

هوش مصنوعی

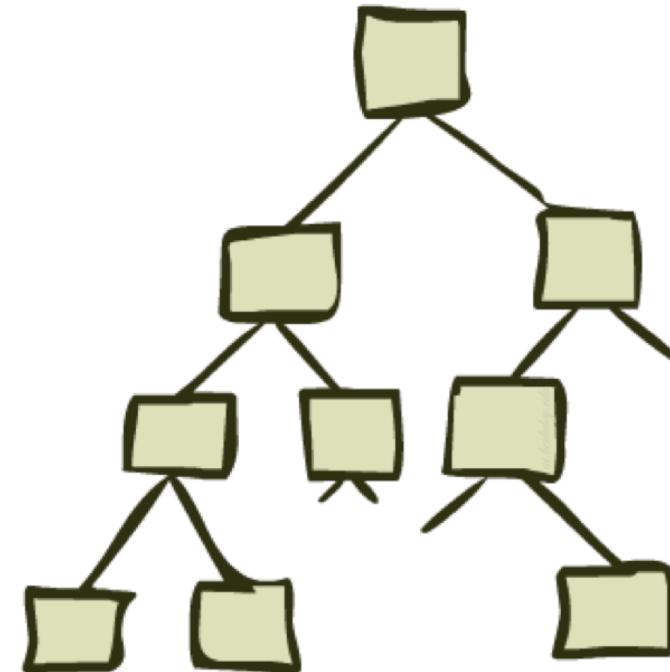
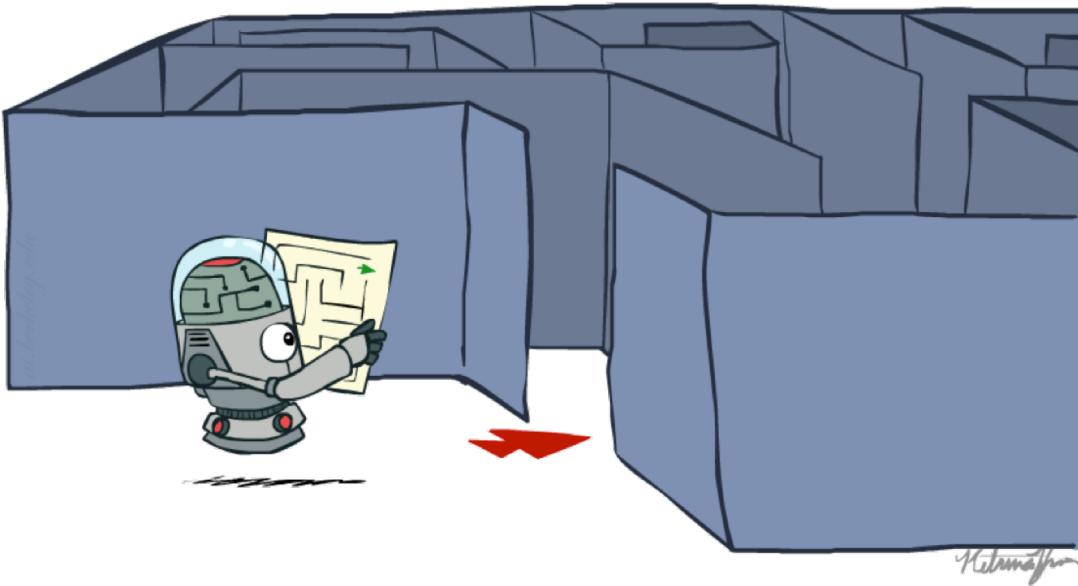
مسئل ارضای محدودیت

Constraint Satisfaction Problems

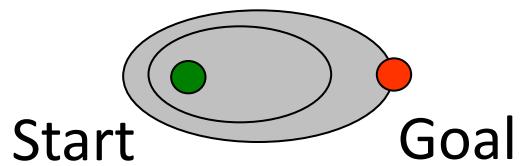
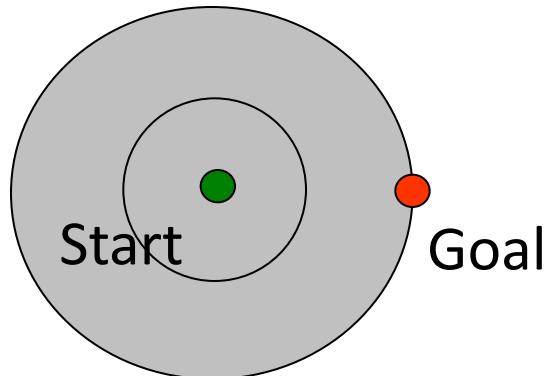


محمد طاهر پیلهور

جلسات قبل



جلسات قبل



▪ جستجوی ناآگاهانه

- DFS, BFS, UCS, ... •

▪ جستجوی آگاهانه

- هیوریستیک
- جستجوی حریصانه
- جستجوی A^*

امروز

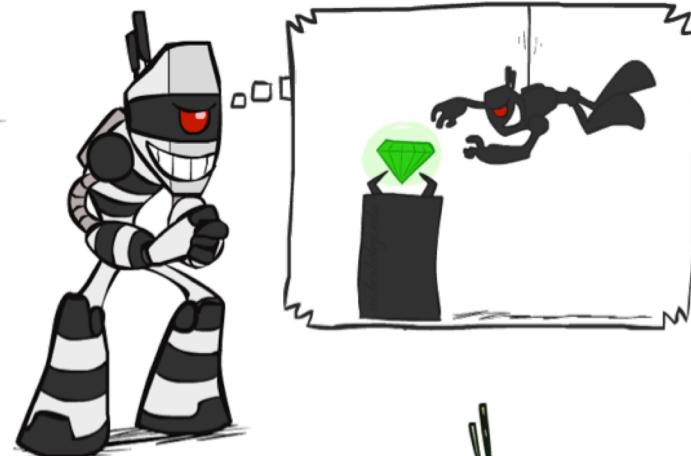
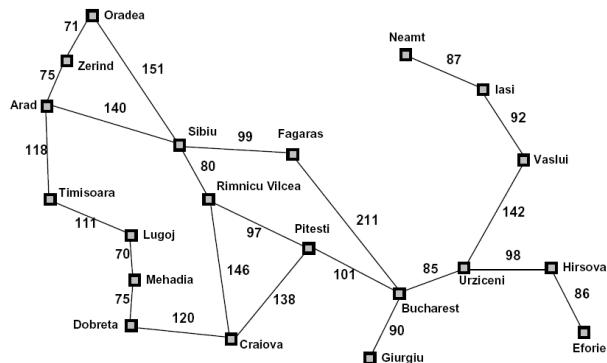
مسائل ارضای محدودیت

Constraint Satisfaction Problems



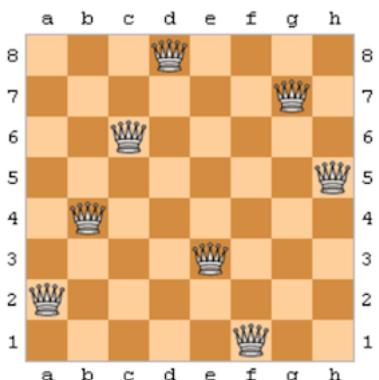
مسائل جستجو

پیشفرضها: تک عاملی، قطعی، کاملا قابل مشاهده، گستته



نقشه‌کشی: دنباله‌ای از تصمیمات

- طریقه رسیدن به هدف (مسیر) اهمیت دارد
- مسیرهای مختلف هزینه‌های مختلفی دارند

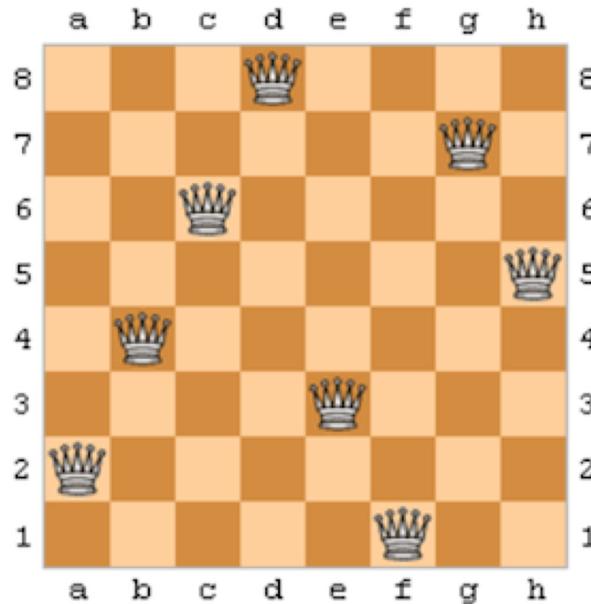


شناسایی: مقداردهی به متغیرها

- هدف مهم است (نه مسیر رسیدن به آن)
- مسیرهای مختلف هزینه‌ی یکسانی دارند

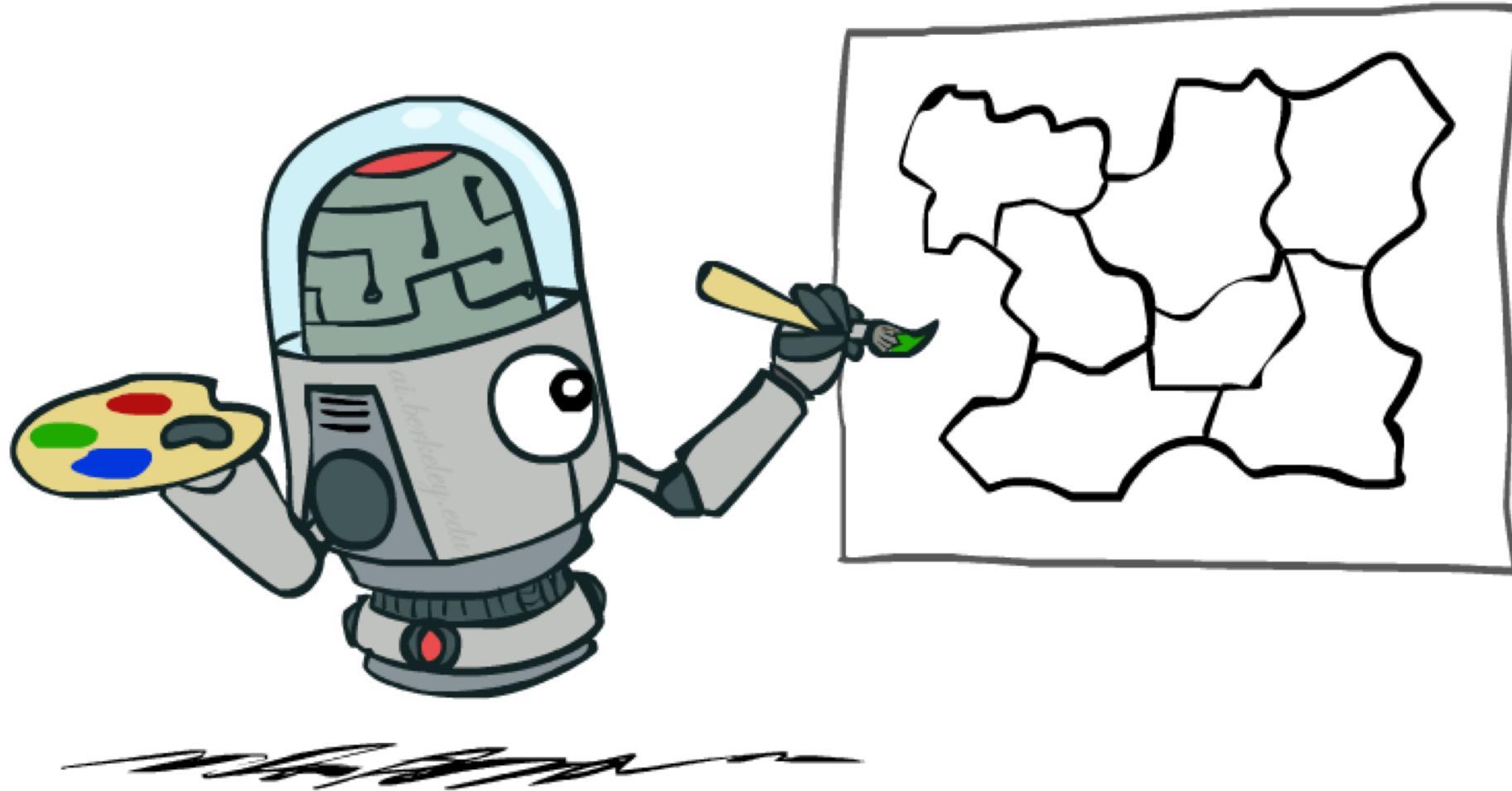
ها مناسب این دسته از مسائل هستند

مسائل ارضی محدودیت - CSP

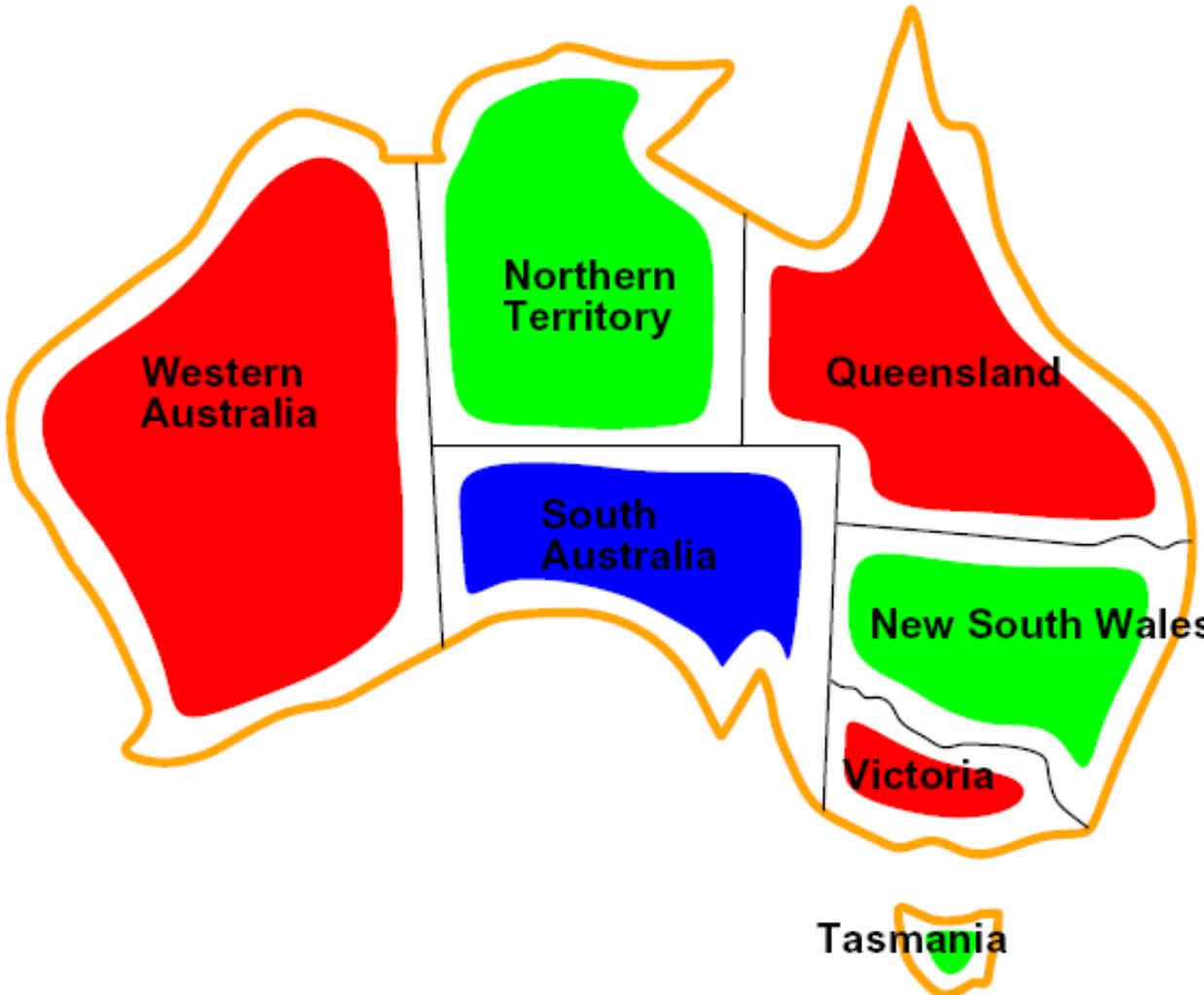


5	3		7				
6			1	9	5		
	9	8				6	
8				6			3
4			8	3			1
7				2		6	
	6				2	8	
		4	1	9			5
			8		7	9	

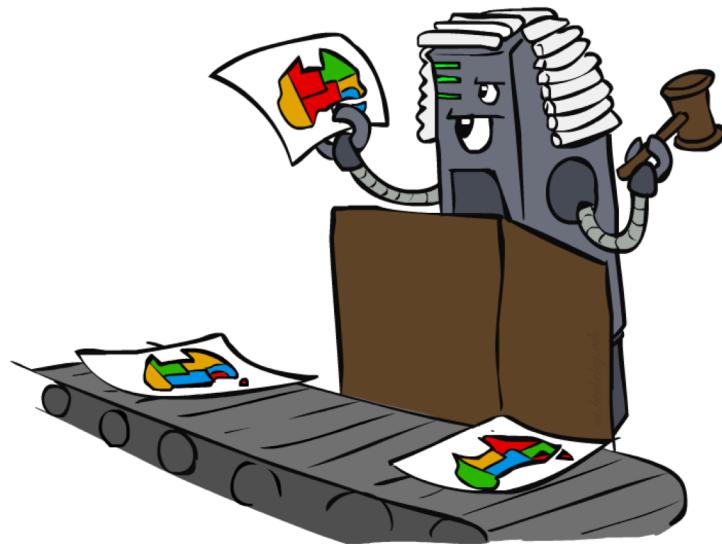
مثال CSP – رنگ زدن نقشه



مثال CSP – رنگ زدن نقشه

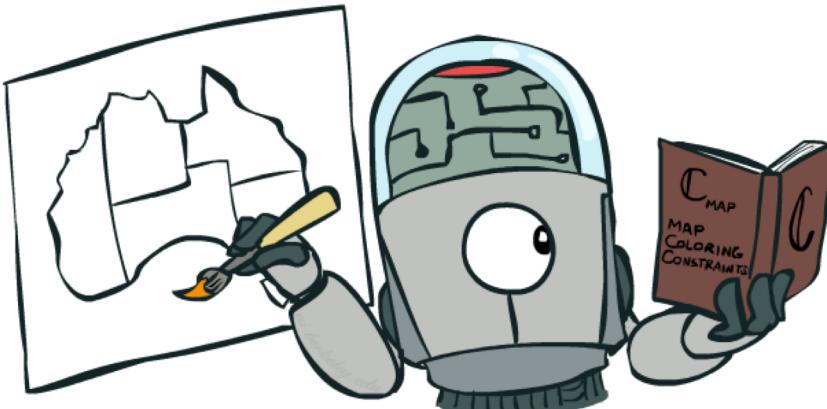


مسائل ارضای محدودیت - CSP



مسائل جستجوی عادی

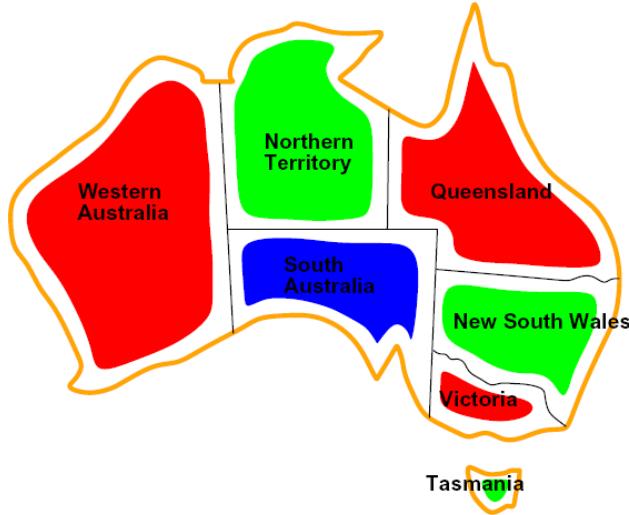
- حالات تعریف منعطفی دارند: هر گونه ساختار داده می‌توان استفاده کرد
- تابع پیشبر یا هدف می‌توانند هر تابعی بر روی حالات باشد



مسائل ارضای محدودیت

- نوع خاصی از مسائل جستجو
- یک حالت با نحوه مقداردهی به تعدادی متغیرهای X_i از دامنه D تعریف می‌شود
- هدف مقداردهی به تعدادی متغیر است، در حالتی که «محدودیت» ها ارضاء شوند

مثال: رنگ کردن نقشه



WA, NT, Q, NSW, V, SA, T

متغیرها

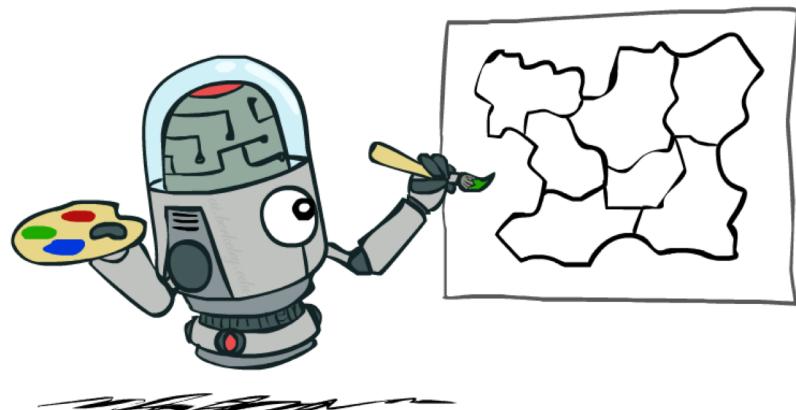
$D = \{\text{red, green, blue}\}$

دامنهها

محدودیت ها: نواحی مجاور رنگ های متفاوتی داشته باشند

Implicit: $WA \neq NT$

Explicit: $(WA, NT) \in \{(red, green), (red, blue), \dots\}$



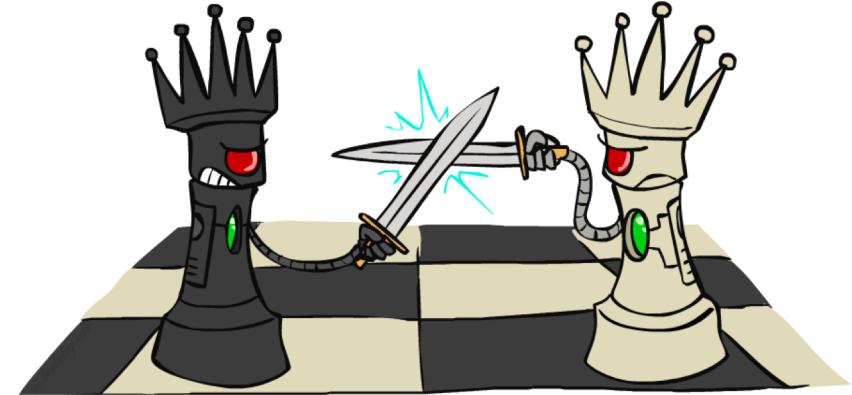
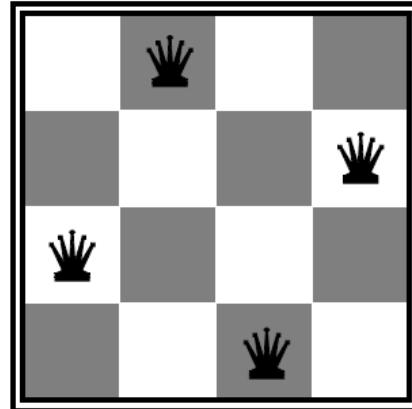
هدف: یک رنگبندی که محدودیتها را در نظر بگیرد، مثلا:

$\{WA=\text{red}, NT=\text{green}, Q=\text{red}, NSW=\text{green}, V=\text{red}, SA=\text{blue}, T=\text{green}\}$

N-Queens :مثال

■ Formulation 1:

- Variables: X_{ij}
- Domains: $\{0, 1\}$
- Constraints



$$\forall i, j, k \quad (X_{ij}, X_{ik}) \in \{(0, 0), (0, 1), (1, 0)\}$$

$$\forall i, j, k \quad (X_{ij}, X_{kj}) \in \{(0, 0), (0, 1), (1, 0)\}$$

$$\forall i, j, k \quad (X_{ij}, X_{i+k,j+k}) \in \{(0, 0), (0, 1), (1, 0)\}$$

$$\forall i, j, k \quad (X_{ij}, X_{i+k,j-k}) \in \{(0, 0), (0, 1), (1, 0)\}$$

$$\sum_{i,j} X_{ij} = N$$

N-Queens : مثال

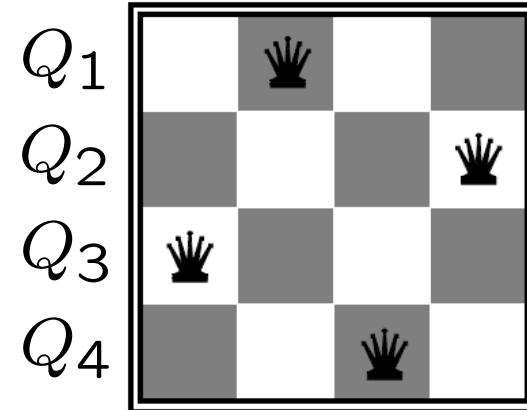
■ Formulation 2:

- Variables: Q_k
- Domains: $\{1, 2, 3, \dots, N\}$
- Constraints:

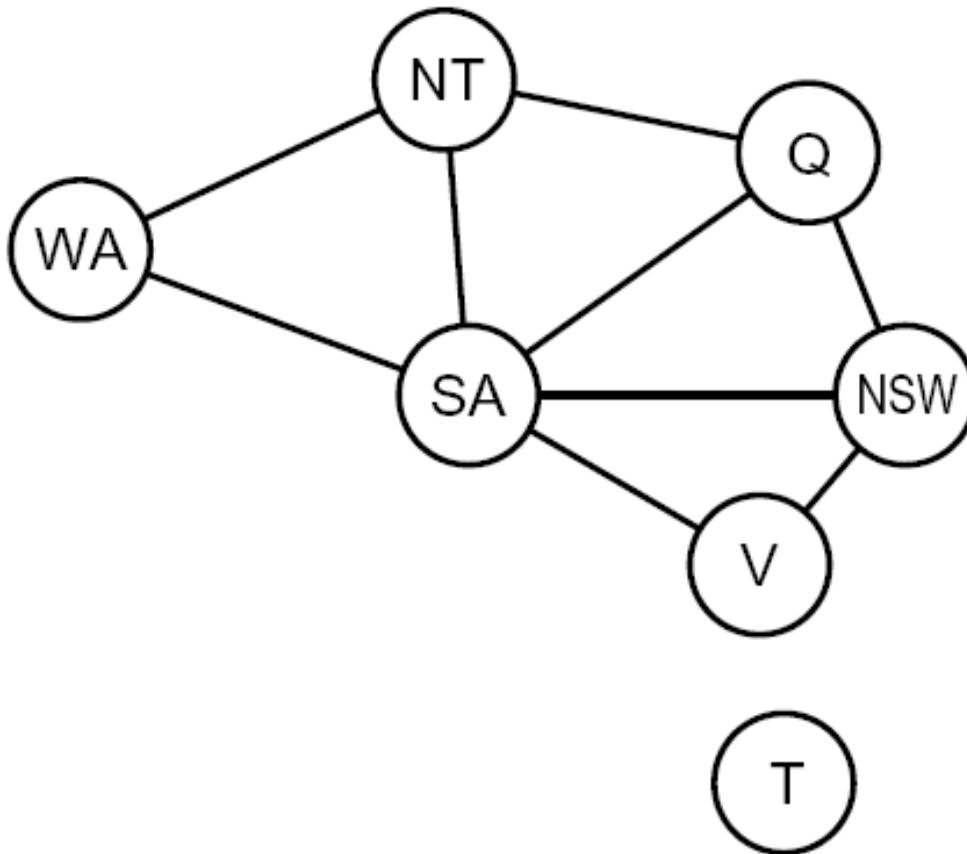
Implicit: $\forall i, j \text{ non-threatening}(Q_i, Q_j)$

Explicit: $(Q_1, Q_2) \in \{(1, 3), (1, 4), \dots\}$

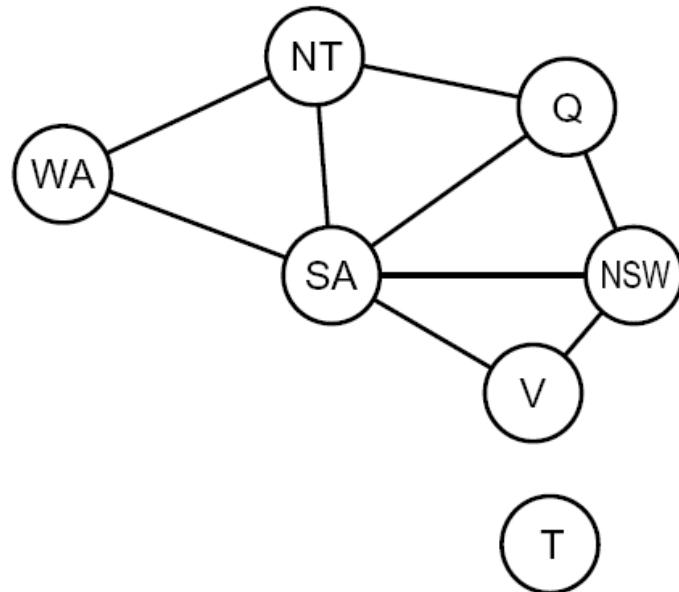
... .



گراف محدودیت – Constraint Graphs



گراف محدودیت – Constraint Graphs



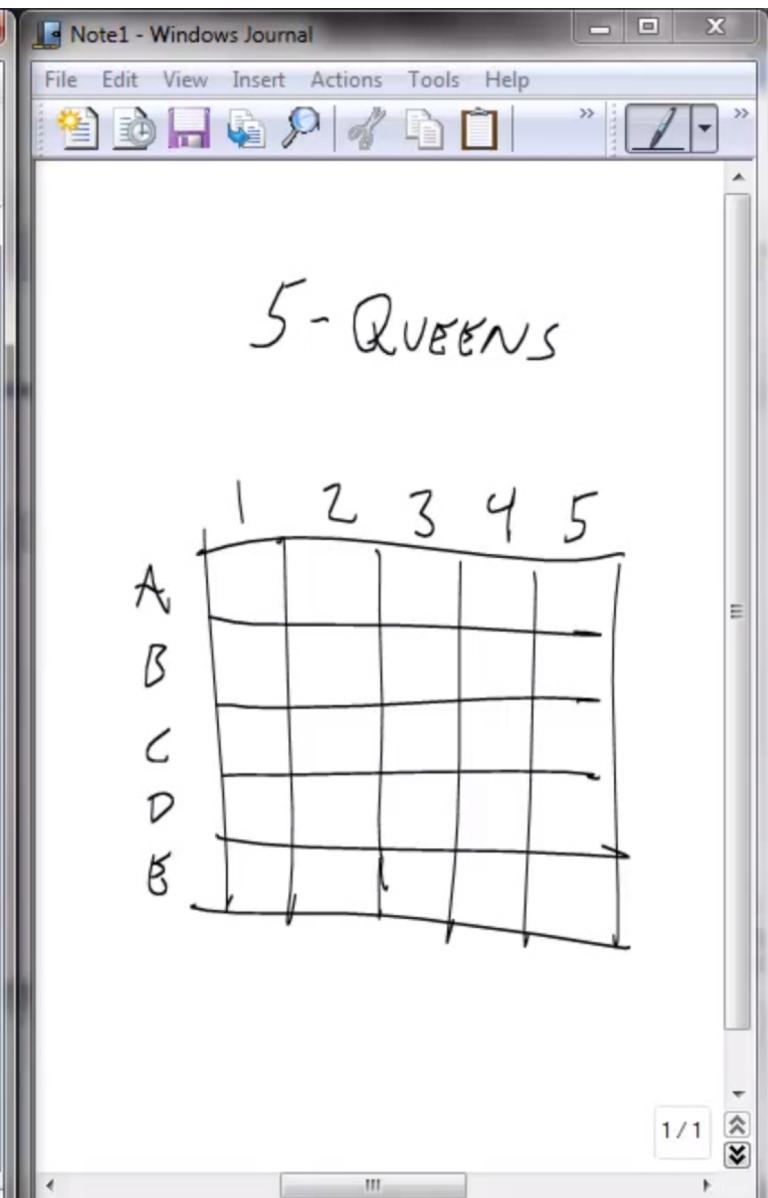
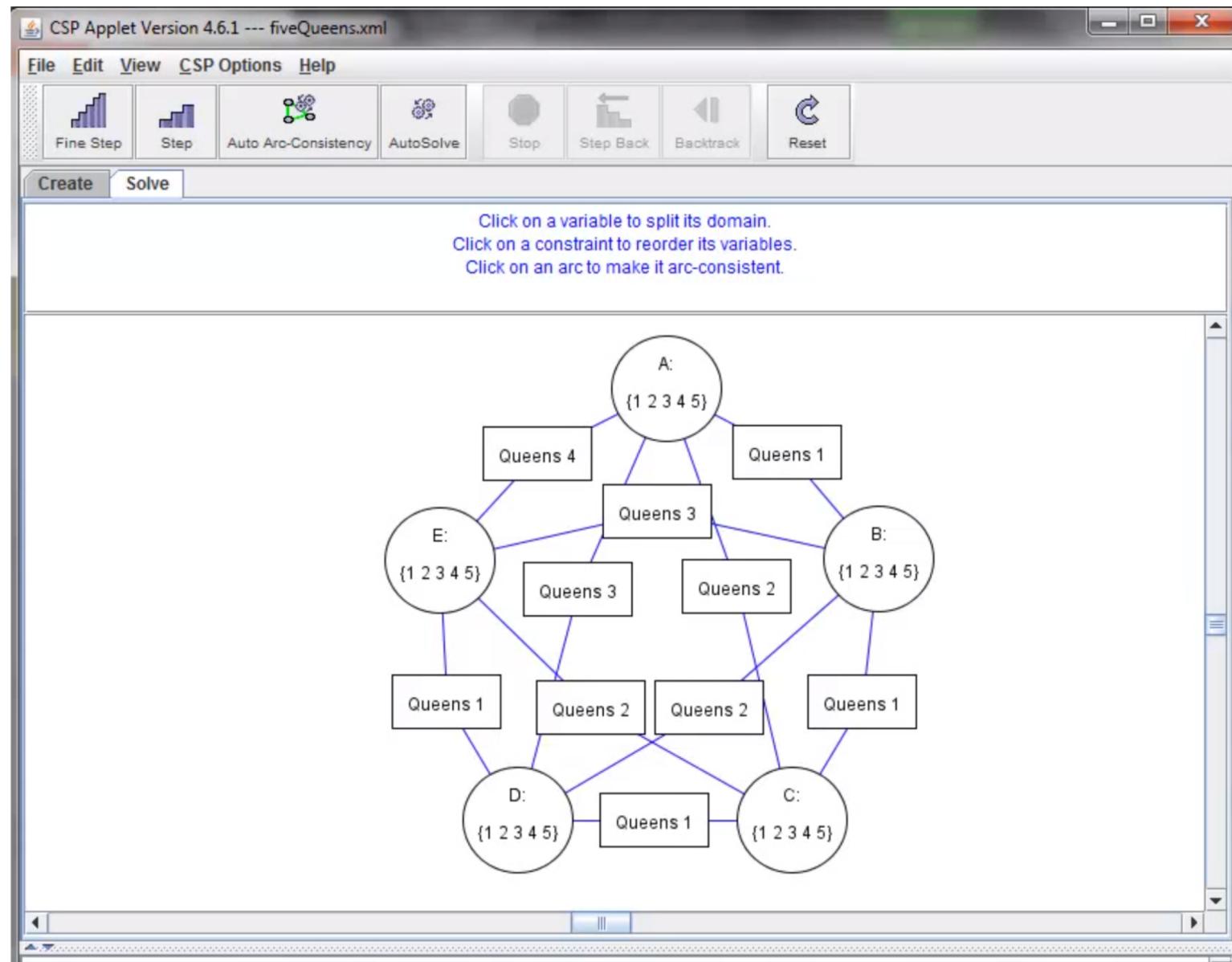
CSP باینری: هر محدودیت نهایتا برای دو متغیر تعریف می‌شود

گراف محدودیت باینری: گره‌ها متغیرها، لبه‌ها محدودیت‌ها

الگوریتم‌ها می‌توانند ساختار گراف را بررسی می‌کنند تا جستجو را تسريع کنند.

مثال: تاسمانیا زیرمساله‌ی مستقلی است.

N-Queens دموی



Cryptarithmetic : مثال

$$\begin{array}{r} \text{T} \quad \text{W} \quad \text{O} \\ + \text{T} \quad \text{W} \quad \text{O} \\ \hline \text{F} \quad \text{O} \quad \text{U} \quad \text{R} \end{array}$$



متغيرها

$F \ T \ U \ W \ R \ O \ X_1 \ X_2 \ X_3$

دامنهها

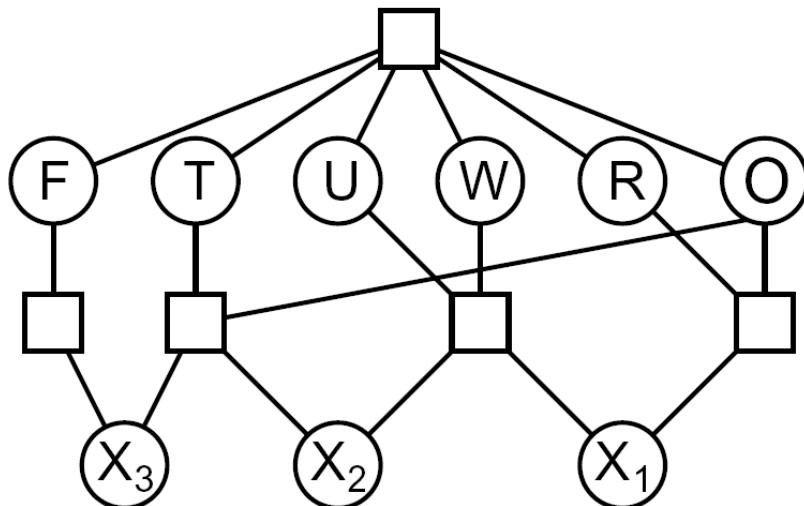
$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

محدودیتها

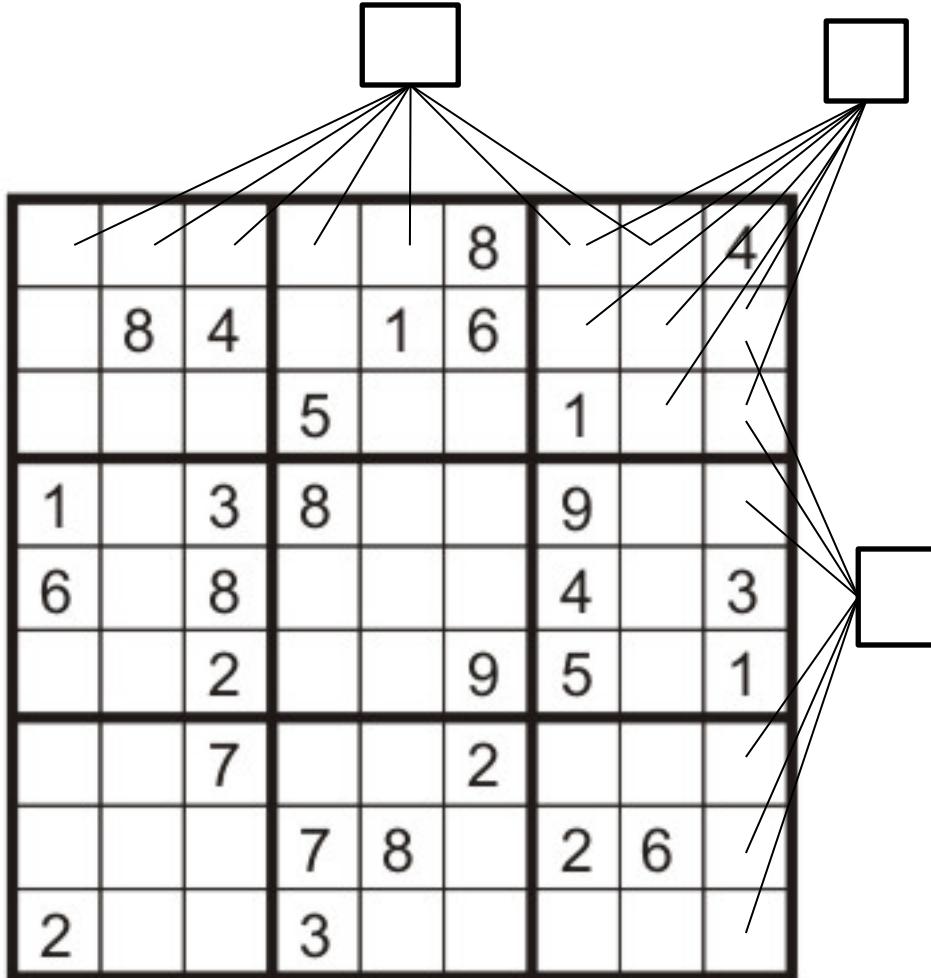
$\text{alldiff}(F, T, U, W, R, O)$

$O + O = R + 10 \cdot X_1$

...



Sudoku : مثال:



متغیرها: خانه‌های خالی

دامنه‌ها: یک عدد صحیح، از ۱ تا ۹

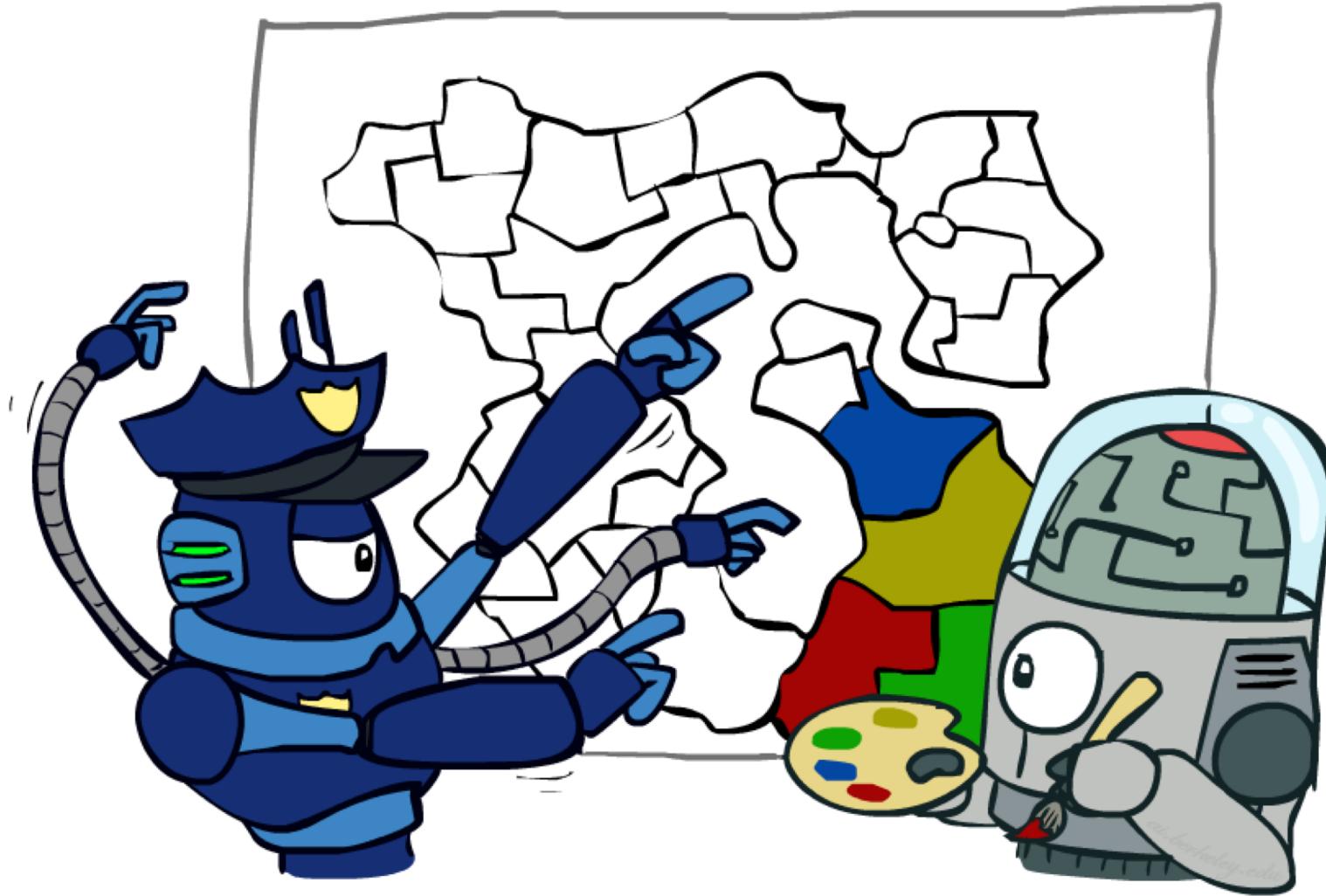
محدودیت‌ها

9-way alldiff for each column

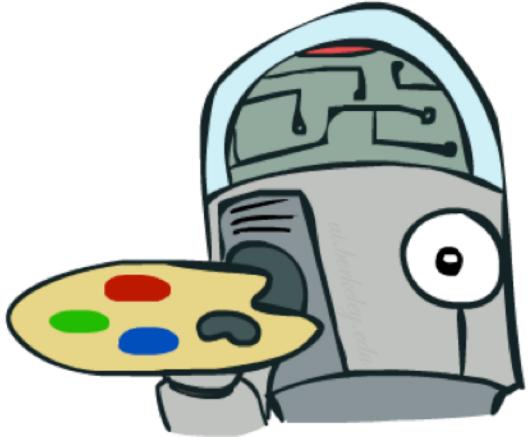
9-way alldiff for each row

9-way alldiff for each region

انواع CSP و محدودیت‌ها



انواع CSP



متغیرهای گستته

دامنه محدود (مثلا ۳ رنگ اصلی)

دامنه نامحدود (مثلا اعداد صحیح)



متغیرهای پیوسته

زمان شروع یا پایان یک پروسه

انواع محدودیت



Unary

محدودیت‌ها فقط بر روی یک متغیر اعمال می‌شوند، مثال:

$SA \neq \text{green}$

Binary

محدودیت‌ها بر روی دو متغیر اعمال می‌شوند، مثال:

$SA \neq WA$

Higher-order

سه یا تعداد بیشتری از متغیرها، مثال:
cryptarithmetic

کاربردهای CSP

تعیین وظیفه: کلاس دادن به استادها

مسائل زمانبندی: تعیین زمان و مکان کلاسها

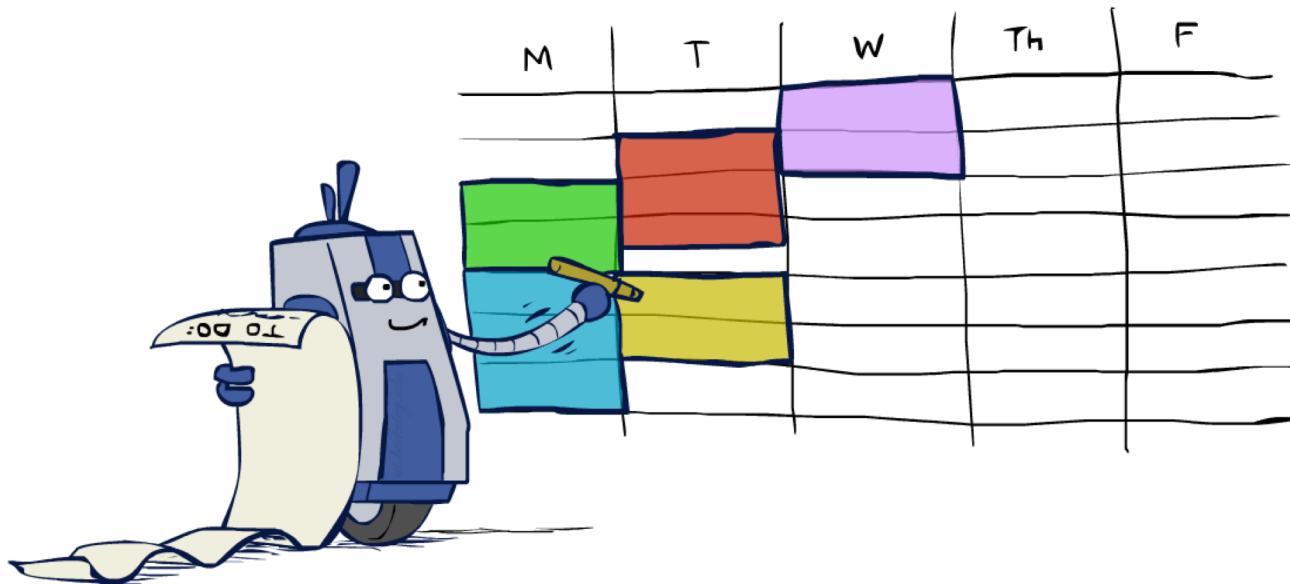
اسمبل کردن کامپیوترها

زمانبندی حمل و نقل

زمانبندی تولید کارخانجات

طراحی مدار

...



حل کردن CSP ها



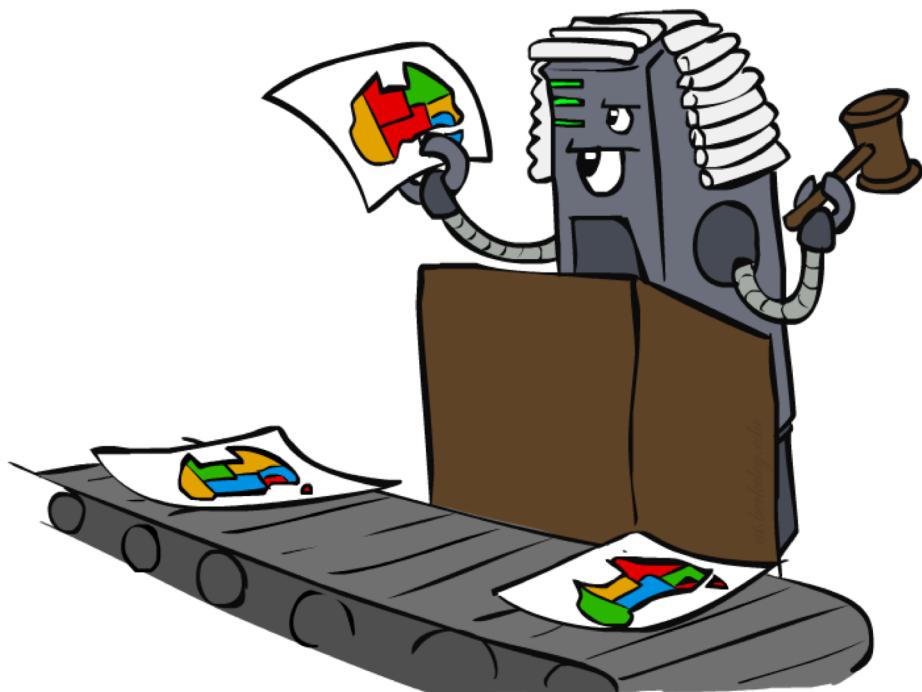
جستجو برای CSP ها

حالات: مقداردهی های انجام شده تا کنون

نقطه‌ی شروع: هیچ مقداری داده نشده - {}

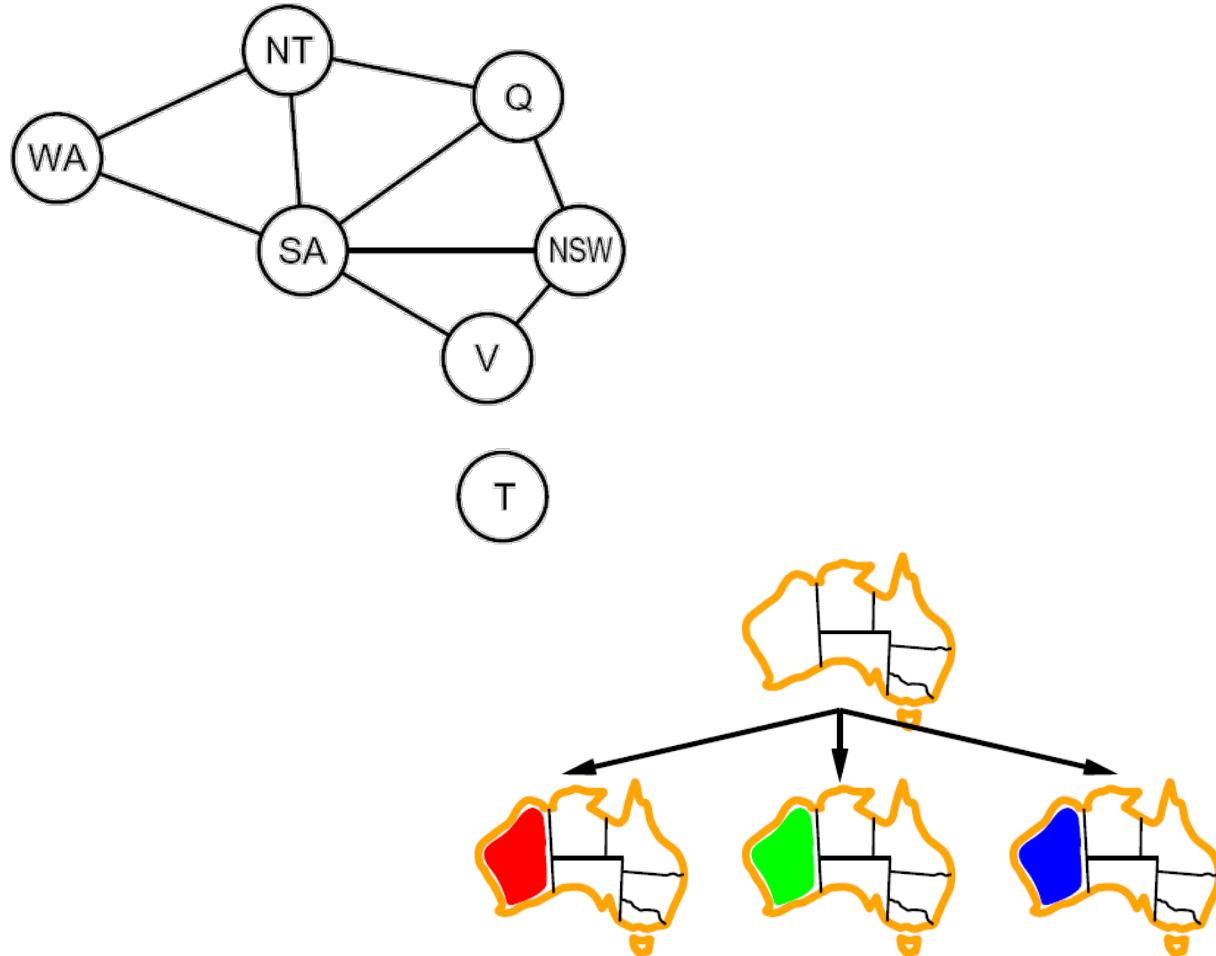
تابع پیشبر: مقداری را به یک متغیر میدهد

نقطه‌ی پایان: تمامی متغیرها مقداردهی شوند در حالی
که از محدودیت‌ها تبعیت شده است



با یک روش ساده شروع می‌کنیم، بعد آن را به
مرور بهبدود می‌دهیم

استفاده از روش جستجو



الگوریتم اول-سطح چطور عمل می کند؟

اول-عمق چطور؟

مشکل کجاست؟

رنگ زدن نقشه - اول عمق

Graph

Simple

Algorithm

Naive Search

Ordering

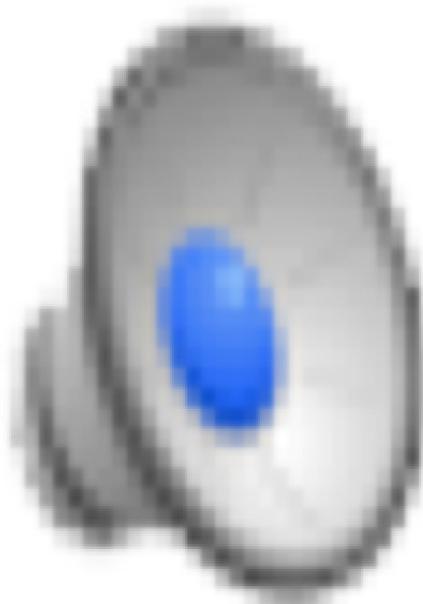
- None
- MRV
- MRV with LCV

Filtering

- None
- Forward Checking
- Arc Consistency

Speed

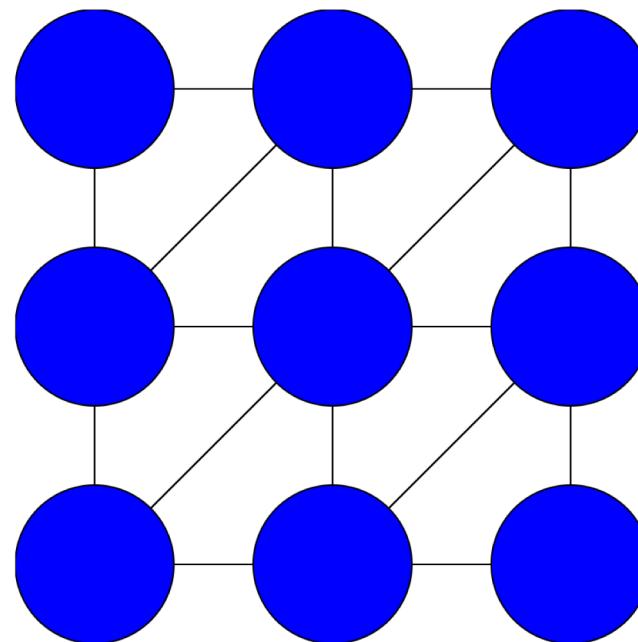
Speedup Frame Delay
1 x 700



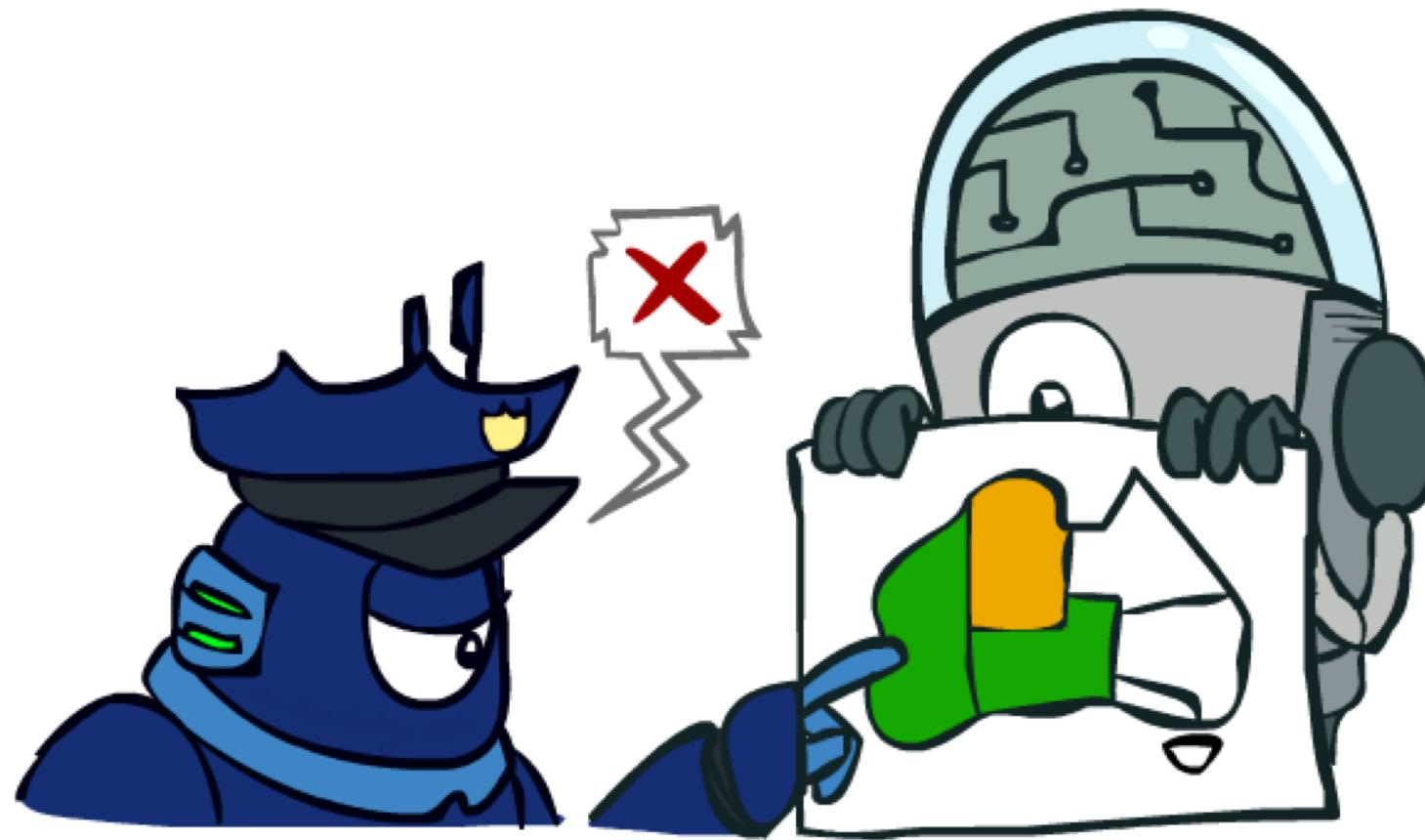
Back tracking demo (javascript): DFS

سوال

برای جلوگیری از چنین حالتی چه کار می‌توان کرد؟



جستجوی عقبگرد – Backtracking



جستجوی عقبگرد – Backtracking

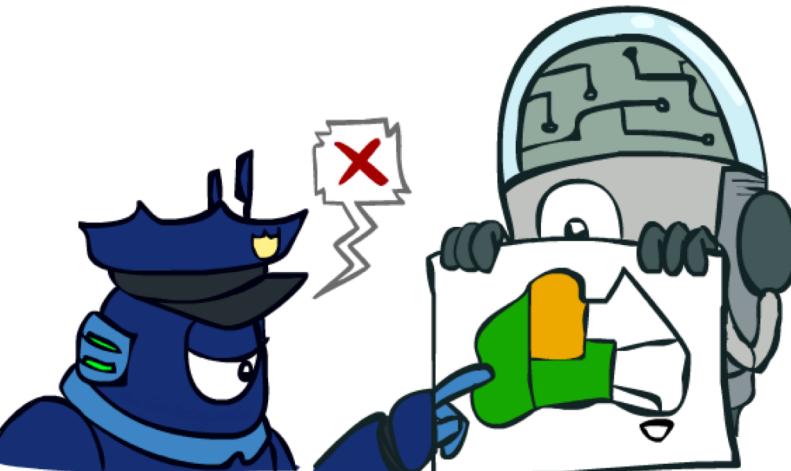
یک روش ابتدایی برای حل مسائل CSP

۱ - یک متغیر را در نظر بگیر و مقداردهی کن

متغیرها را مرتب کن و یکی یکی مقداردهی کن

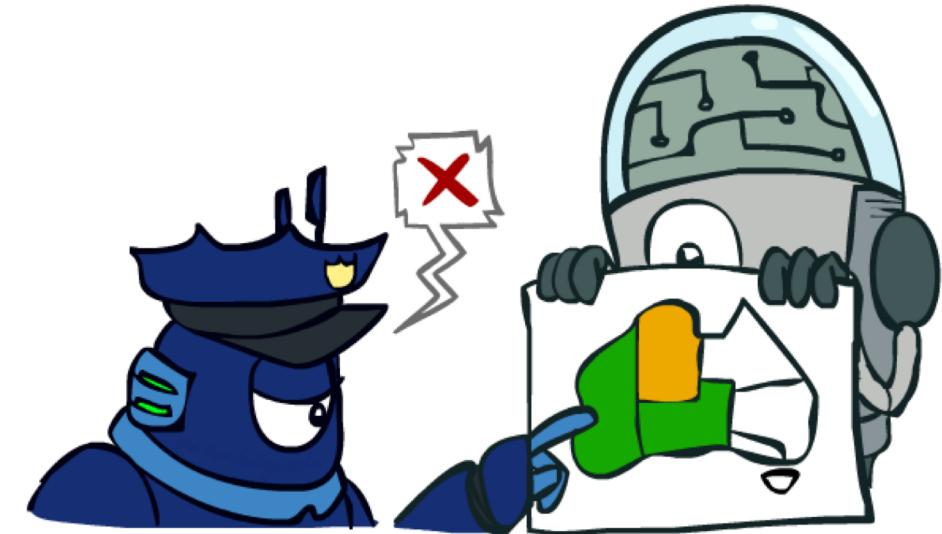
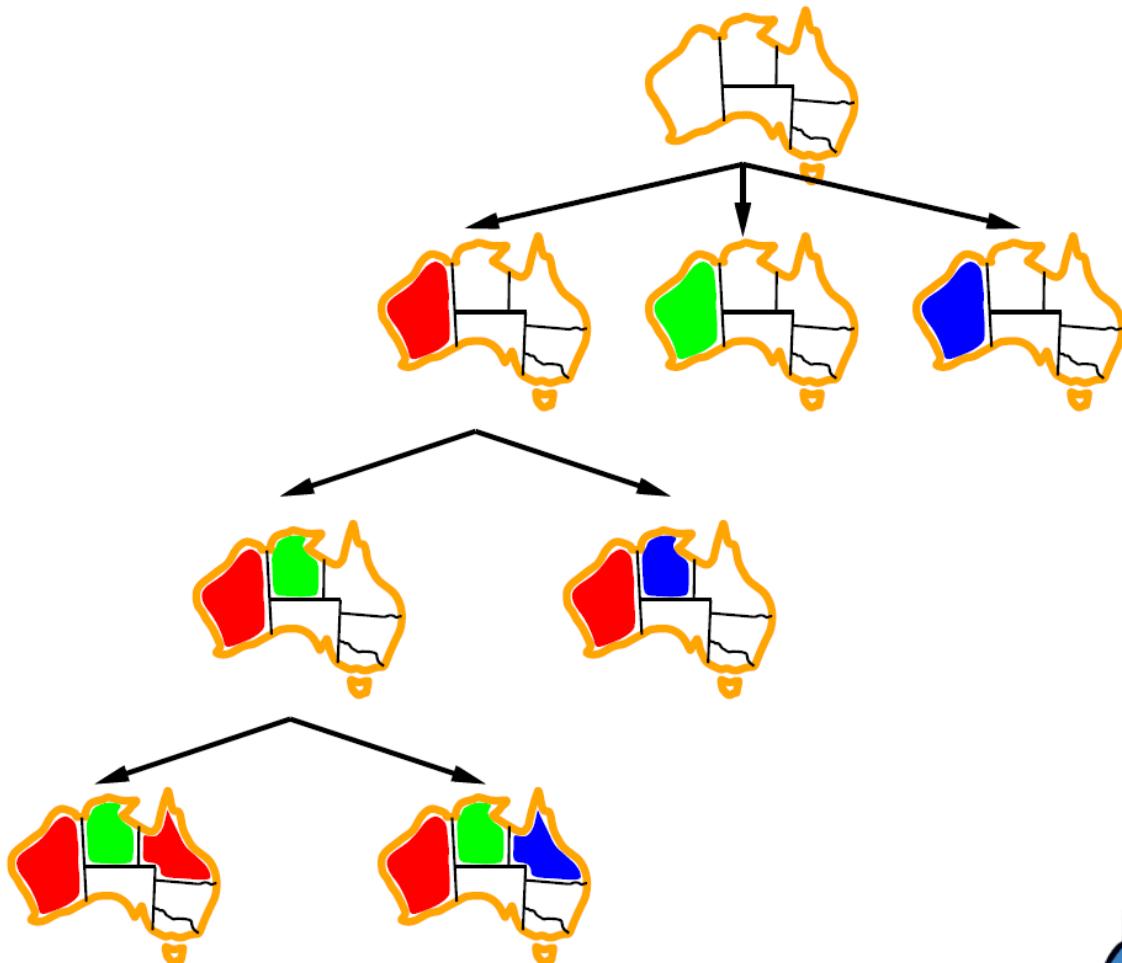
۲ - در حالی که پیش میروی، محدودیتها را چک کن

تنها مقادیری را در نظر بگیر که از محدودیتها تبعیت می‌کنند



جستجوی اول-عمق + ویژگی‌های فوق =
Backtracking search
می‌تواند N-queens را با حدود ۲۵ حرکت حل کند

مثال Backtracking



جستجوی Backtracking

```
function BACKTRACKING-SEARCH(csp) returns solution/failure
    return RECURSIVE-BACKTRACKING({ }, csp)
function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/failure
    if assignment is complete then return assignment
    var  $\leftarrow$  SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment given CONSTRAINTS[csp] then
            add {var = value} to assignment
            result  $\leftarrow$  RECURSIVE-BACKTRACKING(assignment, csp)
            if result  $\neq$  failure then return result
            remove {var = value} from assignment
    return failure
```

- Backtracking = DFS + variable-ordering + fail-on-violation

مثال رنگ زدن - backtracking

Graph

Simple

Algorithm

Backtracking

Ordering

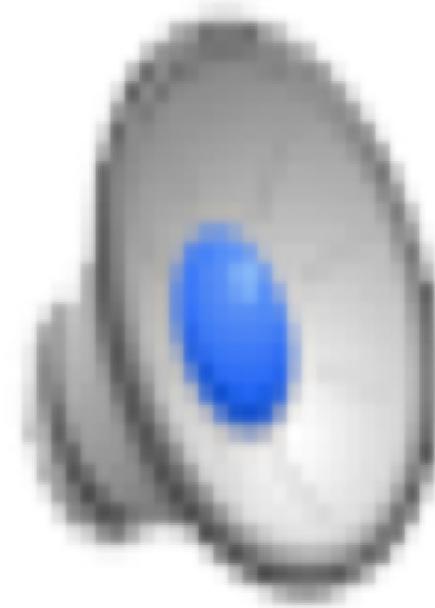
- None
- MRV
- MRV with LCV

Filtering

- None
- Forward Checking
- Arc Consistency

Speed

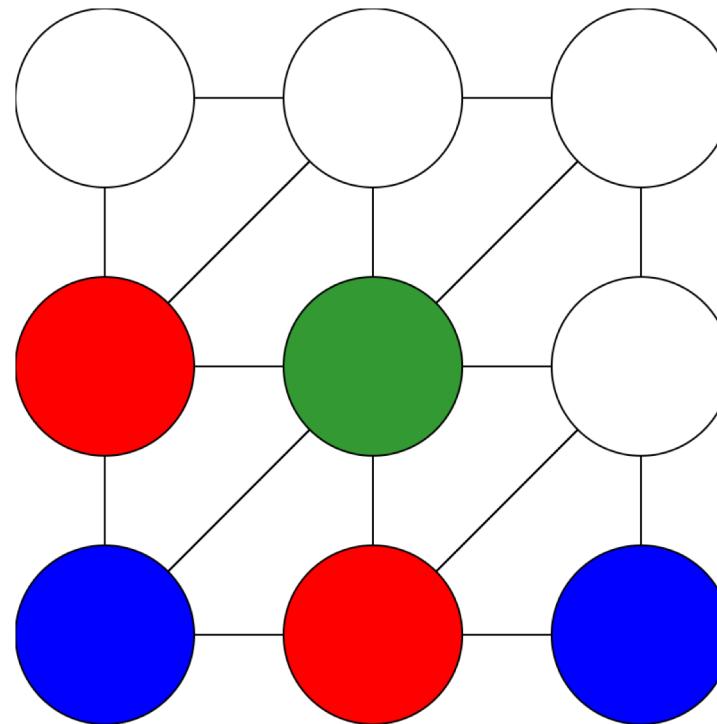
Speedup Frame Delay
1 x 700



Back tracking demo (javascript): Backtracking

Backtracking

آیا می‌توان از بروز چنین مشکلاتی پیشگیری کرد؟



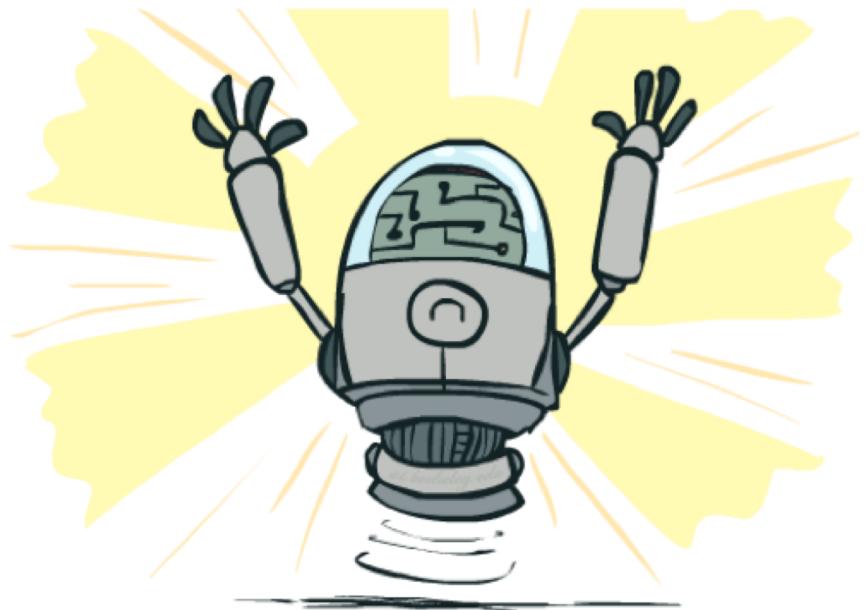
بهبود Backtracking

با استفاده از ایده‌های ساده می‌توان سرعت را به مقدار قابل توجهی بهبود داد

فیلتر کردن: آیا می‌توان از بروز مشکلات پیشگیری کرد؟

مرتب‌سازی: اول کدام متغیر را مقداردهی کنیم؟

ساختار: آیا از ساختار مساله می‌توان حسن استفاده را برد؟

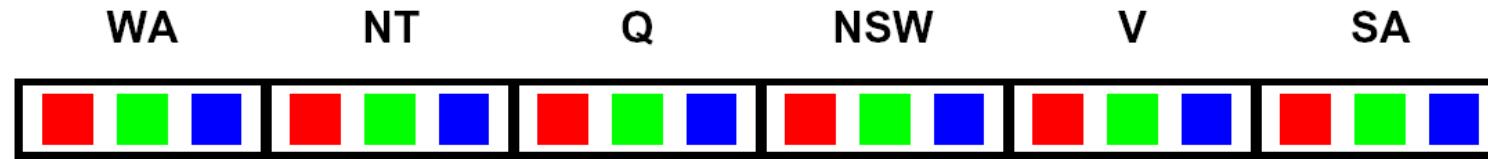


فیلتر کردن



فیلتر کردن با Forward Checking

بعد از هر مقداردهی، مقادیر ناممکن را از دامنه‌های دیگر حذف کن



اطلاعات را از متغیرهای مقداردهی شده به متغیرهای بدون مقدار انتشار می‌دهد

Backtracking with Forward Checking : مثال

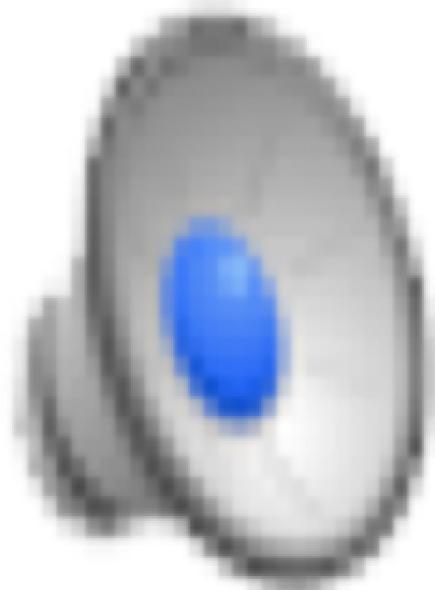
Graph
Simple

Algorithm
Backtracking

Ordering
 None
 MRV
 MRV with LCV

Filtering
 None
 Forward Checking
 Arc Consistency

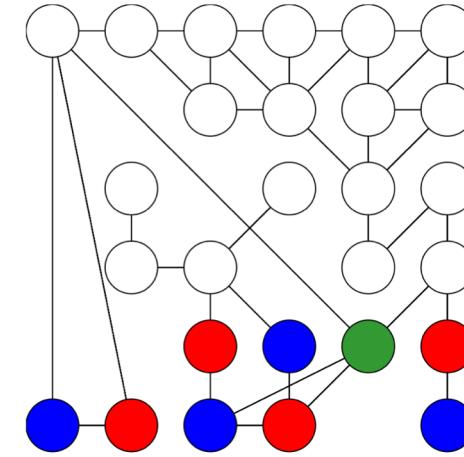
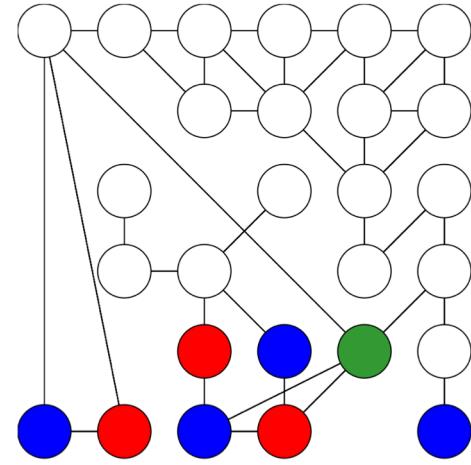
Speed
Speedup Frame Delay
1 x 700



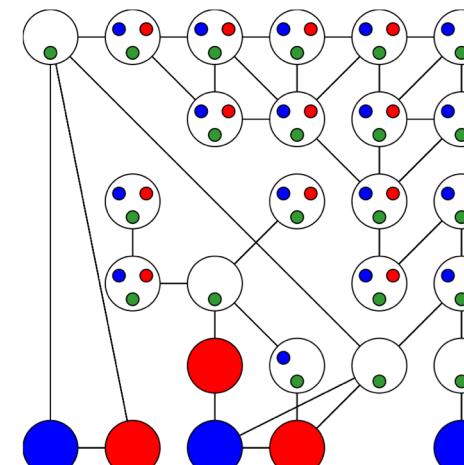
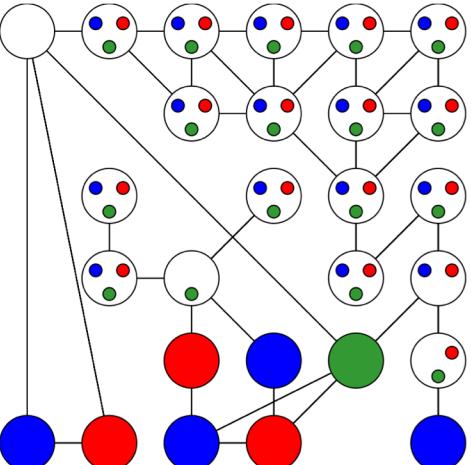
Back tracking demo (javascript): Backtracking with FC

Backtracking with Forward Checking

Backtracking

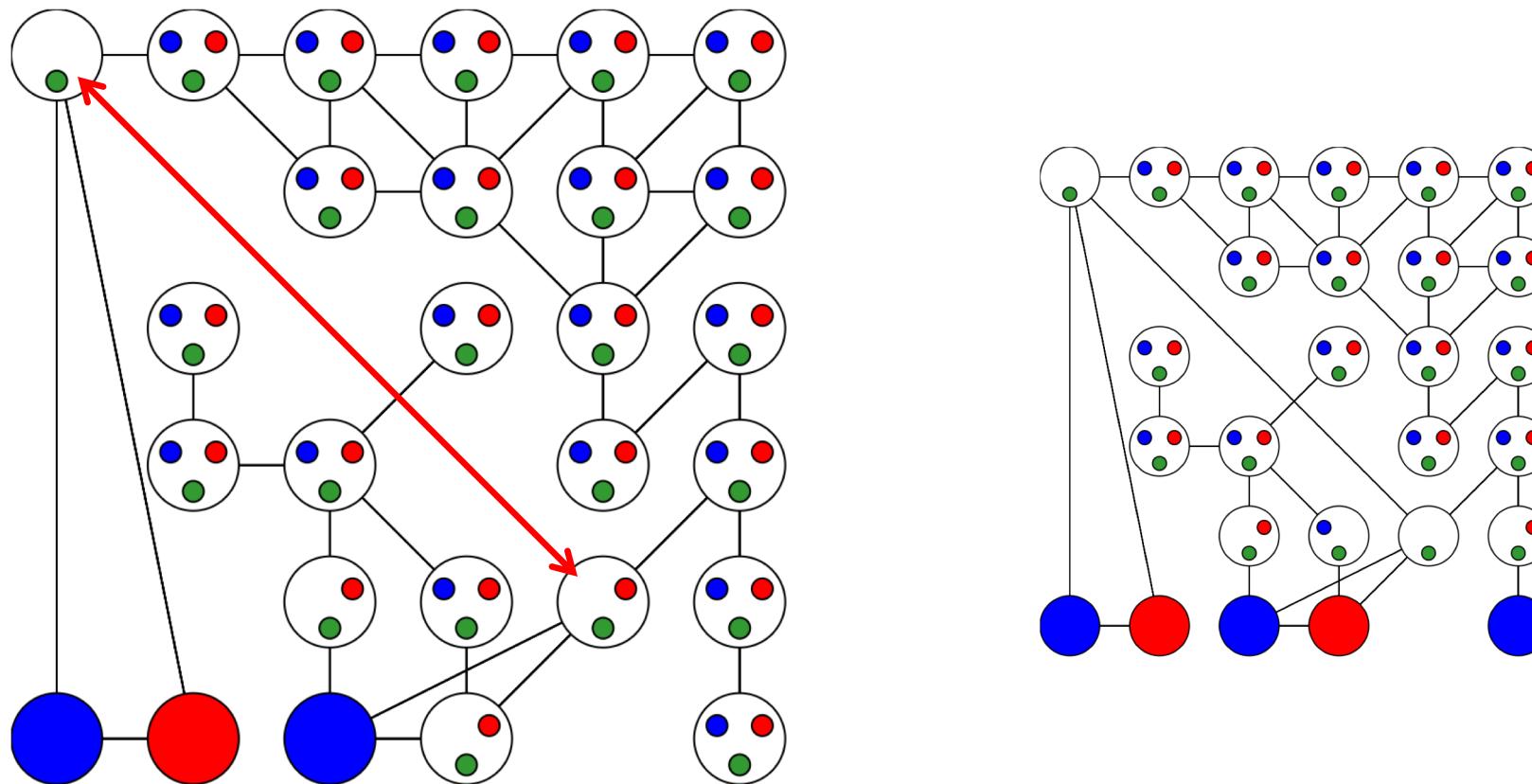


Backtracking with
Forward Checking



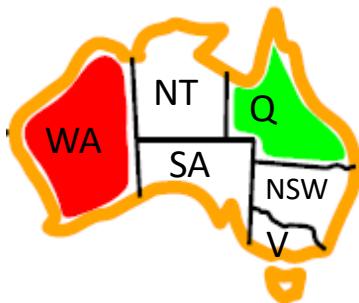
مشکل Forward checking

نمیتواند مشکلات را از قبل پیش بینی کند



مشکل Forward checking

نمیتواند مشکلات را از قبل پیش‌بینی کند Forward checking



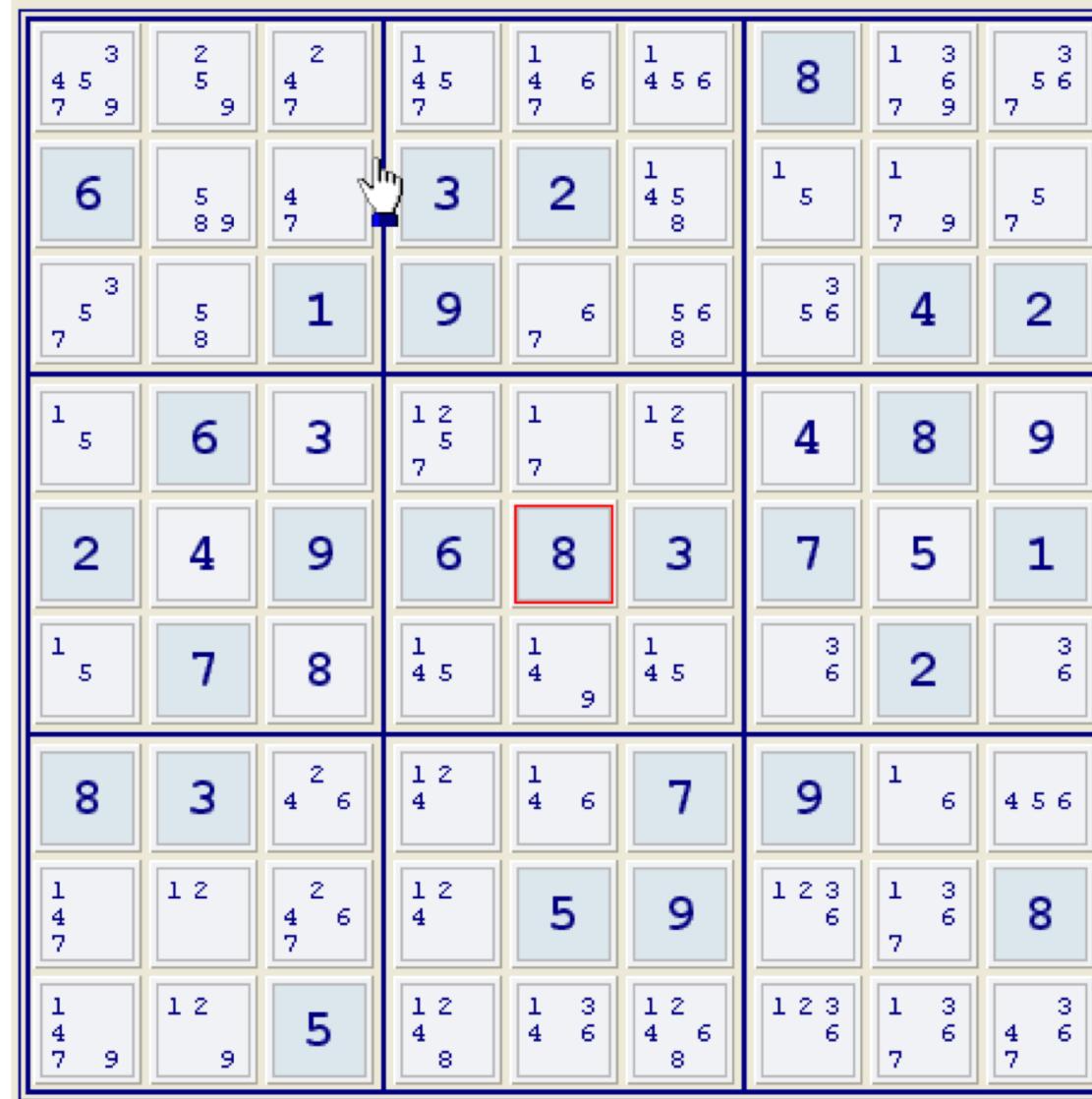
WA	NT	Q	NSW	V	SA
Red	Green	Blue	Red	Green	Blue
Red	White	Green	Blue	Red	Green
Red	White	Blue	Green	Red	Blue

و NT و SA نمیتوانند هر دو آبی باشند.
چرا نتوانستیم این را پیش‌بینی کنیم؟

مشکل اینجاست که فوروارد چکینگ فقط همسایگان مستقیم را در نظر می‌گیرد

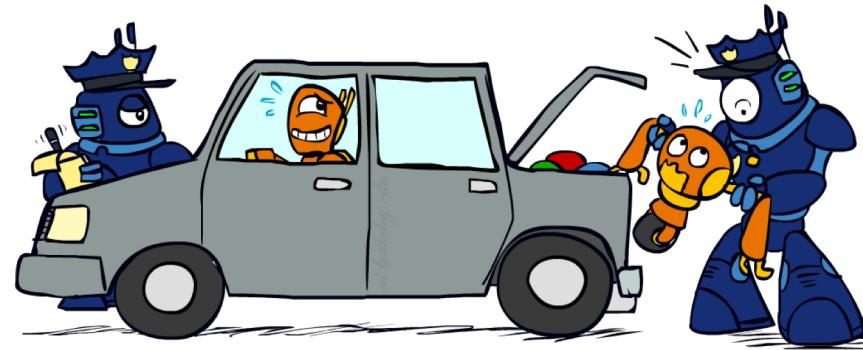
Forward checking - سودوکو

Forward checking - سودوكو



سازگاری کمان – Arc Consistency

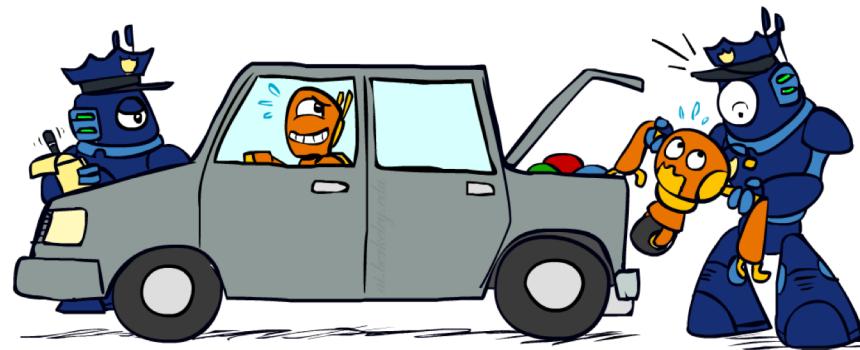
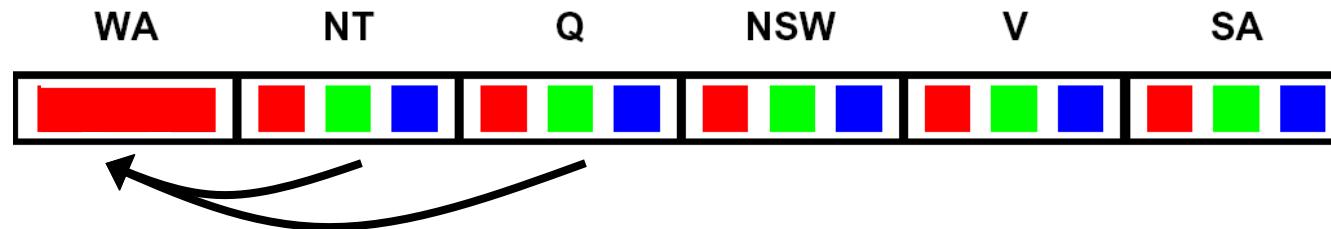
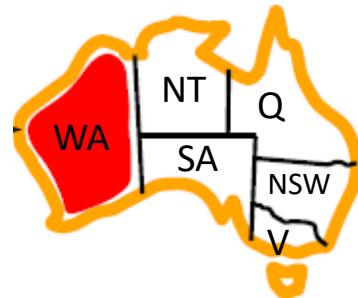
Constraint propagation



«سازگاری کمان» تمام گراف را چک می کند

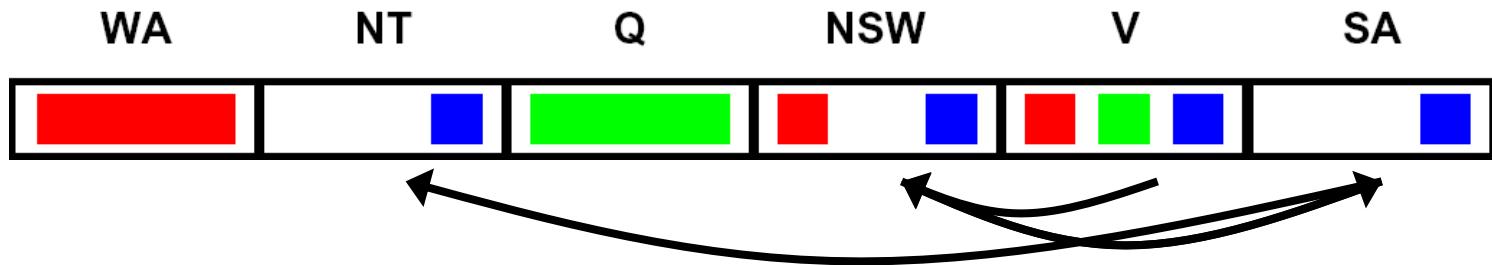
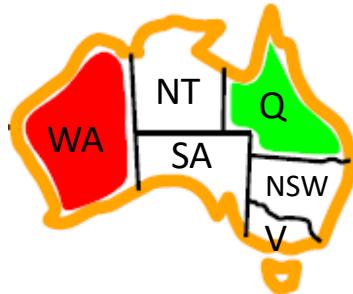
سازگار بودن تک کمان

کمان $Y \rightarrow X$ در صورتی سازگار است که به ازای هر X در انتهای کمان یک y در وجود داشته باشد که از محدودیت تبعیت کند



Delete from the tail!

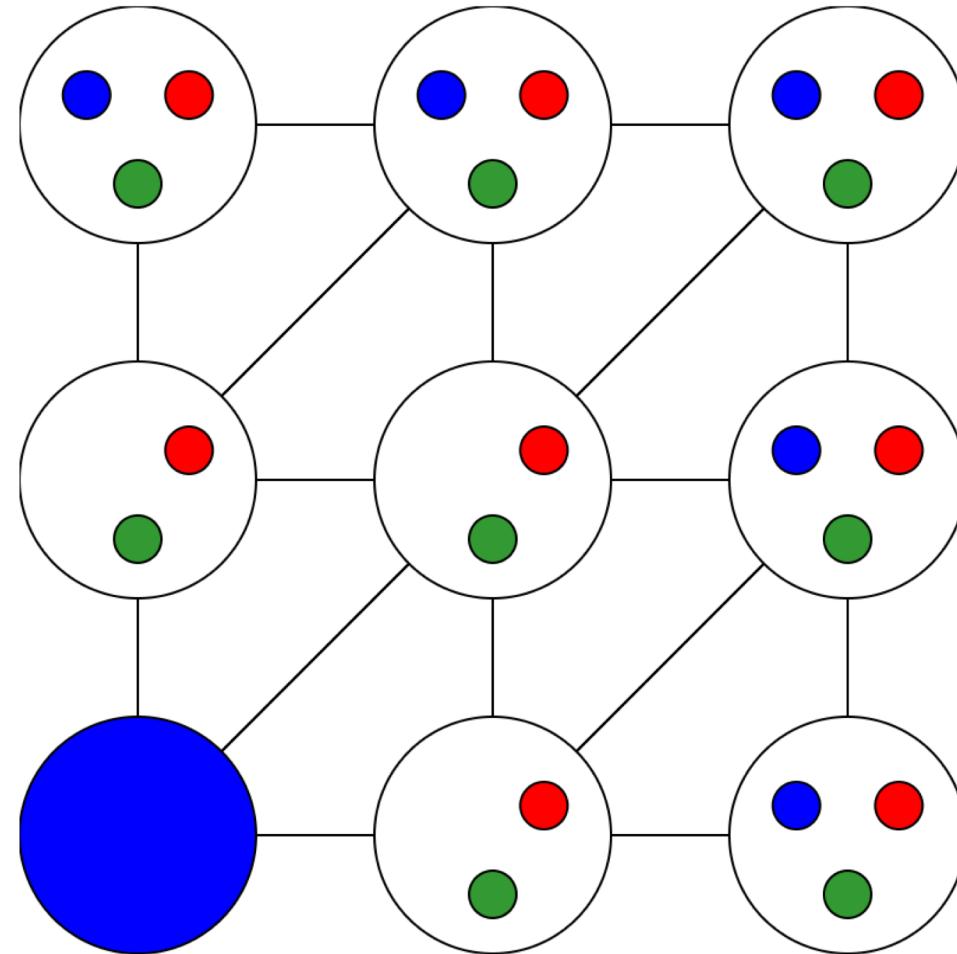
سازگار بودن تمامی کمانها



*Remember: Delete
from the tail!*

اگر X مقداری را از دست دهد، تمامی همسایگان X باید دوباره چک شوند
هر متغیری که مقدار آن عوض می شود، تمامی کمانهای ورودی اش را دوباره چک کن
چک کردن برای سازگاری، زودتر از Forward checking مشکلات را پیش بینی می کند.

تمرین: سازگاری کمان



Forward Checking ↗ Backtracking

Graph

Complex

Algorithm

Backtracking

Ordering

- None
- MRV
- MRV with LCV

Filtering

- None
- Forward Checking
- Arc Consistency

Speed

Speedup Frame Delay
1 x 700



Arc consistency Ł Backtracking

Graph

Algorithm

Ordering

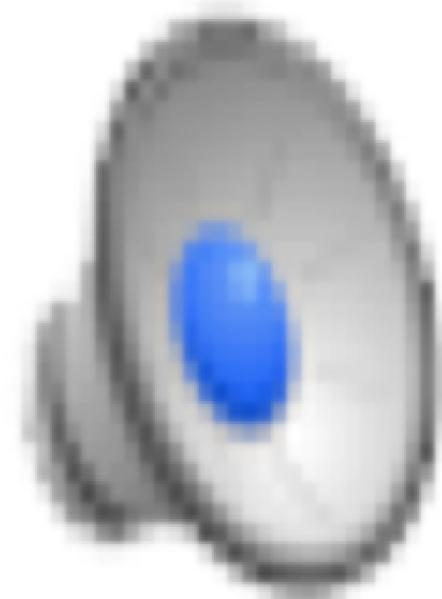
- None
- MRV
- MRV with LCV

Filtering

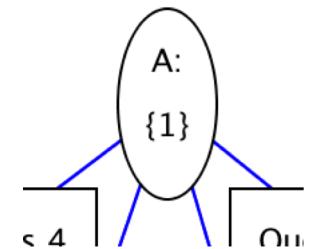
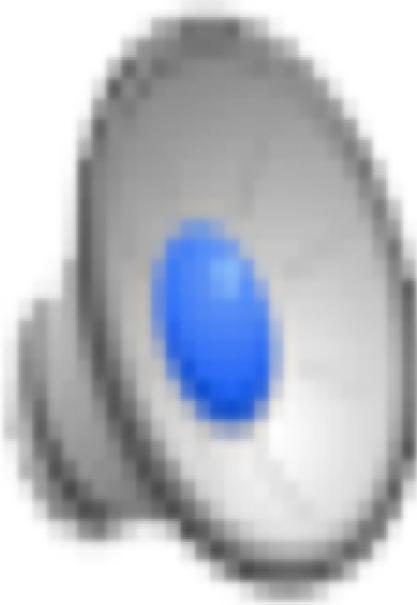
- None
- Forward Checking
- Arc Consistency

Speed

Speedup	Frame Delay
<input type="text" value="1"/> x	<input type="text" value="700"/>



مثال سازگاری کمان – N-Queens



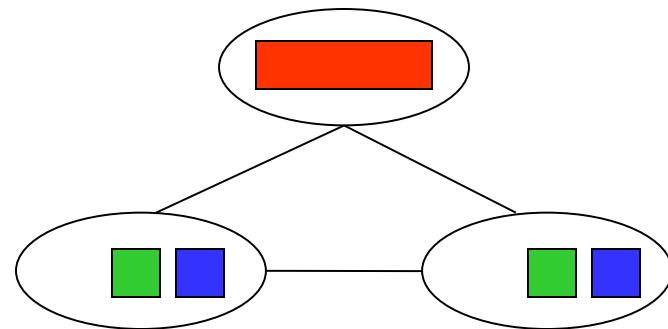
SpaceX



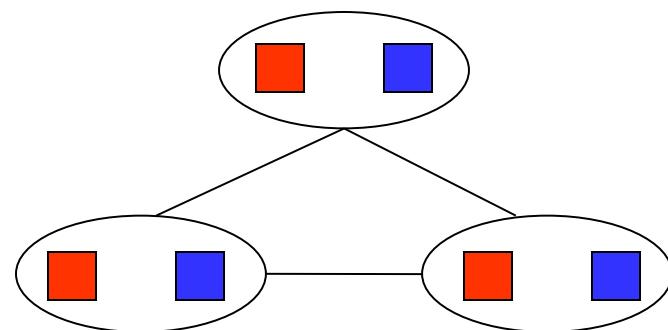
SpaceX



محدودیت‌های سازگاری کمان



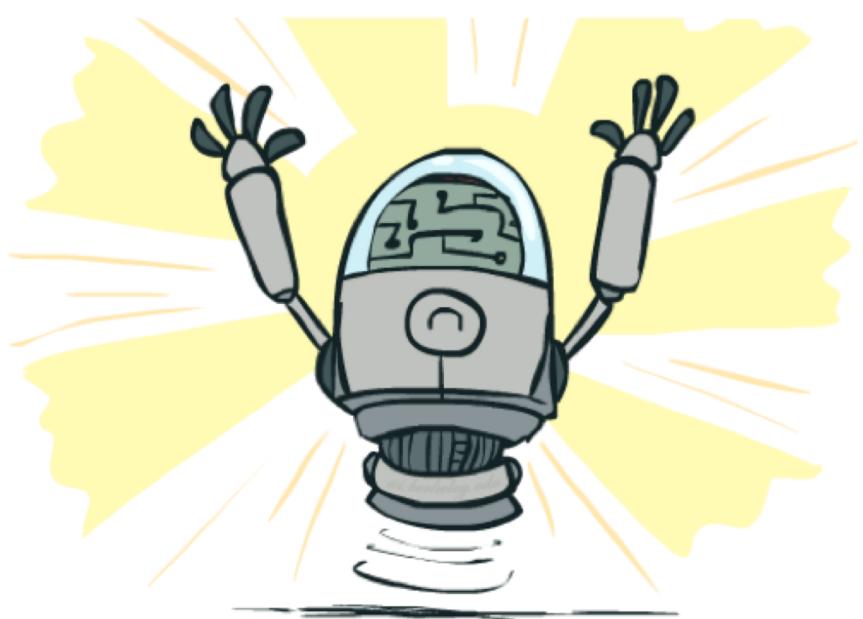
بعد از اعمال سازگاری کمان:
ممکن است جوابی باقی نماند!



کمان‌ها سازگارند، ولی آیا جوابی داریم؟

مشکل کجاست؟

بهبود Backtracking



فیلتر کردن: آیا می‌توان از بروز مشکلات پیشگیری کرد؟

- Forward checking
- Arc consistency

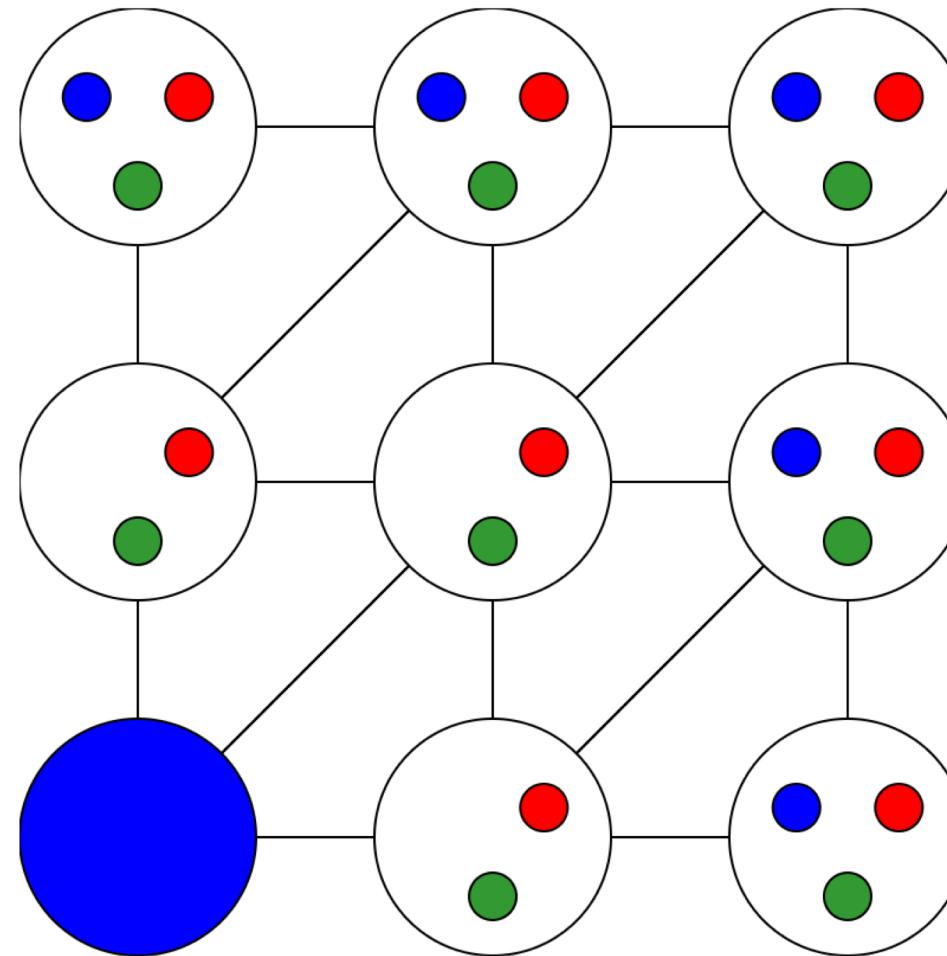
مرتب سازی: اول کدام متغیر را مقداردهی کنیم؟

ساختار: آیا از ساختار مساله می‌توان حسن استفاده را برد؟

مرتبسازی – Ordering



مرتبسازی – Ordering



Minimum Remaining Values: مرتبسازی

متغیری را برای مقداردهی برگزین که کمترین تعداد انتخاب را دارد (کوچکترین دامنه ممکن)



چرا کمترین؟

تمرین - MRV

به ترتیب، کدام گره‌ها را اول بسط بدھیم؟

The visualization shows a complex constraint satisfaction problem graph with nodes and arcs. The graph consists of several nodes arranged in layers, connected by arcs. Some nodes contain three colored dots (blue, red, green), while others are empty circles or filled with a single color. A legend indicates that blue dots represent blue values, red dots represent red values, and green dots represent green values.

Graph: Complex

Algorithm: Backtracking

Ordering:

- None
- MRV
- MRV with LCV

Filtering:

- None
- Forward Checking
- Arc Consistency

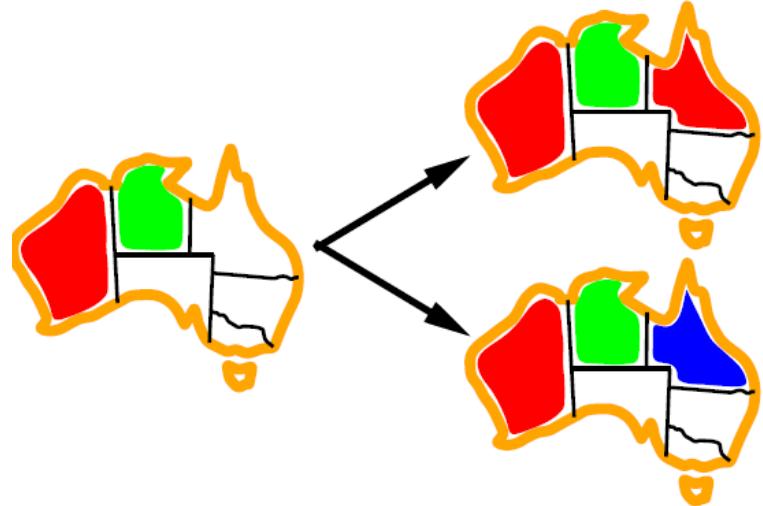
Speed:

Speedup X Frame Delay

Control buttons at the bottom:

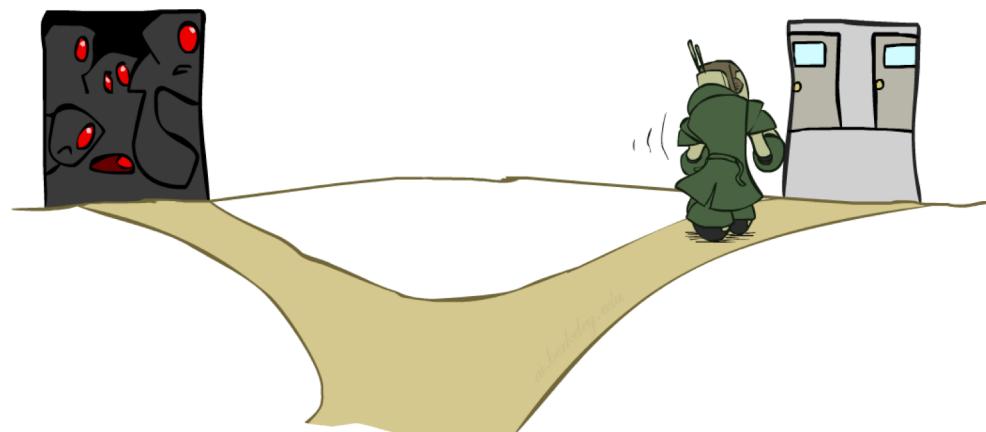
- Reset
- Prev
- Pause
- Next
- Play
- Faster

مرتب سازی: Least Constraining Value



از بین مقادیر ممکن برای یک متغیر MRV
مقداری را انتخاب کن که برای متغیرهای دیگر کمترین
حدودیت ایجاد می‌کند
یعنی، برای متغیرهای دیگر انتخاب‌های بیشتری باقی می‌گذارد

چرا کمترین؟



با استفاده از این ایده‌های ساده می‌توان ۱۰۰۰-کوپینز را
هم حل کرد!

Backtracking + Forward Checking + Ordering

Graph

Simple

Algorithm

Backtracking

Ordering

- None
- MRV
- MRV with LCV

Filtering

- None
- Forward Checking
- Arc Consistency

Speed

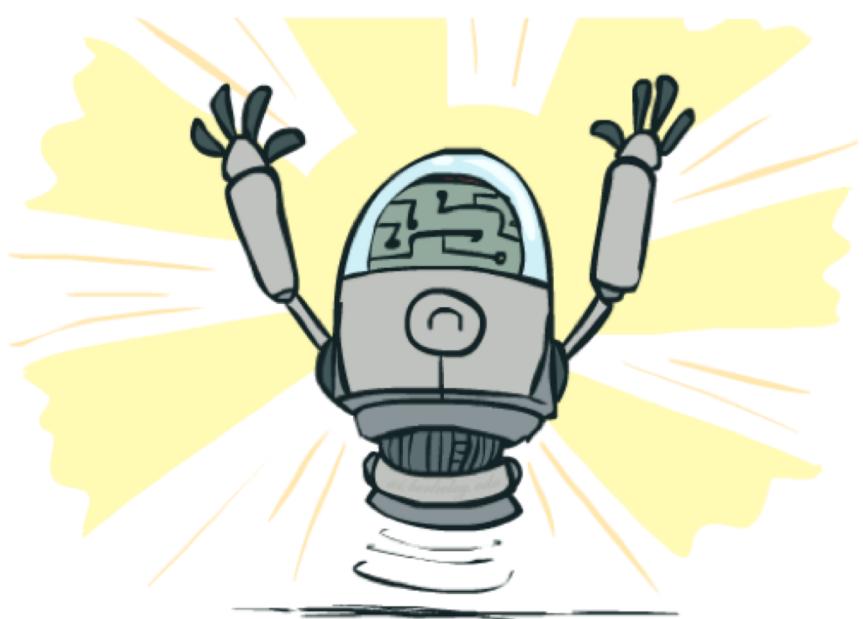
Speedup Frame Delay
1 x 700

کوییز

8	5		E			3	7
2		C	D	4			
	A	B	6	1		5	
4			1	9			
		3			1		
		3	7			4	
6			7	2			
		8				7	
4	1			9	6		

- ۱- مقادیر ممکن برای هر یک از خانه‌های رنگی را مشخص کنید
- ۲- کدامیک از خانه‌ها **Minimum Remaining Value** هستند؟
- ۳- از میان این خانه‌ها (یکی را انتخاب کرده) و مقدار **Least Constraining** را (برای همان بلوک) بیابید.

بهبود Backtracking



فیلتر کردن: آیا می‌توان از بروز مشکلات پیشگیری کرد؟

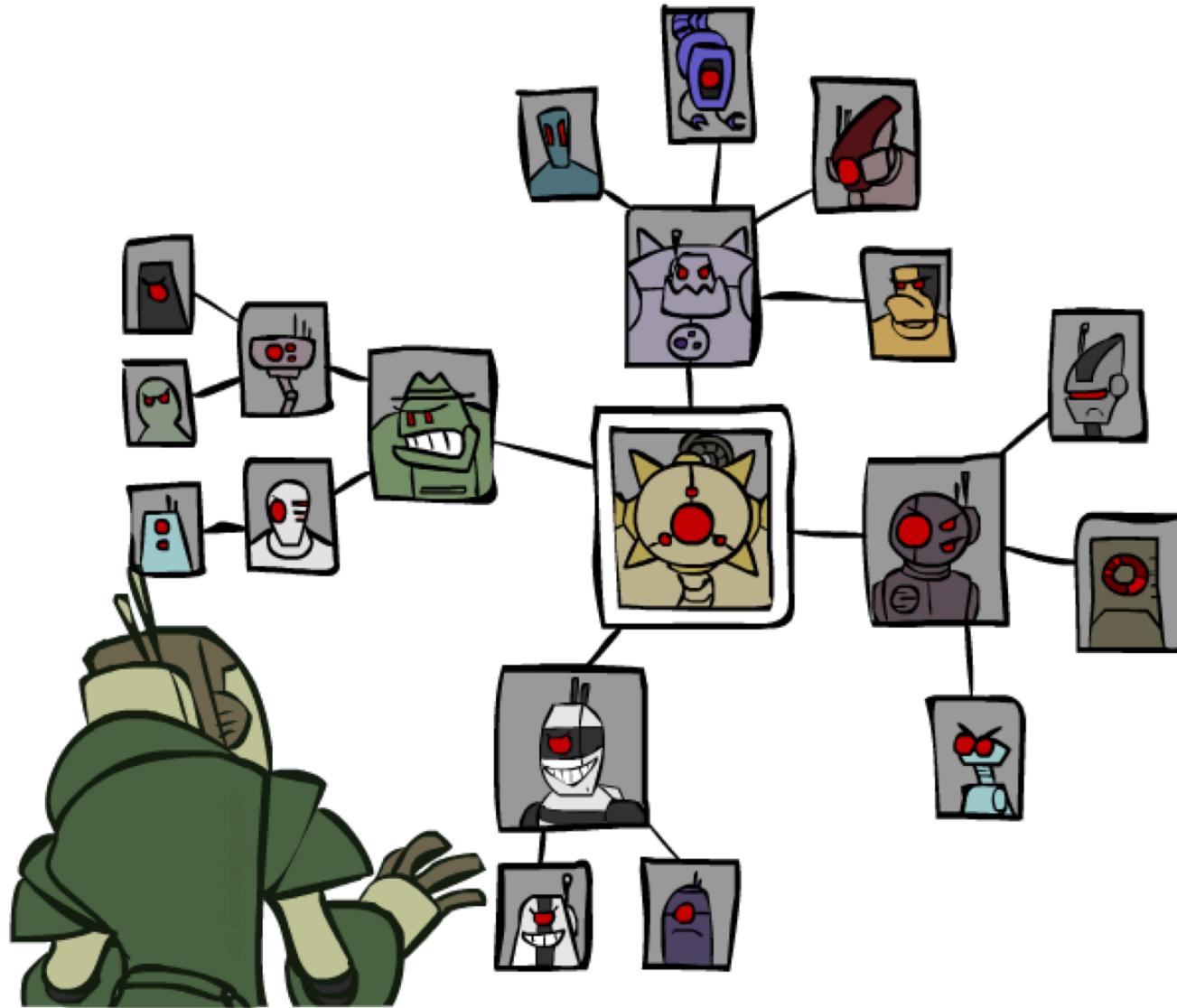
- Forward checking
- Arc consistency

مرتب‌سازی: اول کدام متغیر را مقداردهی کنیم؟

- MRV
- LCV

ساختار: آیا از ساختار مساله می‌توان حسن استفاده را برد؟

ساختار



ساختار مساله

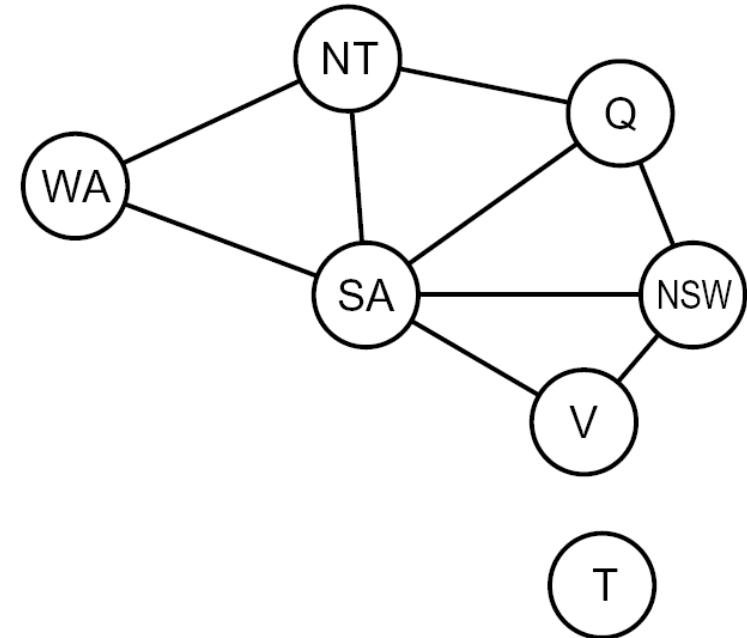
دنبال مسائل مستقل می‌گردیم

مثال: تاسمانیا محدودیتی (اتصالی) ندارد

در صورت تبدیل یک مساله با n گره به c مساله مستقل (دامنه d)

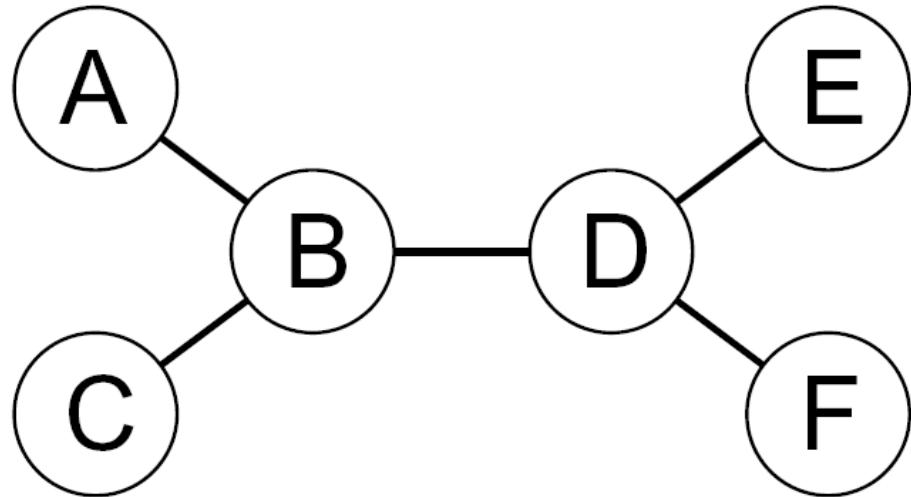
هزینه‌ی بدترین حالت: $O((n/c)(d^c))$, linear in n

مثال: $n = 80, d = 2, c = 20$



- $2^{80} = 4 \text{ billion years at 10 million nodes/sec}$
- $(4)(2^{20}) = 0.4 \text{ seconds at 10 million nodes/sec}$

CSP ها با ساختار درختی



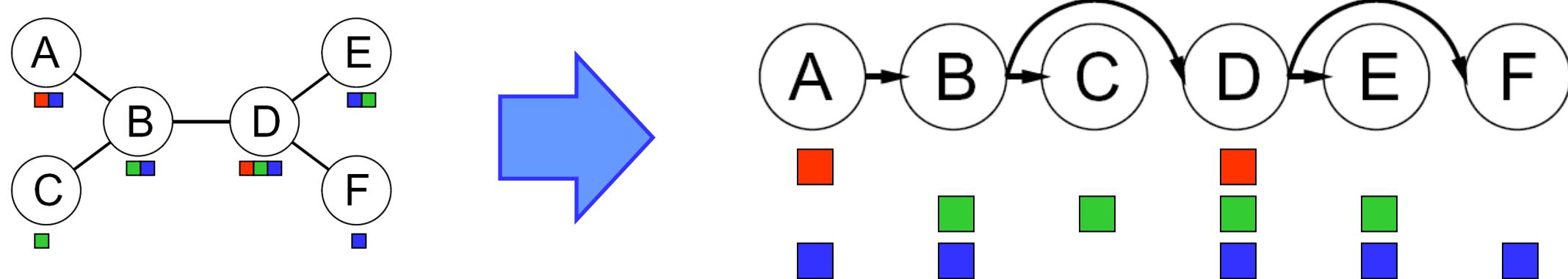
قضیه: در صورتی که لوپی نداشته باشیم، CSP را می‌توان در زمان $O(n d^2)$ حل کرد

در مقایسه با حالت کلی که $O(d^n)$

CSP ها با ساختار درختی

الگوریتم:

- Order: Choose a root variable, order variables so that parents precede children



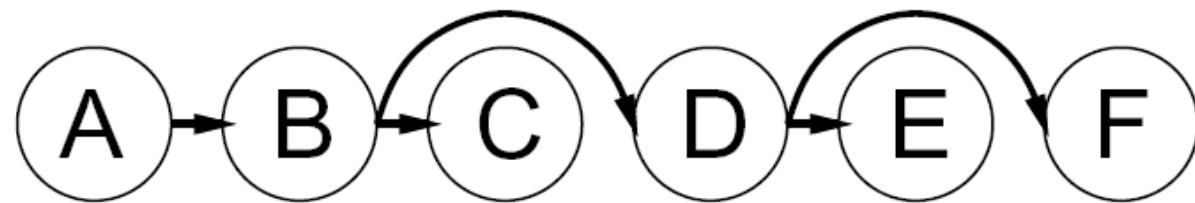
- Remove backward: For $i = n : 2$, apply RemoveInconsistent($\text{Parent}(X_i), X_i$)
- Assign forward: For $i = 1 : n$, assign X_i consistently with $\text{Parent}(X_i)$

هزینه: $O(n d^2)$



CSP ها با ساختار درختی

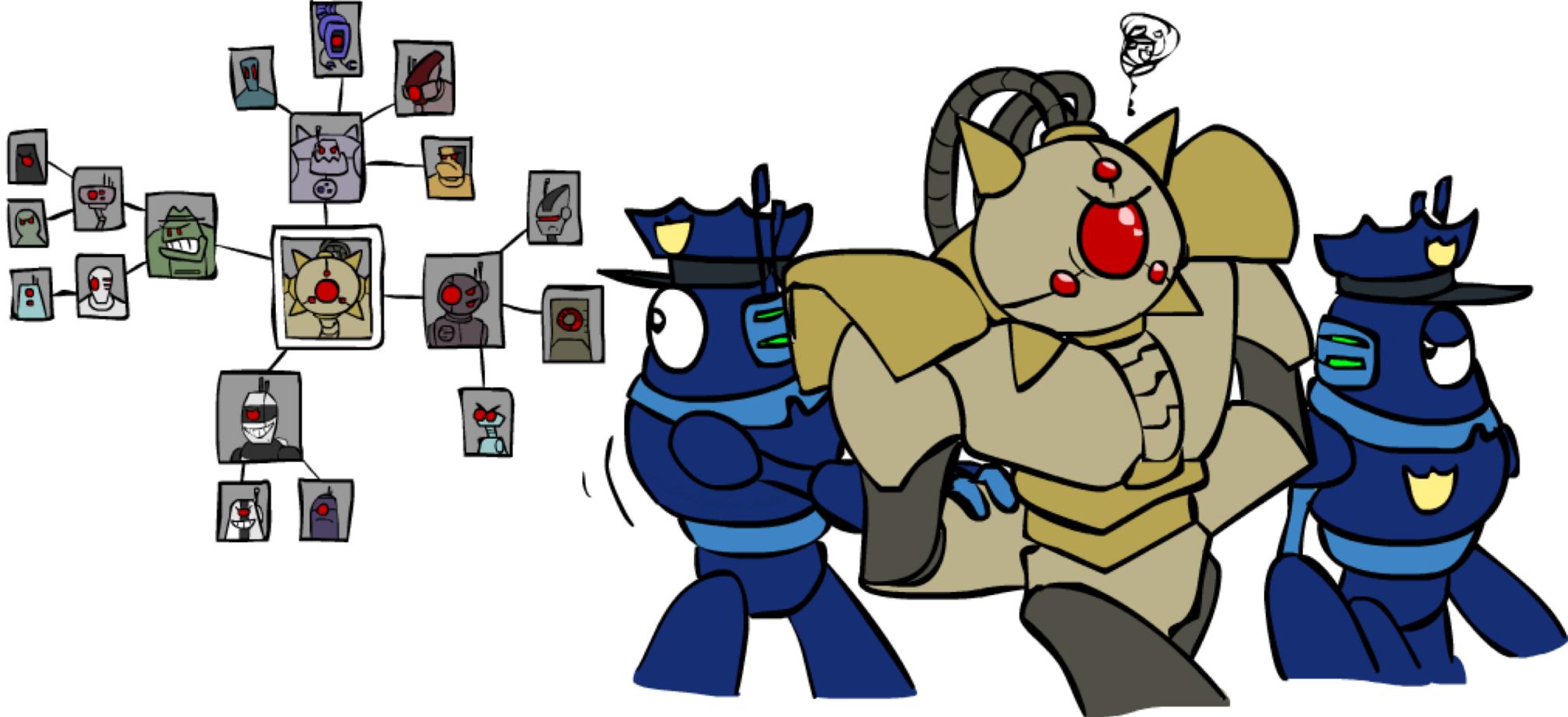
- Claim 1: After backward pass, all root-to-leaf arcs are consistent
 - Proof: Each $X \rightarrow Y$ was made consistent at one point and Y 's domain could not have been reduced thereafter (because Y 's children were processed before Y)



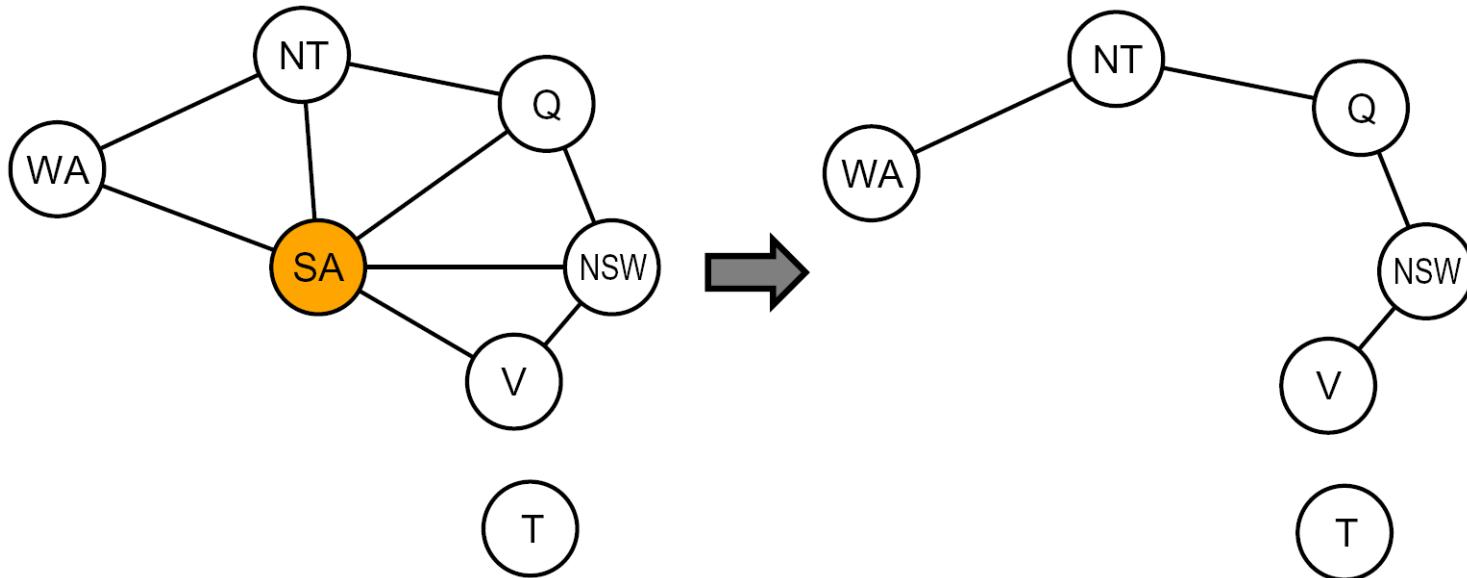
- Claim 2: If root-to-leaf arcs are consistent, forward assignment will not backtrack

چرا این الگوریتم روی گراف (درخت لوب دار) کار نمی کند؟

بھبود ساختار



CSP ها با ساختار نزدیک به درخت



- Cutset conditioning:

تمام مقادیر ممکن SA را در نظر بگیر
 SA را حذف کن
 چند مساله‌ی مستقل از هم داریم، جدأگانه حلشان کن!

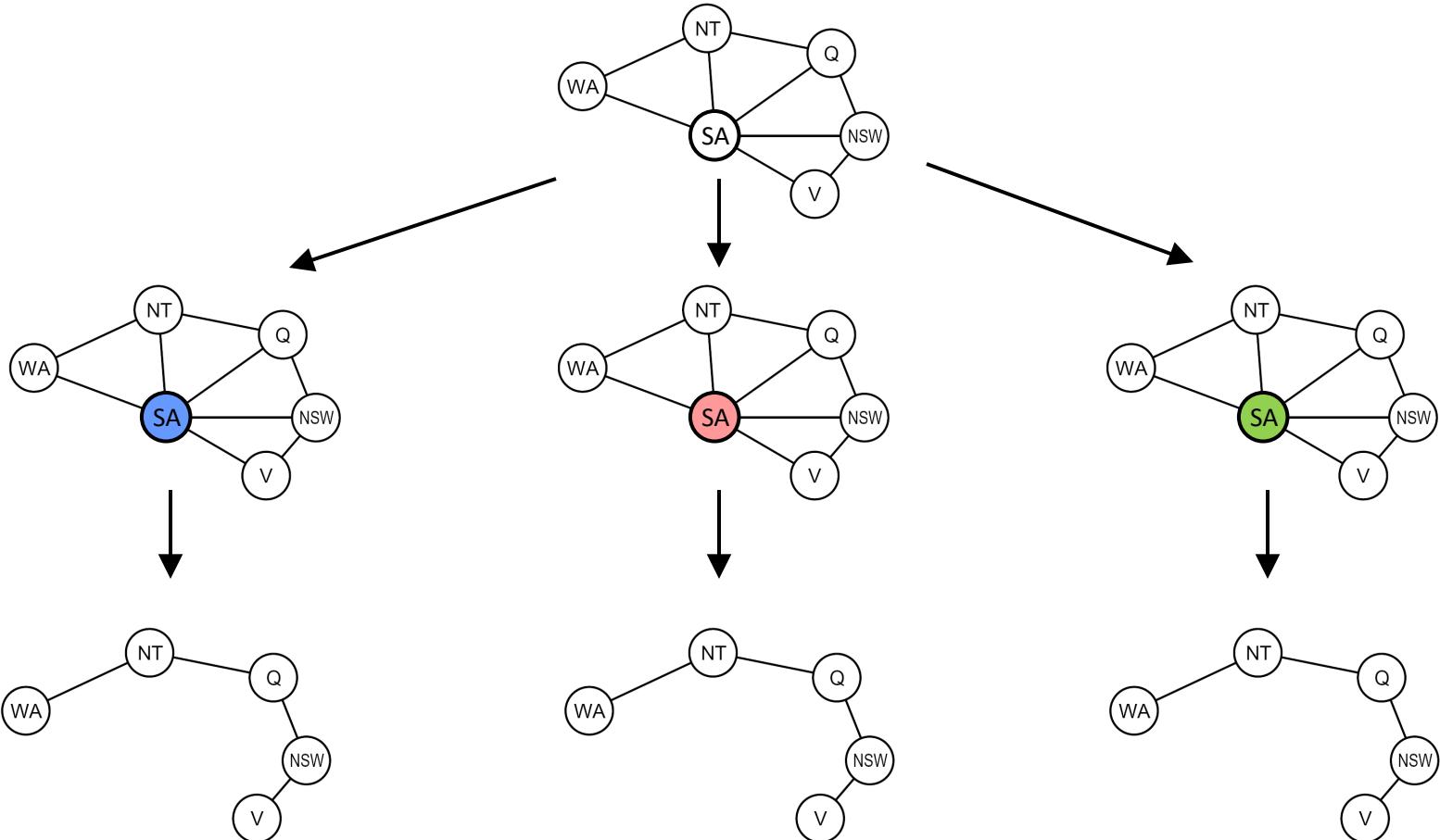
Cutset Conditioning

یک کاتست در نظر بگیر

تمام حالت ممکن کاتست را
مقداردهی کن

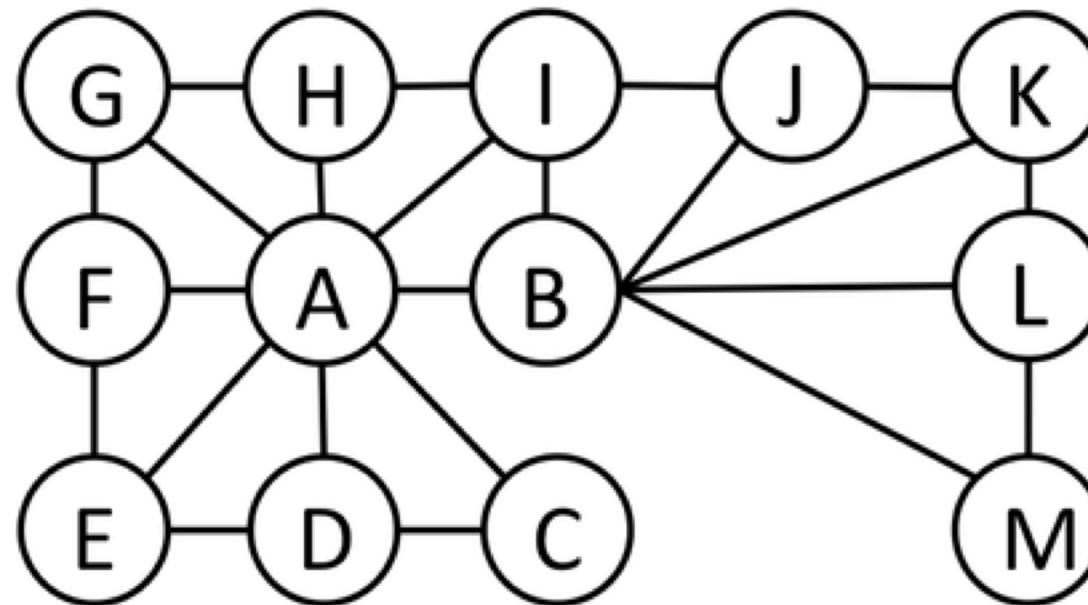
Compute residual CSP
for each assignment

Solve the residual CSPs
(tree structured)



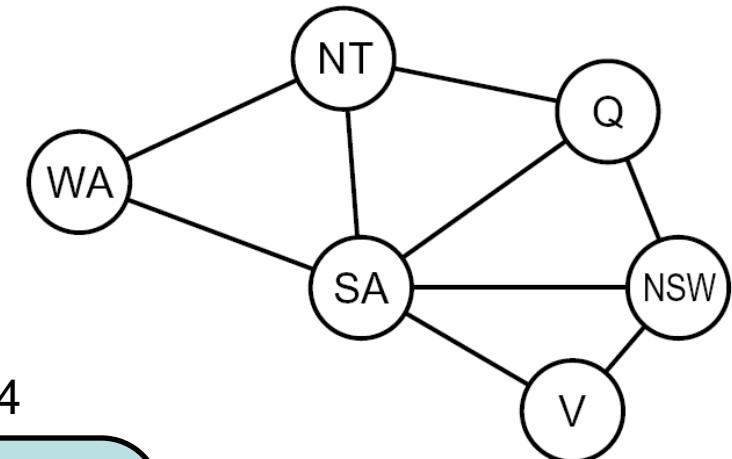
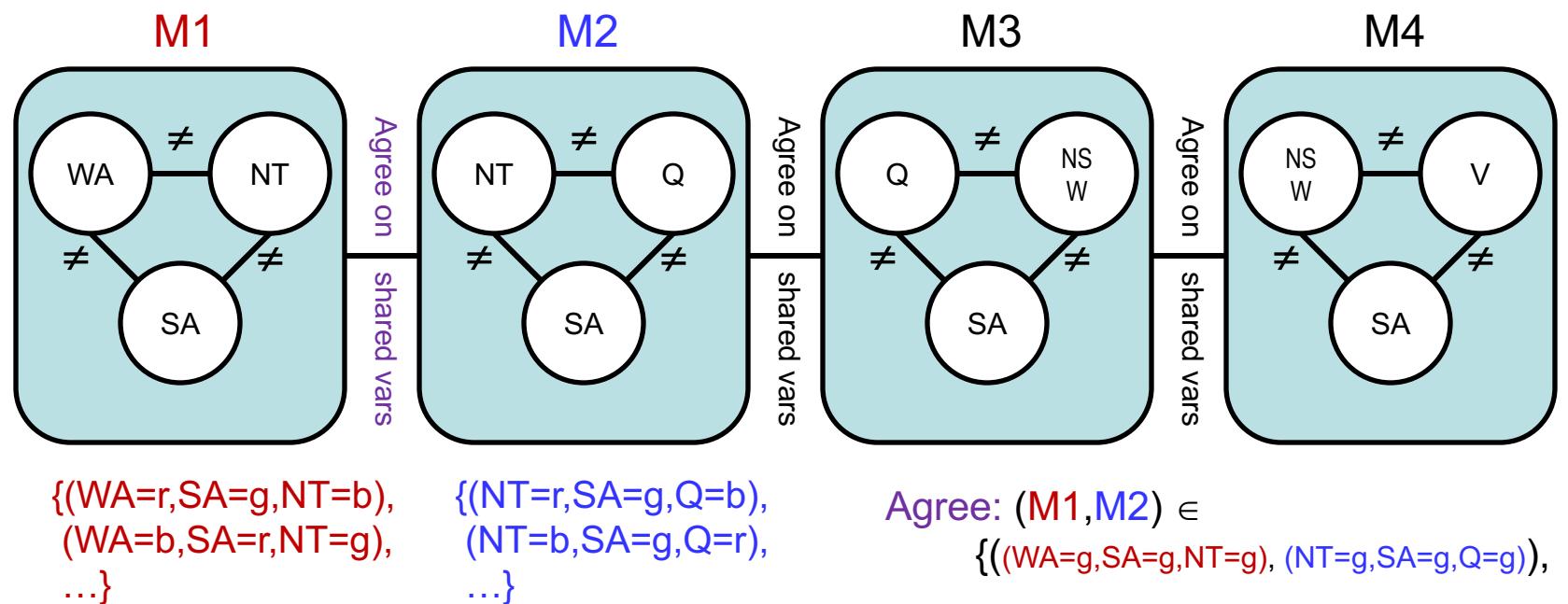
Cutset کوییز

کوچکترین کاتست را برای این گراف پیدا کنید

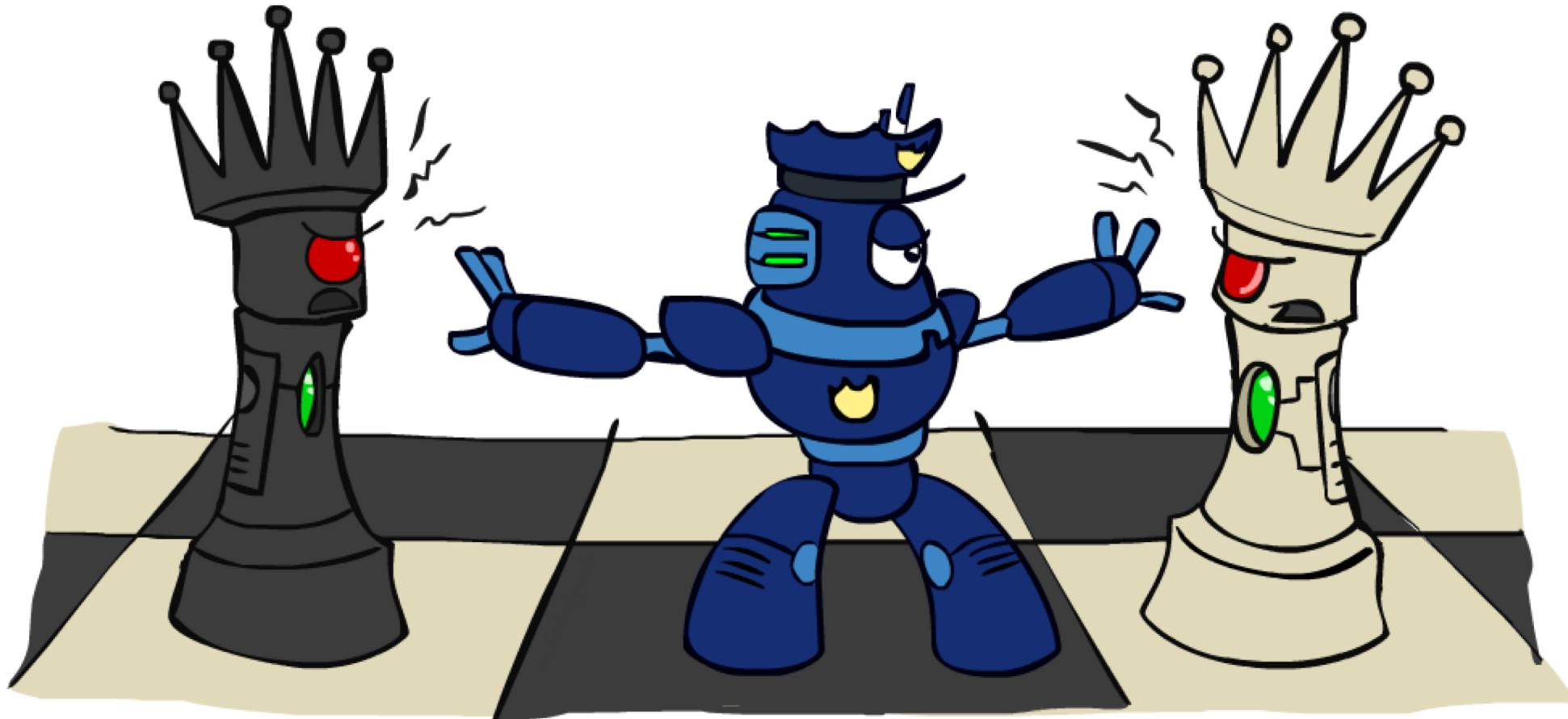


تجزیه درخت - Tree Decomposition

- Idea: create a tree-structured graph of mega-variables
- Each mega-variable encodes part of the original CSP
- Subproblems overlap to ensure consistent solutions

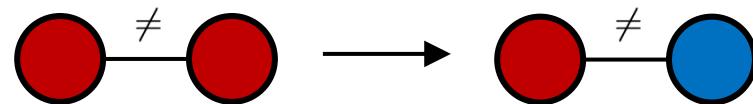


بیوبود تکراری – Iterative Improvement



حل CSP ها توسط Iterative Algorithms

روش‌های جستجوی محلی معمولاً از مساله‌ی کامل شروع به کار می‌کنند: یعنی حالتی که تمامی متغیرها مقداری دارند

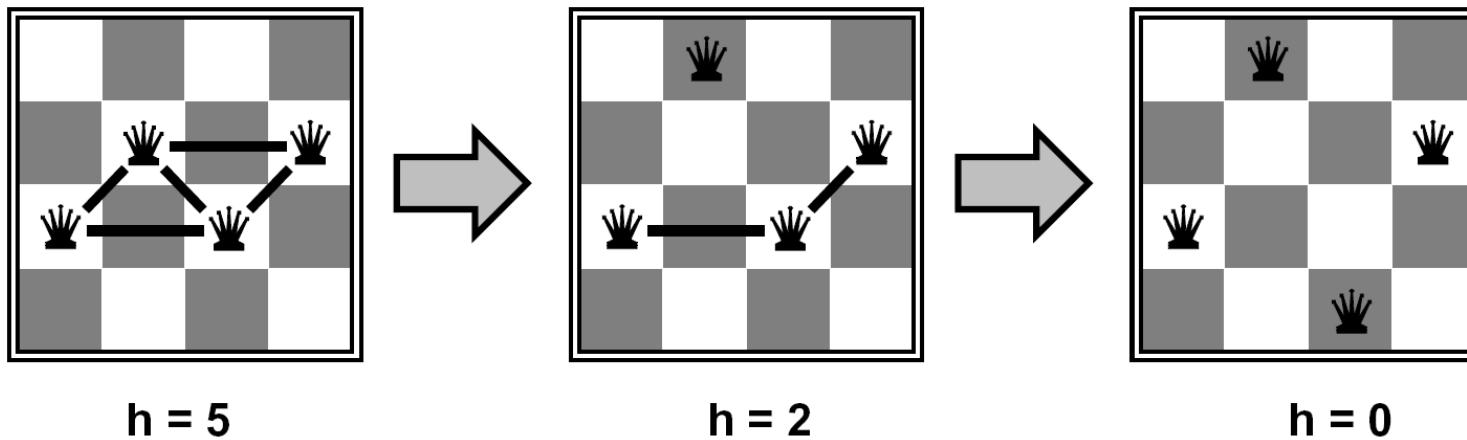


در حالت CSP ها

- یکی از متغیرهایی که از محدودیت پیروی نمی‌کند را انتخاب کن
- مقدارش را عوض کن
- فرینجی نداریم

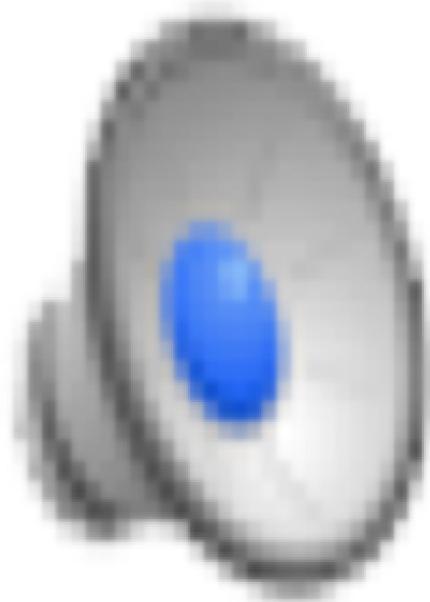
- Algorithm: While not solved,
 - Variable selection: randomly select any conflicted variable
 - Value selection: **min-conflicts heuristic:**
 - Choose a value that violates the fewest constraints
 - I.e., hill climb with $h(n) = \text{total number of violated constraints}$

مُثَالٌ : 4-Queens



- States: 4 queens in 4 columns ($4^4 = 256$ states)
- Operators: move queen in column
- Goal test: no attacks
- Evaluation: $c(n) = \text{number of attacks}$

Iterative Improvement – n Queens



ویدئو

رنگ کردن نقشه – Iterative Improvement

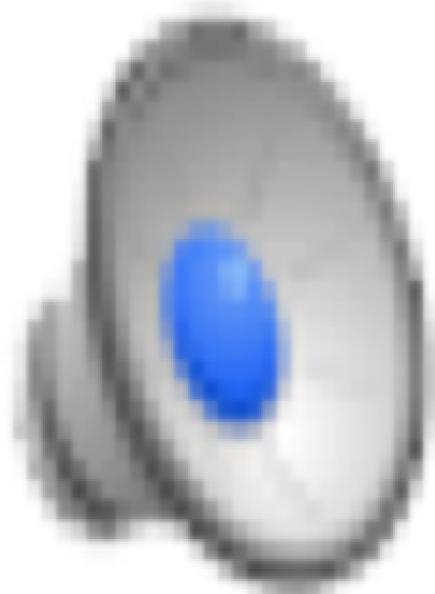
Graph
Complex

Algorithm
Iterative Improvement

Ordering
 None
 MRV
 MRV with LCV

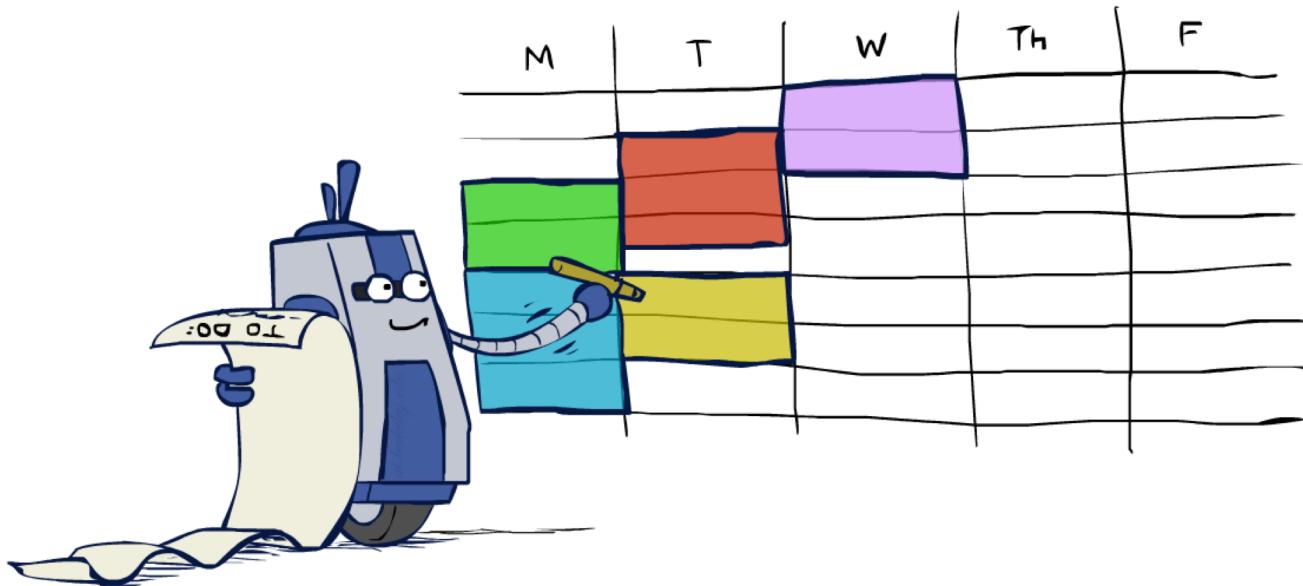
Filtering
 None
 Forward Checking
 Arc Consistency

Speed
Speedup x Frame Delay



CSP - خلاصه

نوع خاصی از مسائل جستجو هستند
حالات: مقادیر بخشی از متغیرها
هدف با کمک محدودیت‌ها تعریف می‌شود



روش ساده: backtracking

بهبود:

فیلتر کردن
مرتب سازی
ساختار

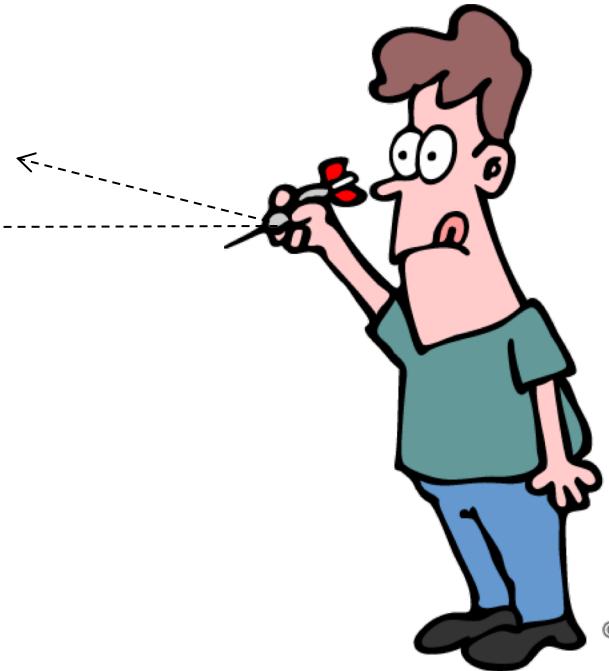
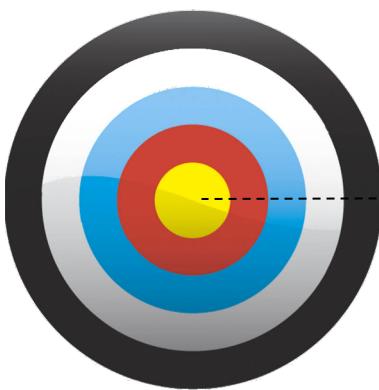
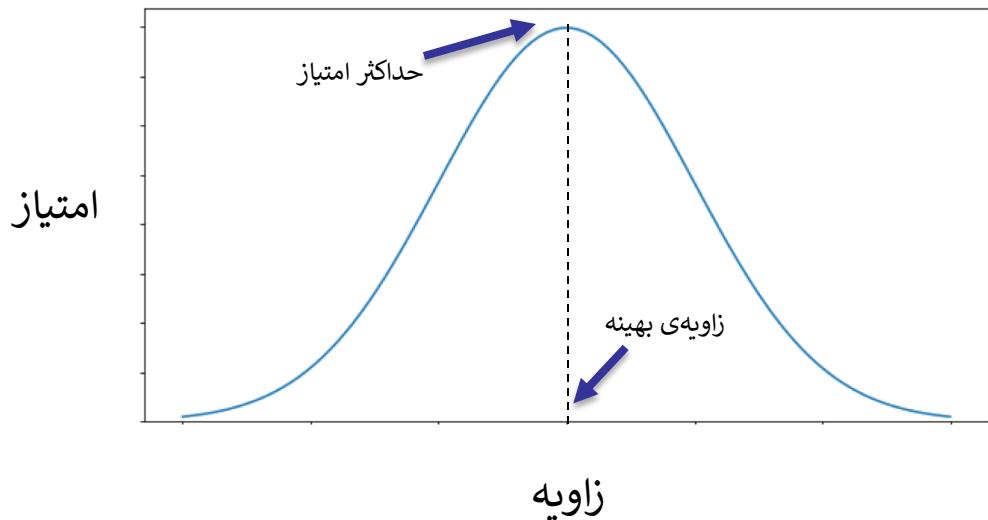
در عمل، روش Min-conflict برای خیلی از مسائل موثر است

جستجوی محلی - Local Search



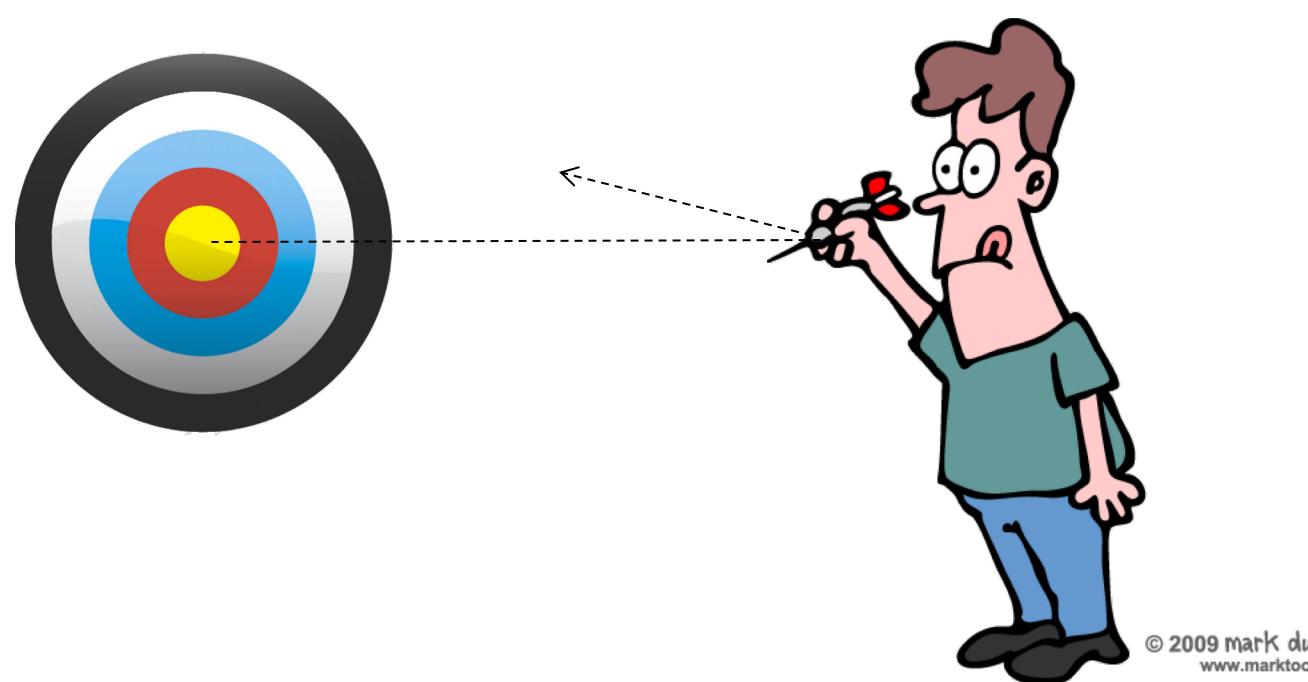
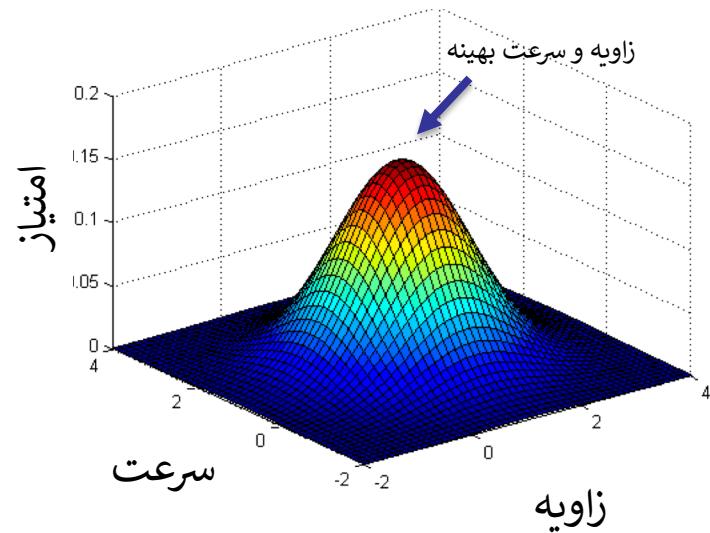
پیش‌زمینه - بهینه‌سازی

Objective function

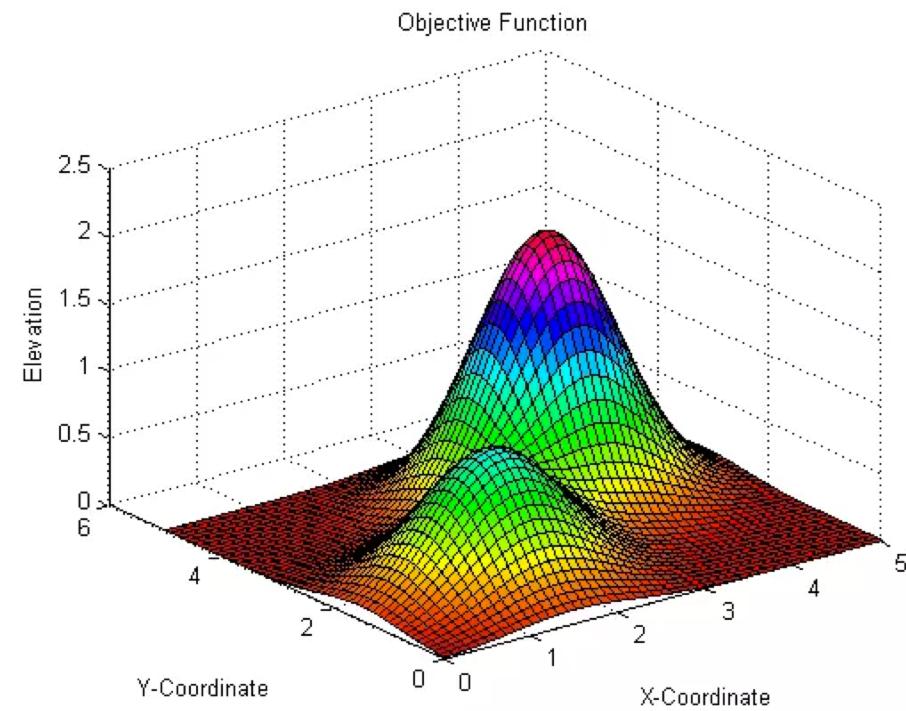


پیش زمینه - بهینه سازی

Objective function



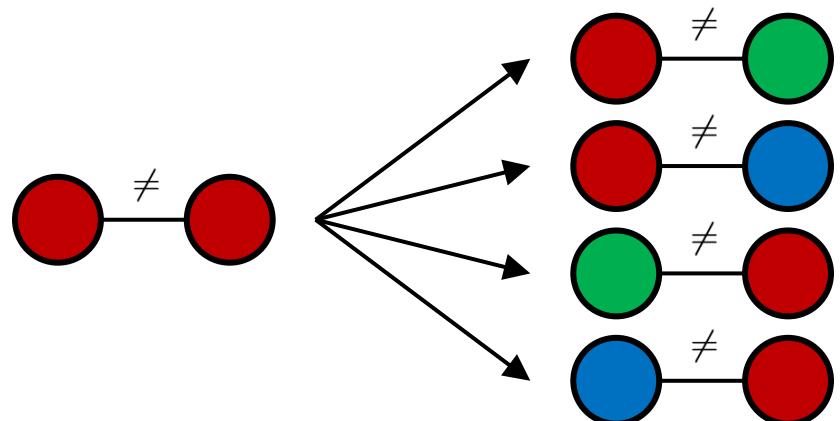
جستجو در قالب بهینه‌سازی



جستجوی محلی - Local Search

در جستجوی درختی، یک فرینج داشتیم و تمامی حالات جستجو نشده را در آن نگه میداشتیم

در جستجوی محلی فرینج نداریم! در هر مرحله، سعی می کنیم بخش از راه حل را بهبود بدهیم



تابع پیش بر جدید: بهبودهای کوچک محلی

معمولًا بسیار سریع تر و نیازمند به حافظه کمتر

Hill Climbing

ایده‌ی کلی:

از هر جا که خواستی شروع کن
تکرار کن: به بهترین حالت همسایه برو
اگر همسایه بهتری وجود ندارد، تلاش را متوقف کن

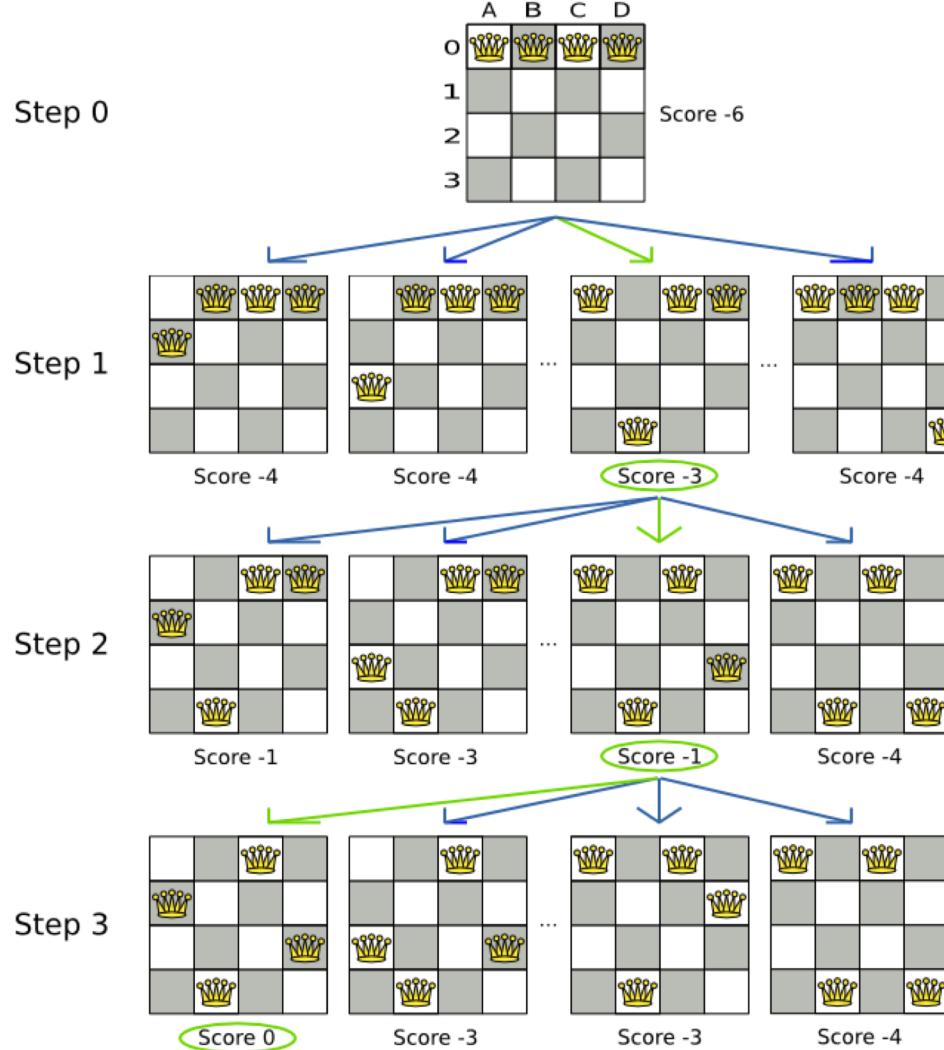


مشکل این روش کجاست؟

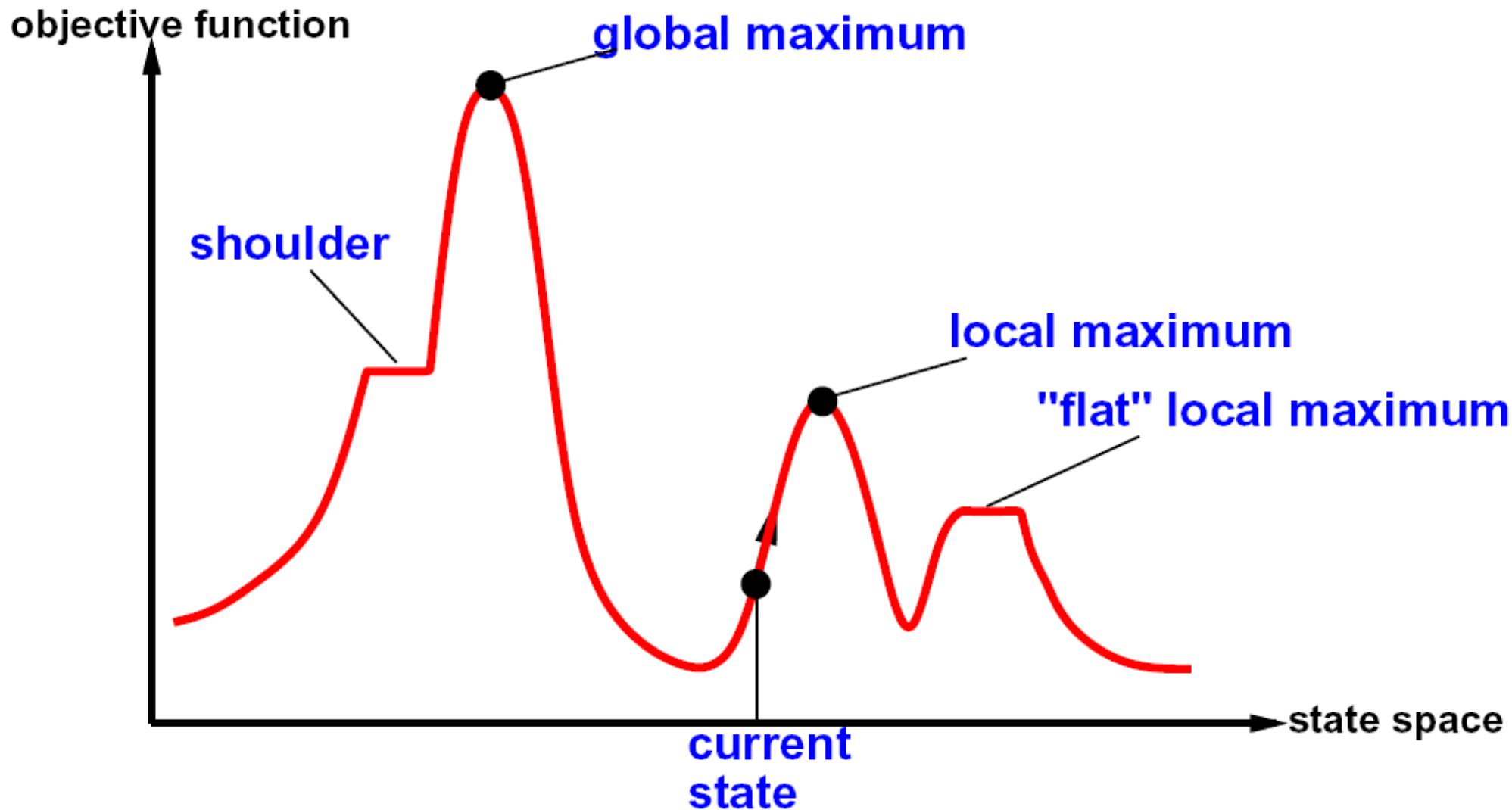
- Complete?
- Optimal?

خوبی این روش چیست؟

Hill Climbing – N-Queens

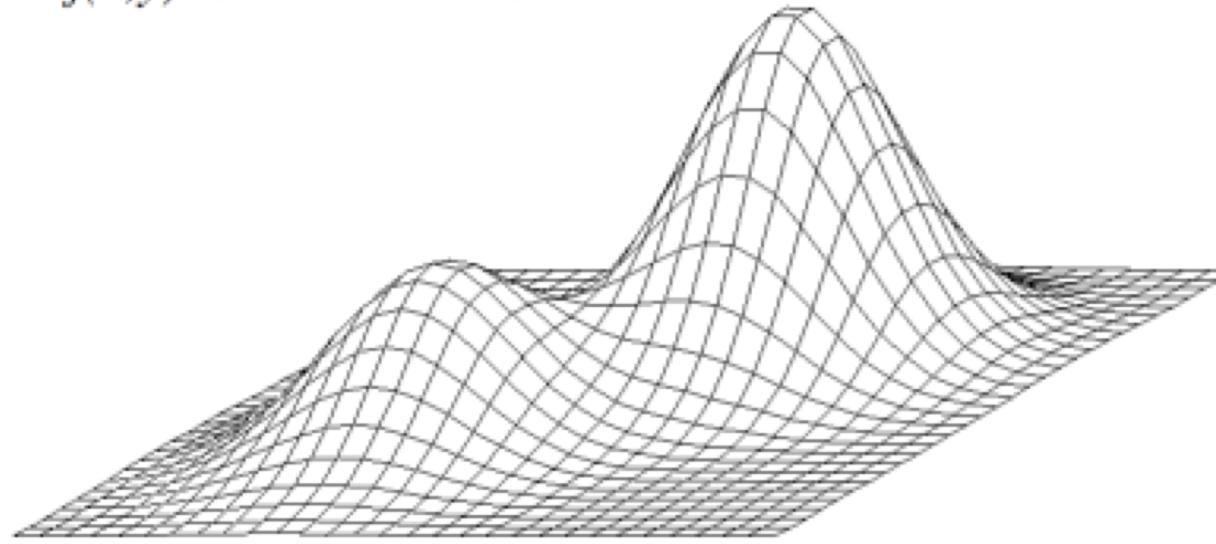


Hill Climbing دیاگر ارم

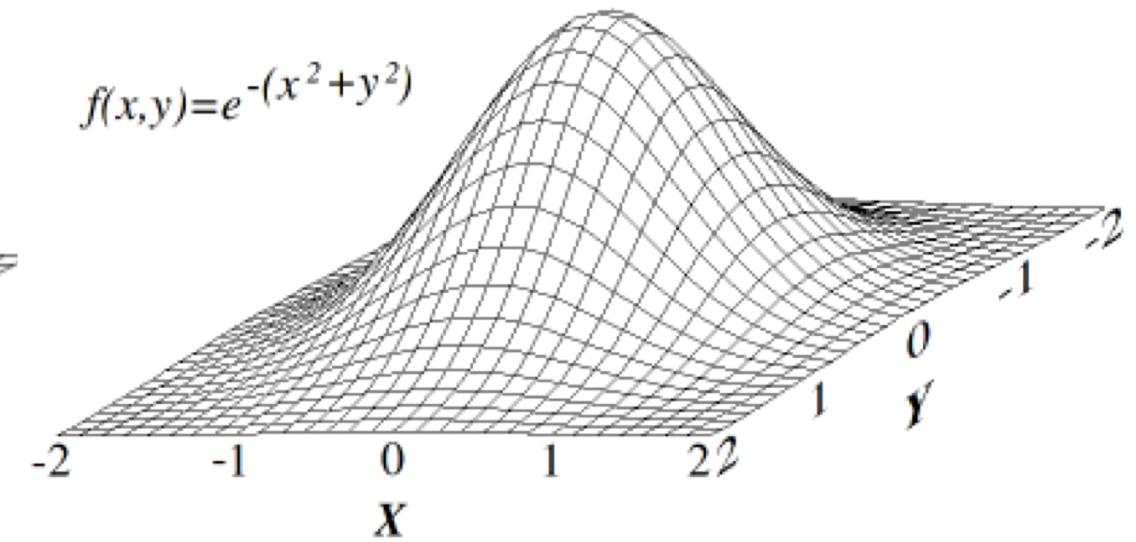


دیاگرام سه بعدی Hill Climbing

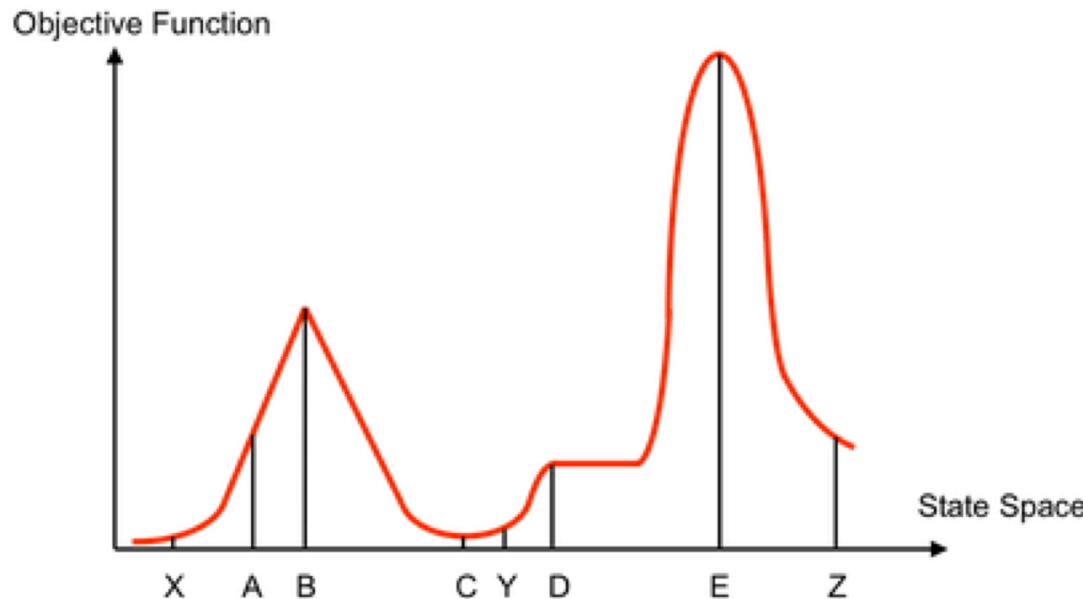
$$f(x,y) = e^{-(x^2+y^2)} + 2e^{-((x-1.7)^2+(y-1.7)^2)}$$



$$f(x,y) = e^{-(x^2+y^2)}$$



Hill Climbing کوییز

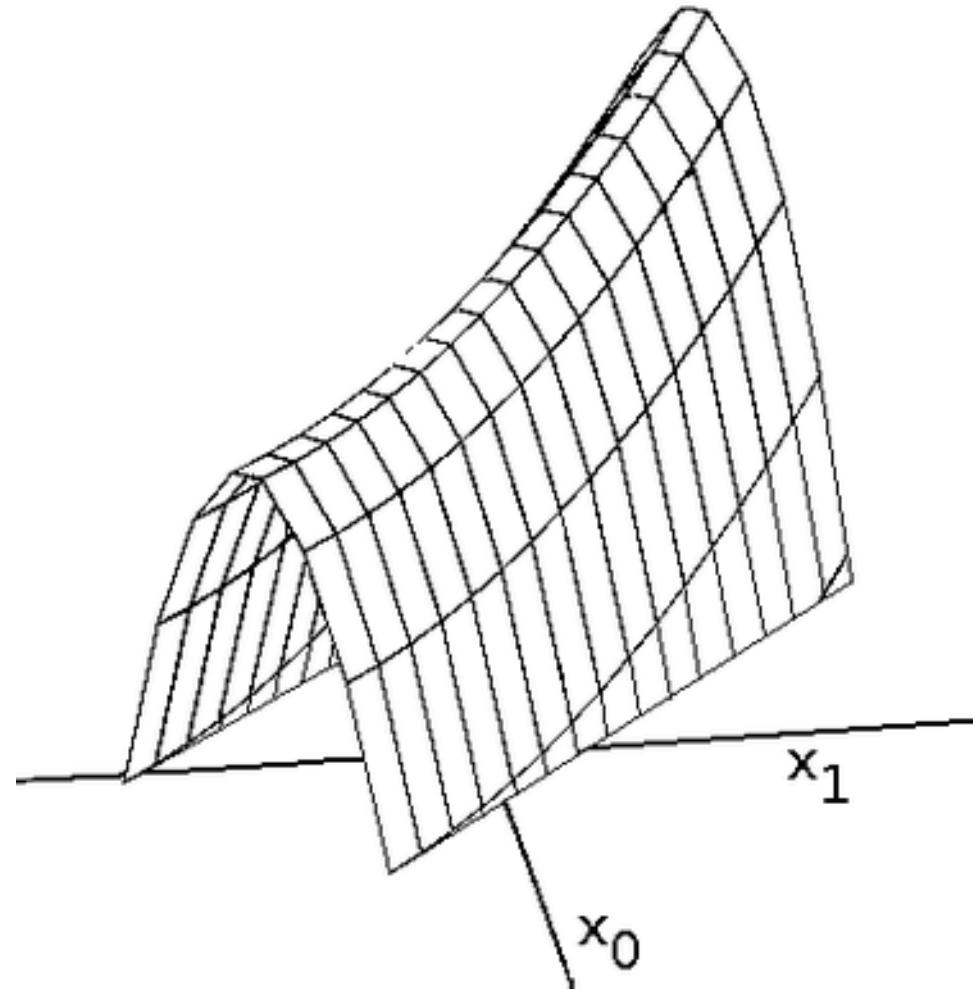


Starting from X, where do you end up ?

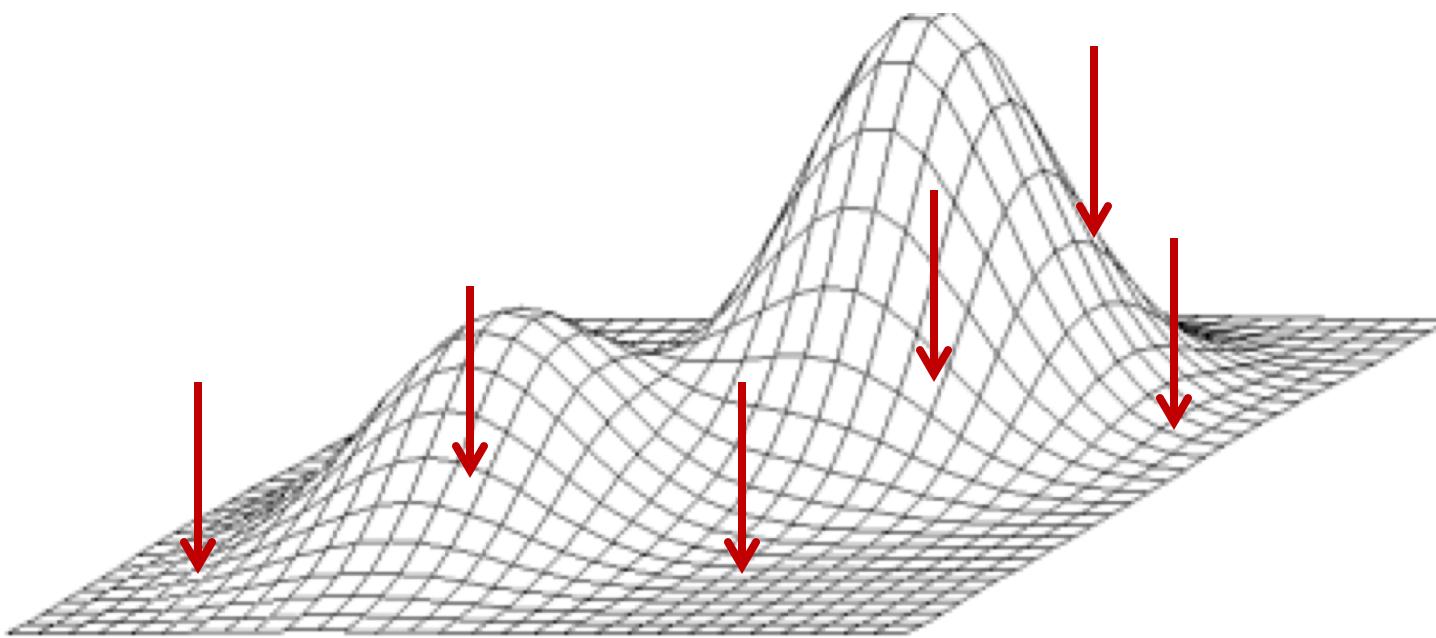
Starting from Y, where do you end up ?

Starting from Z, where do you end up ?

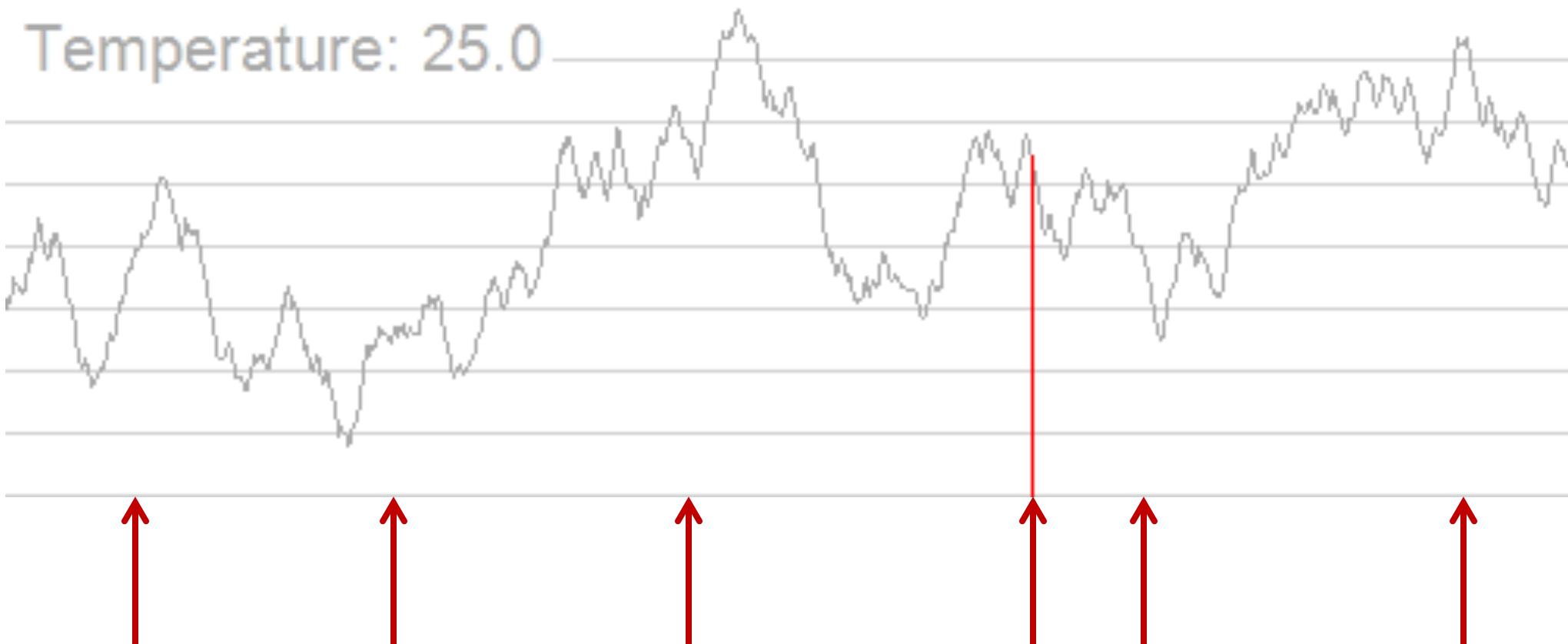
خط الراس – Ridge



Stochastic Hill Climbing



Stochastic Hill Climbing



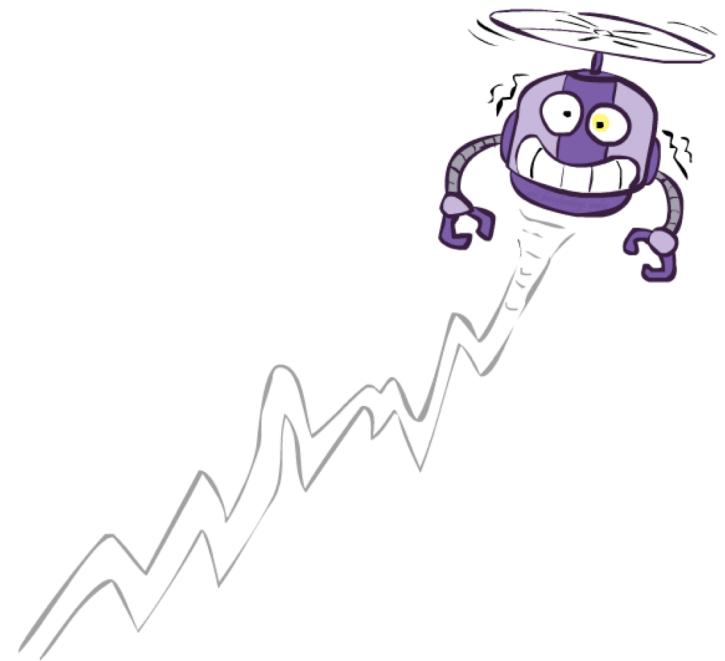
Simulated Annealing



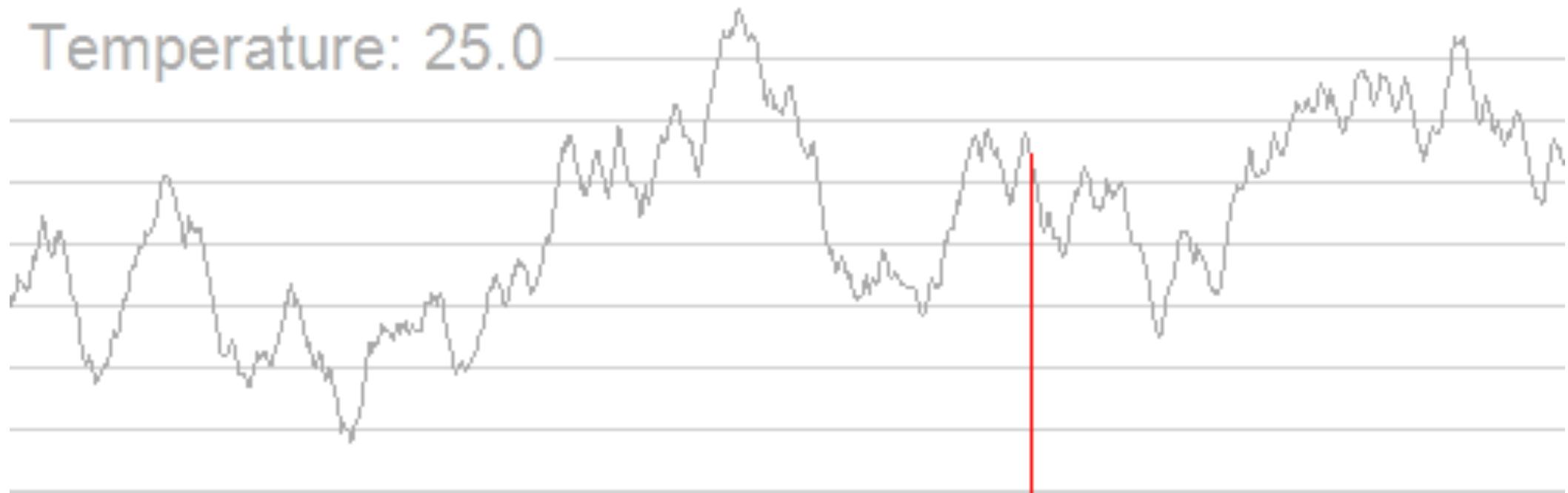
Simulated Annealing

ایده: از ماکزیمم محلی بگریز. چطور؟ میتوانی در جهت سرپاپینی هم حرکت کنی!
ولی، به مرور زمان این کار را کمتر انجام بده

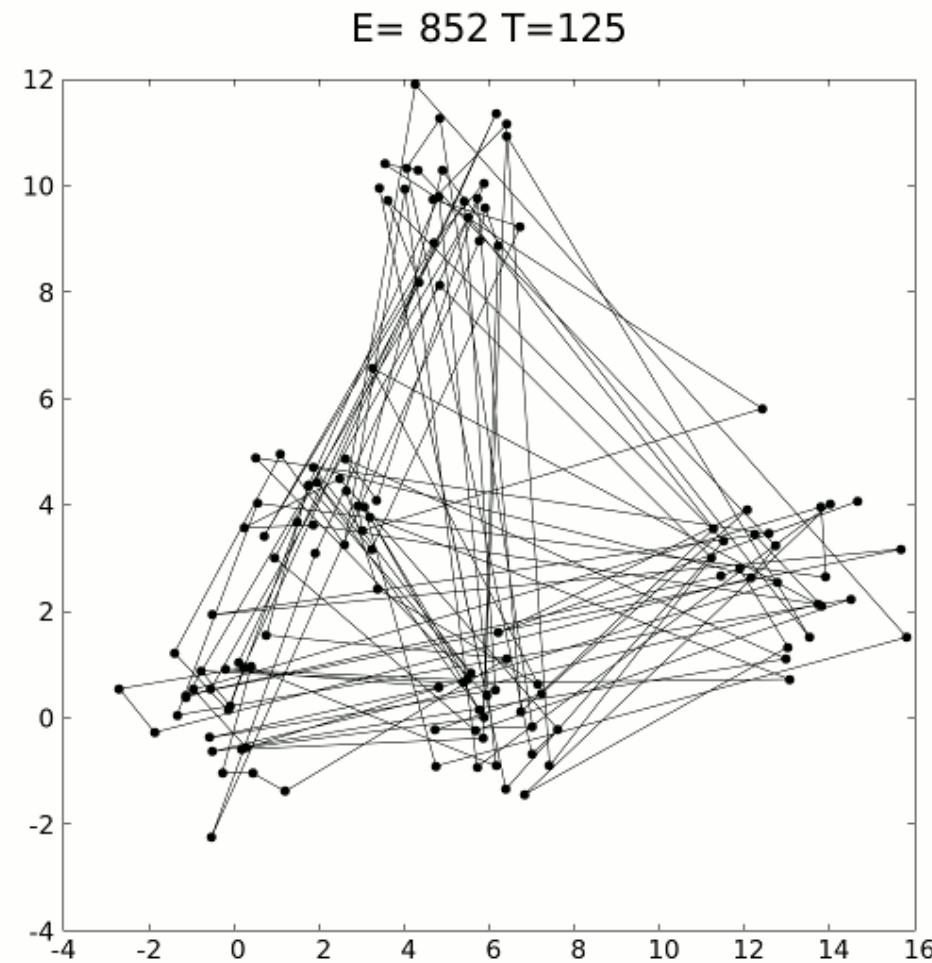
```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
          schedule, a mapping from time to “temperature”
  local variables: current, a node
                    next, a node
                    T, a “temperature” controlling prob. of downward steps
  current  $\leftarrow$  MAKE-NODE(INITIAL-STATE[problem])
  for t  $\leftarrow$  1 to  $\infty$  do
    T  $\leftarrow$  schedule[t]
    if T = 0 then return current
    next  $\leftarrow$  a randomly selected successor of current
     $\Delta E \leftarrow$  VALUE[next] – VALUE[current]
    if  $\Delta E > 0$  then current  $\leftarrow$  next
    else current  $\leftarrow$  next only with probability  $e^{\Delta E/T}$ 
```



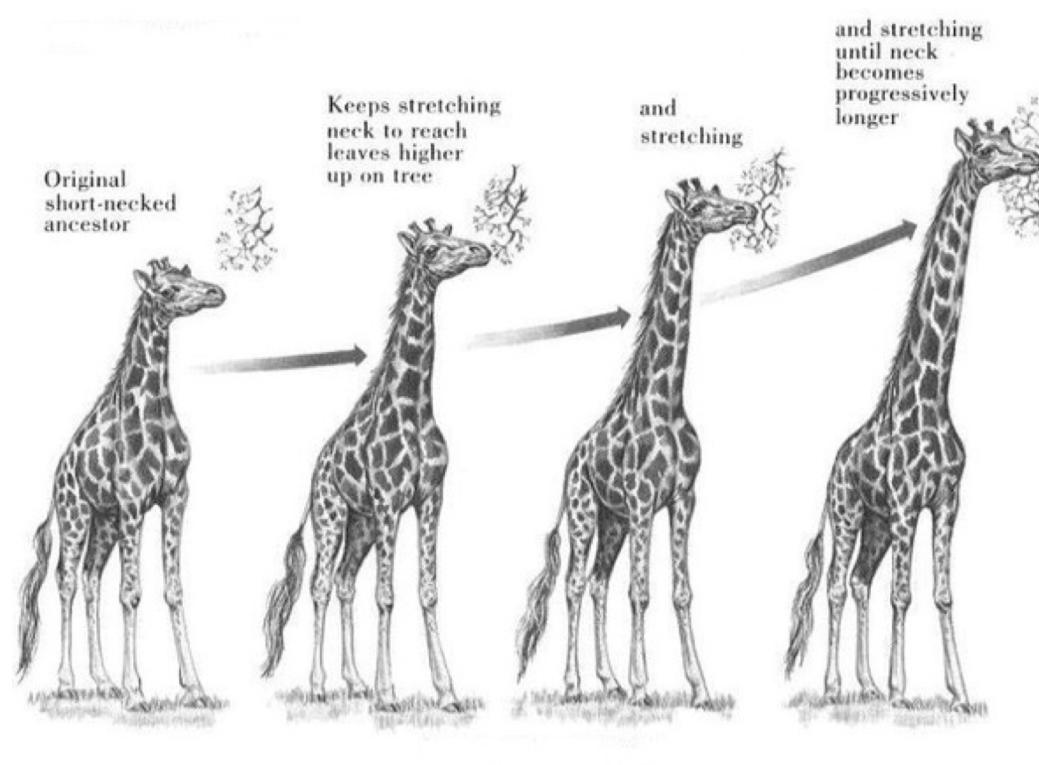
Simulated Annealing



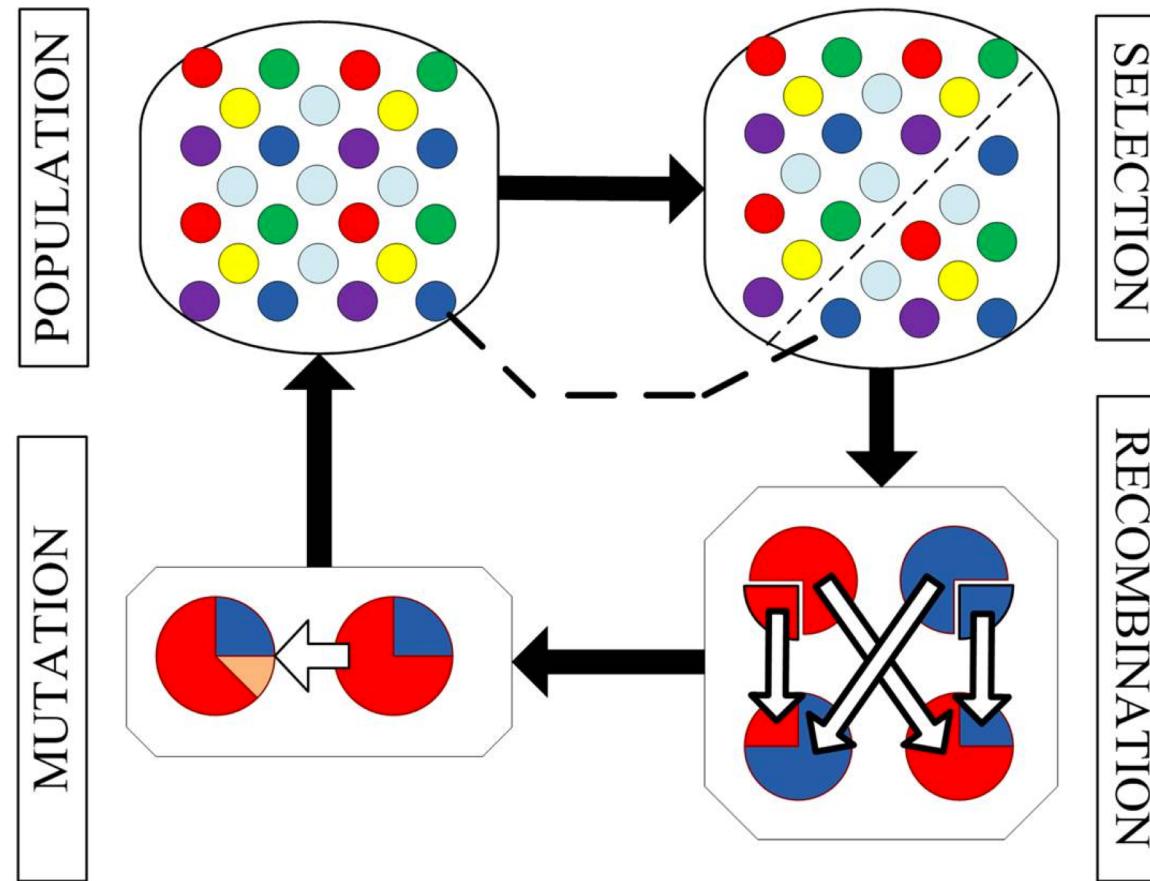
Simulated Annealing



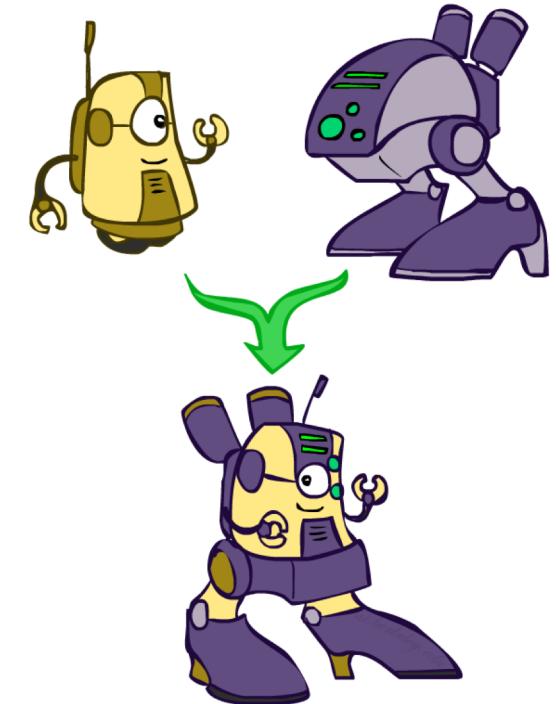
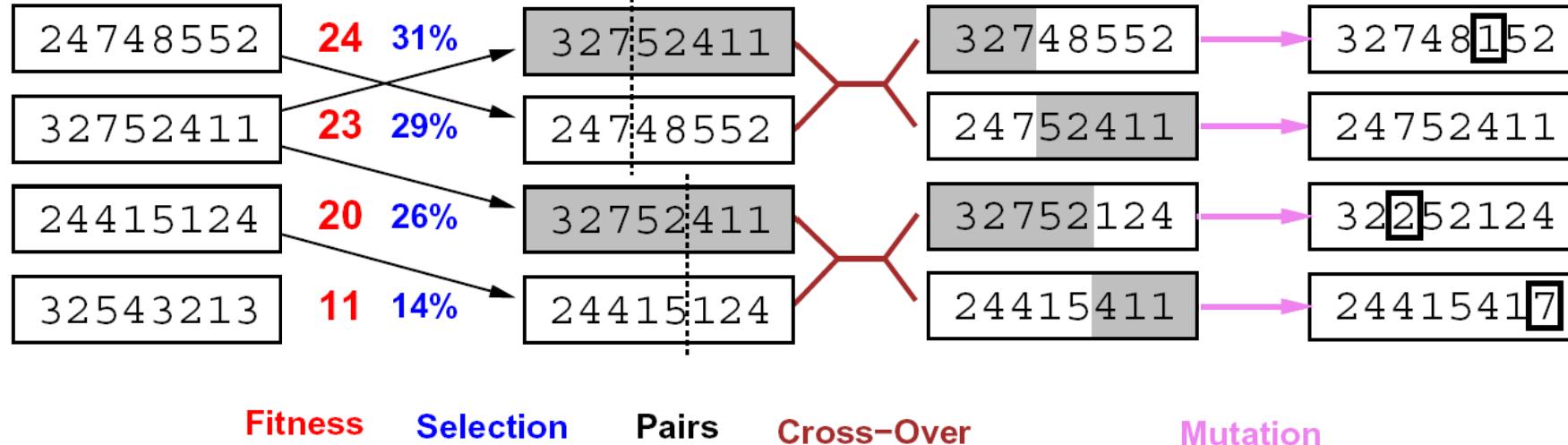
الگوريتم ڙتيك



الگوريتم ژنتيک

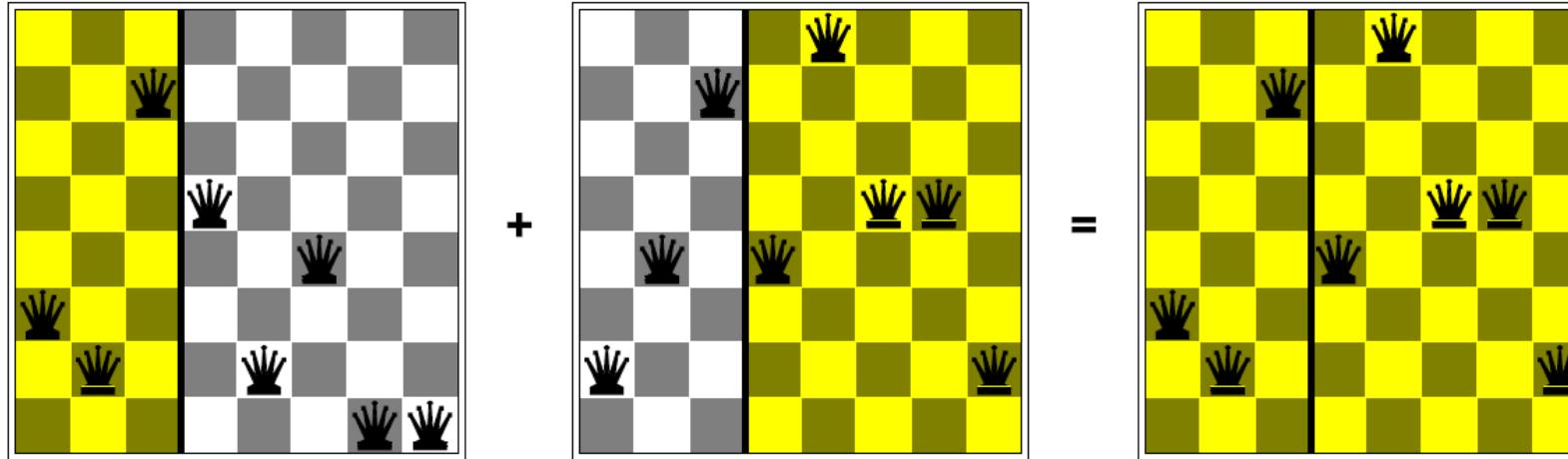


الگوريتم ژنتيک



- Genetic algorithms use a natural selection metaphor
 - Keep best N hypotheses at each step (selection) based on a fitness function
 - Also have pairwise crossover operators, with optional mutation to give variety

Example: N-Queens



- Why does crossover make sense here?
- When wouldn't it make sense?
- What would mutation be?
- What would a good fitness function be?

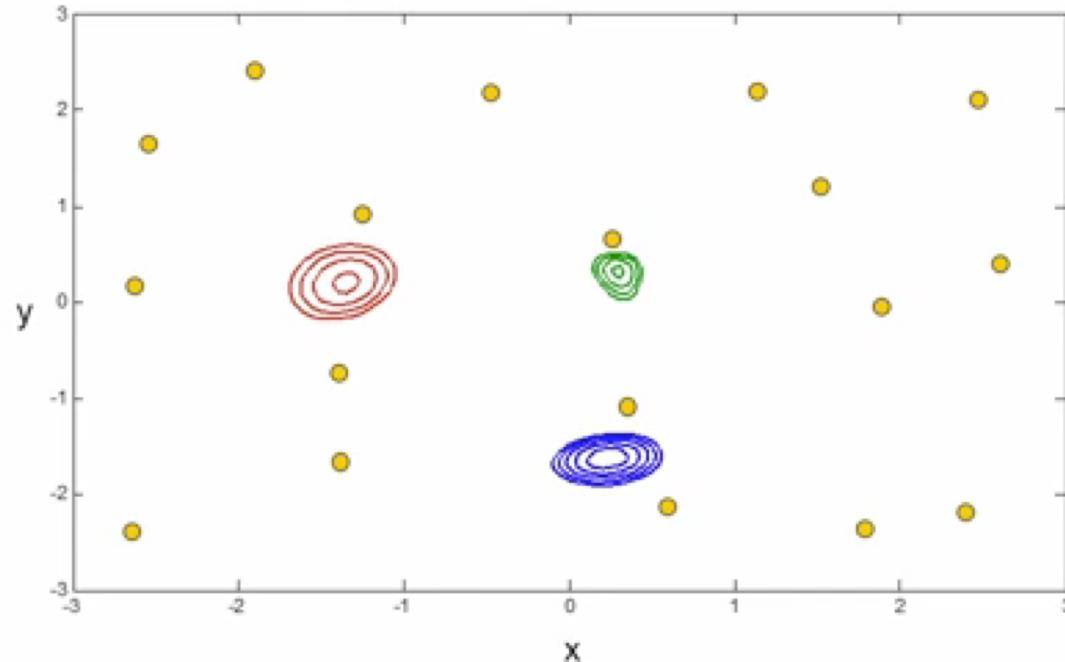
مثال مار – الگوريتم ژنتيک



الگوريتم ژنتيک - بهينه سازي

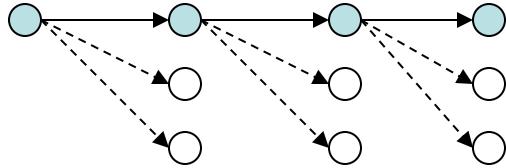


Genetic Algorithm – Iteration 1
Evaluate initial population

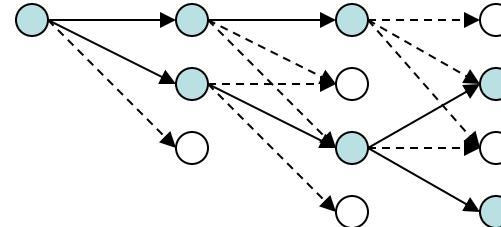


Beam Search – جستجوی بیم

- Like greedy hillclimbing search, but keep K states at all times:



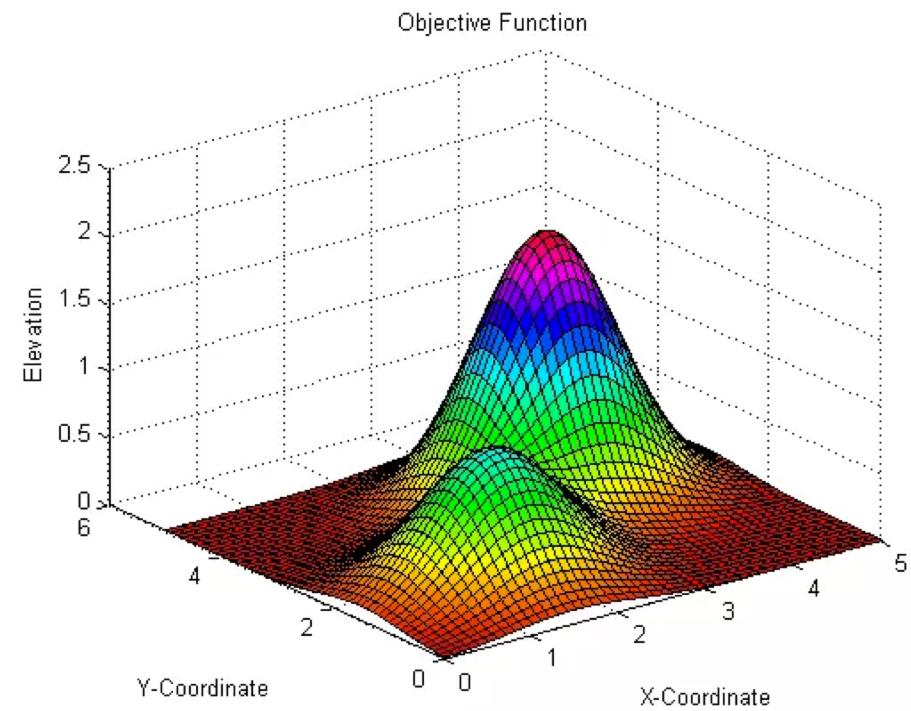
Greedy Search



Beam Search

- Variables: beam size, encourage diversity?
- The best choice in MANY practical settings
- Complete? Optimal?

سؤال؟



بوستون داینامیکس - اطلس

