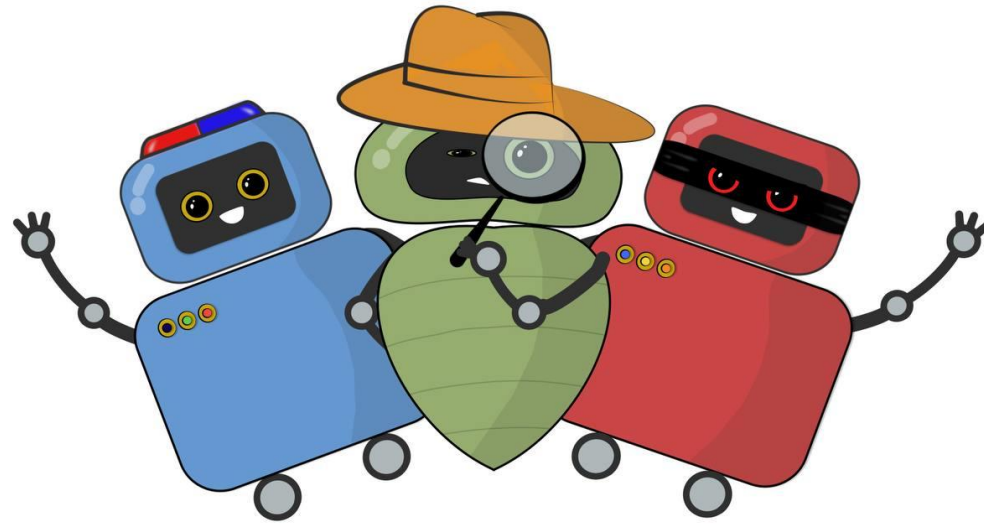


مبانی و کاربردهای هوش مصنوعی

مسائل ارضای محدودیت (فصل 6.1 الی 6.5)



مدرس: مهدی جوانمردی

دانشکده مهندسی کامپیوتر دانشگاه صنعتی امیرکبیر



مسائل ارضای محدودیت (Constraint Satisfaction Problems)

N متغیر
D دامنه
محدودیت‌ها

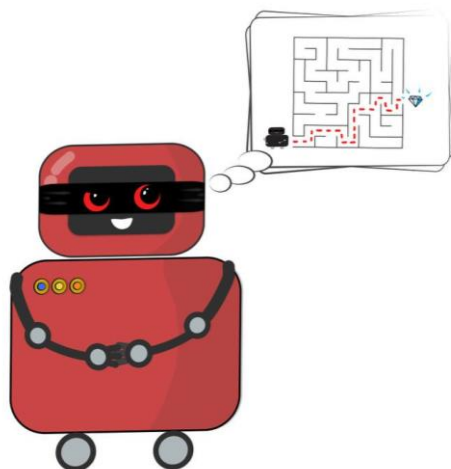
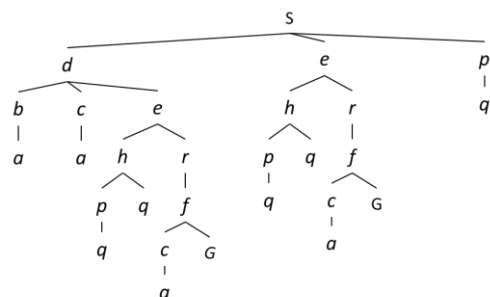


تابع پسین
تخصیص یک مقدار
به یک متغیر

آزمون هدف
تخصیص کامل،
ارضای محدودیت‌ها

حالت‌ها
تخصیص جزئی

جستجو برای چه هدفی؟

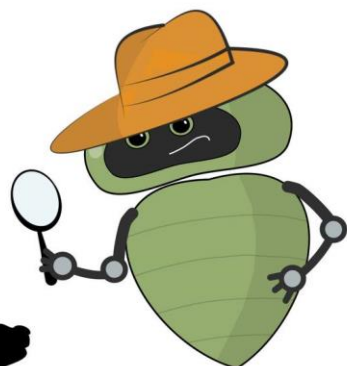
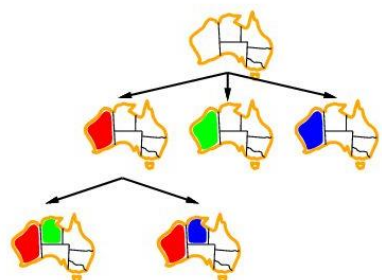


- مفروضات در مورد جهان:

تک عامل، اقدامات قطعی، حالت کاملاً مشاهده شده، فضای حالت گسسته

- برنامه‌ریزی (planning): دنباله‌ای از اقدامات

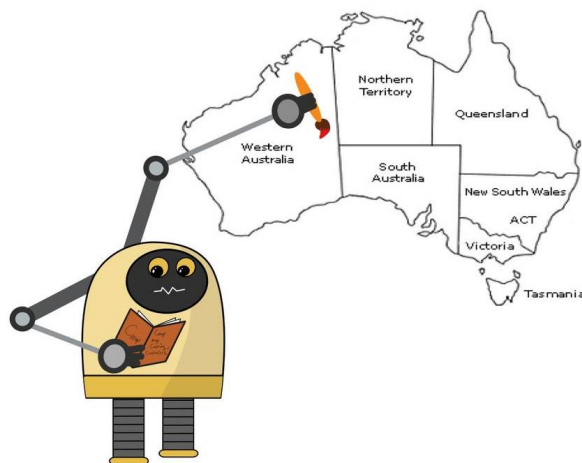
- مسیر رسیدن به هدف مهم است.
- مسیرها عمق و هزینه متفاوتی دارند.
- هیوریستیک‌ها راهنمایی‌های مخصوصی برای مسئله ارائه می‌دهند.



- شناسایی (identification): تخصیص مقادیر به متغیرها

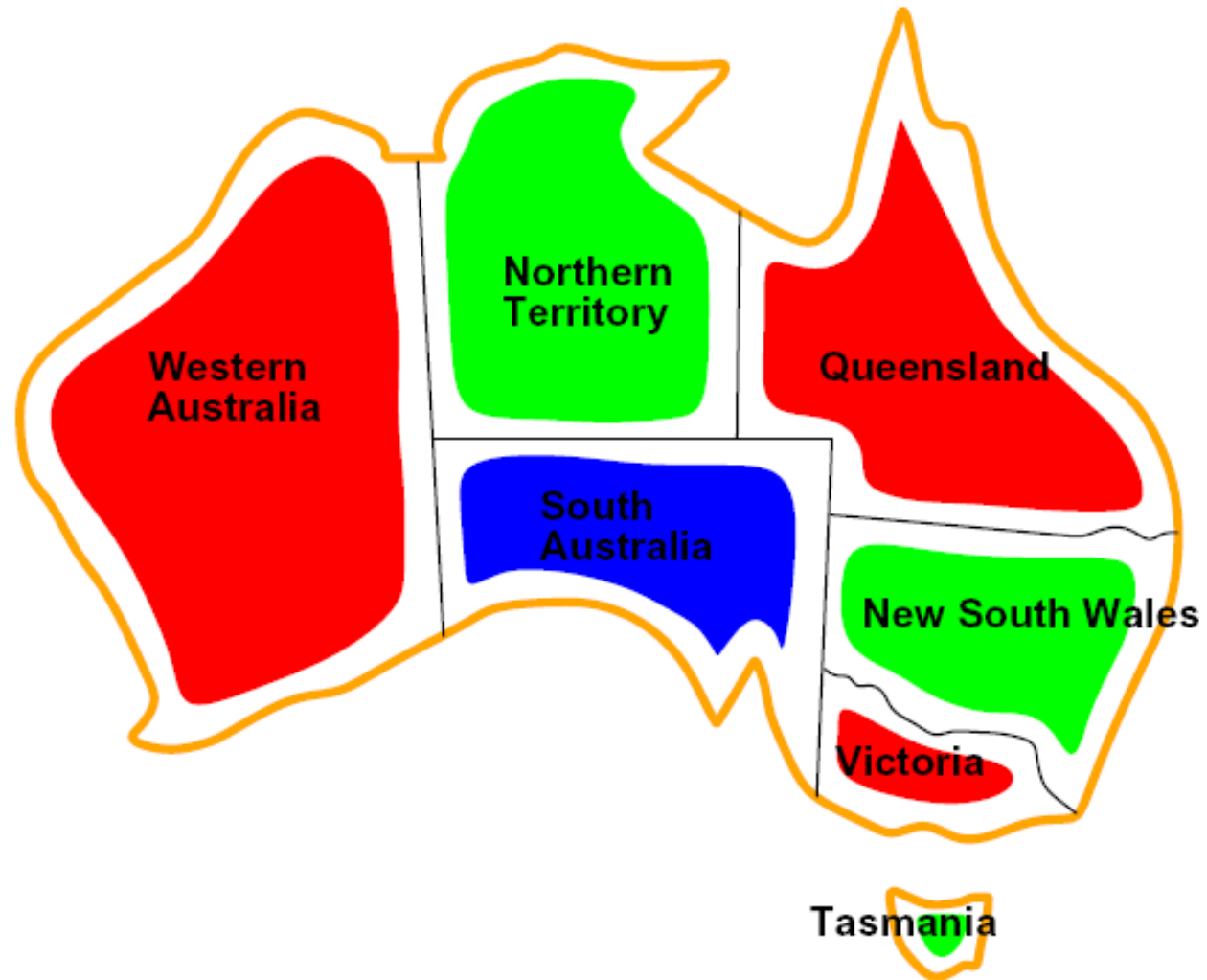
- خود هدف مهم است نه مسیر
- معمولاً همه مسیرهای هدف در عمق یکسان (برای برخی از فرمولاسیون‌ها)
- مسائل ارضای محدودیت (CSP) اختصاصاً برای مسائل شناسایی طراحی شده است

مسائل ارضای محدودیت

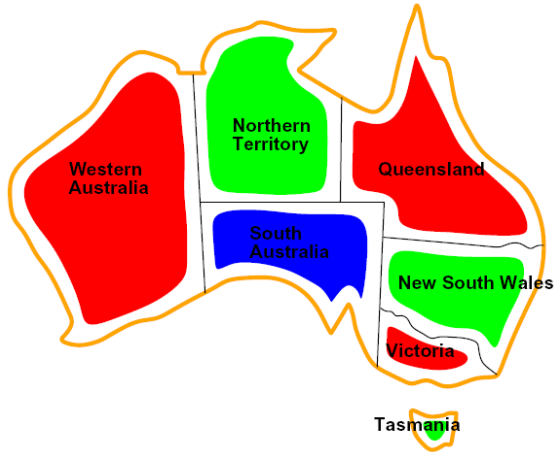


- مسائل جستجوی استاندارد:
 - حالت (state) یک "جعبه سیاه" است: ساختار داده دلخواه
 - آزمون هدف می‌تواند هر تابعی بر روی حالت‌ها باشد.
 - تابع پسین نیز می‌تواند هر چیزی باشد.
- مسائل ارضای محدودیت (CSPs):
 - یک زیرمجموعه خاص از مسائل جستجو
 - حالت توسط **متغیرهای X_i** با مقادیر **دامنه D** تعریف می‌شود. (گاهی اوقات D به i بستگی دارد)
 - آزمون هدف **مجموعه‌ای از محدودیت‌ها** است که ترکیبات مجاز مقادیر را برای زیرمجموعه‌های متغیرها مشخص می‌کند.
- مجموعه‌ای از الگوریتم‌های همه‌منظوره را ارائه می‌دهد که از الگوریتم‌های جستجوی استاندارد قوی‌تر است.

مثال‌های CSP



مثال: رنگ آمیزی نقشه



- متغیرها:

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

- دامنه‌ها:

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

- محدودیت‌ها: مناطق مجاور باید رنگ‌های متفاوتی داشته باشند.

$WA \neq NT$

ضمنی:

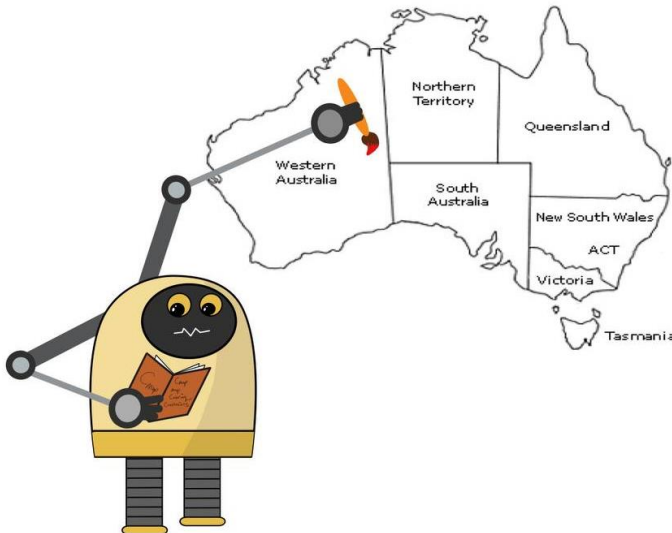
$(WA, NT) \in \{(red, green), (red, blue), \dots\}$

صریح:

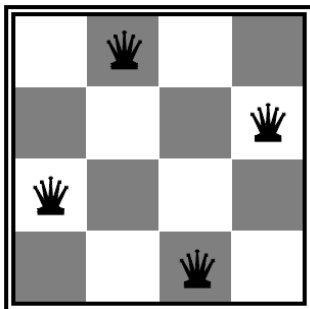
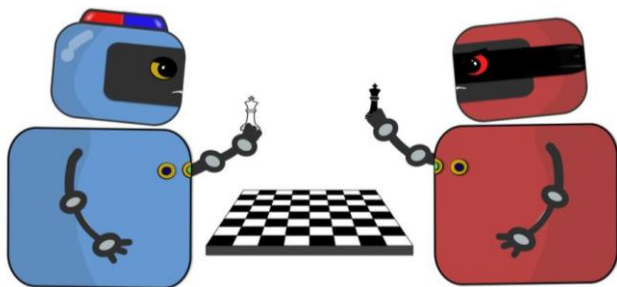
- راه‌حل‌ها انواع تخصیص‌هایی هستند که تمام محدودیت‌ها را برآورده می‌کنند،

به عنوان مثال:

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



مثال: N وزیر



- فرمول بندی 1:

- متغیرها:

X_{ij}

- دامنه‌ها:

$\{0, 1\}$

- محدودیت‌ها:

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

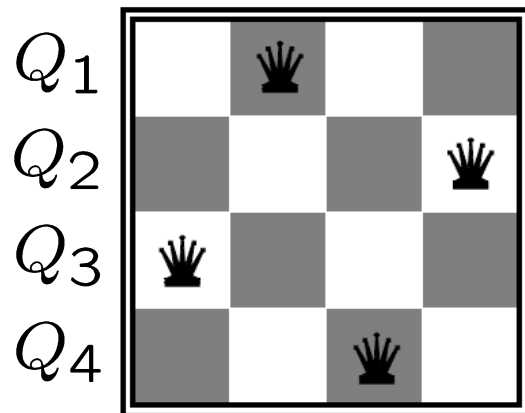
$$\forall i, j, k \quad (X_{ij}, X_{kj}) \in \{(0, 0), (0, 1), (1, 0)\}$$

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

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

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

مثال: N وزیر



- فرمول بندی 2:

- متغیرها:

 Q_k

- دامنه ها:

 $\{1, 2, 3, \dots, N\}$

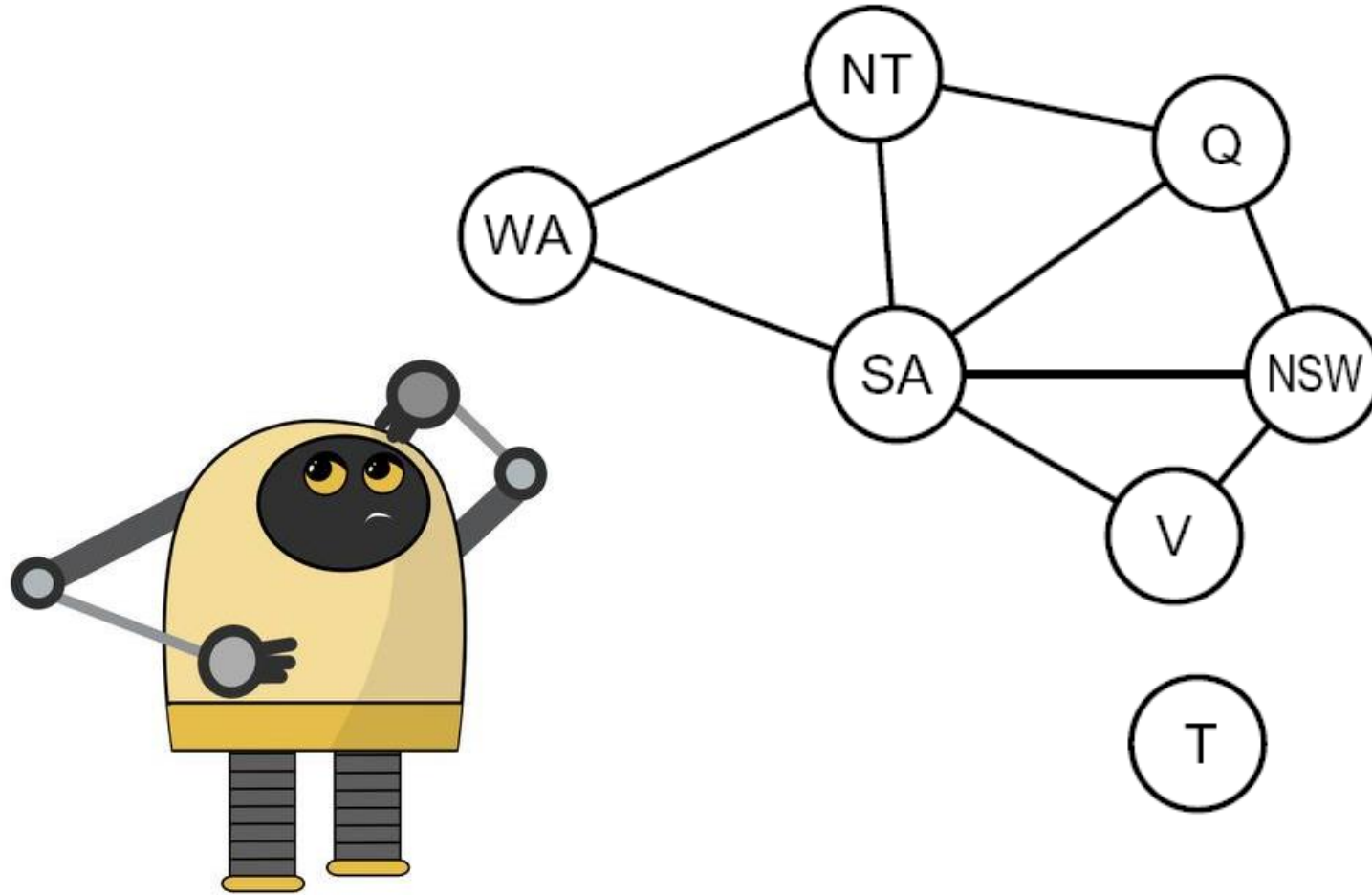
- محدودیت ها:

$\forall i, j \text{ non_threatening}(Q_i, Q_j)$ ضمنی:

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

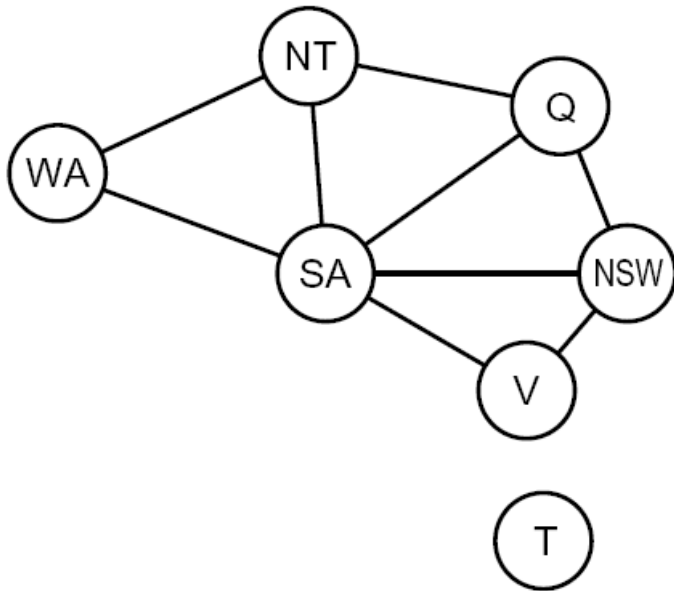
...

گراف محدودیت (Constraint Graph)

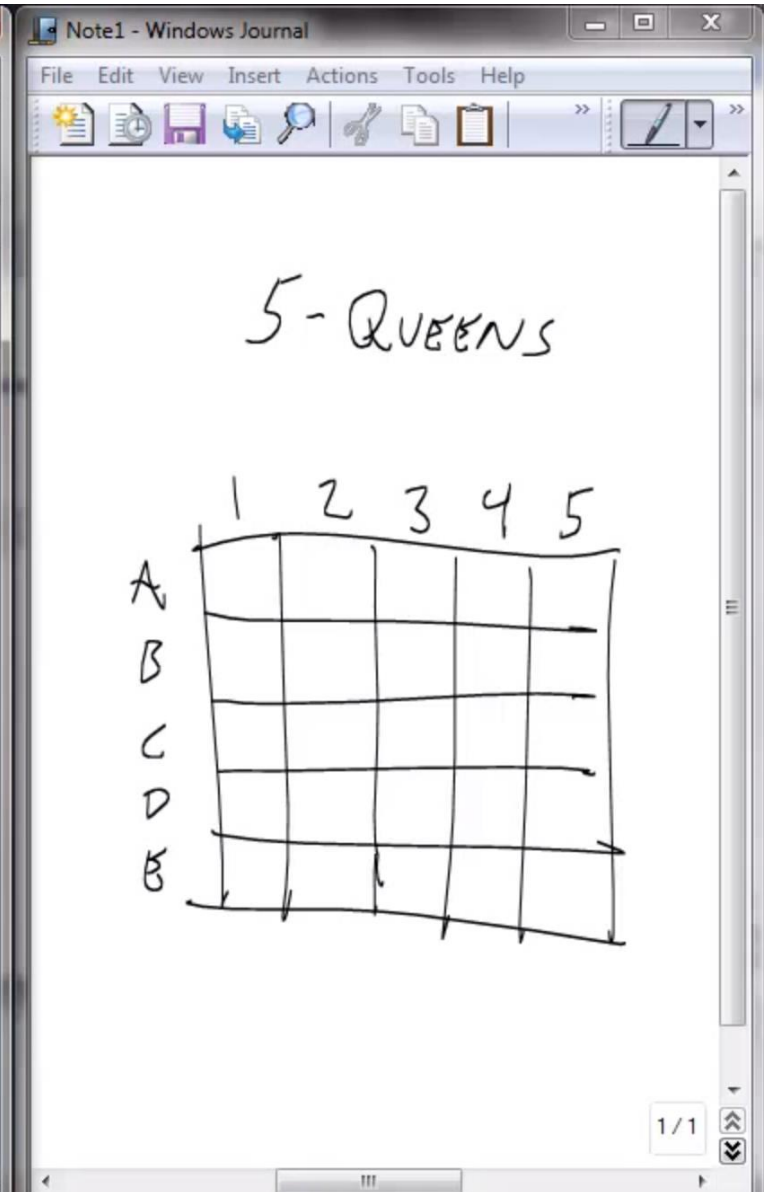
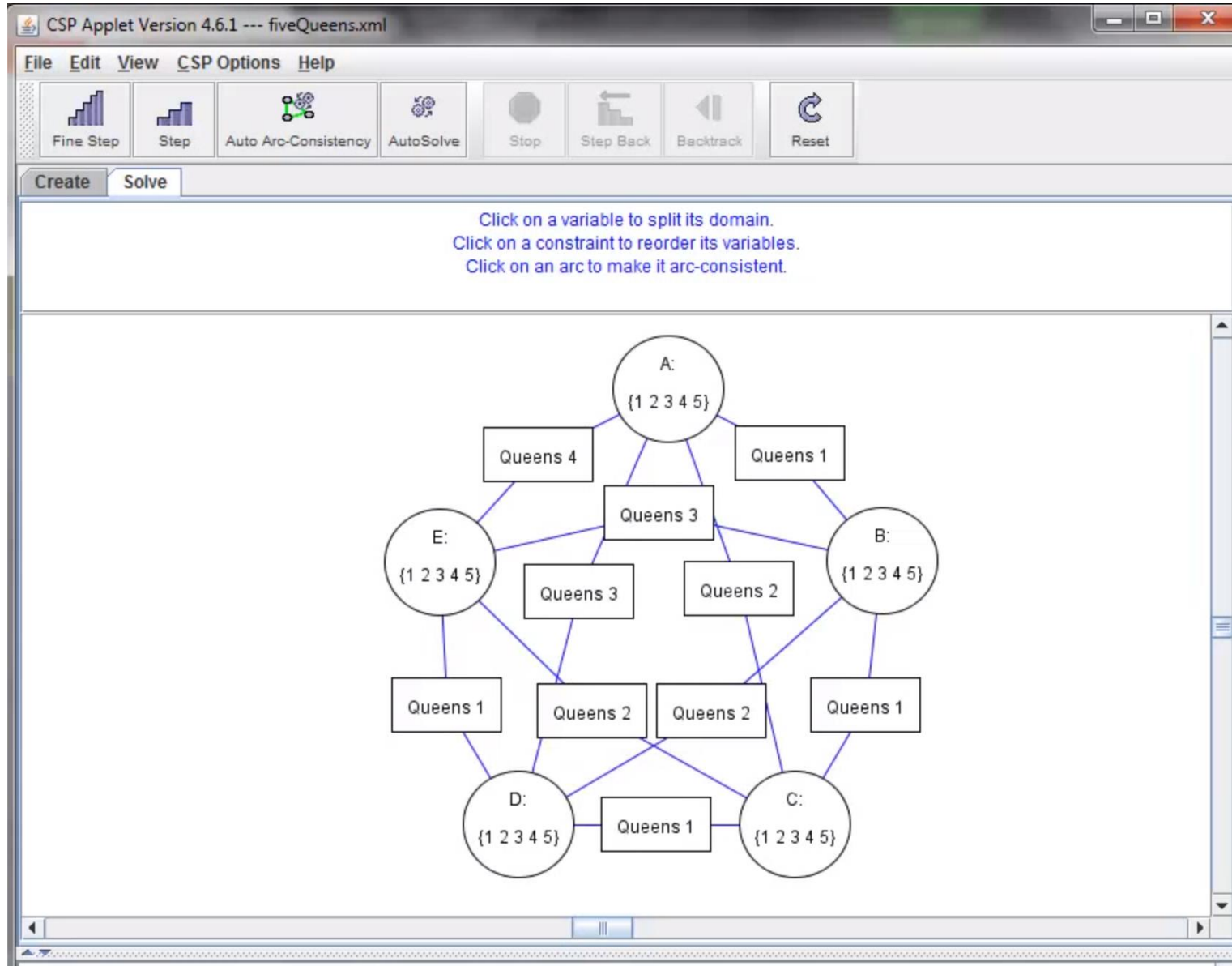


گراف محدودیت

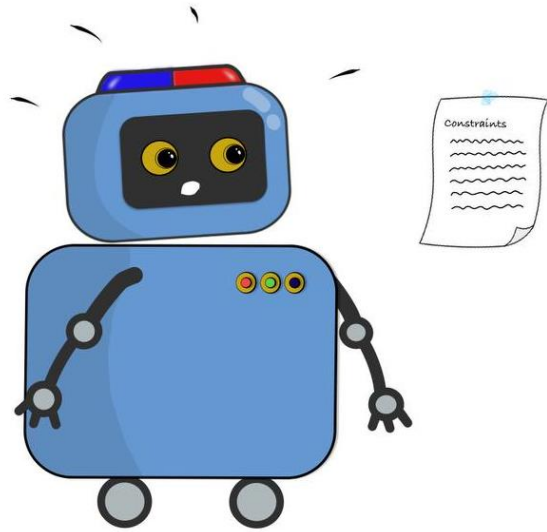
- CSP باینری: هر محدودیت (حداکثر) به دو متغیر مربوط می‌شود.
- گراف محدودیت باینری: گره‌ها متغیر هستند، یال‌ها محدودیت‌ها را نشان می‌دهند.
- الگوریتم‌های CSP همه منظوره از ساختار گراف برای سرعت بخشیدن به جستجو استفاده می‌کنند. به عنوان مثال، تاسمانی (T) یک زیرمسئله مستقل است!



اسکرین شات دموی N وزیر

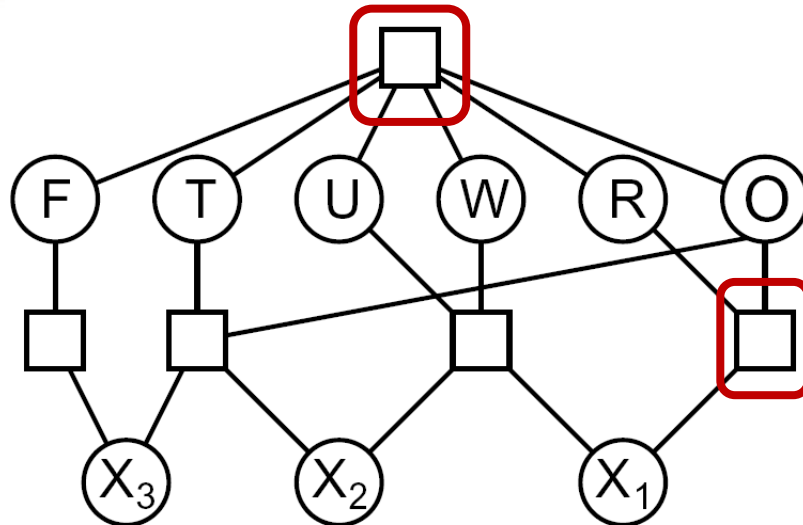


مثال: رمزگذاری



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

X_1



متغیرها:

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

دامنه‌ها:

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

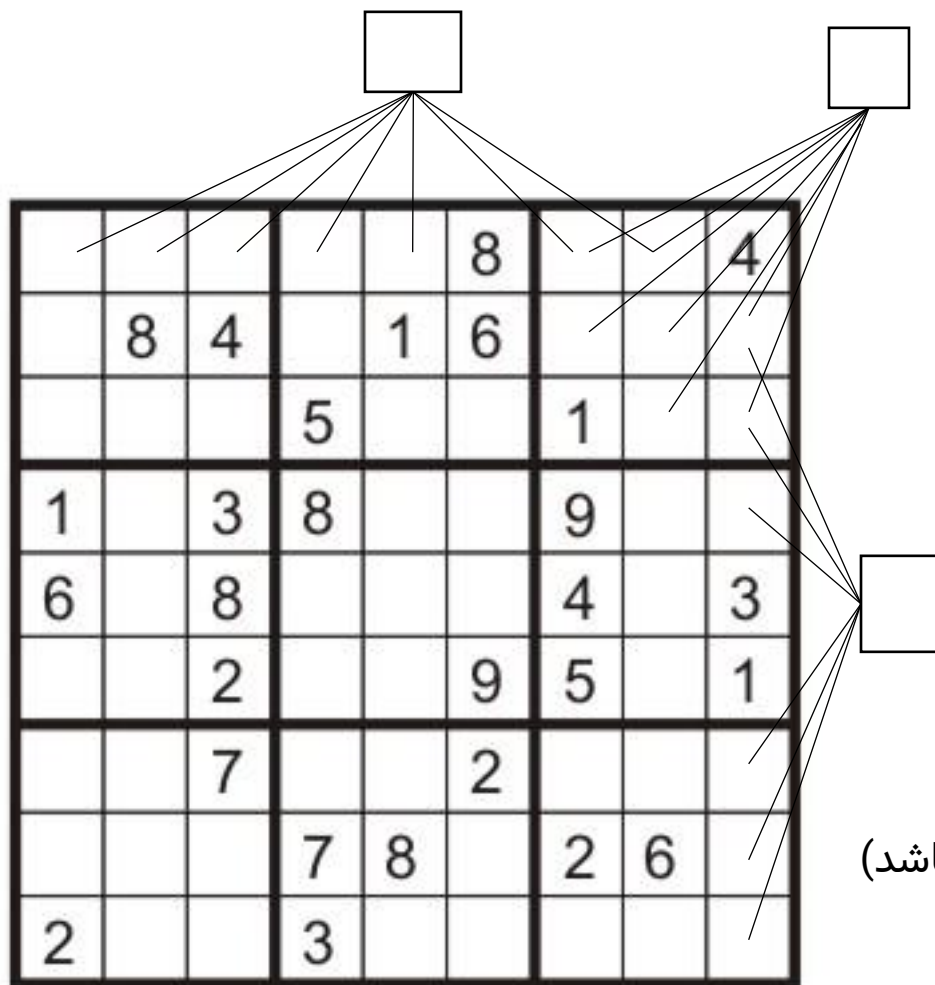
محدودیت‌ها:

$alldif(F, T, U, W, R, O)$

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

...

مثال: سودوکو



- متغیرها:

- هر مربع (خالی)

- دامنه‌ها:

- $\{1, 2, \dots, 9\}$

- محدودیت‌ها:

- 1 تا 9 متمایز (alldiff) برای هر ستون

- 1 تا 9 متمایز برای هر سطر

- 1 تا 9 متمایز برای هر ناحیه

(یا می‌تواند یک دسته از محدودیت‌های نابرابری دو به دو داشته باشد)

CSP های دنیای واقعی

- مسائل برنامه‌ریزی: به عنوان مثال، چه زمانی همه ما می‌توانیم ملاقات کنیم؟
- مسائل زمان‌بندی: به عنوان مثال، کدام کلاس در چه زمانی و در کجا ارائه می‌شود؟
- مسائل تخصیص: به عنوان مثال، چه کسی در چه کلاسی تدریس می‌کند.

- پیکربندی سخت افزار

- برنامه‌ریزی حمل و نقل

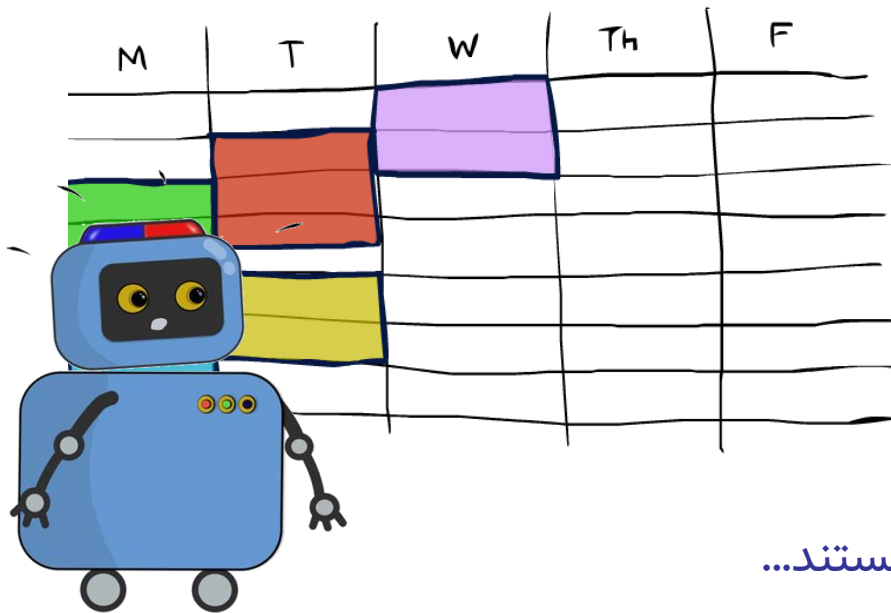
- برنامه‌ریزی کارخانه

- طراحی مدار الکتریکی

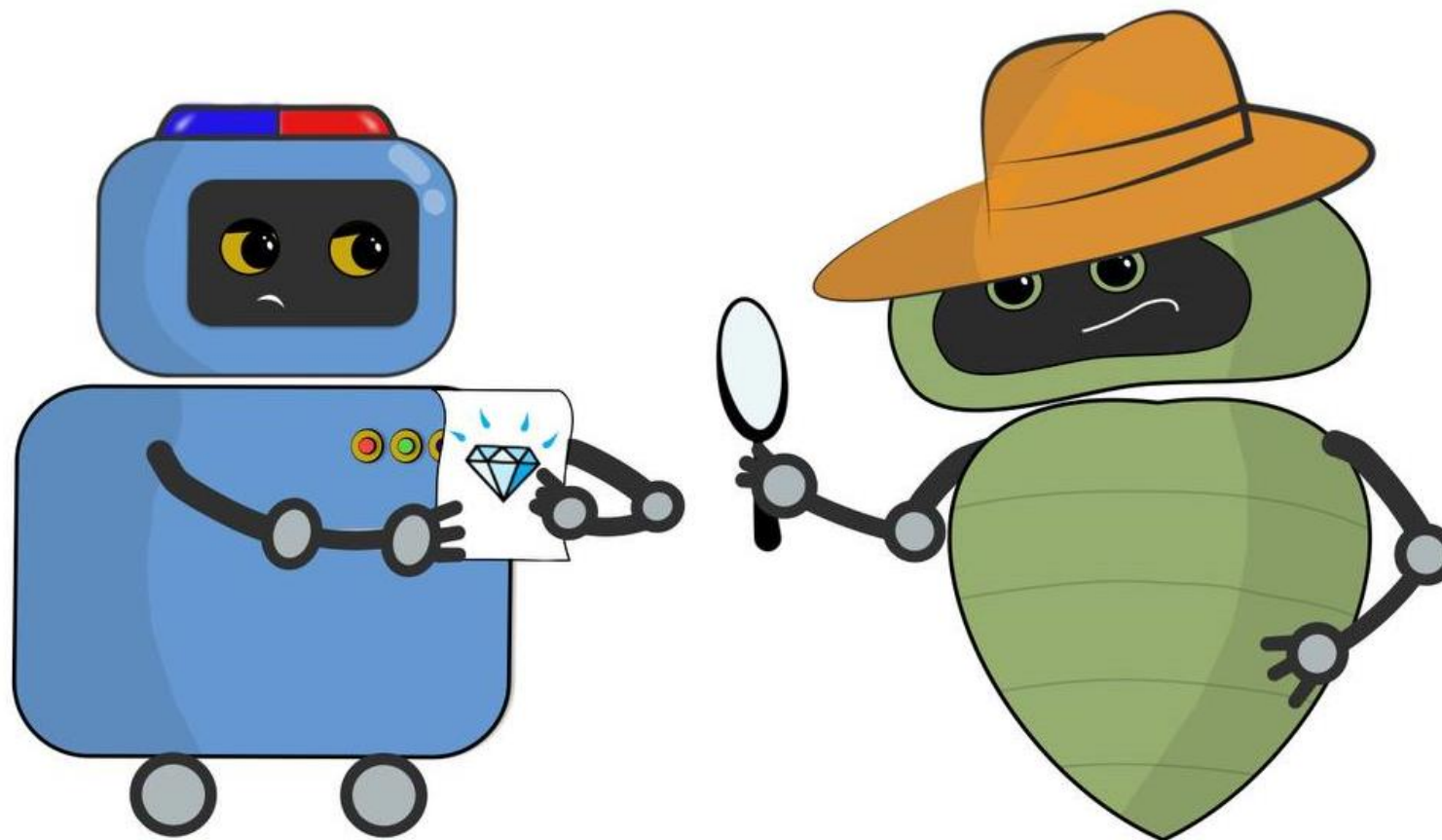
- تشخیص عیب

- ... خیلی بیشتر!

- بسیاری از مسائل دنیای واقعی شامل متغیرهایی با مقادیر حقیقی هستند...

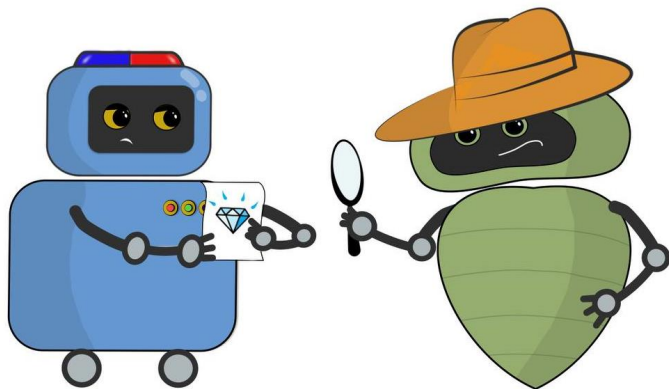


حل کردن CSP ها



فرمول جستجوی استاندارد (Standard Search)

- فرمول جستجوی استاندارد CSP ها
- تعریف حالت: مقادیر اختصاص یافته تاکنون (تخصیص جزئی متغیرها)
 - حالت اولیه: هیچ مقداری به هیچ متغیری اختصاص نیافته است، { }
 - تابع پسین: یک مقدار را به یک متغیر تخصیص نیافته اختصاص دهید
 - آزمون هدف: همه متغیرها تخصیص یافته است (تخصیص کامل)
و تخصیص صورت گرفته تمامی محدودیت ها را برآورده می کند
- ما با یک رویکرد صریح و ساده کار را شروع می کنیم، سپس آن را بهبود می بخشیم

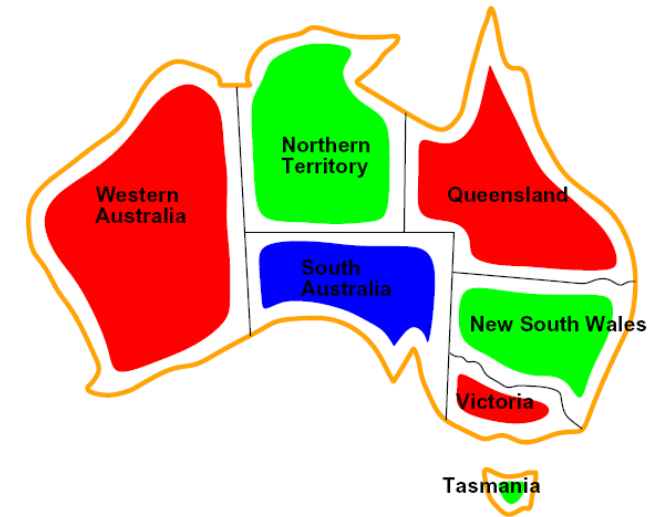


روش‌های جستجو

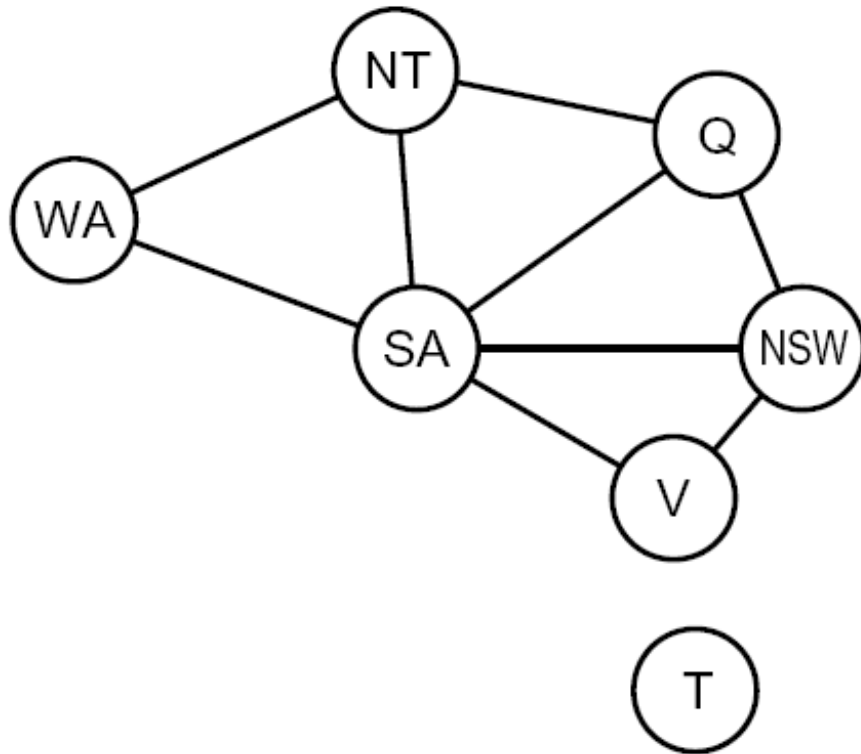
$\{\}$

$\{WA=g\}$ $\{WA=r\}$... $\{NT=g\}$...

• BFS چه کاری انجام می‌دهد؟

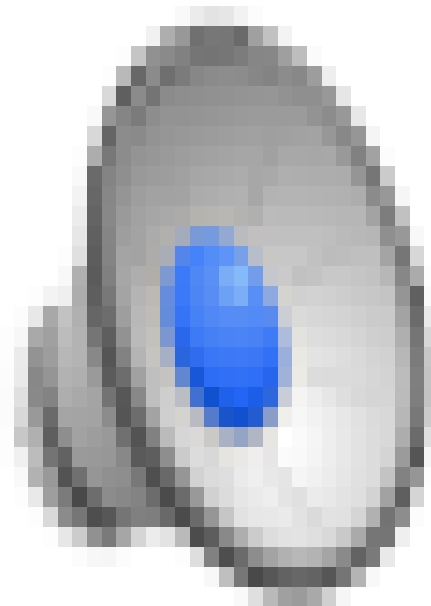


روش‌های جستجو

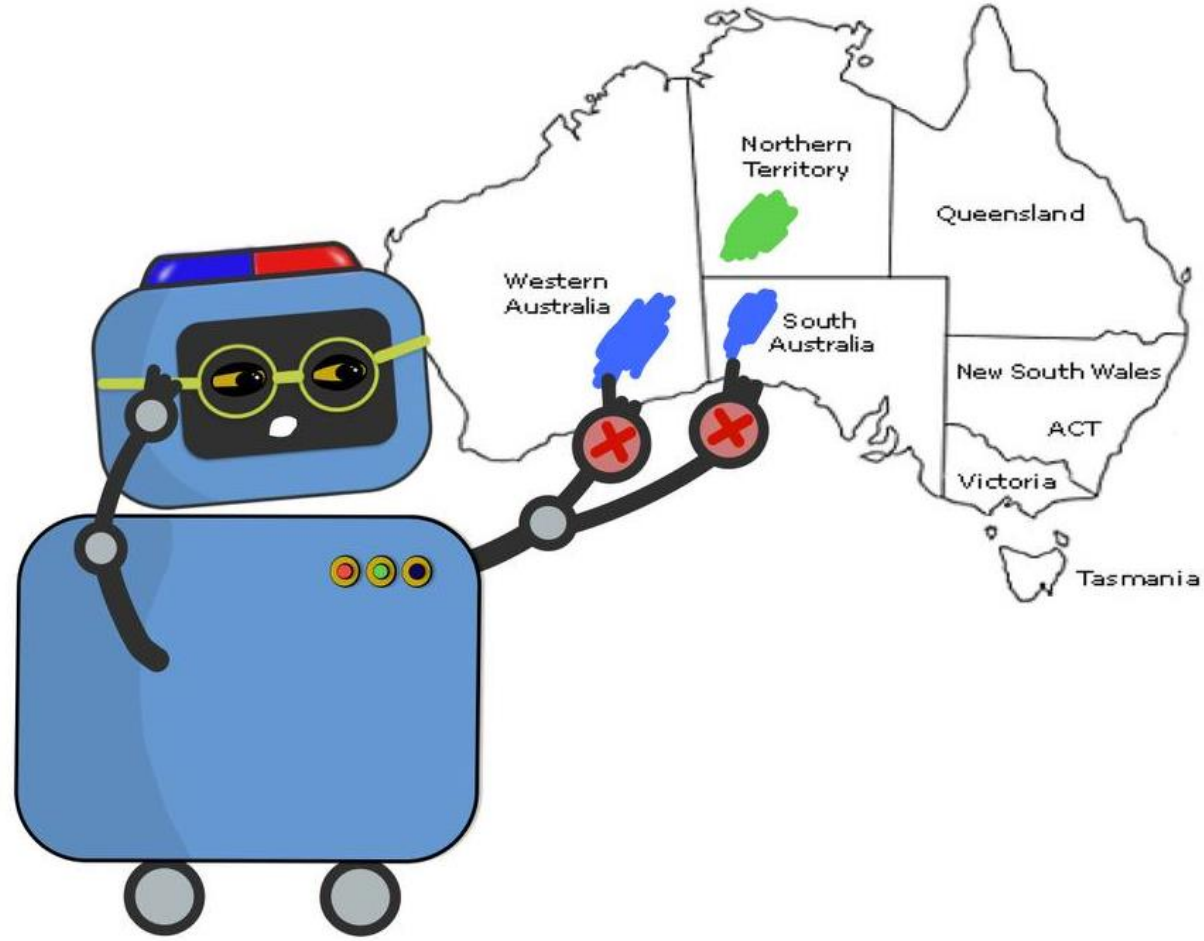


- BFS چه کاری انجام می‌دهد؟
- DFS چه کاری انجام می‌دهد؟
- جستجوی ساده چه مشکلاتی دارد؟

ویدیوی رنگ آمیزی دمو -- DFS



جستجوی عقبگرد (Backtracking Search)



جستجوی عقبگرد (پس‌گرد)

Backtrack search

- جستجوی عقبگرد یک الگوریتم ناآگاهانه پایه برای حل CSP ها است

- ایده 1: یک متغیر در هر گام

- مقداردهی متغیرها جابه‌جایی پذیر است، بنابراین یک ترتیب مشخص کنید.

- یعنی $[WA = red \text{ then } NT = green]$ با $[NT = green \text{ then } WA = red]$ برابر است

- در هر گام صرفاً لازم است تا تخصیص مقدار به یک متغیر را در نظر بگیریم

- ایده 2: محدودیت‌ها را در حین تخصیص، بررسی کنید

- یعنی تنها مقادیری را در نظر بگیرید که با تخصیص‌های قبلی مغایرت ندارند

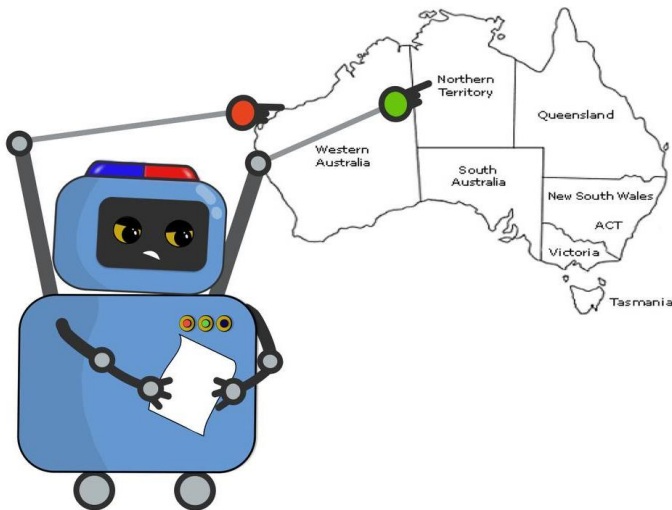
- ممکن است برای بررسی محدودیت‌ها، مقداری محاسبات انجام شود

- "آزمون هدف افزایشی"

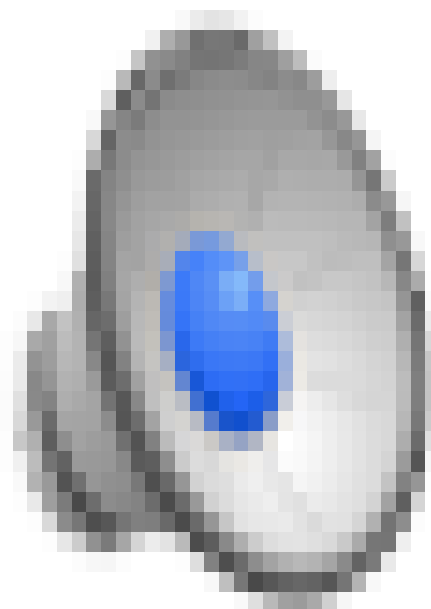
- جستجوی اول عمق (DFS) با این دو بهبود، جستجوی عقبگرد نامیده می‌شود (بهترین

نام نیست).

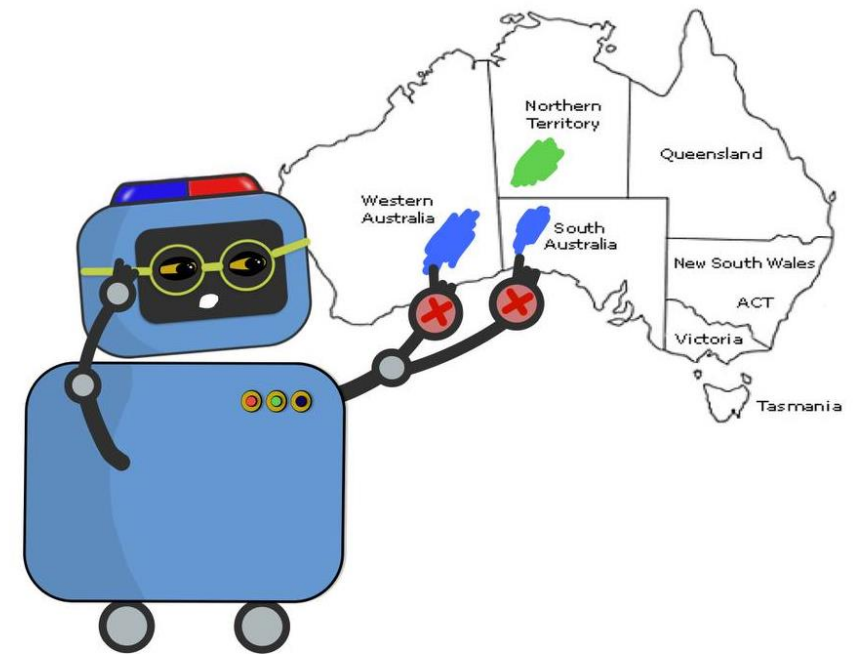
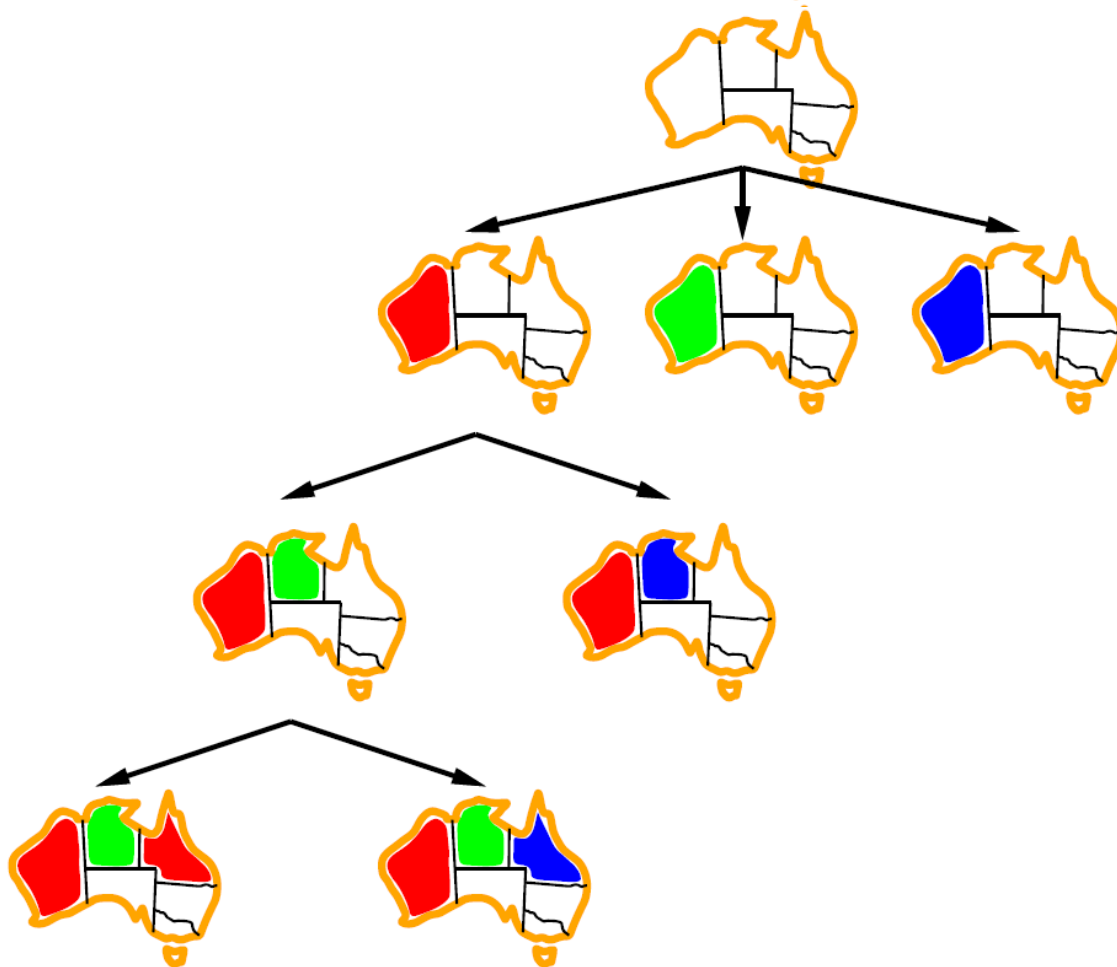
- می‌تواند n وزیر را برای $n \approx 25$ حل کند



ویدیوی رنگ آمیزی دمو -- عقبگرد



مثال عقبگرد



جستجوی عقبگرد

```
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 ← 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 ← RECURSIVE-BACKTRACKING(assignment, csp)
      if result ≠ failure then return result
      remove {var = value} from assignment
  return failure
```

- عقبگرد = DFS + مرتب‌سازی متغیرها (variable ordering) + شکست در هنگام مغایرت (fail-on-violation)
- نقاط انتخاب چه هستند؟

بهبود عقبگرد

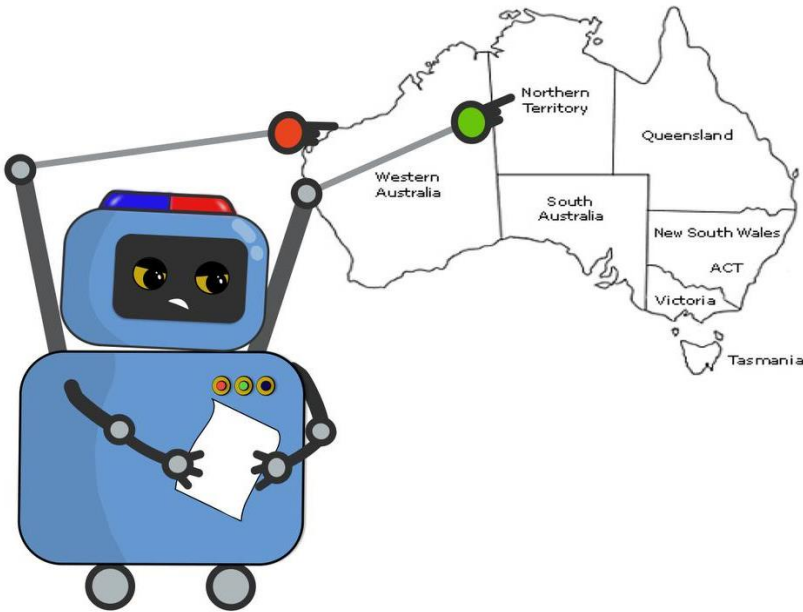
- ایده‌های همه منظوره دستاوردهای بزرگی در سرعت به همراه دارند

- مرتب سازی:

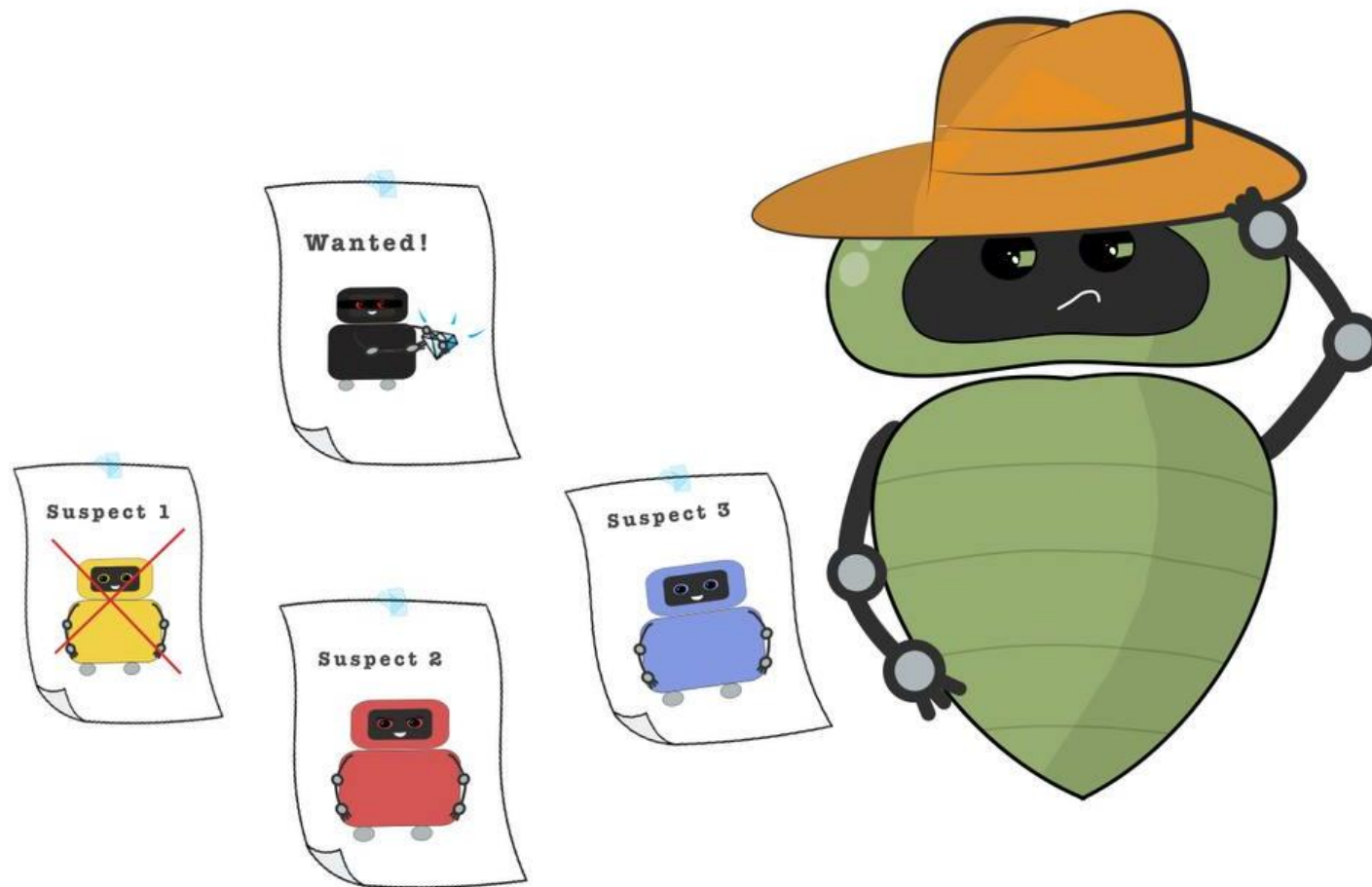
- کدام متغیر باید در مرحله بعد مقدار دهی شود؟
- مقادیر ممکن آن را به چه ترتیبی باید امتحان کرد؟

- فیلتر کردن: آیا می‌توانیم شکست اجتناب ناپذیر را زود تشخیص دهیم؟

- ساختار: آیا می‌توانیم از ساختار مسئله بهره برداری کنیم؟

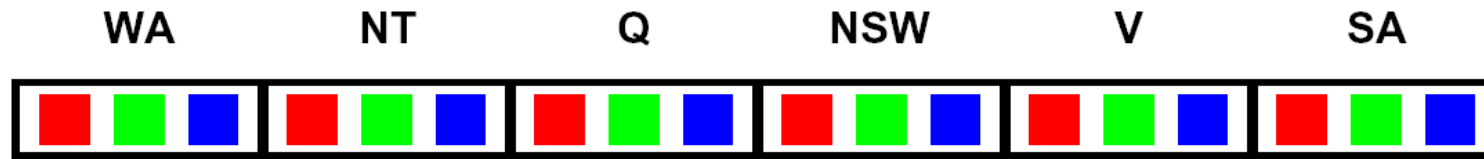


فیلتر کردن

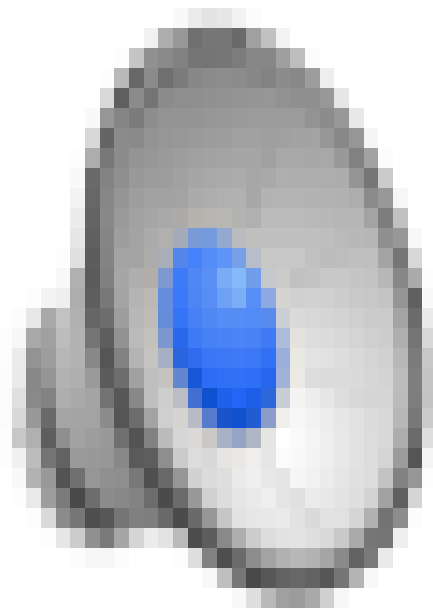


فیلتر کردن: بررسی رو به جلو (Forward Checking)

- فیلتر کردن: دامنه‌های متغیرهای تخصیص نیافته را ردیابی کنید و گزینه های نامناسب را کنار بگذارید
- بررسی رو به جلو: خط زدن مقادیری که با تخصیص‌شان به تخصیص‌های فعلی محدودیتی نقض می‌شود از دامنه

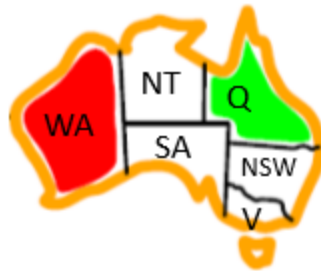


ویدیوی رنگ آمیزی دمو - عقبگرد با بررسی رو به جلو



فیلتر کردن: انتشار محدودیت (Constraint Propagation)

- "بررسی رو به جلو" اطلاعات را از متغیرهای اختصاص یافته به متغیرهای اختصاص نیافته انتشار می دهد، اما نمی تواند تشخیص زودهنگام را برای تمام حالت های شکست ها فراهم کند:

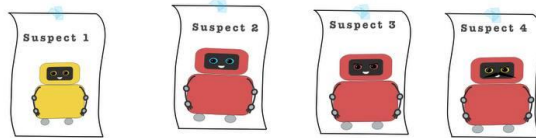
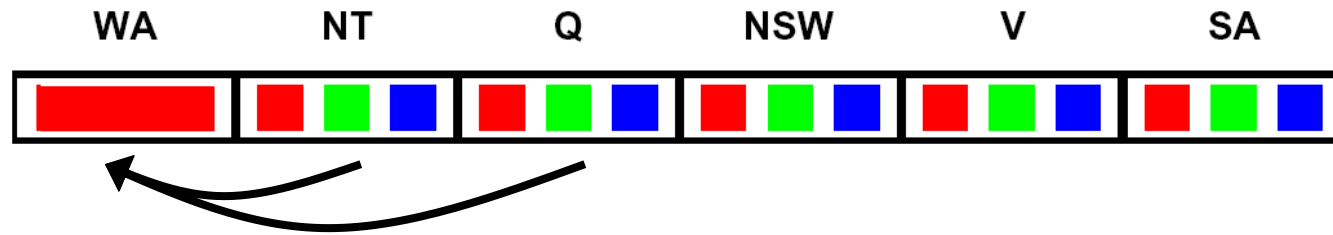
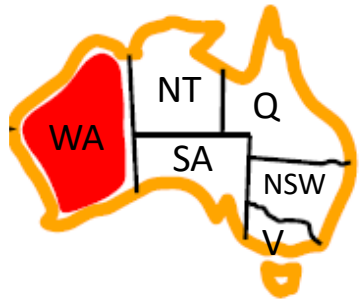


WA	NT	Q	NSW	V	SA
<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>

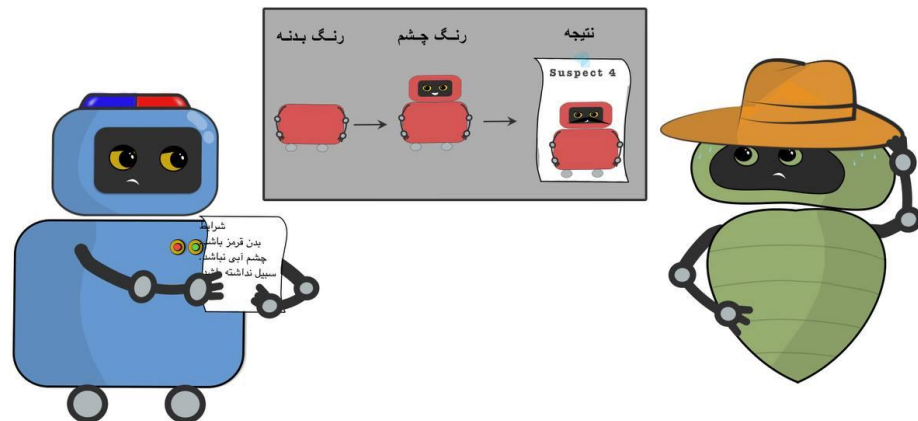
- NT و SA نمی توانند هر دو آبی باشند!
- چرا ما تاکنون این مشکل را تشخیص ندادیم؟
- انتشار محدودیت: استدلال در مورد یک محدودیت از طریق محدودیت دیگر

سازگاری یک یال (Arc Consistency)

- یال $X \rightarrow Y$ سازگار (consistent) است اگر و تنها اگر به ازای هر مقدار x در دامنه X ، حداقل یک مقدار y در دامنه Y وجود داشته باشد که بتواند بدون نقض محدودیتی بین دو متغیر X و Y ، اختصاص داده شود



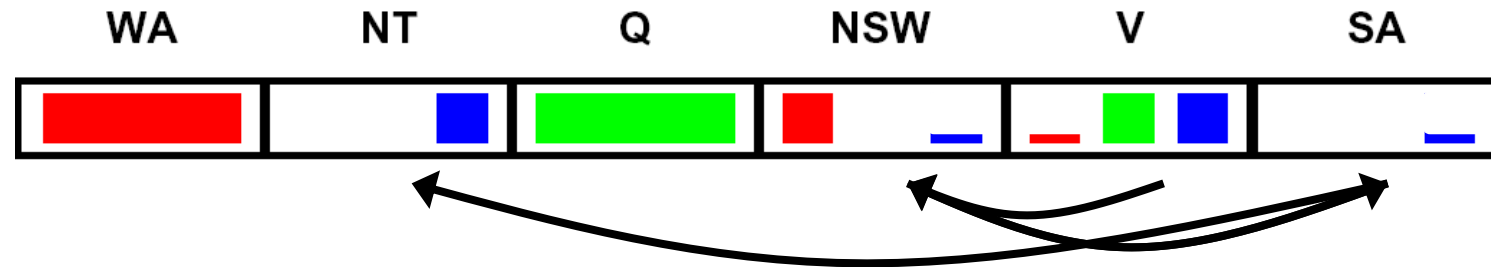
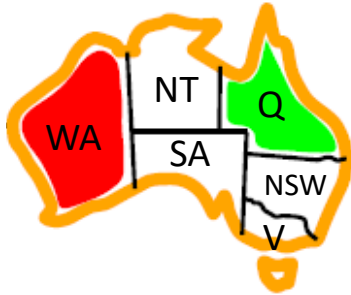
- اگر سازگاری وجود نداشت از ته (X) مقدار ناسازگار را حذف کنید!



- بررسی رو به جلو؟
اعمال سازگاری یال‌هایی که به تخصیص جدید اشاره می کنند

سازگاری یال یک CSP کامل

- یک مدل ساده از انتشار محدودیت، اطمینان حاصل می کند که تمام یال ها سازگار هستند:



به یاد داشته باشید: حذف از ته!

- نکته مهم: اگر X مقداری را از دست داد، همسایگان X باید دوباره بررسی شوند!
- "سازگاری یال"، شکست را زودتر از "بررسی رو به جلو" تشخیص می دهد
- می تواند به عنوان یک پیش پردازنده یا بعد از هر تخصیص اجرا شود.
- ضعف اعمال سازگاری یال چیست؟

اعمال سازگاری یال در یک CSP

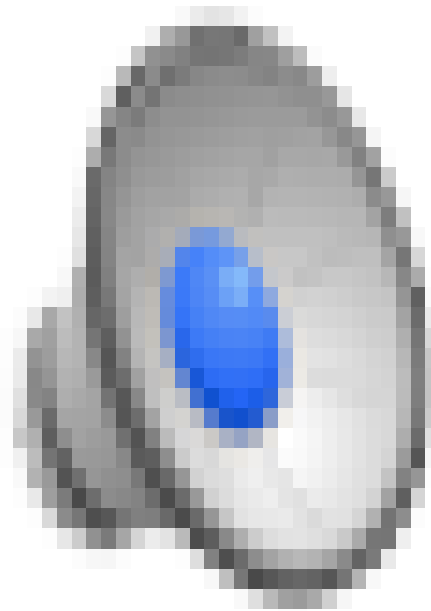
```
function AC-3(csp) returns the CSP, possibly with reduced domains
inputs: csp, a binary CSP with variables  $\{X_1, X_2, \dots, X_n\}$ 
local variables: queue, a queue of arcs, initially all the arcs in csp

while queue is not empty do
     $(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(\text{queue})$ 
    if REMOVE-INCONSISTENT-VALUES( $X_i, X_j$ ) then
        for each  $X_k$  in NEIGHBORS[ $X_i$ ] do
            add  $(X_k, X_i)$  to queue
```

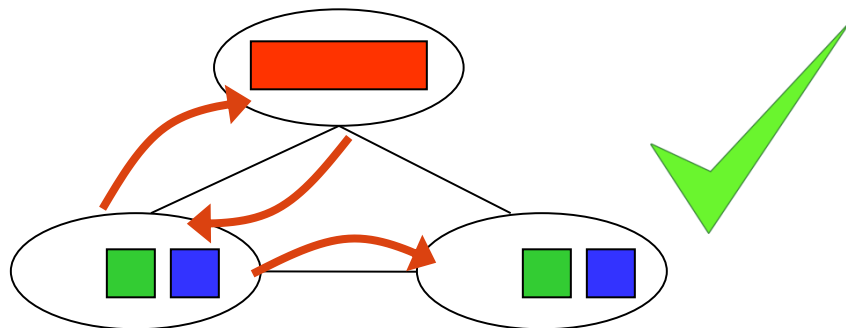
```
function REMOVE-INCONSISTENT-VALUES( $X_i, X_j$ ) returns true iff succeeds
    removed  $\leftarrow$  false
    for each  $x$  in DOMAIN[ $X_i$ ] do
        if no value  $y$  in DOMAIN[ $X_j$ ] allows  $(x, y)$  to satisfy the constraint  $X_i \leftrightarrow X_j$ 
            then delete  $x$  from DOMAIN[ $X_i$ ]; removed  $\leftarrow$  true
    return removed
```

- زمان اجرا: $O(n^2d^3)$ ، می‌تواند به $O(n^2d^2)$ کاهش یابد (AC-4 algorithm (Mohr and Henderson, 1986))
- ... اما تشخیص همه مشکلات احتمالی آینده NP-hard است - چرا؟

ویدیوی دمو سازگاری یال – CSP Applet – N – وزیر

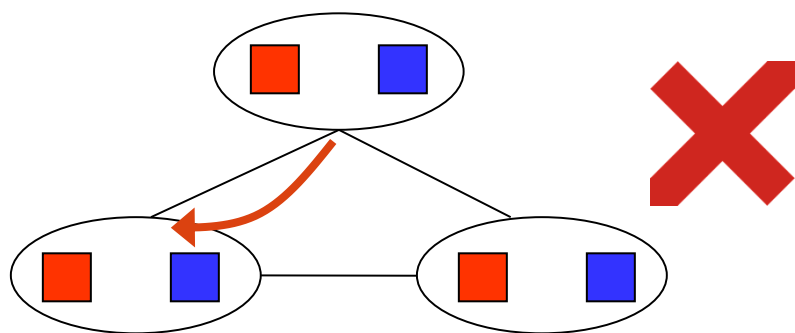


محدودیت‌های سازگاری یال



- پس از اعمال سازگاری یال:

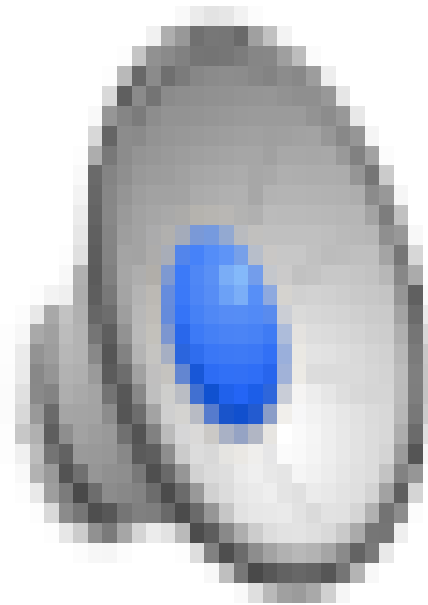
- می‌تواند یک راه حل باقی بماند
- می‌تواند چندین راه حل باقی مانده باشد
- می‌تواند هیچ راه حلی باقی نمانده باشد (و الگوریتم جستجو نداند)



- سازگاری یال همچنان در داخل جستجوی عقبگرد اجرا می‌شود!

اینجا چه اشتباهی
رخ داده است؟

ویدیوی رنگ آمیزی دمو - عقبگرد با بررسی رو به جلو - گراف پیچیده



ویدیوی رنگ آمیزی دمو - عقبگرد با سازگاری یال - گراف پیچیده

