x,y \mid z) = p(x \mid z)\; p(y \mid z)$$

$$p(x \mid y,z) = p(x \mid z)$$

مثال

$$x \longrightarrow z \longrightarrow y \longrightarrow w \longrightarrow v$$

در این زنجیر: همه با هم وابسته‌اند:

$x \sim y$

$z \sim v$

$\vdots$

اما اگر تواهری داشته باشیم:

$$x \longrightarrow z \longrightarrow y \longrightarrow \underset{w=1}{w} \longrightarrow v$$

با شرط دانستن w، x و z مستقل‌اند

$$p(x \mid w, z) = p(x \mid w)$$

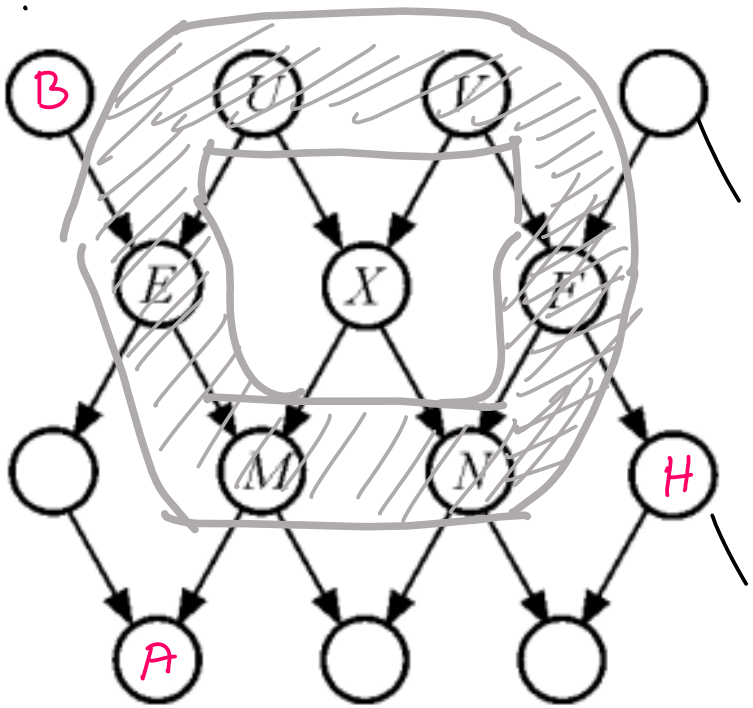
با شرط $z=0$، x و w مستقل‌اند:

$$x \longrightarrow \underset{z=0}{z} \longrightarrow y \longrightarrow w \longrightarrow v$$

$$p(x \mid z, w) = p(x \mid z)$$

پوشش مارکوفیِ X یعنی:

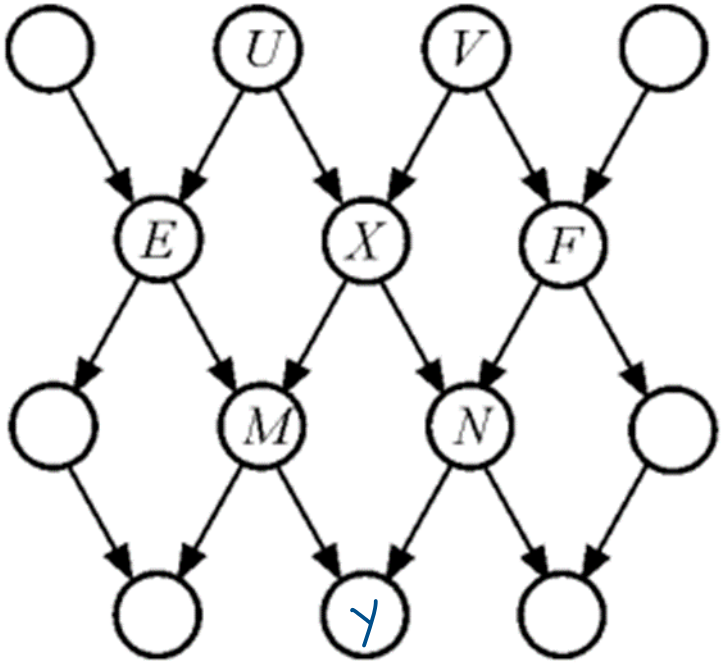
- والدینِ X
- فرزندانِ X
- دیگرِ والدینِ فرزندانِ X



با شرط داشتنِ پوششِ مارکوفیِ X، این گره از سایرین مستقل می‌شود.

مثلاً

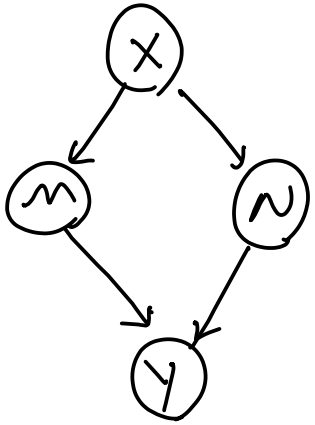
پوشش مارکوفی X	$(X \perp A)$
پوشش مارکوفی X	$(X \perp H)$
پوشش مارکوفی X	$(X \perp B)$



آیا درست است؟

$$p(N, M \mid X) = p(N \mid X)\, p(M \mid X)$$

بله، True به شرط دانستن X، متغیرهای N و M مستقل‌اند.



و الله را می‌دانیم. (شرط X). درمورد Y هم درحالت نداشتنِ Y، M و N مستقل‌اند.

۷۷- در نمونه‌برداری Gibbs از متغیرهای تصادفی دودویی در مدل شبکه بیزی زیر، متغیرهای $A=0$، $B=1$ و $D=0$ تا اینجای کار نمونه‌برداری شده‌اند. در لحظه بعد که متغیر C قرار است نمونه‌برداری شود، به چه احتمالی مقدار آن صفر خواهد شد؟

$$A \rightarrow B \rightarrow C \rightarrow D$$

$P(A=0)=0/5$ $P(C=0|B=1)=0/1$ $P(D=0|C=1)=0/1$

$P(B=0|A=0)=0/8$ $P(C=0|B=0)=0/8$ $P(D=0|C=0)=0/8$

$P(B=0|A=1)=0/1$

۱) $\frac{9}{17}$ ۲) $\frac{8}{9}$

۳) $\frac{8}{17}$ ۴) $\frac{1}{9}$

مهندسی کامپیوتر ۱۴۰۱

$P(C=0 \mid A=0, B=1, D=0)$

هدف ← ↓ شواهد

$A \rightarrow B \rightarrow C \rightarrow D$

توضیحات اولیه درباره‌ی اطلاعات داده‌شده:

$p(A=0) = \frac{1}{2}$

A	B=0	B=1
0	0/8	0.2
1	0/1	0.9

B	C=0	C=1
0	0/8	0.2
1	0/1	0.9

C	D=0	D=1
0	0/8	0.2
1	0/1	0.9

احتمال توام با توجه به شکل:

$$p(A,B,C,D) = p(A)\,p(B|A)\,p(C|B)\,p(D|C)$$

حل مسئله

$$P(C=0 \mid A=0, B=1, D=0)$$

هدف ← $C=0$

مشاهده ← $A=0, B=1, D=0$

در این مثال متغیر پنهان نداریم همه متغیرها یا مشاهده هستند یا هدف

برای یافتن پاسخ استنتاج، باید احتمالات توام را با توجه به شکل شبکه بنویسیم:

$$P(A,B,C,D) = P(A)\,P(B|A)\,P(C|B)\,P(D|B)$$

حالا مقدیر مشاهده را قرار دهیم.

اما برای هدف (و برای متغیرهای پنهان) هر دو حالت 0 و 1 را در نظر بگیریم.

در نهایت :

$$P(C=0 \mid A=0,\ B=1,\ D=0) = \frac{\text{مجموع احتمالات توامی که در آنها } C=0 \text{ است.}}{\text{مجموع همه‌ی احتمالات توام به دست آمده.}}$$

$$P(A,B,C,D) = P(\overset{0}{A})\, P(\overset{1}{B}|\overset{0}{A})\, P(C|\overset{1}{B})\, P(\overset{0}{D}|C)$$

$C=0$ $\quad P(A=0, B=1, C=0, D=0) = ۰/۷ \quad ۰/۲ \quad ۰/۱ \quad ۰/۸ = ۰/۰۰۸۰$

$C=1$ $\quad P(A=0, B=1, C=1, D=0) = ۰/۷ \quad ۰/۲ \quad ۰/۹ \quad ۰/۱ = ۰/۰۰۹۰$

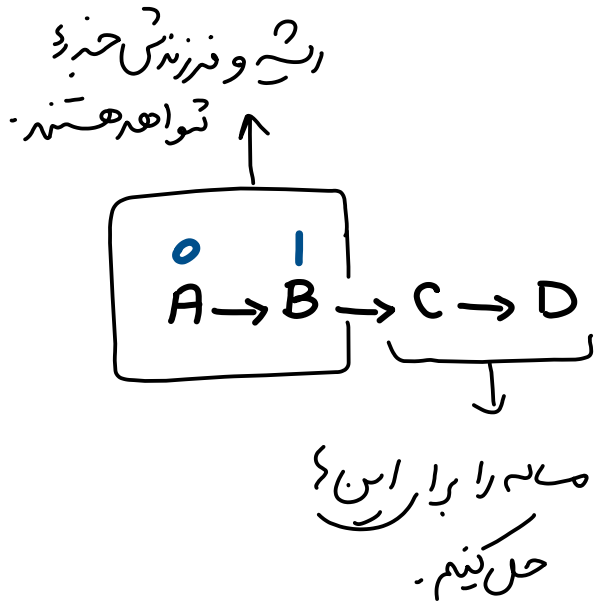
$$\text{جواب} = \frac{\text{حالتی که } C=0 \text{ است}}{\text{مجموع دو}} = \frac{۸۰}{۸۰+۹۰} = \frac{۸}{۱۷}$$

در مساله قبل استثنائاً ، می‌توانیم A و B را وارد محاسبات نکنیم.

نکته

در مساله قبل چون گره‌های والد (A و B) جزء نخواهد بود نمی‌توانیم

مساله را فقط برای متغیرهای C و D حل کنیم.



$$p(C=0 \mid D=0, \underbrace{A=0, B=1}_{\text{چشم‌پوشی}})$$

$$C=0 \quad p(C=0|B=1)\,p(D=0|C=0) = (0.1)(0.11)$$

$$C=1 \quad p(C=1|B=1)\,p(D=0|C=1) = (0.9)(0.1)$$

$$\text{جواب} = \frac{0.11}{0.09+0.11} = \frac{11}{20}$$

۷۹- فرض کنید چهار متغیر تصادفی دودویی A، B، C و D داریم که با شبکه بیزی زیر مدل شده‌اند. مقدار $P(A=1|B=0\ ,\ D=1)$ چقدر است؟

$A \rightarrow B$

$\downarrow \qquad \downarrow$

$C \rightarrow D$

$P(A=0)=0/5$ $\quad P(B=0|A=0)=0/8$ $\quad P(D=0|B=1\ ,\ C=0)=0/5$

$P(C=0|A=0)=0/1$ $\quad P(B=0|A=1)=0/2$ $\quad P(D=0|B=0\ ,\ C=1)=0/5$

$P(C=0|A=1)=0/8$ $\quad P(D=0|B=0\ ,\ C=0)=1$ $\quad P(D=0|B=1\ ,\ C=1)=0$

۱) $\frac{18}{100}$

۲) $\frac{1}{100}$

۳) $\frac{1}{19}$

۴) صفر

مهندسی کامپیوتر ۱۴۰۱

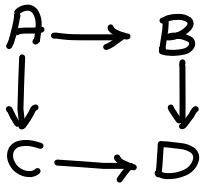
$$p(A=1 \mid B=0, D=1) = ?$$

هدف ← $A=1$

خواهه ← $B=0, D=1$

متغیر C متغیر پنهان است.

ابتدا با توجه به شکل شبکه احتمال توام را بنویسیم:



$$p(A,B,C,D) = p(A)\,p(B|A)\,p(C|A)\,p(D|B,C)$$

به جای خواهه $B=0$ و $D=1$ قرار دهیم. اما برای A و C همه‌ی مقادیر ممکن را در نظر بگیریم:

$$p(A)\,p(\overset{0}{B}|A)\,p(C|A)\,p(\overset{1}{D}|\overset{0}{B},C)$$

A=0	C=0	۰/۵	۰/۸	۰/۱	۰	⇝	۰
A=0	C=1	۰/۵	۰/۸	۰/۹	۰/۵	⇝	۵×۸×۹×۵
A=1	C=0	۰/۵	۰/۲	۰/۸	۰	⇝	۰
A=1	C=1	۰/۵	۰/۲	۰/۲	۰/۵	⇝	۵×۲×۲×۵

} حالت‌های A=0

} حالت‌های A=1

$$P(A=1 \mid B=0, D=1) = \frac{(5\times2\times2\times5)}{(5\times8\times9\times5)+(5\times2\times2\times5)} = \frac{1}{18+1} = \frac{1}{19}$$

$P(A=0)=0.5$

$P(C=0|A=0)=0.1$
$P(C=0|A=1)=0.8$

$P(B=0|A=0)=0.8$
$P(B=0|A=1)=0.2$

$P(D=0|B=0, C=0)=1$
$P(D=0|B=1, C=0)=0.5$
$P(D=0|B=0, C=1)=0.5$
$P(D=0|B=1, C=1)=0$

0.1 A → B 0
↓ ↓
0.1 C → D 1

$P(A=1|B=0, D=1)$

A=0 C=0
A=0 C=1

A=1 C=0
A=1 C=1

فرمول استنتاج به روش شمارش :

هدف	A	→	B
	↓		↓
متغیر پنهان	C	→	D

تواهد

تواهد

بایی مجموع احتمالات تمام را بیابیم که در آنها مقادیر تواهد ثابت و معلوم هستند اما مقادیر هدف و متغیر پنهان تغییر می کنند :

$$p(A|B,D) \propto \sum_{a} \sum_{c} p(A=a, B=b, C=c, D=d)$$

a: هدف — c: پنهان

(متناسب است)

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

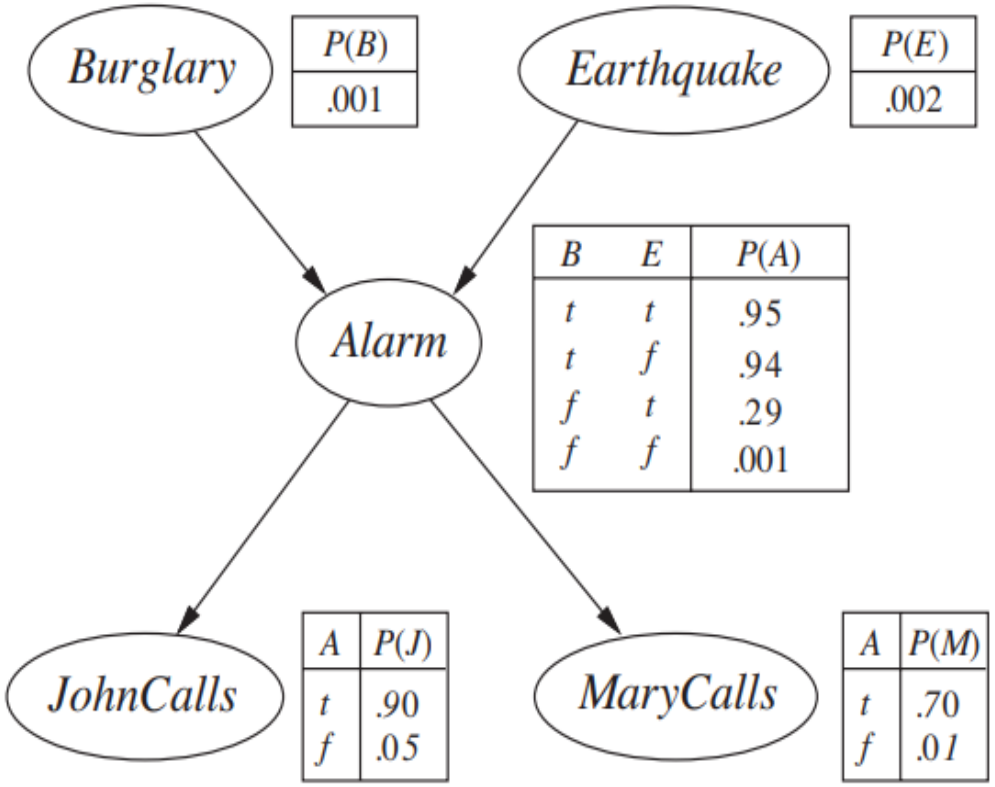
ابوالفضل گیلک
گروه بابان

جلسه ۲۳ مباحث منتخب فصل های ۱۴ تا ۱۷

شبکه بیز – زنجیر مارکوف

در شبکه بیز، روابط، روابط علت و معلول قطعی نیست، بلکه نشان دهنده تاثیر یک متغیر بر احتمال رخ دادن دیگری است.

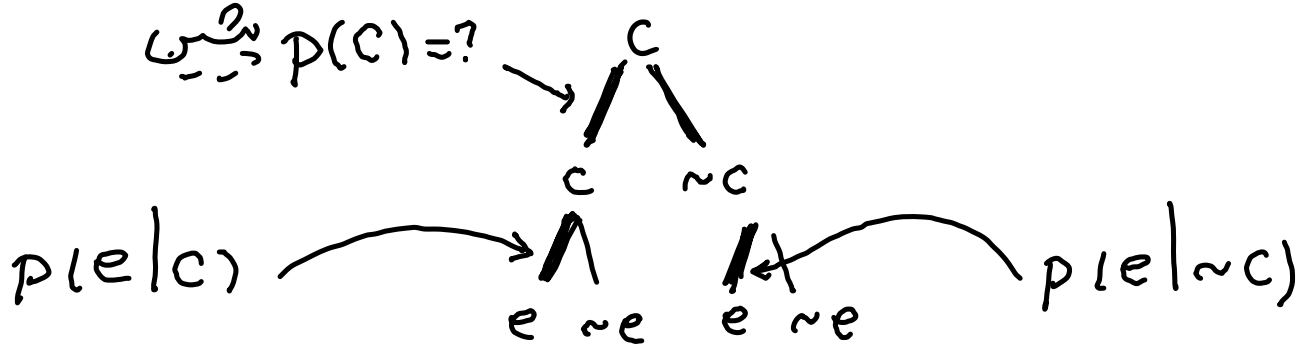
رابطه‌ی وابستگی و احتمال شرطی:



Applying Bayes' rule: The simple case

$$P(cause \mid effect) = \frac{P(effect \mid cause)P(cause)}{P(effect)}$$

احتمال پسین

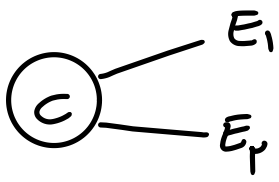


$$P(e) = P(C)P(e|C) + P(\sim C)P(e|\sim C)$$

Using Bayes' rule: Combining evidence

$$\mathbf{P}(Cause, Effect_1, \ldots, Effect_n) = \mathbf{P}(Cause) \prod_i \mathbf{P}(Effect_i \mid Cause) .$$

$$P(C, E_1, E_2) = P(C)\ P(E_1, E_2 \mid C)$$
$$= P(C)\ P(E_1 \mid C)\ P(E_2 \mid C)$$



: ordering

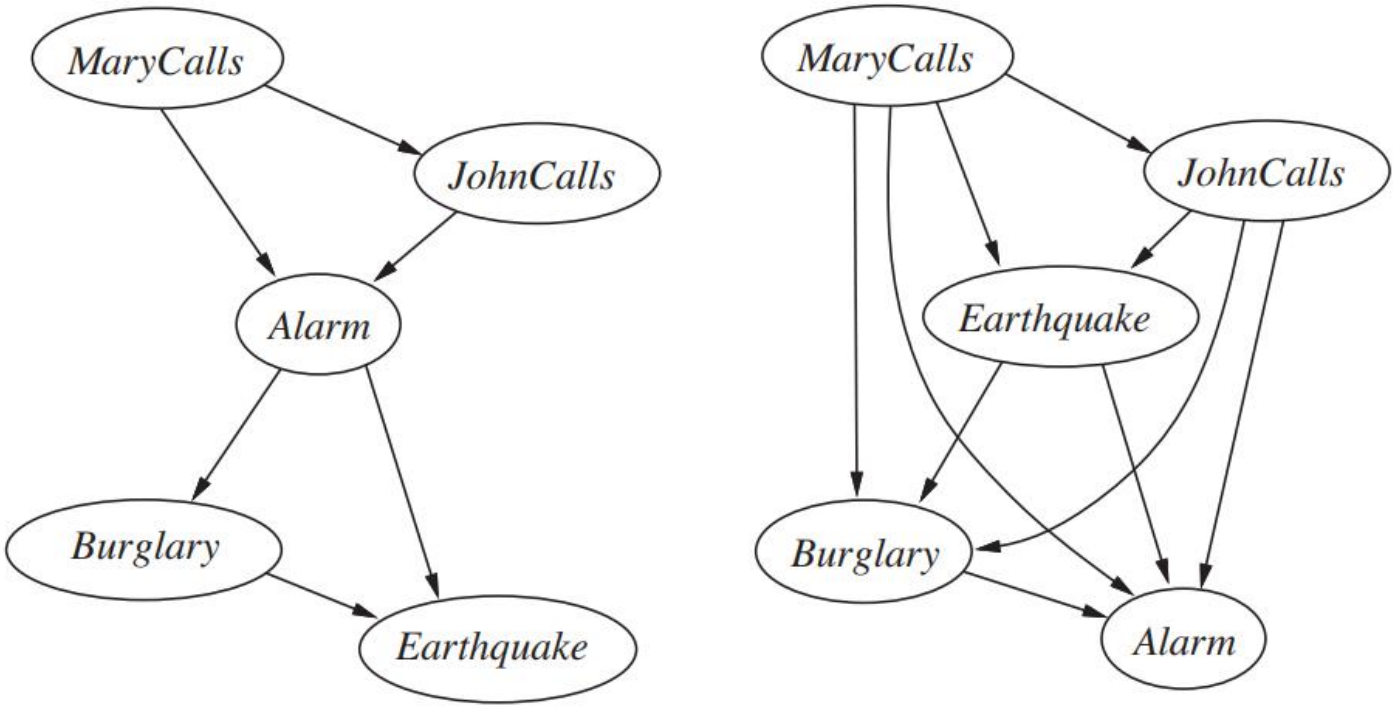
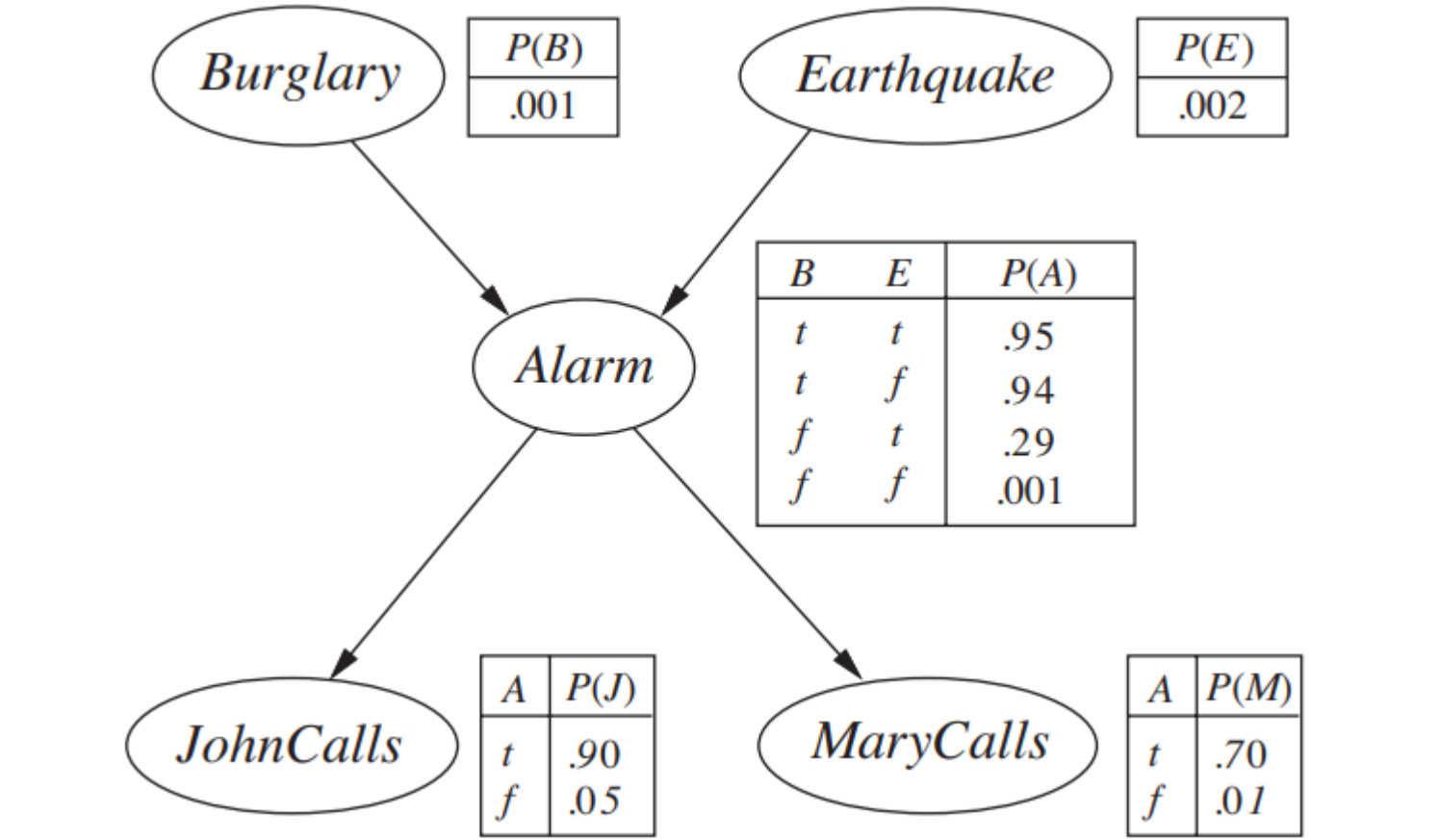


Figure 14.3 Network structure depends on order of introduction. In each network, we have introduced nodes in top-to-bottom order.



distinction between **causal** and **diagnostic**

Conditional independence relations in Bayesian networks

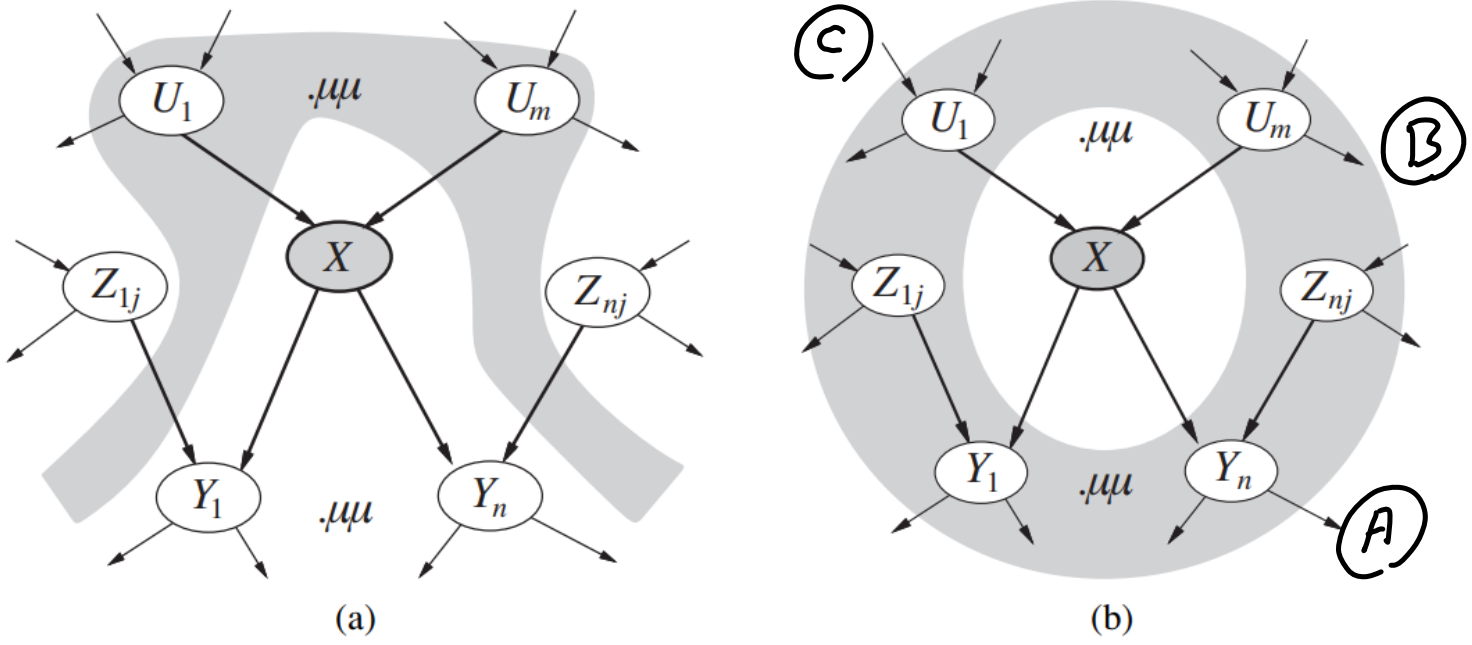
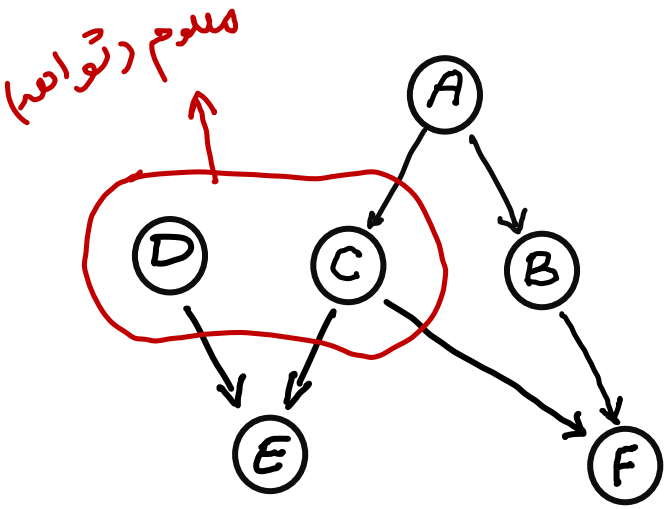


Figure 14.4 (a) A node X is conditionally independent of its non-descendants (e.g., the Z_{ij}s) given its parents (the U_is shown in the gray area). (b) A node X is conditionally independent of all other nodes in the network given its Markov blanket (the gray area).

$$p(x \mid y, e) = p(x \mid e)$$

$$p(x, y \mid e) = p(x \mid e)\, p(y \mid e)$$

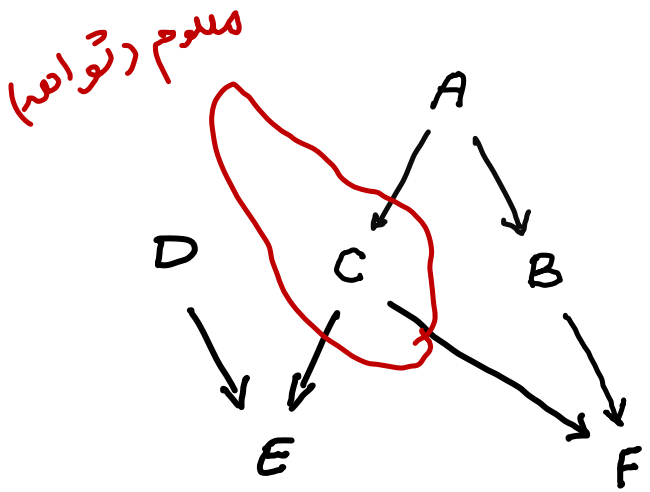


$$P(E \mid A, B, C, D, F) \overset{?}{=} P(E \mid D, C)$$

True

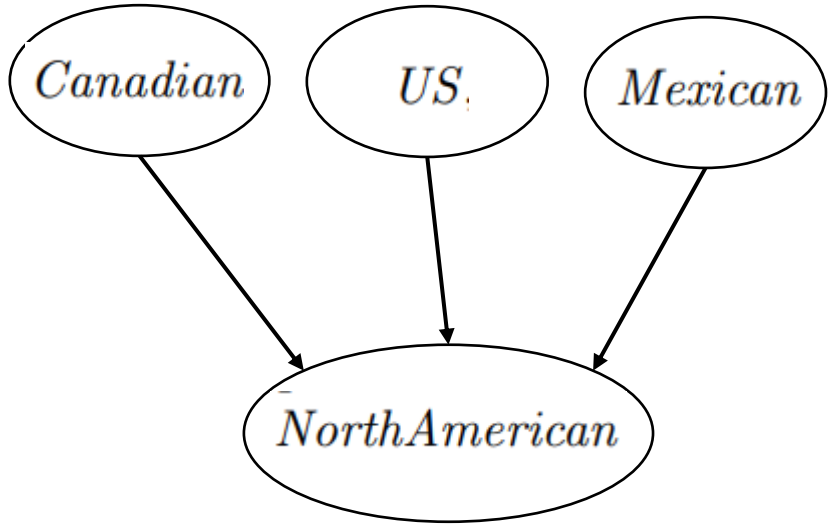
$$P(E, A \mid C) = P(E \mid C)\, P(A \mid C)$$

True

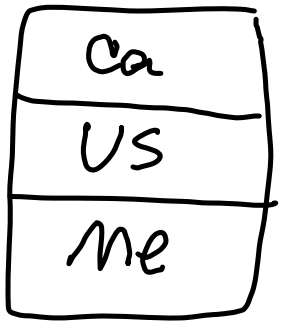


deterministic nodes.

$$Canadian \vee US \vee Mexican \Leftrightarrow \overline{NorthAmerican}$$



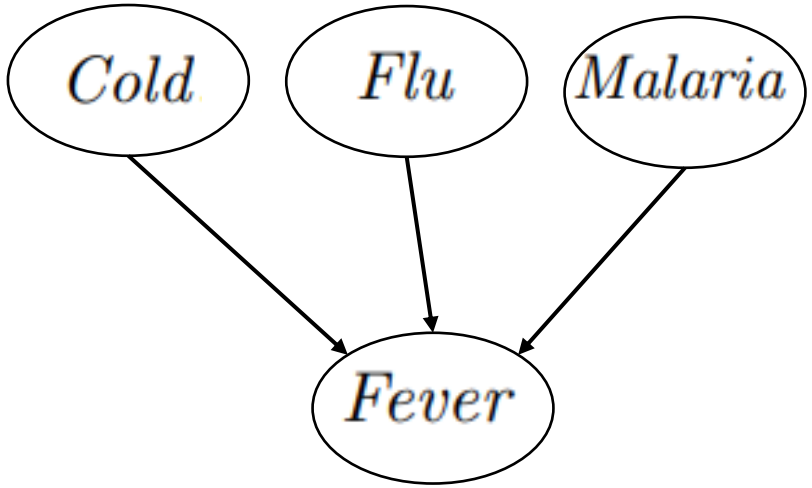
N.A



noisy-OR منفوشی OR

$$Cold \vee Flu \vee Malaria \Rightarrow Fever$$

نه از ۱۰۰



سرماخوردگی، آنفولانزا، مالاریا هرکدام می‌توانند

احتمال تب را افزایش دهند اما هیچکدام قطعی نیستند.

دو اشکال وجود دارد:

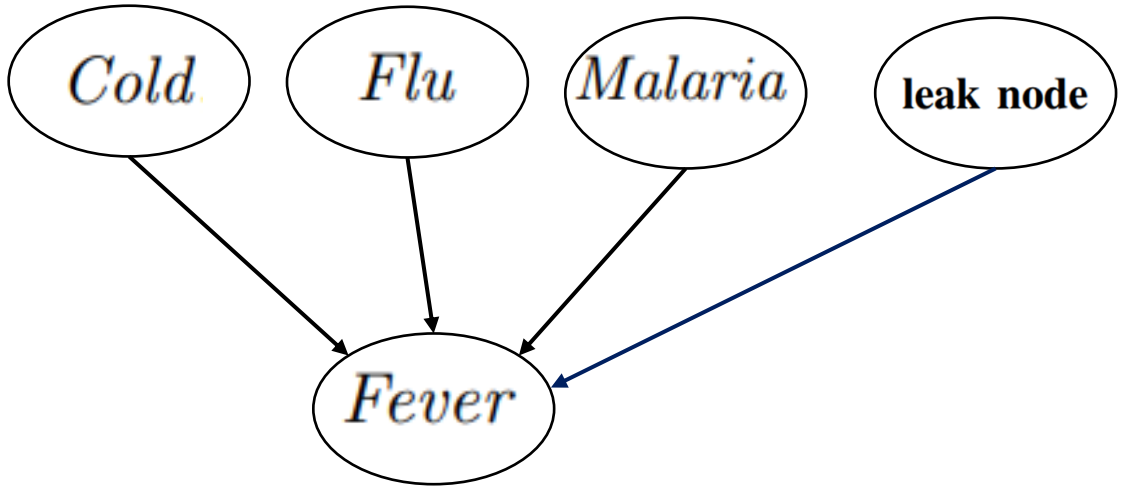
① ممکن است سرماخوردگی، باعث تب نشود (موثر نبودن در تب)

② " آنفولانزا

③ " مالاریا

اشکال دوم { ④ تب ممکن است به دلایل به جز این‌ها اتفاق بیفتد.

leak node



یک راه برای رفع مشکل
در نظر گرفتن گره نشتی
leak node
برای سایر عوامل تب است.

راه دیگر استفاده از این فرض برای حذف سایر عوامل است:

assumes that inhibition of each parent is independent of inhibition of any other parents:

پیشامدهای شکست هر بیماری در به وجود آمدن تب را مستقل فرض کنیم:

$$q_{\text{cold}} = P(\neg fever \mid cold, \neg flu, \neg malaria) = 0.6 \ ,$$
$$q_{\text{flu}} = P(\neg fever \mid \neg cold, flu, \neg malaria) = 0.2 \ ,$$
$$q_{\text{malaria}} = P(\neg fever \mid \neg cold, \neg flu, malaria) = 0.1 \ .$$

با ضرب احتمالات شکست، می‌توان احتمال عدم تب برای ۲ یا ۳ بیماری را یافت.

پس با متمم گرفتن احتمال تب به دست می‌آید:

$$P(x_i \mid parents(X_i)) = 1 - \prod_{\{j:X_j = true\}} q_j$$

Cold	*Flu*	*Malaria*	$P(Fever)$	$P(\neg Fever)$
F	F	F	0.0	1.0
F	F	T	0.9	**0.1** → q_{ma}
F	T	F	0.8	**0.2** → q_{Flu}
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	**0.6** → q_{Cold}
T	F	T	0.94	$0.06 = 0.6 \times 0.1$
T	T	F	0.88	$0.12 = 0.6 \times 0.2$
T	T	T	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

In general, noisy logical relationships in which a variable depends on k parents can be described using $O(k)$ parameters instead of $O(2^k)$ for the full conditional probability table.

با این روش به جای $O(2^k)$ فقط نیاز به $O(k)$ پارامتر داریم.

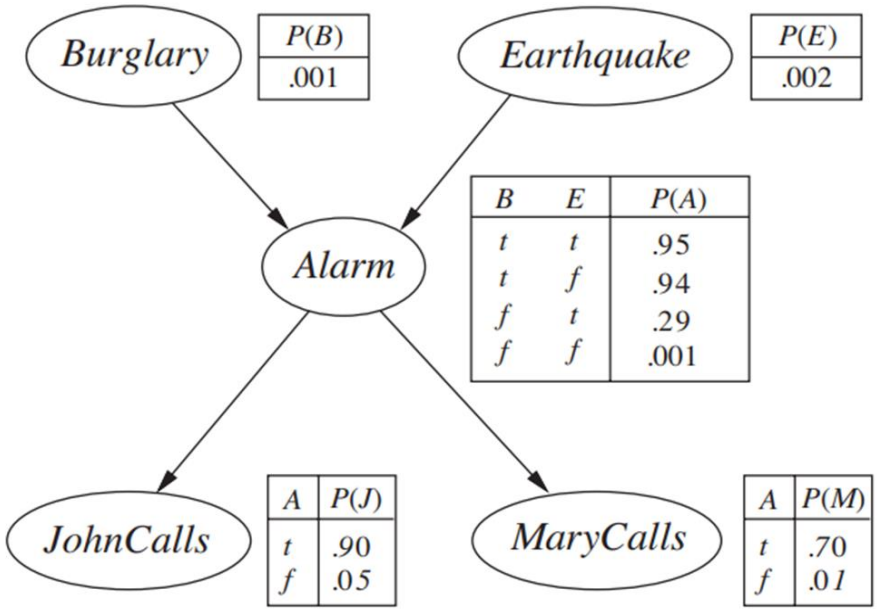
یادآوری تناسب و زمان سازی :

$$p(\beta \mid y, m) = \alpha \left(\cdot/\Delta\Lambda\circ \quad , \quad \cdot/\Delta\circ \right)$$

$$= \left(\frac{\Lambda\circ}{\Lambda\circ + 9\circ} \quad , \quad \frac{9\circ}{\Lambda\circ + 9\circ} \right)$$

Inference by enumeration

یادآوری استنتاج بیشمارش.



$$\mathbf{P}(B \mid j, m) = \alpha \mathbf{P}(B, j, m) = \alpha \sum_e \sum_a \mathbf{P}(B, j, m, e, a,)$$

$P(b)$ از هردو $\sum_e$ و $\sum_a$ خارج می‌شود.

$$P(b \mid j, m) = \alpha \sum_e \sum_a P(b)P(e)P(a \mid b, e)P(j \mid a)P(m \mid a) .$$

$P(e)$ از $\sum_a$ خارج می‌شود.

$$P(b \mid j, m) = \alpha P(b) \sum_e P(e) \sum_a P(a \mid b, e)P(j \mid a)P(m \mid a) .$$

خارج کردن جمله‌ی ثابت، تعداد محاسبات را کاهش می‌دهد.

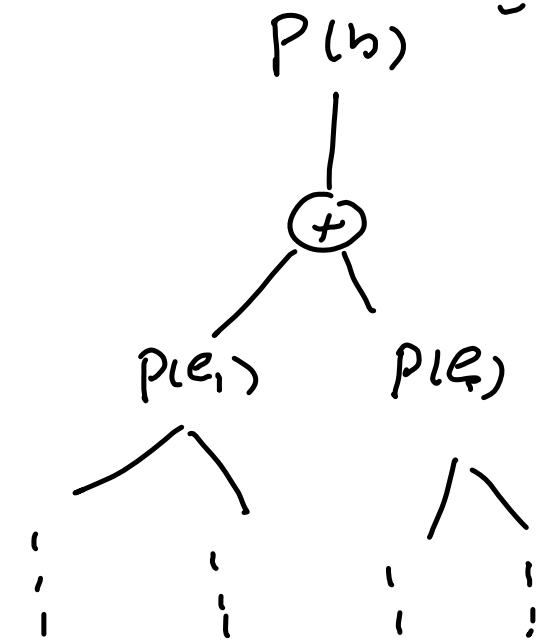
$$\sum_{x_i} p(b)\,p(x_i) = p(b)\,p(x_1) + p(b)\,p(x_2) + \cdots + p(b)\,p(x_n)$$

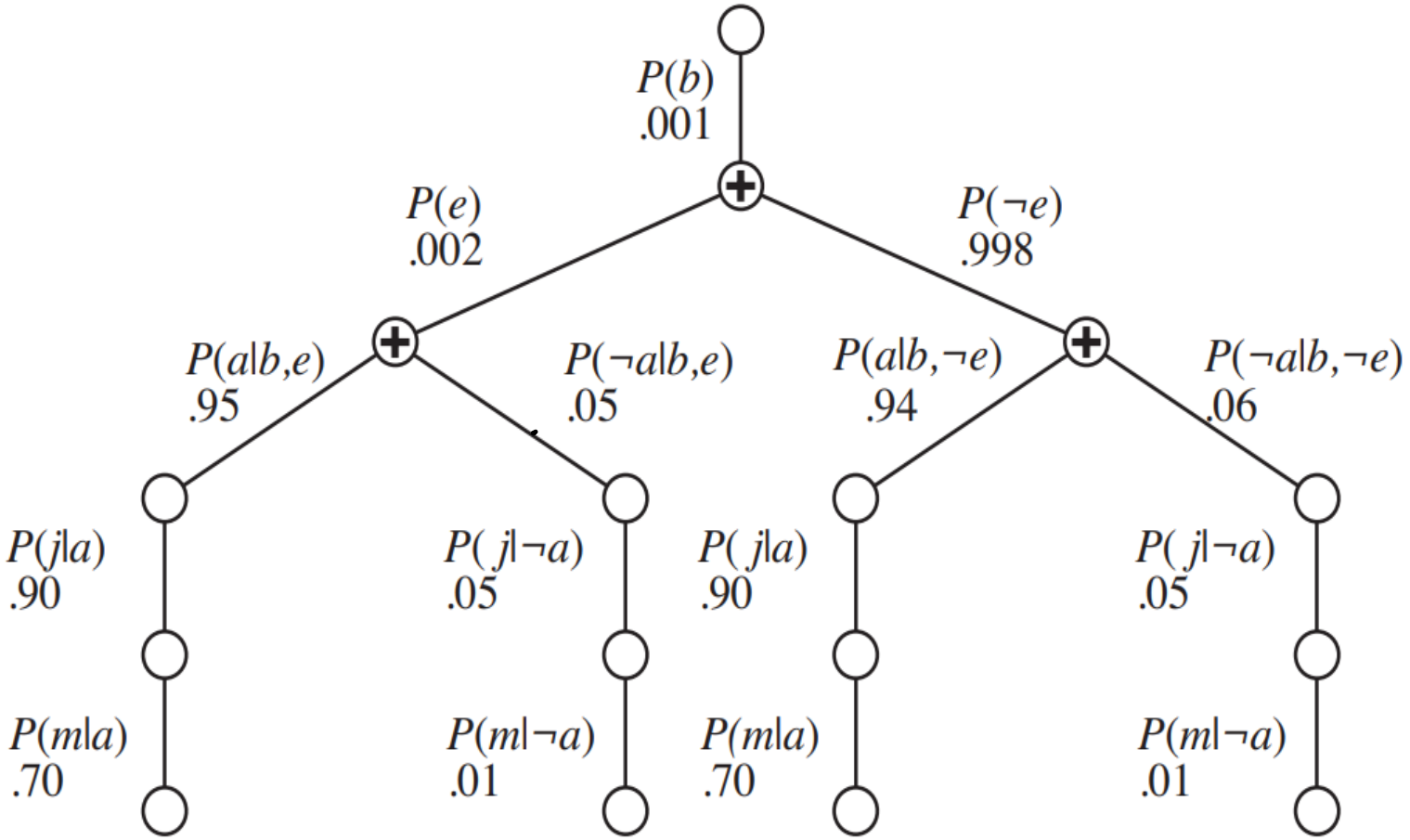
$$p(b)\sum_{x_i} p(x_i) = p(b)\cdot\Big(p(x_1)+\cdots+p(x_n)\Big)$$

$$P(b \mid j, m) = \alpha P(b) \sum_e P(e) \sum_a P(a \mid b, e) P(j \mid a) P(m \mid a) .$$

for a network with n Boolean variables is $O(n2^n)$ مرتبه زمانی:

درعمل از DFS در این درخت استفاده می‌کنیم. (فضای مورد نیاز O(n))




P(b)
.001
P(e)
.002
P(¬e)
.998
P(a|b,e)
.95
P(¬a|b,e)
.05
P(a|b,¬e)
.94
P(¬a|b,¬e)
.06
P(j|a)
.90
P(j|¬a)
.05
P(j|a)
.90
P(j|¬a)
.05
P(m|a)
.70
P(m|¬a)
.01
P(m|a)
.70
P(m|¬a)
.01

The variable elimination algorithm

الگوریتم حذف متغیر

$$\mathbf{P}(B \mid j, m) = \alpha \underbrace{\mathbf{P}(B)}_{\mathbf{f}_1(B)} \sum_e \underbrace{P(e)}_{\mathbf{f}_2(E)} \sum_a \underbrace{\mathbf{P}(a \mid B, e)}_{\mathbf{f}_3(A,B,E)} \underbrace{P(j \mid a)}_{\mathbf{f}_4(A)} \underbrace{P(m \mid a)}_{\mathbf{f}_5(A)} .$$

$$\mathbf{f}_4(A) = \begin{pmatrix} P(j \mid a) \\ P(j \mid \neg a) \end{pmatrix} = \begin{pmatrix} 0.90 \\ 0.05 \end{pmatrix} \qquad \mathbf{f}_5(A) = \begin{pmatrix} P(m \mid a) \\ P(m \mid \neg a) \end{pmatrix} = \begin{pmatrix} 0.70 \\ 0.01 \end{pmatrix} .$$

$\mathbf{f}_3(A, B, E)$ will be a $2 \times 2 \times 2$ matrix,

یک توضیح:

$$\sum_{z=0}^{1} \underbrace{x+y+2z}_{f(x,y,z)} = (x+y+0)+(x+y+2)$$

$$= \underbrace{2x+2y+2}_{f(x,y)}$$

- First, we sum out A from the product of $\mathbf{f}_3$, $\mathbf{f}_4$, and $\mathbf{f}_5$. This gives us a new 2×2 factor $\mathbf{f}_6(B, E)$ whose indices range over just B and E:

$$\begin{aligned}\mathbf{f}_6(B, E) &= \sum_a \mathbf{f}_3(A, B, E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A) \\ &= (\mathbf{f}_3(a, B, E) \times \mathbf{f}_4(a) \times \mathbf{f}_5(a)) + (\mathbf{f}_3(\neg a, B, E) \times \mathbf{f}_4(\neg a) \times \mathbf{f}_5(\neg a)) \,.\end{aligned}$$

  Now we are left with the expression

$$\mathbf{P}(B \,|\, j, m) = \alpha \, \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(E) \times \mathbf{f}_6(B, E) \,.$$

- Next, we sum out E from the product of $\mathbf{f}_2$ and $\mathbf{f}_6$:

$$\begin{aligned}\mathbf{f}_7(B) &= \sum_e \mathbf{f}_2(E) \times \mathbf{f}_6(B, E) \\ &= \mathbf{f}_2(e) \times \mathbf{f}_6(B, e) + \mathbf{f}_2(\neg e) \times \mathbf{f}_6(B, \neg e) \,.\end{aligned}$$

  This leaves the expression

$$\mathbf{P}(B \,|\, j, m) = \alpha \, \mathbf{f}_1(B) \times \mathbf{f}_7(B)$$

Operations on factors

اعمال روی فاکتورها:

$$\begin{aligned}\mathbf{f}_7(B) &= \sum_e \mathbf{f}_2(E) \times \mathbf{f}_6(B,E) \\ &= \mathbf{f}_2(e) \times \mathbf{f}_6(B,e) + \mathbf{f}_2(\neg e) \times \mathbf{f}_6(B,\neg e).\end{aligned}$$

ضرب نقطه‌ای

This leaves the expression

$$\mathbf{P}(B|j,m) = \alpha\, \mathbf{f}_1(B) \times \mathbf{f}_7(B)$$

نحوه محاسبه فرب‌ها:

$2^j 2^k = 2^{j+k}$ $\quad 2^k 2^\ell = 2^{k+\ell}$ $\quad 2^j \times 2^k \times 2^\ell = 2^{j+k+\ell}$

$$\mathbf{f}(X_1 \dots X_j, Y_1 \dots Y_k) \times \mathbf{g}(Y_1 \dots Y_k, Z_1, \dots Z_\ell) = \mathbf{h}(X_1 \dots X_j, Y_1 \dots Y_k, Z_1 \dots Z_\ell)$$

X	Y	$\mathbf{f}(X,Y)$
t	t	.3
t	f	.7
f	t	.9
f	f	.1

Y	Z	$\mathbf{g}(Y,Z)$
t	t	.2
t	f	.8
f	t	.6
f	f	.4

X	Y	Z	$\mathbf{h}(X,Y,Z)$
t	t	t	$.3 \times .2 = .06$
t	t	f	$.3 \times .8 = .24$
t	f	t	$.7 \times .6 = .42$
t	f	f	$.7 \times .4 = .28$
f	t	t	$.9 \times .2 = .18$
f	t	f	$.9 \times .8 = .72$
f	f	t	$.1 \times .6 = .06$
f	f	f	$.1 \times .4 = .04$

Illustrating pointwise multiplication: $\mathbf{f}(X,Y) \times \mathbf{g}(Y,Z) = \mathbf{h}(X,Y,Z)$.

• محاسبه‌ی جمع‌های ماتریسی:

$$\mathbf{h}_2(Y,Z) = \sum_x \mathbf{h}(X,Y,Z) = \mathbf{h}(x,Y,Z) + \mathbf{h}(\neg x,Y,Z)$$

$$= \begin{pmatrix} .06 & .24 \\ .42 & .28 \end{pmatrix} + \begin{pmatrix} .18 & .72 \\ .06 & .04 \end{pmatrix} = \begin{pmatrix} .24 & .96 \\ .48 & .32 \end{pmatrix}.$$

برای x: $h(x, Y, Z)$

برای $\neg x$: $h(\neg x, Y, Z)$

حاصل $h(Y, Z)$ است.

(قانون جمع)

The only trick is to notice that any factor that does *not* depend on the variable to be summed out can be moved outside the summation. For example, to sum out X from the product of $\mathbf{f}$ and $\mathbf{g}$, we can move g outside the summation:

جمله‌ای که نسبت به x ثابت است را خارج کنیم:

$$\sum_x \mathbf{f}(X,Y) \times \mathbf{g}(Y,Z) = \mathbf{g}(Y,Z) \times \sum_x \mathbf{f}(X,Y).$$

This is potentially much more efficient than computing the larger pointwise product $\mathbf{h}$ first and then summing X out from that.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

پیچیدگی استنتاج در شبکه بیز:

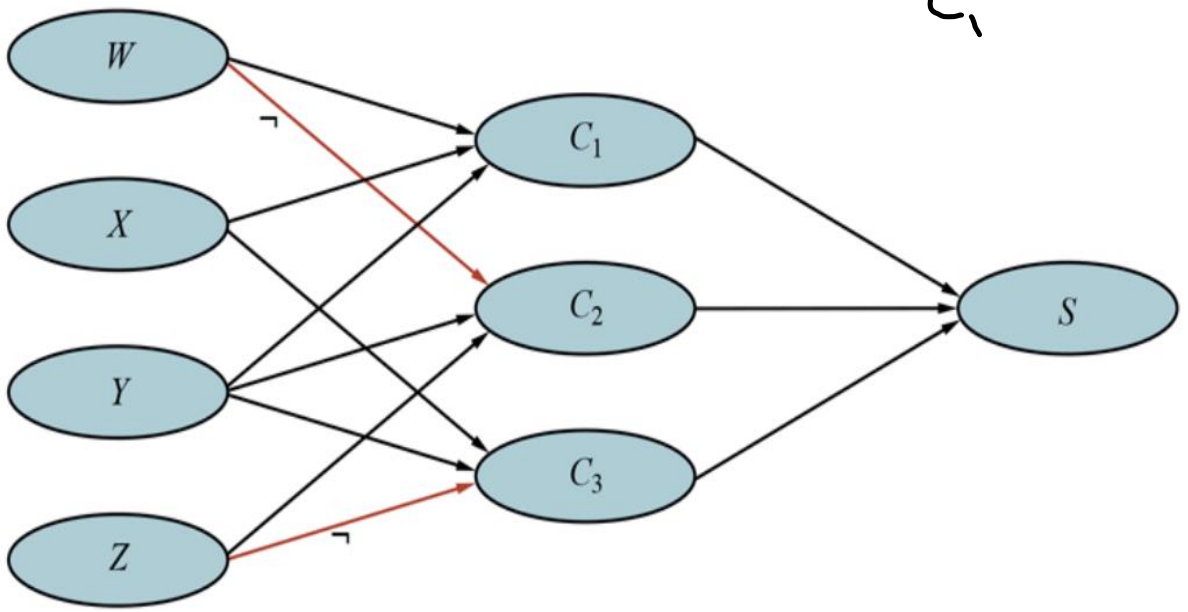
قضیه: استنتاج و رضایت‌پذیری در منطق گزاره‌ای که قابل تبدیل به استنتاج در بیز است NP-hard است.

از آنجایی که مسئله SAT، NP-hard است پس استنتاج بیز هم NP-hard است.

The complexity of exact inference

$S' = (W \vee X \vee Y) \wedge (\neg W \vee Y \vee Z) \wedge (X \vee Y \vee \neg Z)$

$C_1 \quad C_2 \quad C_3$



Bayes net encoding of the 3-CNP sentence $(W \vee X \vee Y) \wedge (\neg W \vee Y \vee Z) \wedge (X \vee Y \vee \neg Z)$.

$P(S' = true) = 1$ valid

$P(S = true) > 0$ for a satisfiable sentence.

$P(S = true) = 0$ for an unsatisfiable sentence:

we can use Bayes net inference to solve 3-SAT problems; ⟶ Bayes net inference is NP-hard.

x, y, z, w هر متغیر

با احتمال $\frac{1}{2}$ ممکن است

true یا false باشد.

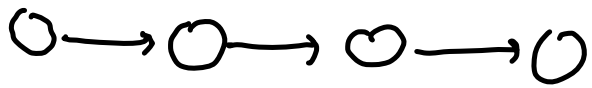
حالا در این شبکه:

اگر ارضاپذیر است حداقل

$P(S = true) > 0$

باشد.

singly connected networks or **polytrees**,



The time

and space complexity of exact inference in polytrees is linear in the size of the network.

برای شبکه‌ای همبند مرتبه‌یک (درخت چندگانه) پیچیدگی فضایی از مرتبه خطی برحسب اندازه شبکه است:

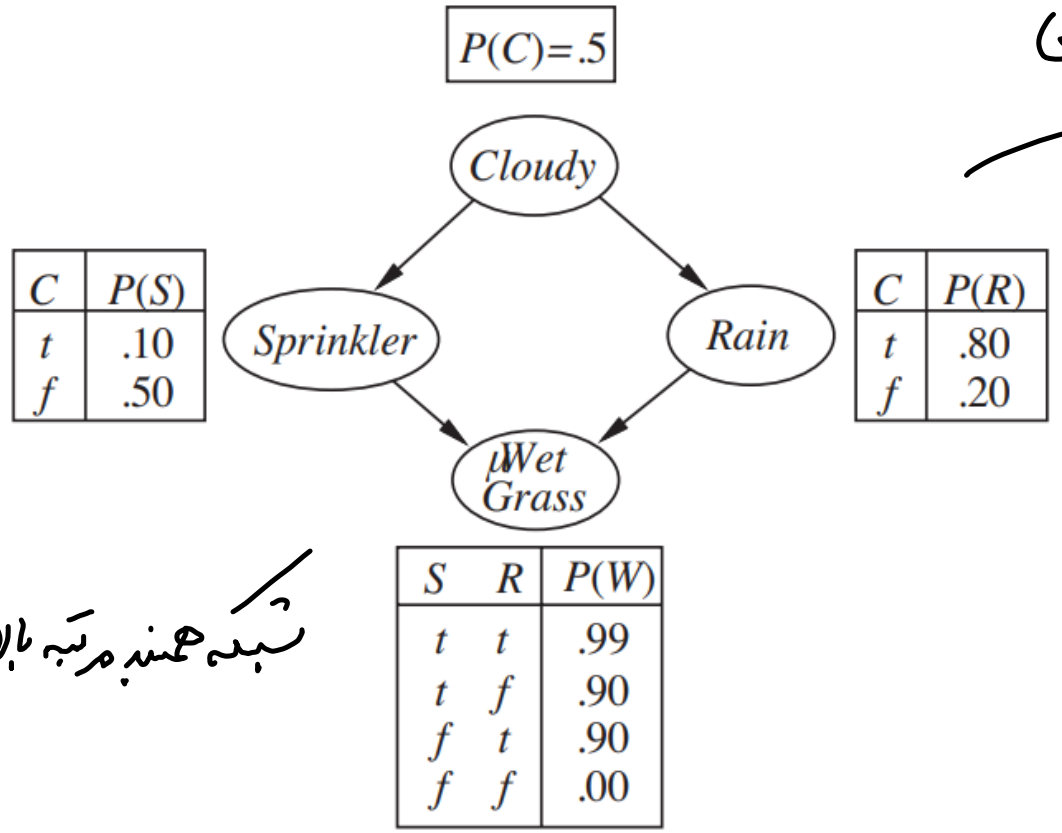
$O(n)$

multiply connected networks,

exponential time and space complexity in the worst case,

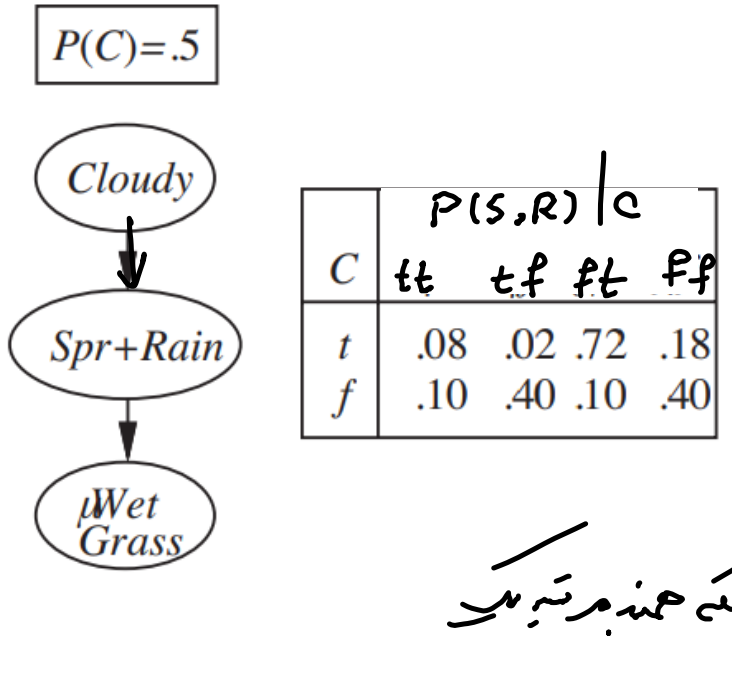
includes inference in propositional logic as a special case, inference in Bayesian networks is NP-hard.

اما در حالت کلی از مرتبه نمایی و NP-Hard است: $O(n2^n)$



خوشه بندی

گره خوشه را با S+R یا همان (S,R) نشان می دهیم.



شبکه همبند مرتبه بالاتر

شبکه همبند مرتبه یک

multiply connected networks,

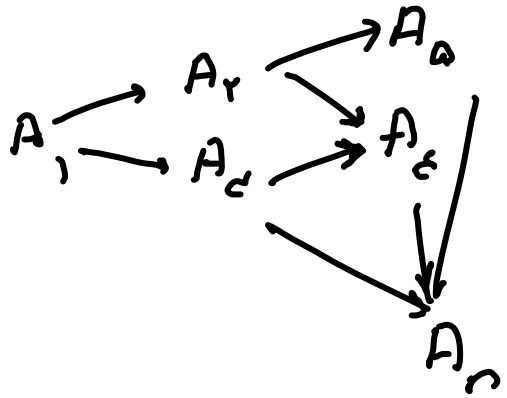
singly connected networks or **polytrees**,

خوشه بندی:

اندازه شبکه

کمترین حالت: $O(n)$

$A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \cdots \rightarrow A_n$



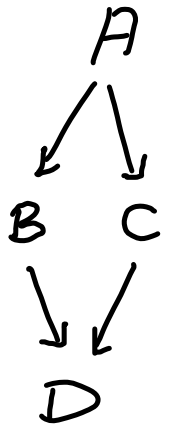
کمترین حالت ممکن:

$O(n^2)$

$n(n-1) = O(n^2)$

تمرین: با خوشه بندی شبکه را دسته‌بندی کنید

$P(A=1) = 0.1$



A	$P(B=1 \mid A)$
0	0.2
1	0.5

A	$P(C=1 \mid A)$
0	0.6
1	0.2

B,C	$P(D=1 \mid B,C)$
0 0	0.1
0 1	0.3
1 0	0.2
1 1	0.5

$P(A=1) = ۰/۱$

A
↓
(B,C)
↓
D

A	B,C | A = 0 0	B,C | A = 0 1	B,C | A = 1 0	B,C | A = 1 1
0	۰/۳۲	۰/۴۸	۰/۰۸	۰/۱۲
1	۰/۴۰	۰/۱۰	۰/۴۰	۰/۱۰

این‌ها باید محاسبه بشوند با استفاده از استقلال B و C به شرط A.

B,C	P(D=1 | B,C)
0 0	۰/۱
0 1	۰/۳
1 0	۰/۲
1 1	۰/۵

مثلاً:

$$P(B=1, C=0 \mid A=0) = P(B=1 \mid A=0)\ P(C=0 \mid A=0)$$

$$= (0.2)(1-0.6)$$

$$= 0.2 \times 0.4$$

$$= 0.08$$

تکمیلی

⊗ در خوشه بندی می‌توان از ترکیب چند متغیر دودویی + یک متغیر غیر دودویی رسید

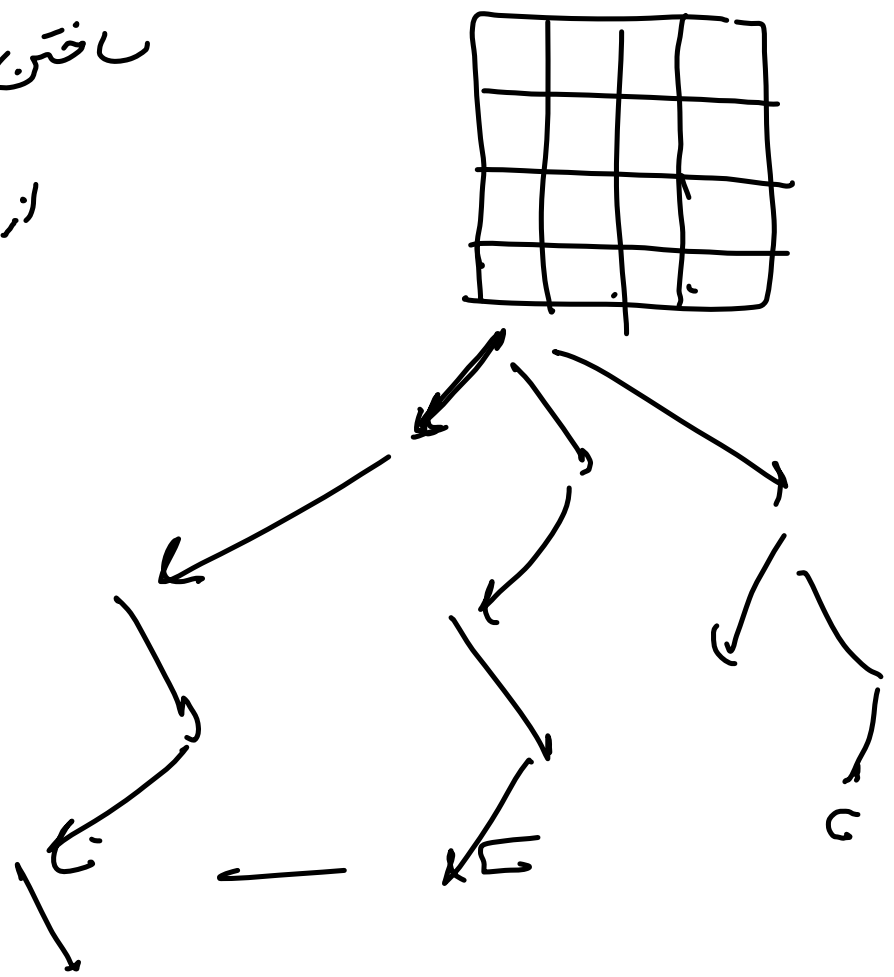
مثلاً (B,C) دیگر متغیر دودویی نیست بلکه ۴ مقدار دارد.

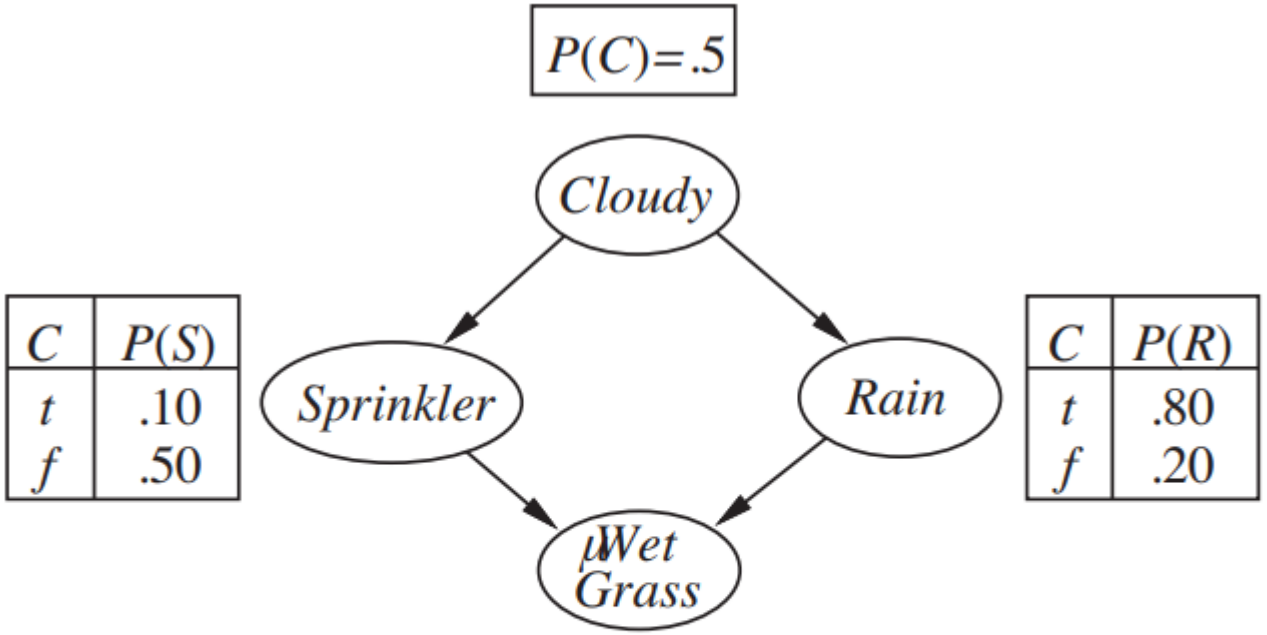
⊗ می‌توان از متغیر مشترک در چند خوشه استفاده کرد. مثلاً یک خوشه‌ی (X,Y) و یک خوشه‌ی (X,W) داشت.

Approximate Inference for Bayesian Networks

روش‌های استنتاج تقریبی حجمی از ایده مشترکی مانند **Monte Carlo** algorithms, استفاده می‌کنند.

ساختن تعداد کافی نمونه و سپس استنتاج (یادگیری) از روی نمونه‌های آموزشی.

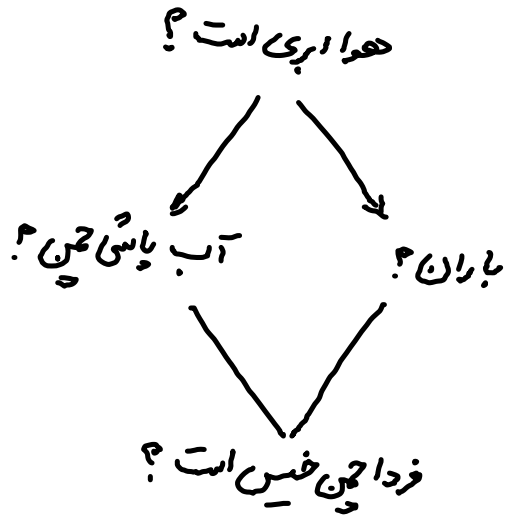




C	P(S)
t	.10
f	.50

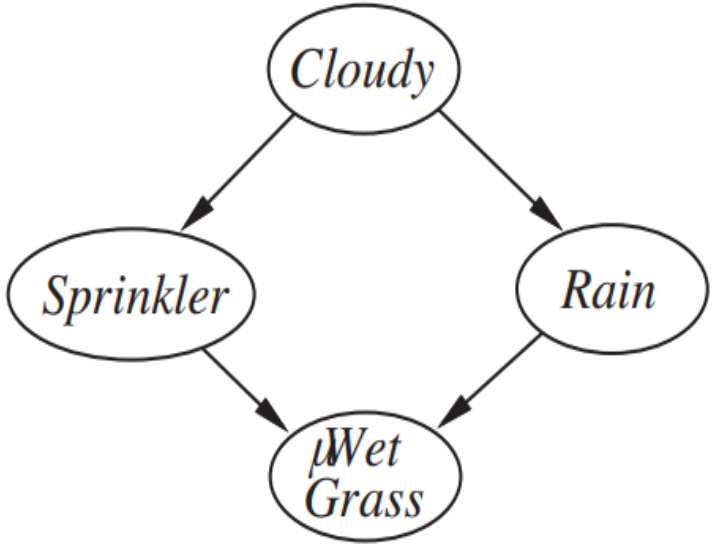
C	P(R)
t	.80
f	.20

S	R	P(W)
t	t	.99
t	f	.90
f	t	.90
f	f	.00



Direct sampling methods

نمونه‌گیری مستقیم.



ابتدا: یک ترتیب مناسب (ordering) انتخاب می‌کنیم. (توپولوژیک)

$$[Cloudy, Sprinkler, Rain, WetGrass]$$

نحوه یک نمونه گیری مستقیم:

1. Sample from $\mathbf{P}(Cloudy)$ value is *true*.
2. Sample from $\mathbf{P}(Sprinkler \mid Cloudy = true)$, value is *false*.
3. Sample from $\mathbf{P}(Rain \mid Cloudy = true)$ value is *true*.
4. Sample from $\mathbf{P}(WetGrass \mid Sprinkler = false, Rain = true)$ value is *true*.

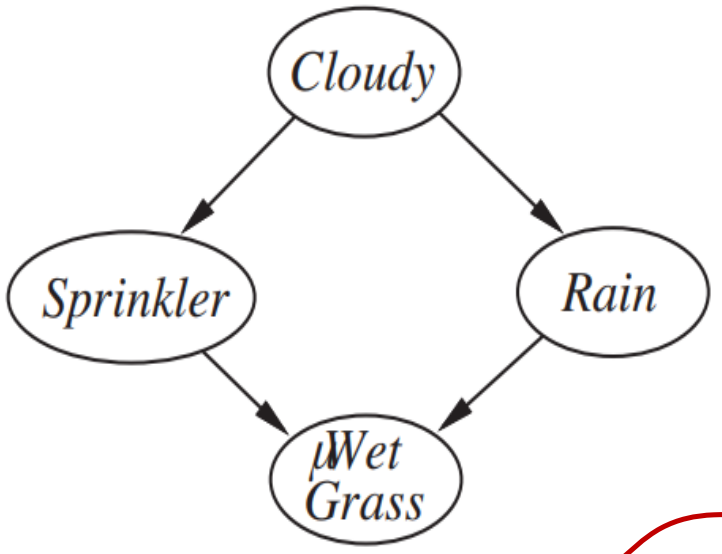
In this case, PRIOR-SAMPLE returns the event $[true, false, true, true]$.

دقت کنید هر نمونه با شش احتمال خواهد قبلی گرفته شده.

در پایان ما یک نمونه گرفته ایم: $[C, S, R, W] = [t, f, t, t]$

باید این فرایند را تکرار کنیم تا نمونه های بعدی به دست آید.

محاسبه احتمال توام با داشتن نمونه ها:



$$P(C=t, S=f, R=t, W=t) = \frac{2}{8}$$

t f t t

$[Cloudy, Sprinkler, Rain, WetGrass]$

۱	t	f	t	t	$\frac{1}{8}$
۲	f	t	f	f	$\frac{1}{8}$
۳	t	f	t	f	$\frac{1}{8}$
	t	f	t	t	$\frac{1}{8}$
	f	f	f	f	$\frac{1}{8}$
	t	t	t	t	$\frac{1}{8}$
	f	t	f	t	$\frac{1}{8}$
	f	f	t	t	$\frac{1}{8}$

تعداد دفعات مشاهده این نمونه

$$\lim_{N\to\infty} \frac{N_{PS}(x_1,\ldots,x_n)}{N} = S_{PS}(x_1,\ldots,x_n) = P(x_1,\ldots,x_n)\,.$$

تعداد کل نمونه‌ها

هنگامی که تعداد نمونه‌ها زیاد شود $(N\to\infty)$ این تخمین، دقیق می‌شود.

$$p(C, \neg S, R, W) = p(t, f, t, t) \cong \frac{2}{8}$$

[در این روش ما فرض کرده‌ایم همه‌ی نمونه‌ها به یک اندازه مهم هستند. وزن هر کدام $\frac{1}{N}$ است]

Rejection sampling in Bayesian networks

روشی رد نمونه :

$$\hat{\mathbf{P}}(X \mid \mathbf{e}) = \alpha \mathbf{N}_{PS}(X, \mathbf{e}) = \frac{\mathbf{N}_{PS}(X, \mathbf{e})}{N_{PS}(\mathbf{e})} .$$

اگر استنتاج تخمینی به صورت $p(X \mid e)$ از روی نمونه‌ها بخواهیم بایم

نمونه‌هایی که در آنها e رخ نداده است را دور بریزیم. فقط از بین نمونه‌هایی دارای مقدار e

احتمال X را حساب کنیم.

$\frac{1}{\text{(تعداد کل دفعات مشاهده } e\text{)}} \times$ تعداد دفعات مشاهده‌ی نمونه‌ی (X, e)

$p(R \mid S=t)$ = ? (هدف / تواهد)

$$\mathbf{P}(Rain \mid Sprinkler = true) \simeq \frac{1}{3}$$

فقط به توزیعی که شامل $S = true$ است در آنها هستیم را نگه داریم.

تعداد آنها ۳ تا است.

در بین آنها R چه داده.

$[Cloudy, Sprinkler, Rain, WetGrass]$

t	f	t	t
f	t	f	f
t	f	t	f
t	f	t	t
f	f	f	f
t	t	t	t
f	t	f	t
f	f	t	t

$\frac{1}{3}$ $\frac{1}{3}$ $\frac{1}{3}$

مثال:

اگر از ۱۰۰ نمونه مستقیم گرفته شده در ۷۳ مورد $S = false$ و در ۲۷ مورد $S = true$ بوده که از این ۲۷ تا در ۸ مورد $R = true$ است. خواهیم داشت:

$$\mathbf{P}(Rain \mid Sprinkler = true) \approx \text{NORMALIZE}(\langle 8, 19 \rangle) = \langle 0.296, 0.704 \rangle .$$

$$\left\langle \frac{8}{8+19} , \frac{19}{8+19} \right\rangle$$

$$R \mid S=t \qquad \sim R \mid S=t$$

(*) در روش‌های نمونه‌گیری Gibbs و likelihood که در ادامه خواهیم خواند، به جای استفاده از (رد نمونه)، از همان ابتدا مقدار متغیرهای (شواهد) را ثابت در نظر می‌گیریم و فقط از سایر متغیرها نمونه می‌گیریم.

در پایان فقط نمونه‌هایی داریم که در آنها e رخ داده است.

Likelihood weighting

درروشی درست نمایی likelihood
به نمونه یک وزن w می دهیم.

هدف

شواهد

هدف

$$\mathbf{P}(Rain \mid Cloudy = true, WetGrass = true)$$

انتخاب ترتیب:

$$[Cloudy, Sprinkler, Rain, WetGrass]$$

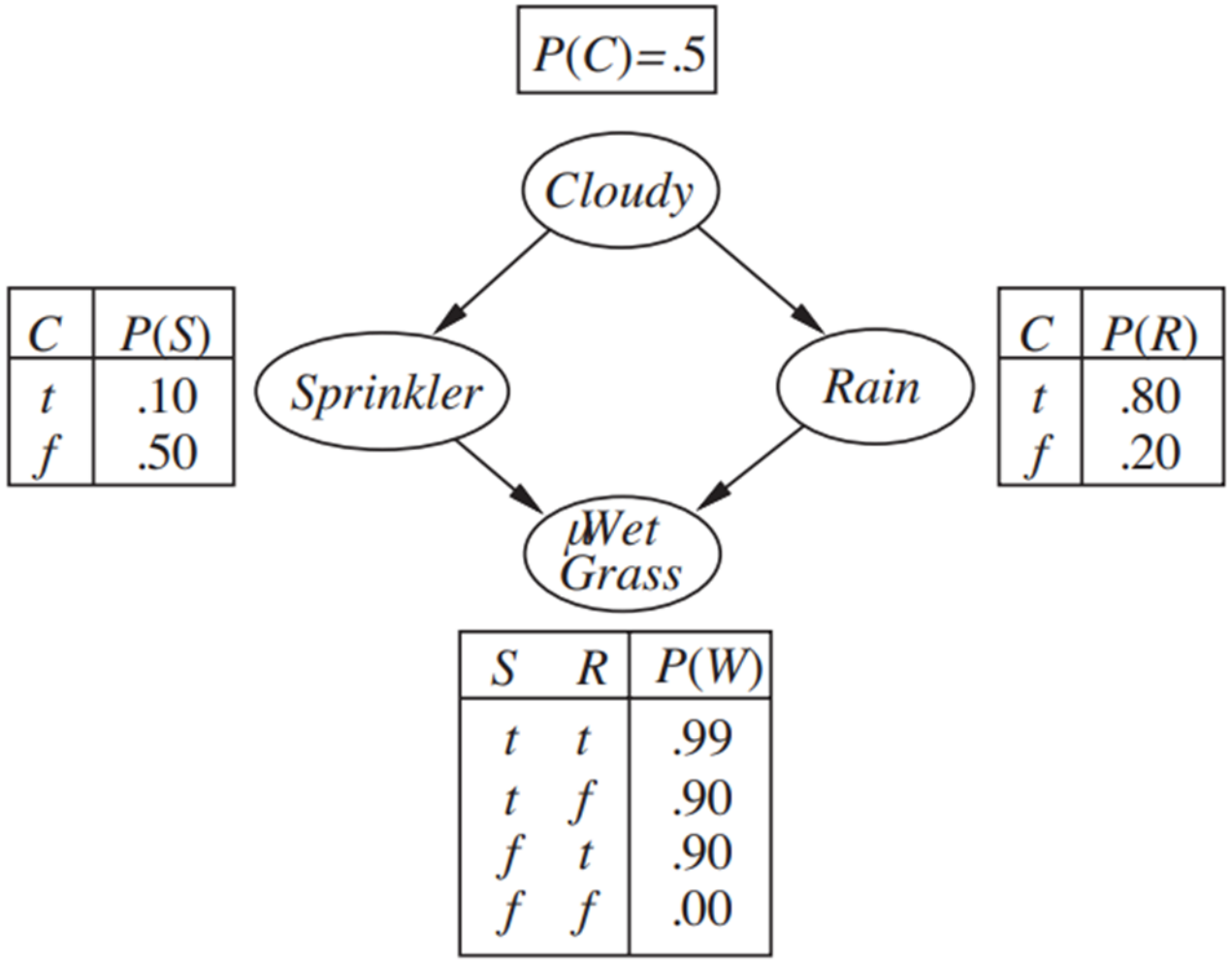
true true

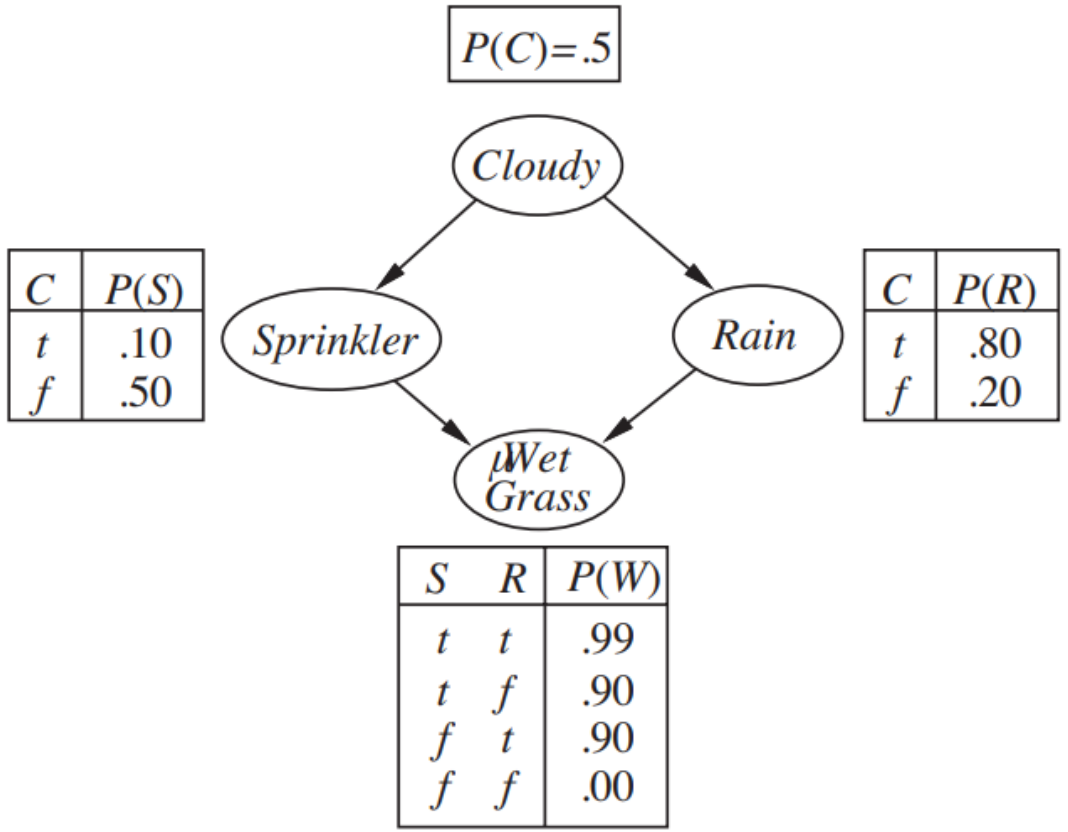
در نمونه گیری، شواهد را fix می کنیم. فقط از بر متغیرهای نمونه می گیریم. اما این بار

هر نمونه (حالت) دارای یک وزن است.

C, S, R, W

[t, ?, ?, t]


P(C)=.5
Cloudy
Sprinkler
Rain
Wet Grass
C P(S)
t .10
f .50
C P(R)
t .80
f .20
S R P(W)
t t .99
t f .90
f t .90
f f .00



$$\mathbf{P}(Rain \mid \overbrace{Cloudy = true, WetGrass = true}^{\text{شواهد}})$$

مثلاً می‌خواهیم وزنِ این نمونه را حساب کنیم:

$$[true, false, true, true]$$

$$[Cloudy, Sprinkler, Rain, WetGrass]$$

فقط احتمالِ رخ‌دادنِ شواهد به شرطِ والدینِ آن‌ها را در هم ضرب کنید:

$$w(t, f, t, t) = P(Cloudy = true)\, P(WetGrass = true \mid Sp = false, Rain = true)$$

$$= 0.5 \times 0.9 = 0.45$$

محاسبه‌ی وزن نمونه به صورت گام به گام :

First, the weight w is set to 1.0

مقدار اولیه وزن : $w=1$

$W=1$

1. *Cloudy* is an evidence variable with value *true*. Therefore, we set

   $$w \leftarrow w \times P(Cloudy = true) = 0.5 .$$

   $W = 1 \times 0.5 = 0.5$

2. *Sprinkler* is not an evidence variable, so sample from $\mathbf{P}(Sprinkler \mid Cloudy = true) = \langle 0.1, 0.9 \rangle$; suppose this returns *false*.
3. Similarly, sample from $\mathbf{P}(Rain \mid Cloudy = true) = \langle 0.8, 0.2 \rangle$; suppose this returns *true*.
4. *WetGrass* is an evidence variable with value *true*. Therefore, we set

   $$w \leftarrow w \times P(WetGrass = true \mid Sprinkler = false, Rain = true) = 0.45 .$$

Here WEIGHTED-SAMPLE returns the event $[true, false, true, true]$ with weight 0.45, and this is tallied under $Rain = true$.

$W = 0.5 \times 0.9 = 0.45$

استنتاج تقریبی با وزن‌های درست‌نمایی

$$\hat{P}(x \mid \mathbf{e}) = \alpha \sum_{\mathbf{y}} N_{WS}(x, \mathbf{y}, \mathbf{e}) w(x, \mathbf{y}, \mathbf{e}) \qquad \text{from LIKELIHOOD-WEIGHTING}$$

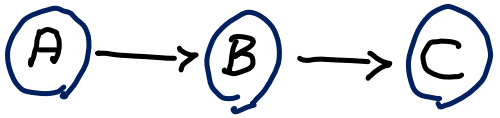
N_{WS}: تعداد این نمونه

w: وزن

در واقع، وزنِ همه‌ی نمونه‌هایی که در آنها X رخ داده را با هم جمع می‌زنیم

وزن نمونه‌ای $\sim X$ را هم با هم جمع می‌زنیم، حالا نرمال‌سازی می‌کنیم.

مثال: استفاده از نمونه‌گیری وزن‌دار برای استنتاج تخمینی:

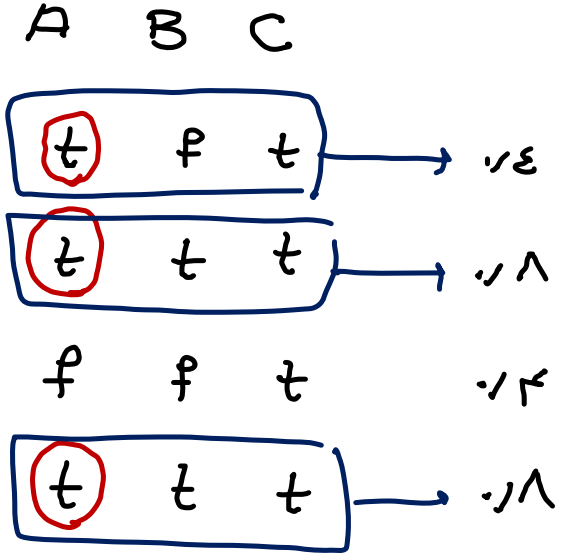


هدف:

$P(A \mid C=true) = ?$

پاسخ: به صورت تقریبی از روی نمونه‌های گرفته شده:

(در هر نمونه شرط $C=true$ رعایت شده است.)



$$\hat{P}(A \mid C=true) = \alpha \langle \underbrace{2(0.8)+0.4}_{A=true \text{ مجموع وزن}} , \underbrace{0.4}_{A=false \text{ مجموع وزن}} \rangle$$

$$\Rightarrow \hat{P}(A \mid C=true) = \alpha \langle 2 , 0.4 \rangle = \langle \frac{2}{2.4} , \frac{0.4}{2.4} \rangle = \langle \frac{5}{6} , \frac{1}{6} \rangle$$

$$P(A \mid C=t) = \left\langle \frac{2}{2,4} \;,\; \frac{0.4}{2,4} \right\rangle$$

$$= \left\langle \frac{a}{4} \;,\; \frac{1}{4} \right\rangle$$

$P(A=t \mid C=t)$ $\qquad$ $P(A=f \mid C=t)$

Gibbs sampling in Bayesian networks

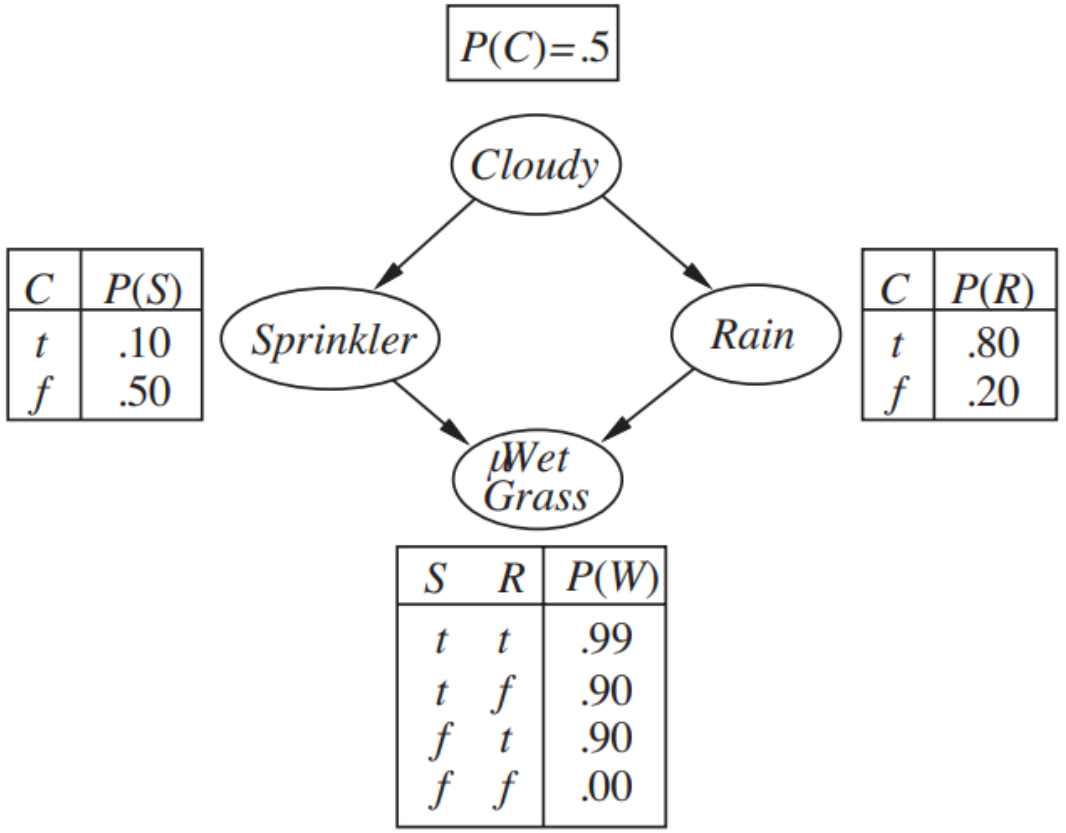
The Gibbs sampling algorithm for Bayesian networks starts with an arbitrary state (with the evidence variables fixed at their observed values) and generates a next state by randomly sampling a value for one of the nonevidence variables X_i. The sampling for X_i is done *conditioned on the current values of the variables in the Markov blanket of* X_i.

برخلاف روش های قبل که از متغیرها تک-تک نمونه می‌گرفتیم.

در این روش از یک state اولیه شروع می‌کنیم. (شواهد در مقدار خود fix شده‌اند)

برای حالت شروع، به سایر متغیرها به صورت تصادفی یک مقدار اولیه می‌دهیم.

سپس هر بار به دلخواه از یک متغیر غیر شاهد، به شرط رعایت مقدار سایرین، نمونه می‌گیریم.


P(C)=.5
Cloudy
Sprinkler
Rain
Wet Grass
C P(S)
t .10
f .50
C P(R)
t .80
f .20
S R P(W)
t t .99
t f .90
f t .90
f f .00

$$\mathbf{P}(Rain \mid Sprinkler = true, WetGrass = true)$$

هدف:

$$[Cloudy, Sprinkler, Rain, WetGrass]$$

Fix Sprinkler → true

Fix WetGrass → true

انتخاب ترتیب:

Fix کردن نمونه‌ها:

The evidence variables *Sprinkler* and *WetGrass* are fixed

Cloudy and *Rain* are initialized randomly- let us say to *true* and *false*

انتخاب تصادفی برای سایرین: مثلاً برای شروع Cl=true و Rain=false باشد.

initial state is $[true, true, false, true]$

حالت اولیه:

حال n بار هر بار به دلخواه از یکی متغیر غیر شاهد، نمونه بگیریم به شرط داشتن مقدار سایرین.

(مارکوف)

1. *Cloudy* is sampled, given the current values of its Markov blanket variables: in this case, we sample from $\mathbf{P}(Cloudy \mid Sprinkler = true, Rain = false)$. (Shortly, we will show how to calculate this distribution.) Suppose the result is $Cloudy = false$. Then the new current state is $[false, true, false, true]$. → نمونه دوم
2. *Rain* is sampled, given the current values of its Markov blanket variables: in this case, we sample from $\mathbf{P}(Rain \mid Cloudy = false, Sprinkler = true, WetGrass = true)$. Suppose this yields $Rain = true$. The new current state is $[false, true, true, true]$. → نمونه سوم

مثلاً فرض کنید بعد از ۸۰ بار نمونه‌گیری داشته باشیم:

If the process visits 20 states where *Rain* is true and 60 states where *Rain* is false, then the answer to the query is $\text{NORMALIZE}(\langle 20, 60 \rangle) = \langle 0.25, 0.75 \rangle$.

گروه بابان
BABAN.IR

بسم الله الرحمن الرحیم

Artificial Intelligence

A Modern Approach

هوش مصنوعی

ابوالفضل گیلک
گروه بابان

بخش دوم جلسه ۲۳ منتخب فصل های ۱۵ تا ۱۷

شبکه بیز – زنجیر مارکوف

15 PROBABILISTIC REASONING OVER TIME

16 MAKING SIMPLE DECISIONS

17 MAKING COMPLEX DECISIONS

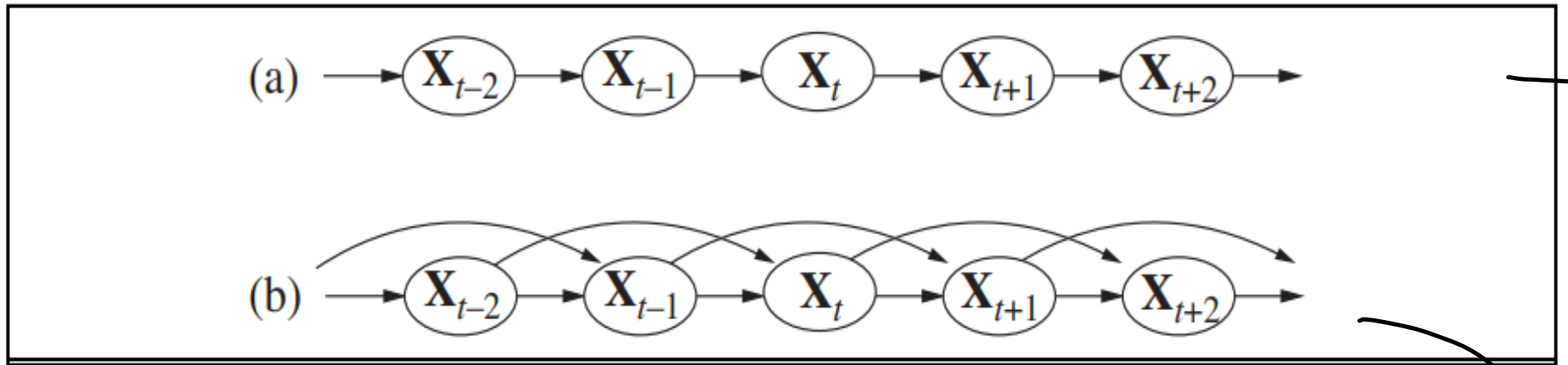


Figure 15.1 (a) Bayesian network structure corresponding to a first-order Markov process with state defined by the variables $\mathbf{X}_t$. (b) A second-order Markov process.

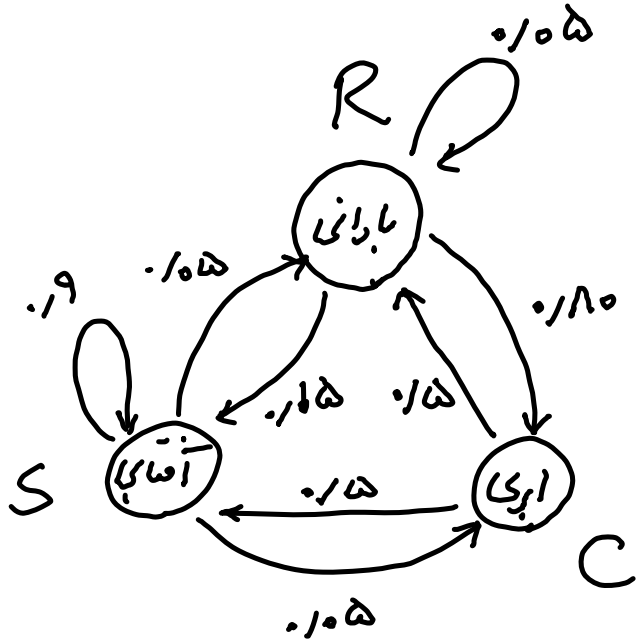
مارکوف مرتبه 1

$P(X_{t+1} \mid X_t)$

مارکوف مرتبه ۲

$P(X_{t+1} \mid X_t, X_{t-1})$

داشتن دو حالت قبل



$$P(X_1 \mid X_0) = \cdots$$

$$P(X_{t+1} \mid X_t) = \cdots$$

افزایش دقت در استنتاج مارکوفی ۲ راه دارد:

① افزایش مرتبه زنجیر مارکوف

$$P(X_{t+1} \mid \overbrace{X_t, X_{t-1}, \ldots}^{\text{تعداد بیشتر}})$$

که باعث پیچیدگی محاسباتی می‌شود.

② استفاده از sensor.

$$P(X_{t+1} \mid X_t, \underbrace{e}_{\text{مشاهده یا برآیند بخشی از محیط}})$$

$\pi_t(S_i)$ = احتمال قرارداشتن فرایند در S_j $= P(X_t = S_i)$

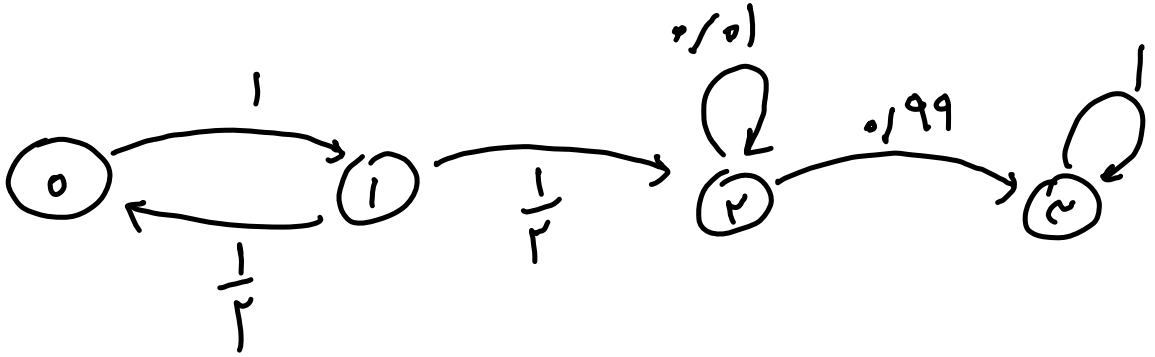
گاهی با گذشت زمان تأثیر حالت اولیه از بین می‌رود و $\pi_t(S_i) = \pi_{t+1}(S_i)$

می‌شود (فرآیند پایدار)

مثلاً: در این مثال با گذشت زمان $t \to \infty$ می‌توان گفت فرایند حتماً در ③ قرار دارد:

$\pi_t(S_3) = 1$

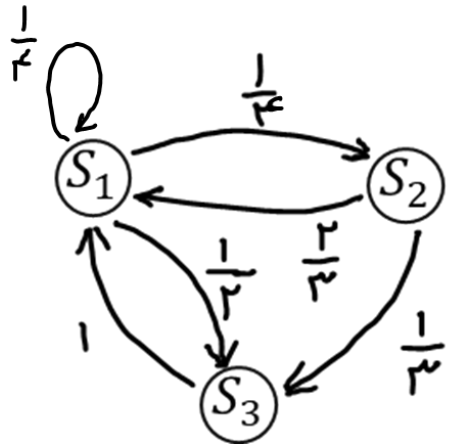
$\pi_t(S_i) = 0 \quad i \neq 3$



S_3 حالت ماندگار (پایدار) است. سایرین حالات گذرا هستند.

[* مجموع خروجی‌های درهم گره‌ی ۱ است.]

فرایند مارکوف ساده و پایدار:



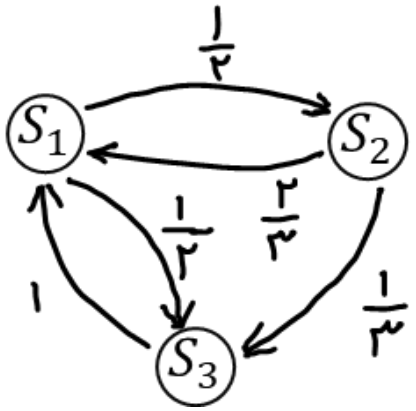
ماتریس انتقال.

$$P = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{2}{3} & 0 & \frac{1}{3} \\ 1 & 0 & 0 \end{bmatrix}$$

(جمع سطرها 1 است. همه درایه‌ها بین 0 و 1 هستند.)

P^k (درایه‌های ماتریس P^k) = (احتمال آن را می‌دهد که اگر اکنون در X_i باشیم k گام بعد در X_j باشیم.)

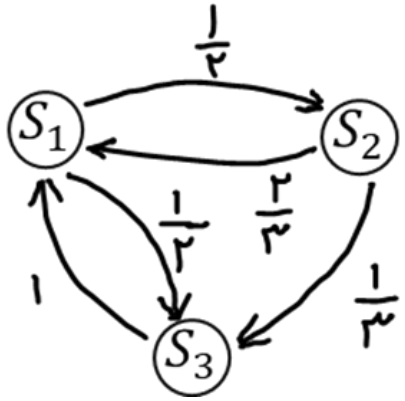
سوال ۱:



فرایند مارکوف زیر در حالت سکون قراردارد. $\{\pi_{t+1}(S_i) = \pi_t(S_i)\}$
احتمالات انتقال داده شده اند. احتمال قرار داشتن فرایند در S_2 کدام است؟

۱- ۰/۲۳
۲- ۰/۳۴
۳- ۰/۳۸
۴- ۰/۴۶

پاسخ: گزینه (۱)



$$\Rightarrow \begin{cases} \pi_1 = \frac{2}{3}\pi_2 + \pi_3 \\ \pi_2 = \frac{1}{2}\pi_1 \\ \pi_3 = \frac{1}{2}\pi_1 + \frac{1}{3}\pi_2 \end{cases} \Rightarrow \begin{cases} \pi_2 = \frac{1}{2}\pi_1 \\ \pi_3 = \frac{2}{3}\pi_1 \end{cases}$$

از طرفی: $\pi_1 + \pi_2 + \pi_3 = 1$ و لی:

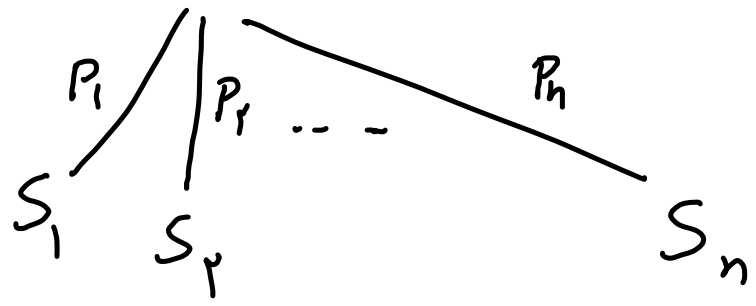
$$\pi_1 + \frac{1}{2}\pi_1 + \frac{2}{3}\pi_1 = 1$$

$$(1 + \frac{1}{2} + \frac{2}{3})\pi_1 = 1$$

$$\pi_1 = \frac{6}{13} \Rightarrow \pi_2 = \frac{1}{2}\frac{6}{13} = \frac{3}{13} \approx 0.23$$

$$L = [p_1, S_1;\ p_2, S_2;\ \ldots\ p_n, S_n]\ .$$

لاتاری :



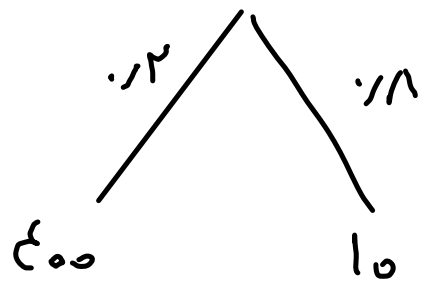
متوسط سودمندی لاتاری $Expect(L) = \sum_{i=1}^{n} p_i U(S_i)$

البته ورودی هم یک هزینه دارد ؛ $Cost(L)$.

انتخاب بین دو لاتاری.

L_1:

Cost = ۱۰۰ هزینه ورودی:



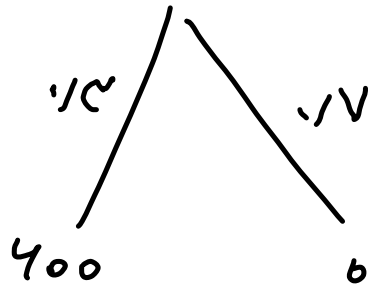
$$EXP(L_1) = \frac{۲}{۱۰}(۴۰۰) + \frac{۸}{۱۰}(۱۰) = ۸۸$$

۱۰۰ دلار هزینه و ۸۸ دلار متوسط سود پس:

$$۸۸ - ۱۰۰ = -۱۲$$

L_2:

Cost = ۱۲۰ هزینه ورودی:

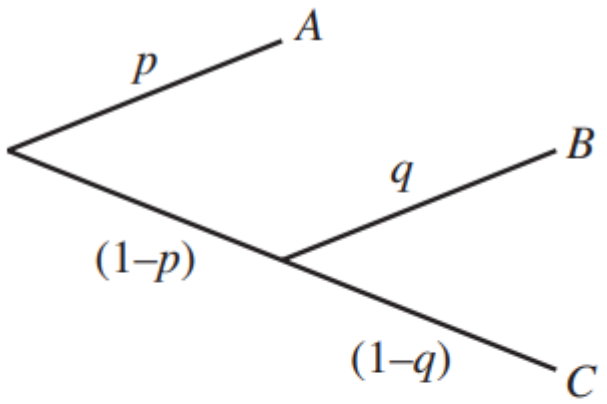


$$EXP(L_2) = \frac{۳}{۱۰}(۴۰۰) + \frac{۷}{۱۰}(۰) = ۱۲۰$$

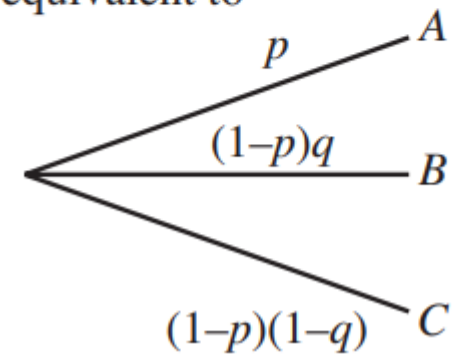
$$۱۲۰ - ۱۲۰ = ۶۰$$

شرکت در L_2 به صرفه است.

معادل سازی لاتاری ها

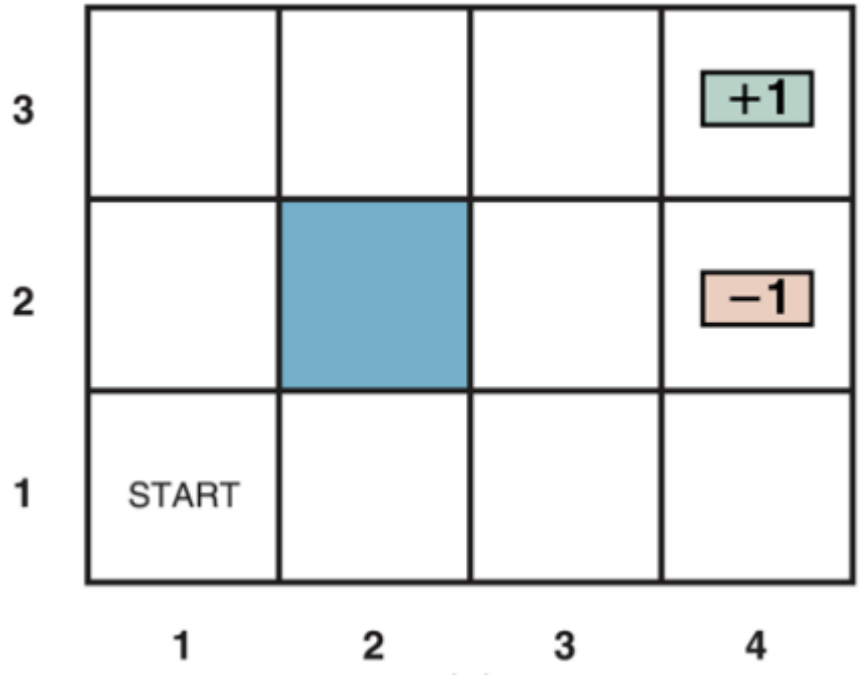


is equivalent to



Markov decision process (MDP)

فرایند تصمیم سازی مارکو (مارکوف)



حالت مطلوب

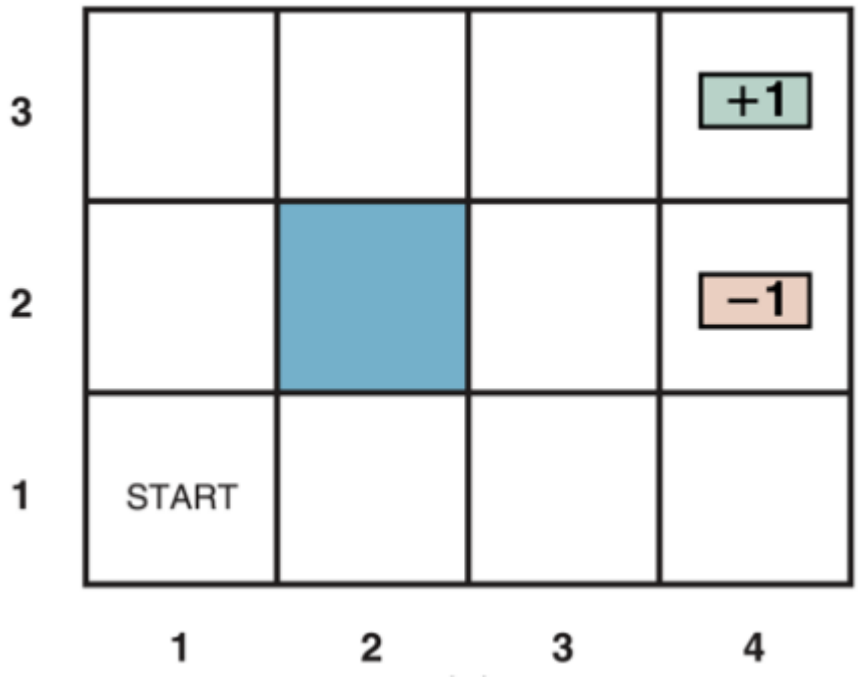
حالت نامطلوب
(چاه)

Markov decision process (MDP)

- مجموعه‌ای از حالات
- حالت اولیه‌ی s'_0
- در هر حالت s، اقدامات قابل اجرای $A(s)$
- یک الگوی انتقال $p(s' \mid s, a)$ برای $a \in A(s)$.
- تابع تشویق $R(s)$ (پاداشی با ورود به s)

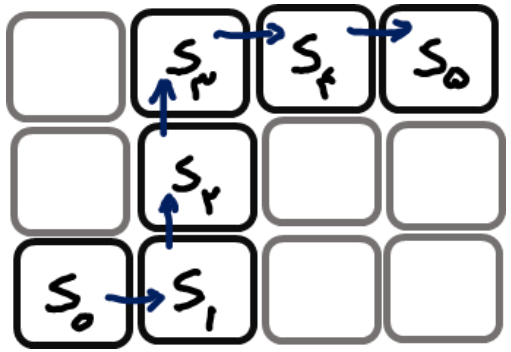
در واقع پاداشی را به اقدام می‌دهیم

$R(s', a, s)$ = پاداشی اقدام a که باعث ورود به s می‌شود.



برای آن که عامل، تشویق به حرکت برای رسیدن به هدف شود برای هر حالت غیرپایانی پاداش -0.04 را در نظر می‌گیریم.

حالات نهایی که اگر به آنها وارد شویم دیگر همان جا متوقف می‌شویم یکی بسیار مطلوب $(+1)$ و دیگری نامطلوب (-1) است.



مسوق‌های تجمیعی (بدون ضریب کاهش) (تخفیف)

$$U_h[s_0, a_0, s_1, a_1, s_2, \dots] = R(s_1) + R(s_2) + R(s_3) + \dots$$

مسوق‌ها دارای ضریب کاهش: $0 < \gamma < 1$

$$U_h[s_0, a_0, s_1, a_1, s_2, \dots] = R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + \dots$$

شکل بهتر فرمول این است: (یادداشتی مربوط به اعلامات هستند)

$$U_h([s_0, a_0, s_1, a_1, s_2, \ldots]) = R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \cdots,$$

علت مفید بودن ضریب تخفیف γ آن است که اجازه نمی‌دهد مجموع به ∞ برسد:

$$U_h([s_0, a_0, s_1, a_1, s_2, \ldots]) = R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \cdots,$$

$s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow s_3 \longrightarrow \cdots$

$R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + \cdots$

اگر R_m بزرگترین پاداش باشد:

$\leq R_m + \gamma R_m + \gamma^2 R_m + \cdots$

$= R_m \left(1 + \gamma + \gamma^2 + \cdots\right) = R_m \frac{1}{1-\gamma}$

$$1 + r + r^2 + r^3 + \cdots = \frac{1}{1 - r}$$

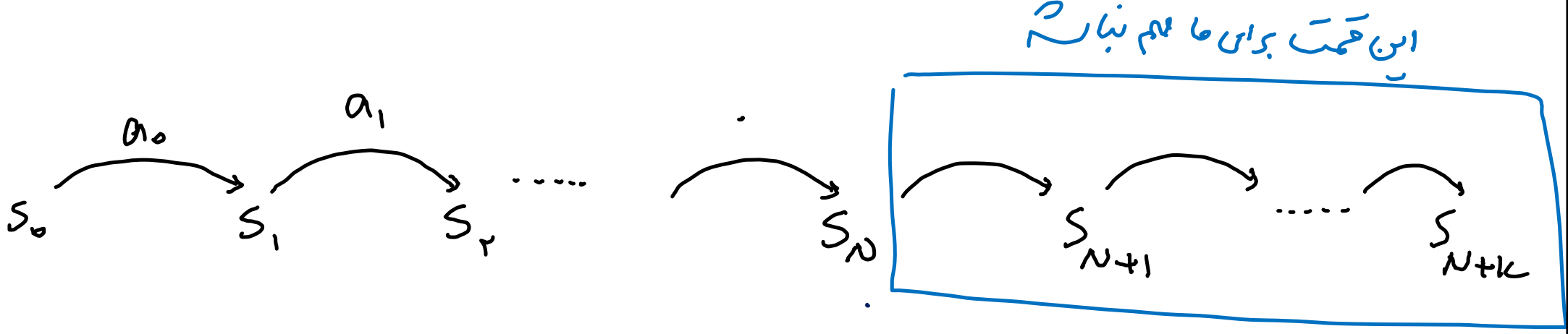
$$U_h([s_0, a_0, s_1, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \leq \sum_{t=0}^{\infty} \gamma^t R_{\max} = \frac{R_{\max}}{1-\gamma}$$

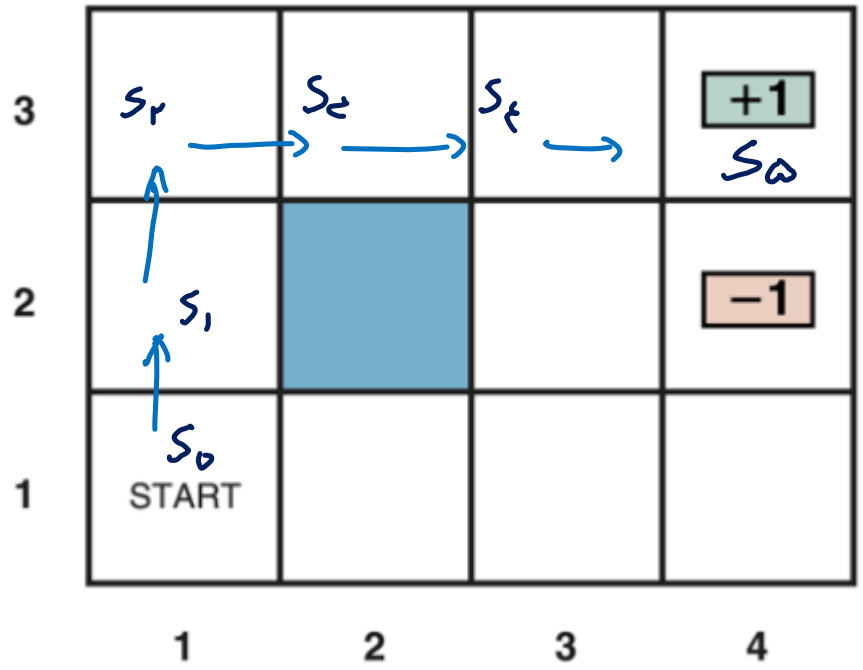
مجموع پاداش‌های روی هر مسیر $\leq R_{max} \frac{1}{1-\gamma}$

یک اصطلاح finite horizon مجموعی پاداشی با افق متناهی N، یعنی فقط تعداد متناهی از اقدامات

برای ما مهم باشد: بعد از ورود به s_N، پاداشی برای اقدامات مهم نباشد:

$$U_h([s_0, a_0, s_1, a_1, \ldots, s_{N+k}]) = U_h([s_0, a_0, s_1, a_1, \ldots, s_N])$$





اگر در یک محیط قطعی قرار داشتیم که هوا قدام

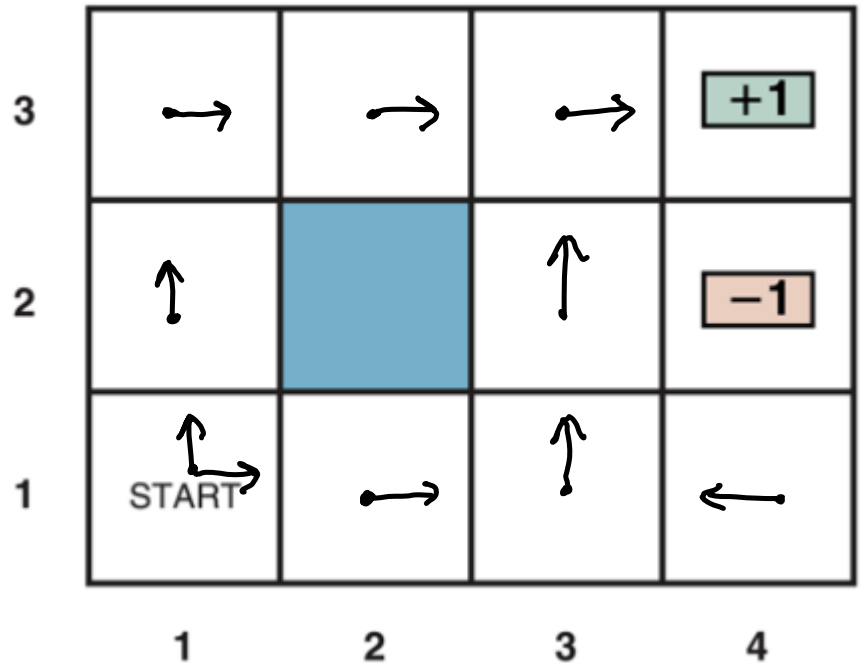
نتیجه‌ی قطعی داشت، راه‌حل مسئله مشخص بود.

up . up . Right, Right. Right

اگر ضریب تخفیف نداشته باشیم ($\gamma = 1$). پاداشی در مجموع روی این مسیر:

$$R(s_1) + \cdots + R(s_5) = ۴(-۰/۰۴) + ۱$$

شبکه‌ی اقدامات π^* چنین است:

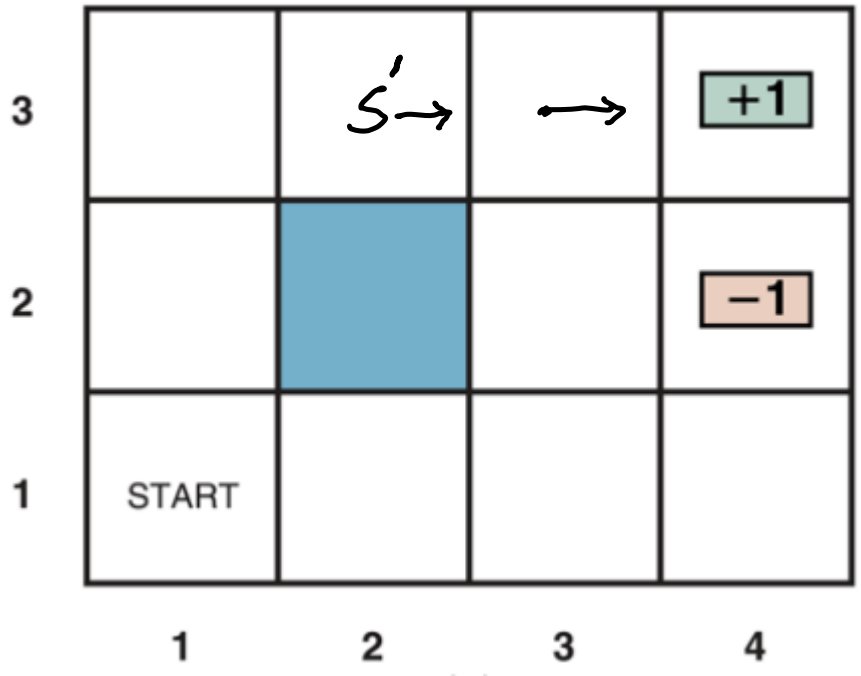


π^*

اقدامات بهینه.

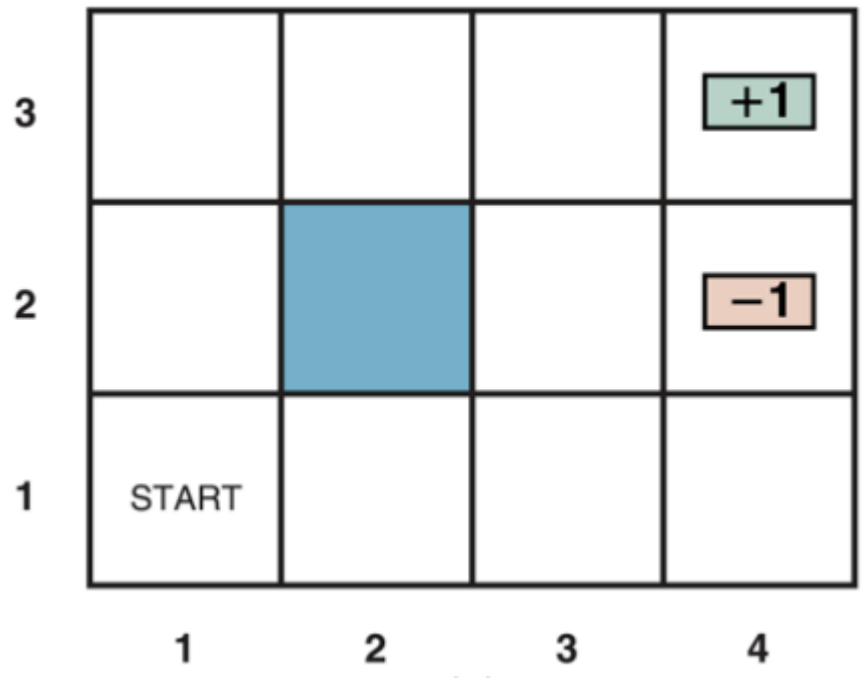
اگر اقدامات ۴ گانه‌ی ↑ → ↓ ← قطعی باشند، π^* در این مثال چنین است:

U^* هم مقدار سودمندی بازی‌ها از هر حالت است.



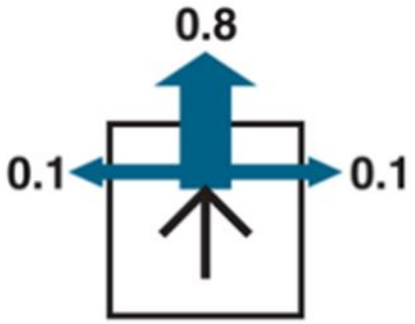
مثلاً برای خانه‌ی s':

$$U^*(s') = -0.04 + \gamma(1)$$

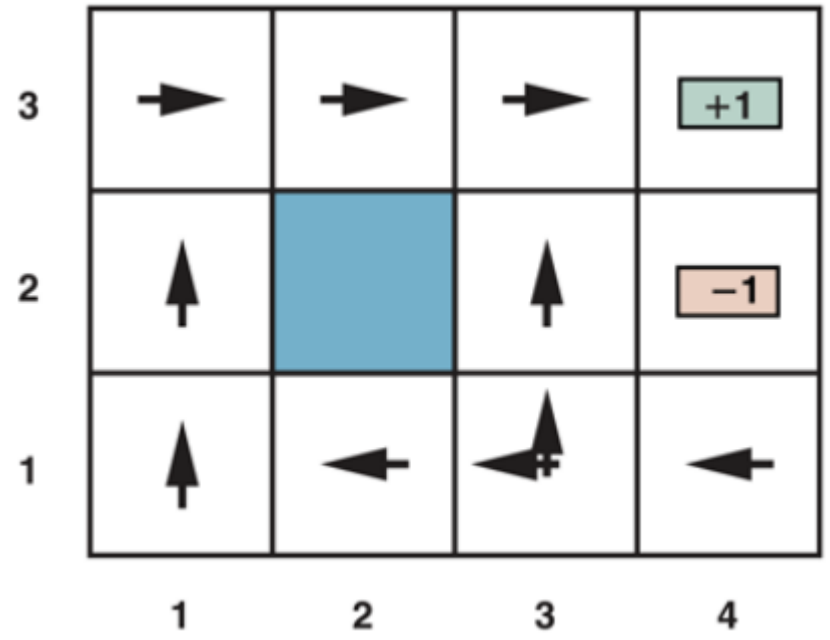


حال فرض کنیم اقدامات با برآیندهای غیرقطعی داریم.

به این صورت:



π^*:

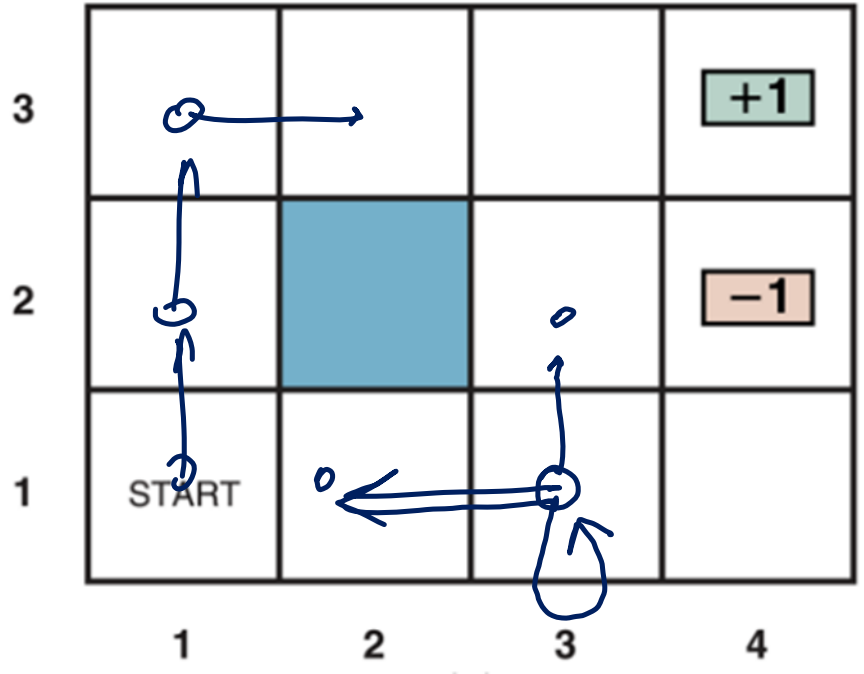


$r = -0.04$

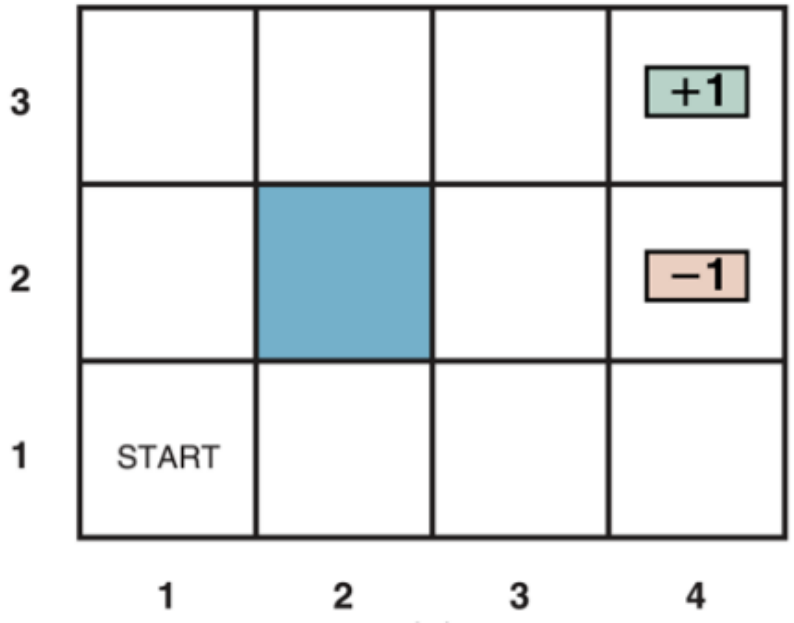
در حالت $\gamma = 1$ با مقادیر پاداشی گفته شده سیاست بهینه برای هر state مطابق شکل است:

π^* : سیاست بهینه در مثال فوق.

(a) The optimal policies for the stochastic environment with $r = -0.04$ for transitions between nonterminal states. There are two policies because in state (3,1) both *Left* and *Up* are optimal.


3
2
1
+1
−1
START
1
2
3
4

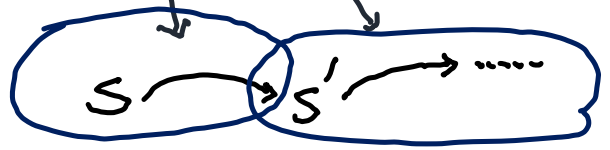
The Bellman equation for utilities

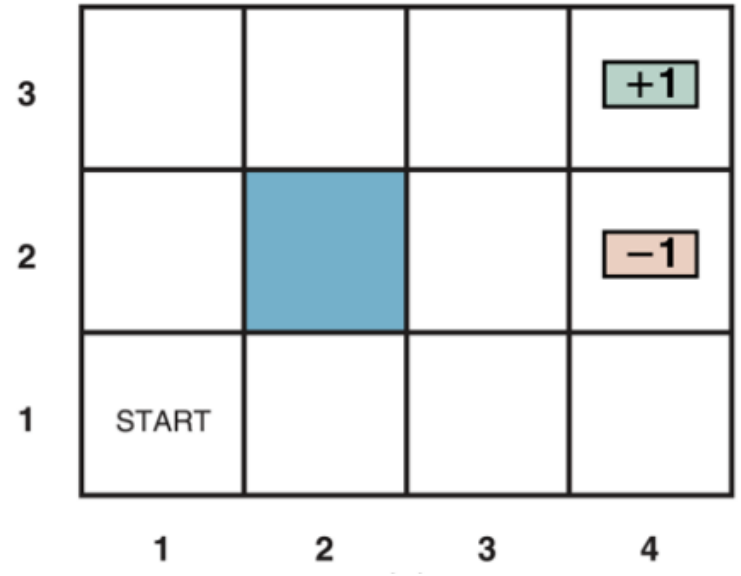


$$U(s) = \max_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')].$$

احتمال آن به‌ازای اقدام وارد s' شود.

بازه همه اقدامات قابل انتخاب

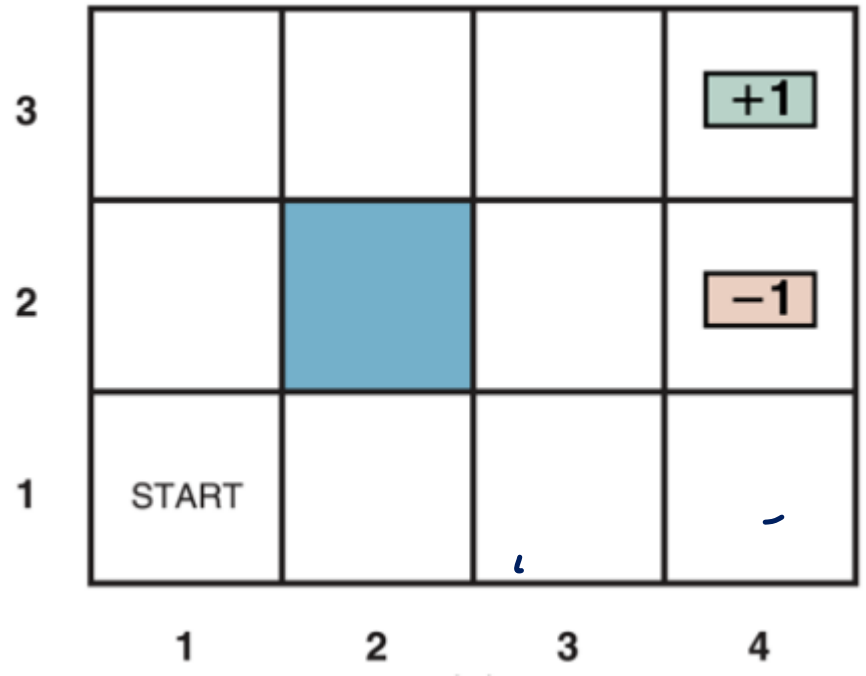




مثلاً برای $U(1,1)$ برای ۴ انتخاب ممکن

Up ، Left ، .... داریم:

$$\max\{ \begin{array}{ll} [0.8(-0.04+\gamma U(1,2))+0.1(-0.04+\gamma U(2,1))+0.1(-0.04+\gamma U(1,1))], & (Up) \\ [0.9(-0.04+\gamma U(1,1))+0.1(-0.04+\gamma U(1,2))], & (Left) \\ [0.9(-0.04+\gamma U(1,1))+0.1(-0.04+\gamma U(2,1))], & (Down) \\ [0.8(-0.04+\gamma U(2,1))+0.1(-0.04+\gamma U(1,2))+0.1(-0.04+\gamma U(1,1))]\} & (Right) \end{array}$$



به این صورت هم می‌توان نوشت:

$$U(1,1) = -0.04 + \gamma \max[\ 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), \quad (Up)$$
$$0.9U(1,1) + 0.1U(1,2), \quad (Left)$$
$$0.9U(1,1) + 0.1U(2,1), \quad (Down)$$
$$0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\ ]. \quad (Right)$$

دقت کنید با نوشتن این معادله برای همه‌ی خانه‌ها

به یک دستگاه بزرگ n معادله n مجهول داریم و با حل آن

x^* ها به دست می‌آیند.

3	0.8516	0.9078	0.9578	+1
2	0.8016		0.7003	−1
1	0.7453	0.6953	0.6514	0.4279
	1	2	3	4

The utilities of the states in the 4×3 world with $\gamma = 1$ and $r = -0.04$ for transitions to nonterminal states.

در حالت $\gamma = 1$

با نوشتن تمام معادلات برای مقادیر $U(i,j)$ به این سودمندی‌ها می‌رسیم:

مقادیر U^* در حالت غیرقطعی

با $\gamma = 1$ و $r = -0.04$

action-utility function, or **Q-function**: $Q(s, a)$:

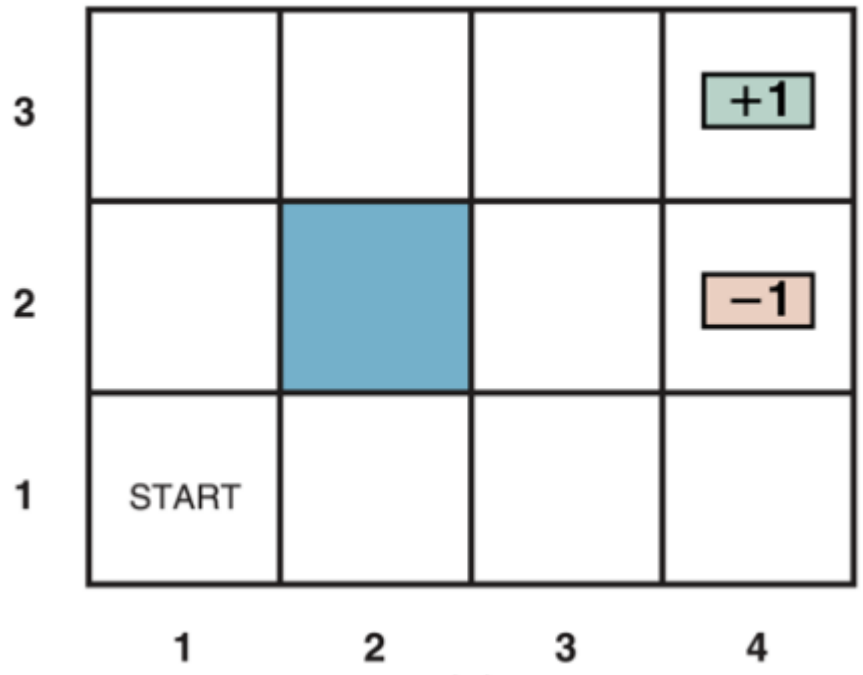
$Q(s,a)$ نشان دهنده‌ی سودمندی بهترین مسیر با شرط آن که اولین اقدام a باشد است.

$$U(s) = \max_a Q(s, a).$$

مقادیر بهینه

$$\pi^*(s) = \operatorname{argmax}_a Q(s, a).$$

اقدامات بهینه.



۷۴- محیط زیر با کنش‌های (action) بالا U، پایین D، چپ L و راست R را در نظر بگیرید. کنش‌هایی که باعث ورود به خانه S۵ می‌شوند پاداش برابر با ۱۰ دارند و خود S۵ خانه وضعیت پایان است. سایر کنش‌ها پاداش ۱- دارند. مقدار ضریب تخفیف (discount factor) برابر $\gamma = ۰/۹$ را در نظر بگیرید. کدام گزینه صحیح است؟

S۱	S۲	S۳
S۴	S۵	S۶
S۷	S۸	S۹

-۱	-۱	-۱
-۱	۱۰	-۱
-۱	-۱	-۱

پاداش ورود

۱) $V^*(S_۱) = ۸$
$Q^*(S_۴, D) = ۶/۲$

۲) $V^*(S_۱) = ۸$
$Q^*(S_۴, D) = ۷/۲$

۳) $V^*(S_۱) = ۹$
$Q^*(S_۴, D) = ۷/۱$

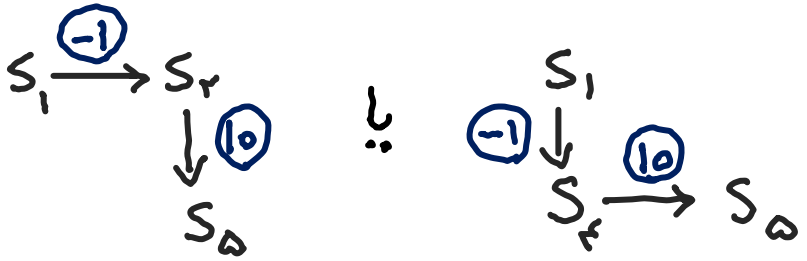
۴) $V^*(S_۱) = ۹$
$Q^*(S_۴, D) = ۸/۱$

$V^*(s_i)$ = پاداش بهترین مسیر با شروع از s_i

$Q^*(s_i, a)$ = پاداش بهترین مسیر با شروع از s_i

و شرط داشتن اولین اقدام: a

دو مسیر بهینه کدام داریم:



S۱	S۲	S۳
S۴	S۵	S۶
S۷	S۸	S۹

$$v^*(s_1) = R_1 + \gamma R_2 \quad = (-1) + (0.9)(10) = 8$$

$Q^*(S_۴, D)$

شروع از $S_۴$ و اولین اقدام به سمت پایین باشد.

$$Q^*(S_۴, D) = (-۱) + \gamma(-۱) + \gamma^۲(۱۰)$$

$$= (-۱) + ۰/۹(-۱) + ۰/۸۱(۱۰)$$

$$= ۶/۲$$

