مدرس: ایمان مختاری فرد
دانشگاه پیام نور شهرکرد

هوش مصنوعی

فصل پنجم

مسائل ارضای محدودیت

Constraint Satisfaction Problems

مسائل ارضای محدودیت

یک مسئله ارضای محدودیت (CSP) به صورت :

- مجموعه ای از متغیرها؛ X1, X2, ..., Xn
- مجموعه ای از محدودیتها؛ C1, C2, ..., Cm
- دامنه های ناتهی از مقادیر برای هر یک از متغیرها؛ D1,D2,...,Dn

هر محدودیت Ci شامل زیرمجموعه ای از متغیرهاست و ترکیبهای ممکن مقادیر را برای آن زیرمجموعه مشخص می سازد.

- یک حالت از مسئله به صورت **انتساب** مقادیر به تعدادی از متغیرهای مسئله یا تمامی آنها تعریف می شود.
- انتسابی که هیچ محدودیتی را نقض نکند، انتساب **سازگار یا مجاز** نام دارد
- در یک انتساب **کامل** تمامی متغیرها مقداردهی می شوند.
- یک **راه حل برای** CSP یک انتساب کامل است که تمام محدودیتها را برآورده سازد.
- بعضی از CSPها به راه حلهایی نیاز دارند که **تابع هدف** را بیشینه کنند

انواع مسائل CSP

- متغیرهای گسسته
  - دامنه های محدود:
    - n متغیر، اندازه هر دامنه d ← تعداد انتساب های کامل $O(d^n)$
    - مثال: CSP های بولین ، n- وزیر ، رنگ آمیزی نقشه و ...
  - دامنه های نامحدود:
    - اعداد صحیح، رشته ها و ...
    - مثال: زمانبندی کارها- متغیرها، زمان شروع/پایان هر کار هستند.
    - نیاز به زبان محدودیت دارند. مثال: $StartJob_1 + 5 \leq StartJob_3$
- متغیرهای پیوسته
  - مثال: زمان های شروع و پایان مشاهدات تلسکوپ فضایی هابل
  - محدودیت های خطی که توسط برنامه ریزی خطی قابل حل در زمان چندجمله ای می باشند.

انواع محدودیتها در مسائل CSP

- **یکانی (Unary)** : روی یک متغیر تعریف می شود،
  - مثال: $SA \neq green$
- **دودویی (Binary)**: محدودیت شامل یک زوج از متغیرها می باشد،
  - مثال: $SA \neq WA$
- **مرتبه بالاتر (Higher-order)**: محدودیت شامل سه یا بیشتر متغیر است،
  - مثال: محدودیت های موجود در ستون های مسائل ریاضیات رمزی
  - در صورت متناهی بودن دامنه، می تواند به تعدادی محدودیت دودیی کاهش یابد.

- محدودیت های مثال های بالا همگی از نوع **مطلق** می باشند. نقض یک محدودیت مطلق به معنای حذف یک راه حل بالقوه می باشد.

محدودیت اولویت دار (نرم): در برخی اولویت بین راه حلها وجود دارد و مطلق نیستند

مثلا، قرمز بر سبز ارجحیت دارد. اغلب بوسیله در نظر گرفتن هزینه برای انتساب متغیرها قابل بازنمایی می باشند. ← Constraint Optimization Problem

یا مثلا استاد X تمایل به تدریس در بعد از ظهر دارد و یا... می توان در تابع هزینه ، هزینه تدریس در صبح را ۲ و بعد از ظهر را ۱ فرض داشت

مثال CSP: رنگ آمیزی نقشه

نقشه ای از نواحی مختلف استرالیا را در اختیار داریم.
می خواهیم این نقشه را با استفاده از رنگهای آبی، قرمز و سبز به گونه ای رنگ آمیزی کنیم که هیچ یک از نواحی مجاور در نقشه دارایرنگهای مشابه نباشند.

متغیرها: WA, NT, Q, NSW, V, SA, T

دامنه: Di = {آبی ، سبز ، قرمز}

محدودیتها: دو منطقه مجاور، همرنگ نباشند

مثال: WA ≠ NT یعنی (WA, NT) عضو

{(قرمز ,سبز) ,(آبی ,قرمز) ,(سبز ,قرمز),
(سبز,آبی),(قرمز,آبی) ,(آبی ,سبز)}

مثال CSP: حساب رمزی

مثال : حروف عبارت زیر را به شکلی به اعداد یک رقمی انتساب دهید که عبارت درست باشد و هر عدد فقط به یک حرف منتسب شده باشد.

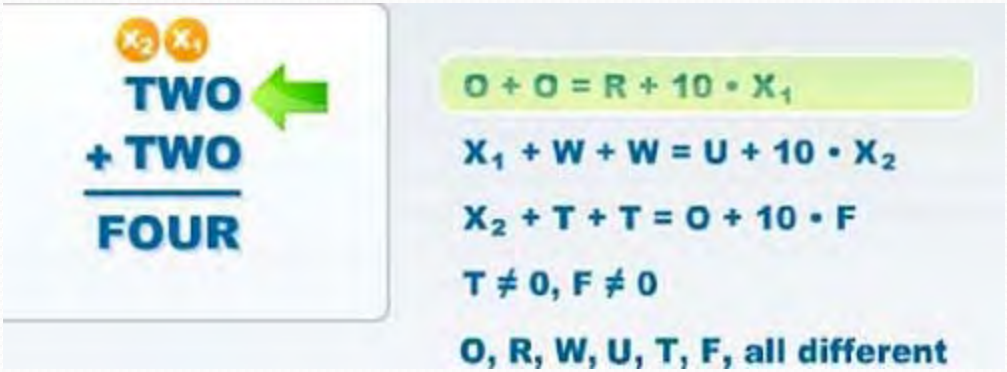
متغیرها: $X_1, X_2, F, T, U, W, R, O$

دو متغیر X_1 و X_2 مجازیند، زیرا جمع ممکن است ده بر یک داشته باشد.

- دامنه‌ی مقادیر:

$O \in [0..9]$, $R \in [0..9]$, $W \in [0..9]$, $U \in [0..9]$, $T \in [0..9]$, $F \in [0..9]$, $X1 \in \{0,1\}$, $X2 \in \{0,1\}$

محدودیت‌ها:

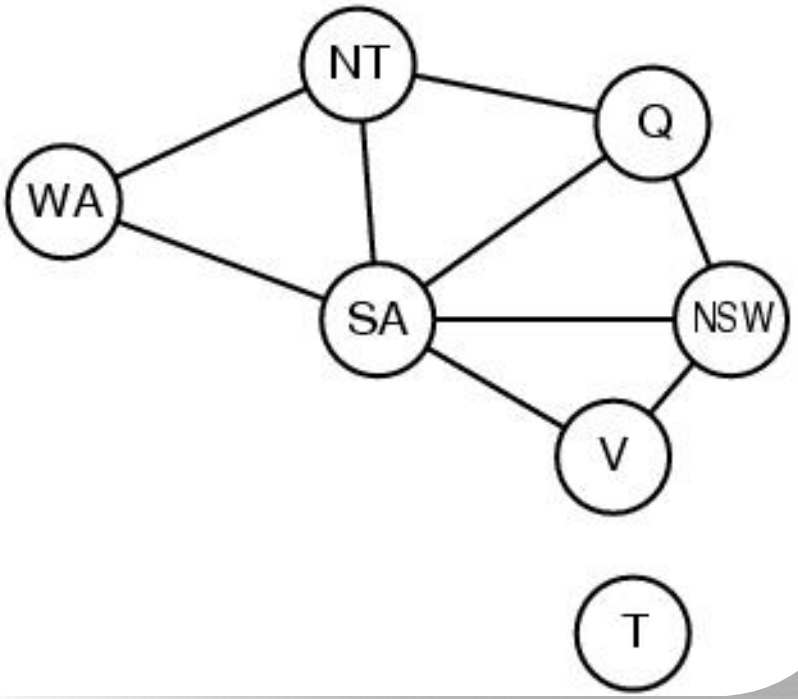


مثال CSP: هشت وزیر

- متغیرها: Q1 تا Q8 (سطر قرار گرفتن وزیری که در ستون اول تا هشتم است).
- مقادیر: هر متغیری می‌تواند ۱ تا ۸ بگیرد.
- محدودیت‌ها:
  - هیچ دو متغیری یک مقدار نداشته باشند (در یک سطر قرار نگیرند).
  - هیچ دو وزیری در یک خط مورب قرار نگیرند.

نمایش یک CSP به صورت گراف محدودیت

- گراف برای ساده تر کردن جست و جو بکار میرود



- در گراف محدودیت:
  - گره ها: متغیرها
  - یال ها: محدودیتها

مزایای در نظر گرفتن یک مسأله به عنوان یک CSP

- از آنجا که بیان حالات در CSP ها از یک الگوی استاندارد (یعنی مجموعه ای از متغیرها با مقادیری برای انتساب) تبعیت می کند، تابع پسین و آزمون هدف می توانند به گونه ای کلی نوشته شوند تا قابل اعمال به تمام CSP ها بشند
- می توان هیوریستیکهای کلی تر و مؤثرتری که به داشتن تخصص ویژه در یک زمینه خاص نیاز ندارند، طراحی نمود.
- ساختار گراف محدودیت می تواند در راستای ساده سازی روند حل مسأله، به طراح کمک فراوان نماید.

برای CSP میتوان تدوین افزایشی به شکل زیر ارائه کرد:

- **حالت اولیه:** انتساب تهی {} که در آن، هیچ متغیری مقداردهی نشده است.
- **تابع پسین:** انتساب یک مقدار به هر متغیر فاقد مقدار، به شرطی که با متغیرهایی که قبلا مقدار گرفتند، تناقض نداشته باشد.
- **آزمون هدف:** آیا انتساب جاری کامل است.
- **هزینه مسیر:** یک هزینه ثابت (مثلا ۱) برای هر مرحله

تدوین CSP

- هر راه‌حل باید یک انتساب کامل باشد، لذا، اگر مسأله دارای nمتغیر باشد در n مرحله انجام خواهد شد و درخت جستجو تنها تا nگام پیش خواهد رفت.

- در مسائل CSP، بیشتر از الگوریتمهای جستجوی اول‌عمق استفاده می‌شود.

- مسیرهایی نیز وجود دارد که به یک راه‌حل بی‌ربط می‌رسد. لذا می‌توان از یک **تدوین حالت کامل** که در آن هر حالت، یک انتساب کامل است که ممکن است در محدودیتها صدق کند یا خیر، استفاده نمود.

- در این تدوین، روشهای جستجوی محلی می‌توانند مفید واقع شوند

خصوصیات جستجوی اول عمق

هر پاسخ در عمق n با n متغیر ظاهر می شود. ← استفاده از جستجوی عمقی

در عمق l، فاکتور انشعاب (b) برابر است با $(n-l)*d$.

بنابراین تعداد برگهای درخت جستجو برابر است با $n! * d^n$!!!!

$(nd) * [(n-1)d] * [(n-2)d] * \ldots d = n! d^n$

n= تعداد متغیر

d = تعداد مقدار

یعنی در مساله ۸-وزیر تعداد فضای حالت برابر است با $8!8^8$

بهینه شدن جستجوی اول عمق

1. ترتیب انتساب مقادیر به متغیرها اهمیت نداشته باشد، پس بسیاری از مسیرها معادل یکدیگرند

پس فضای حالت به d^nکاهش می یابد

2. انتساب های بعدی نمی توانند یک محدودیت نقض شده را تصحیح کنند.

(مثلا اگر در ۸-وزیر دو وزیر اول در یک ردیف قرار بگیرند، جستجوی عمقی 8^6 حالت دیگر رابررسی میکند)

خصوصیات جستجوی پس گرد

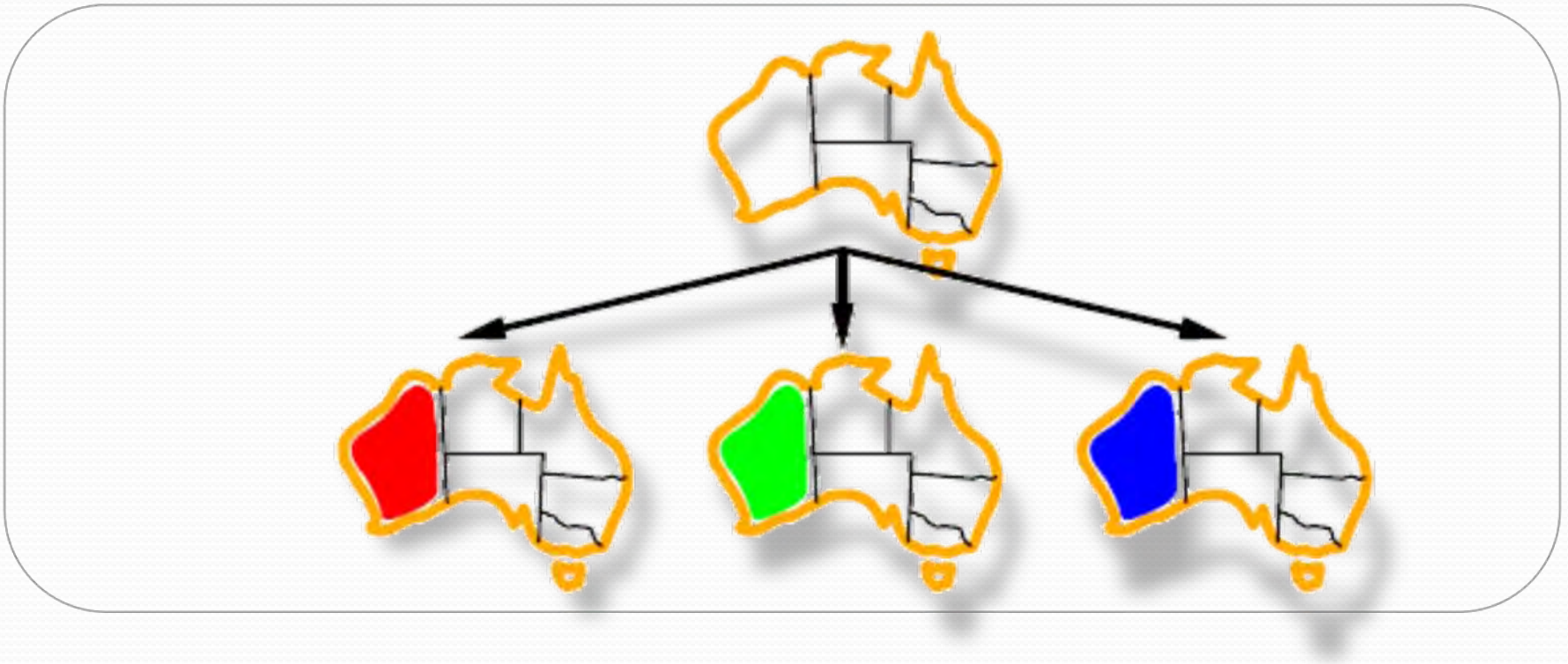
- جست و جوی اول عمقی که در آن :

  در هر زمان برای یک متغیر مقادیری در نظر گرفته می شود و هنگامی که هیچ مقدار مجازی برای انتساب به یک متغیر باقی نمانده باشد به عقب بر می گردد.

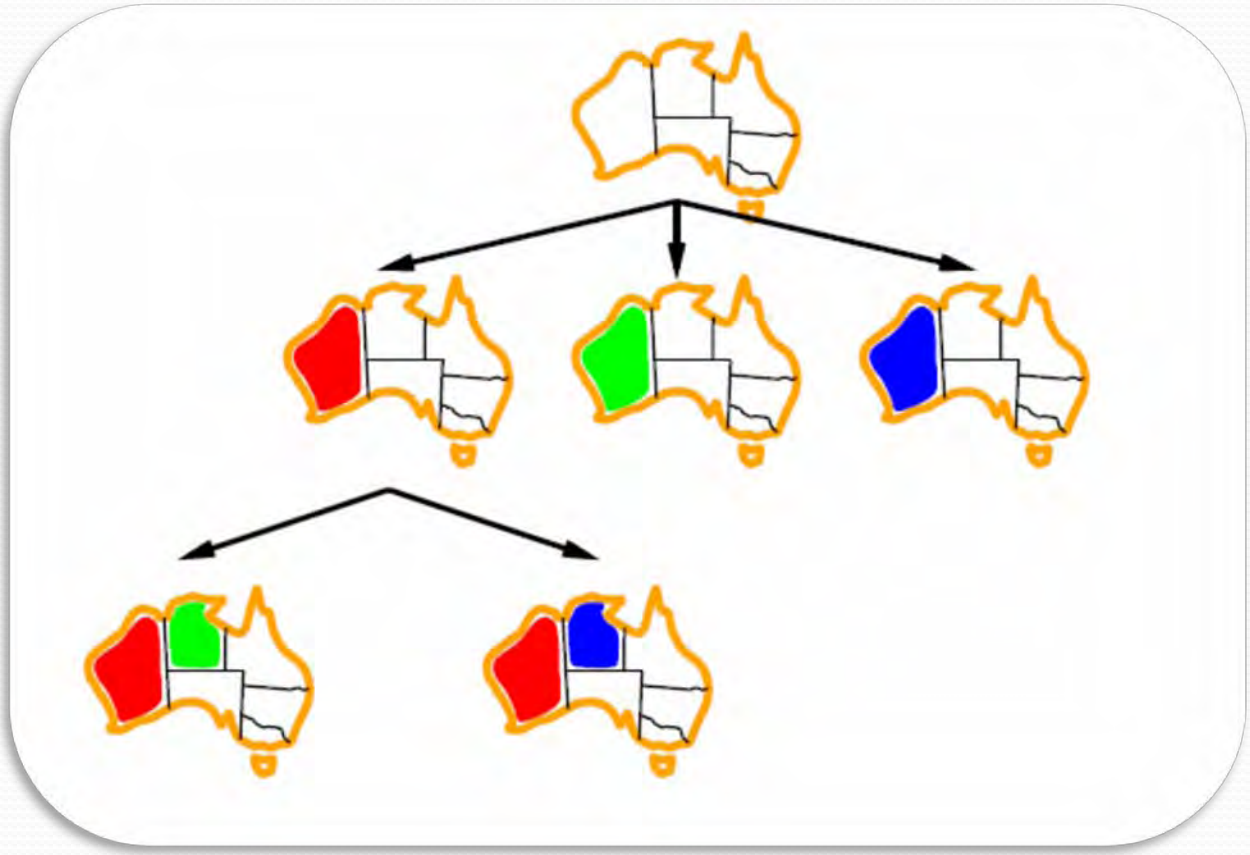
- با توجه به خصوصیت جابجایی پذیری در CSP ها تعداد برگها به d^n کاهش می یابد
- یک روش ناآگاهانه است
- برای مسأله های بزرگ کارآمد نیست

مثال جست و جوی پس گرد در CSP ها

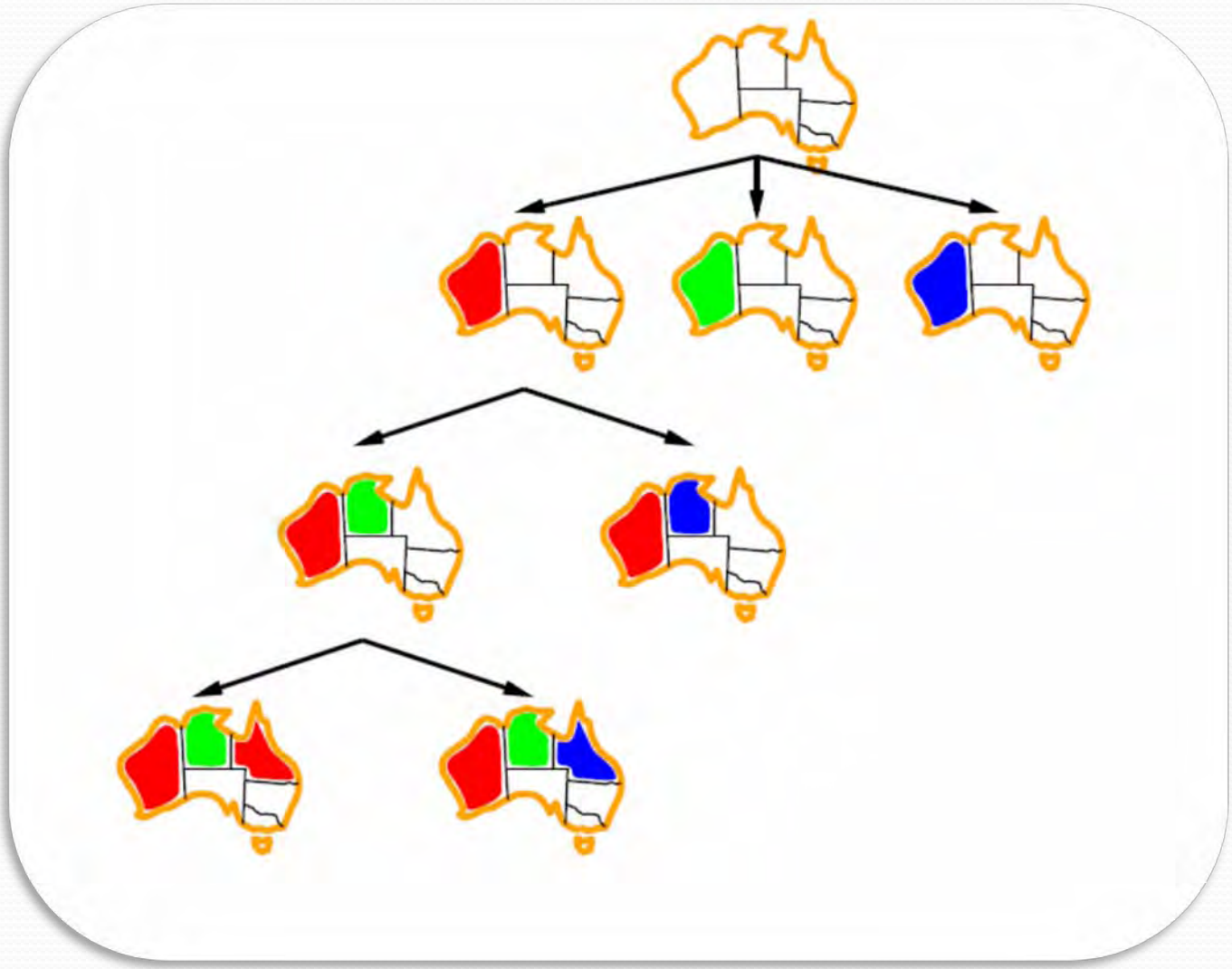
مثال جست و جوی پس گرد در CSP ها



مثال جست و جوی پس گرد در CSP ها



مثال جست و جوی پس گرد در CSP ها

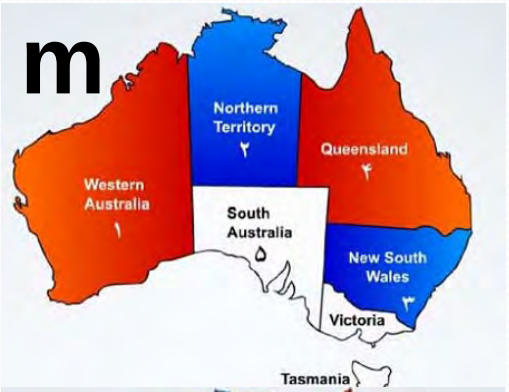


جستجوی پس گرد برای مسئله رنگ آمیزی

جستجوی پس گرد برای مسئله رنگ آمیزی

جستجوی پس گرد برای مسئله رنگ آمیزی

پس گرد دوم



افزایش سرعت توسط روش های جستجوی همه منظوره با پاسخ به:

نا آگاهانه است

۱- در مرحله بعد باید به کدام متغیر مقدار داده شود؟مقادیر آن باید به چه ترتیبی امتحان شوند؟

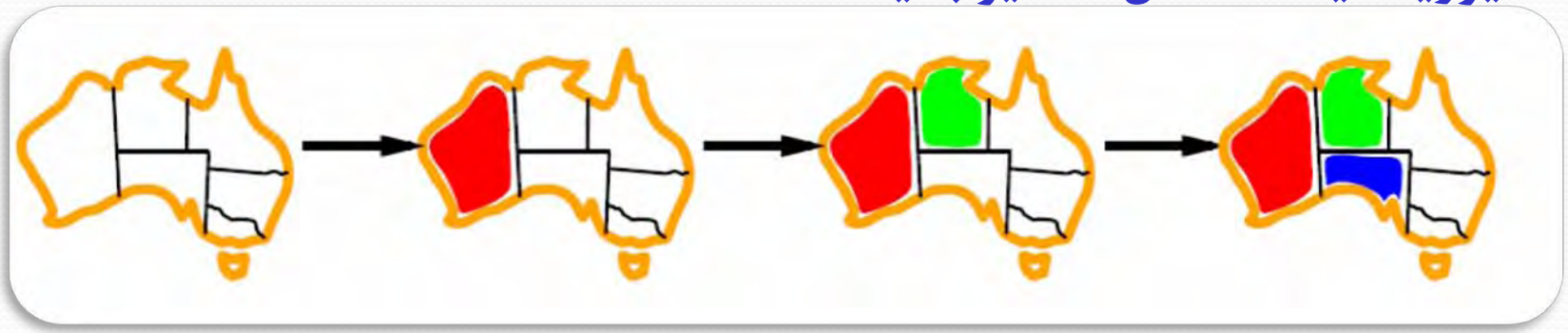
۲- آیا می توان شکست های حتمی را زودتر تشخیص داد؟

فرض کنید در عقبگرد در مسأله ۸- وزیر، ۶ وزیر اول به گونه ای قرار گرفته اند که قرار دادن هشتمین وزیر را غیر ممکن می سازند. عقب گرد تمام ممکنهای ممکن برای وزیر هفتم را چک می کند، اگرچه مسأله غیر قابل حل می باشد.

۳- آیا می توان از ساختار مسأله بهره گرفت؟

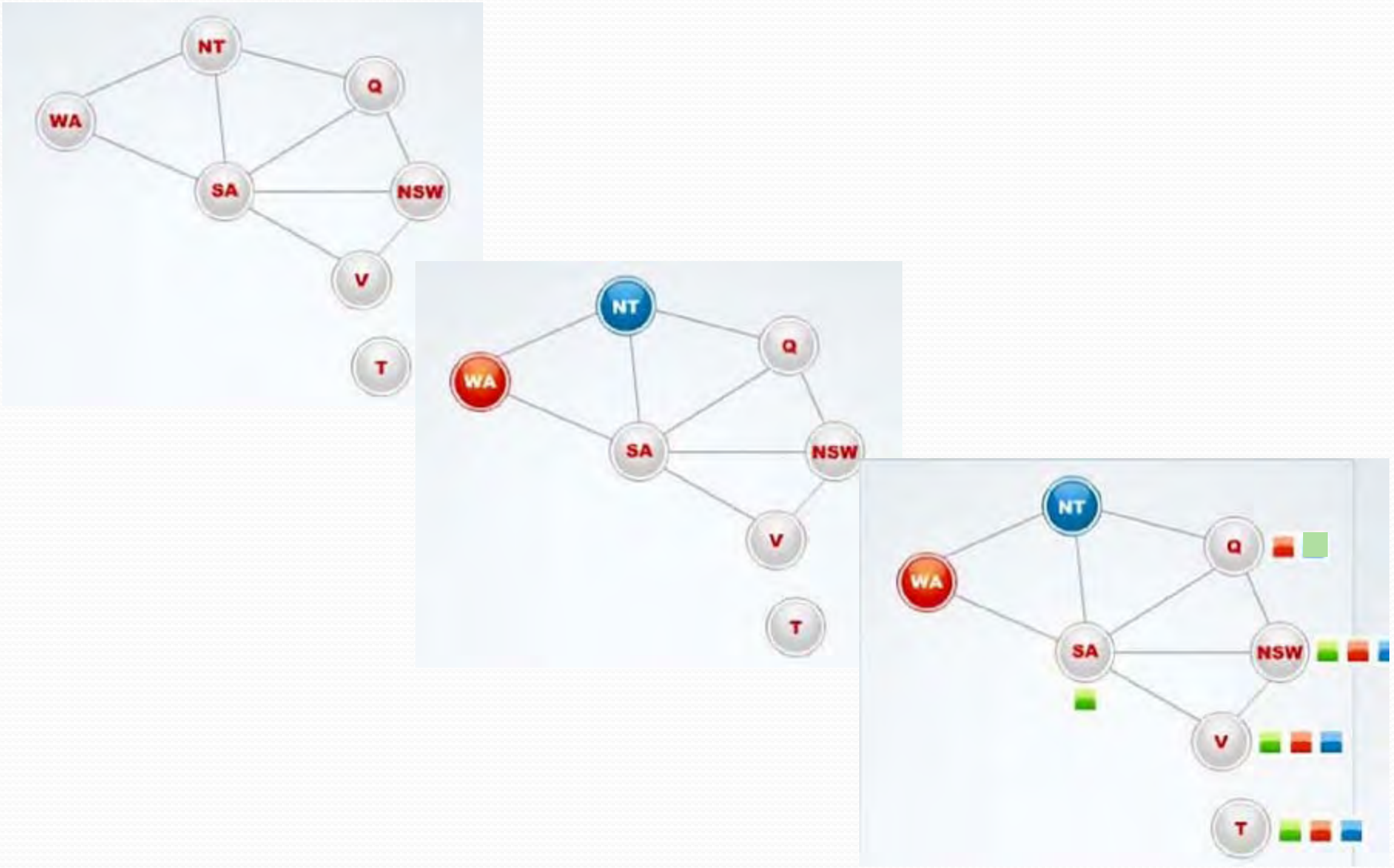
ترتیب متغیر

۱-هیوریستیک حداقل مقادیر باقیمانده (MRV)

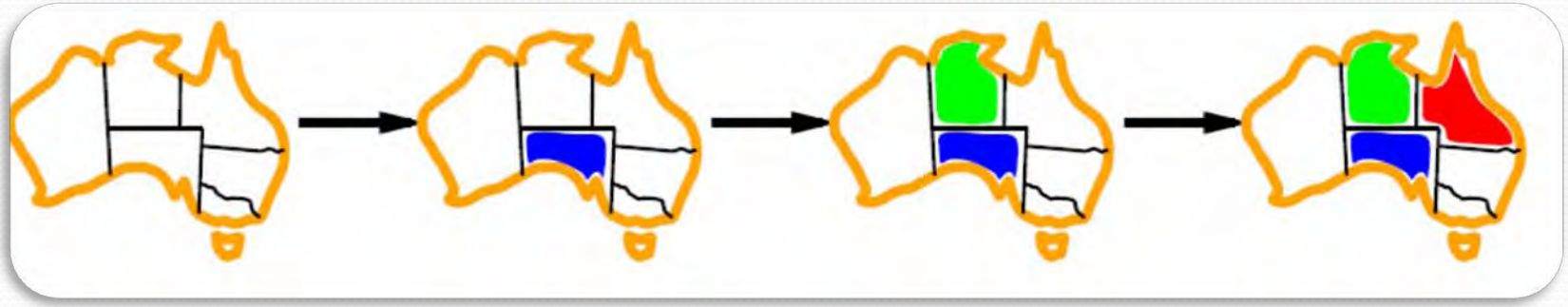


- اختصاص مقدار به متغیری با کمترین تعداد مقادیر مجاز
- نامهای دیگر آن هیوریستیک متغیر با بیشترین محدودیت – هیوریستیک اول شکست
- انتخاب متغیری با بیشترین احتمال ایجاد شکست در مسیر
- متغیری انتخاب می شود که به احتمال زیاد، بزودی با شکست مواجه شده و درخت جست و جو را هرس می کند
- اگر در جریان کار، متغیر X بدون مقدار مجاز تشخیص داده شد،شکست تشخیص داده شده است

مثال : هیوریستیک حداقل مقادیر باقیمانده (MRV)



۲-هیوریستیک درجه



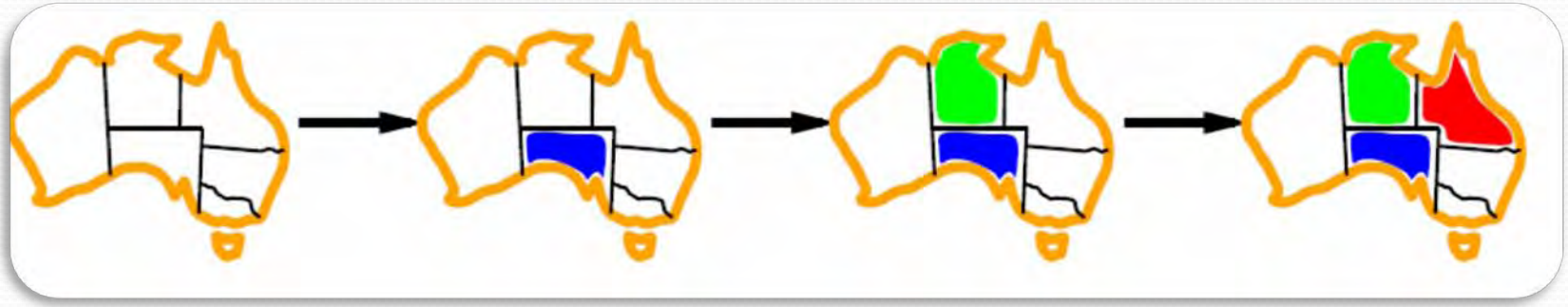
هیوریستیک MRV هیچ کمکی در انتخاب اولین ناحیه برای رنگ آمیزی نقشه استرالیا نمی کند.زیرا تمامی نواحی در مرحله اول دارای سه رنگ مجاز می باشند.در چنین حالاتی از **هیوریستیک درجه** استفاده می کنیم.

هیوریستیک درجه سعی در کاهش ضریب انتخابهای بعدی خواهد داشت.

هیوریستیک درجه متغیری انتخاب می کند که دارای بالاترین **درجه محدودیت** در مقایسه با سایر متغیرهای انتساب نشده است.

درجه محدودیت عبارت است تعداد محدودیتهای یک گره(تعداد همسایگان در مسله رنگ آمیزی)

مثال : هیوریستیک درجه



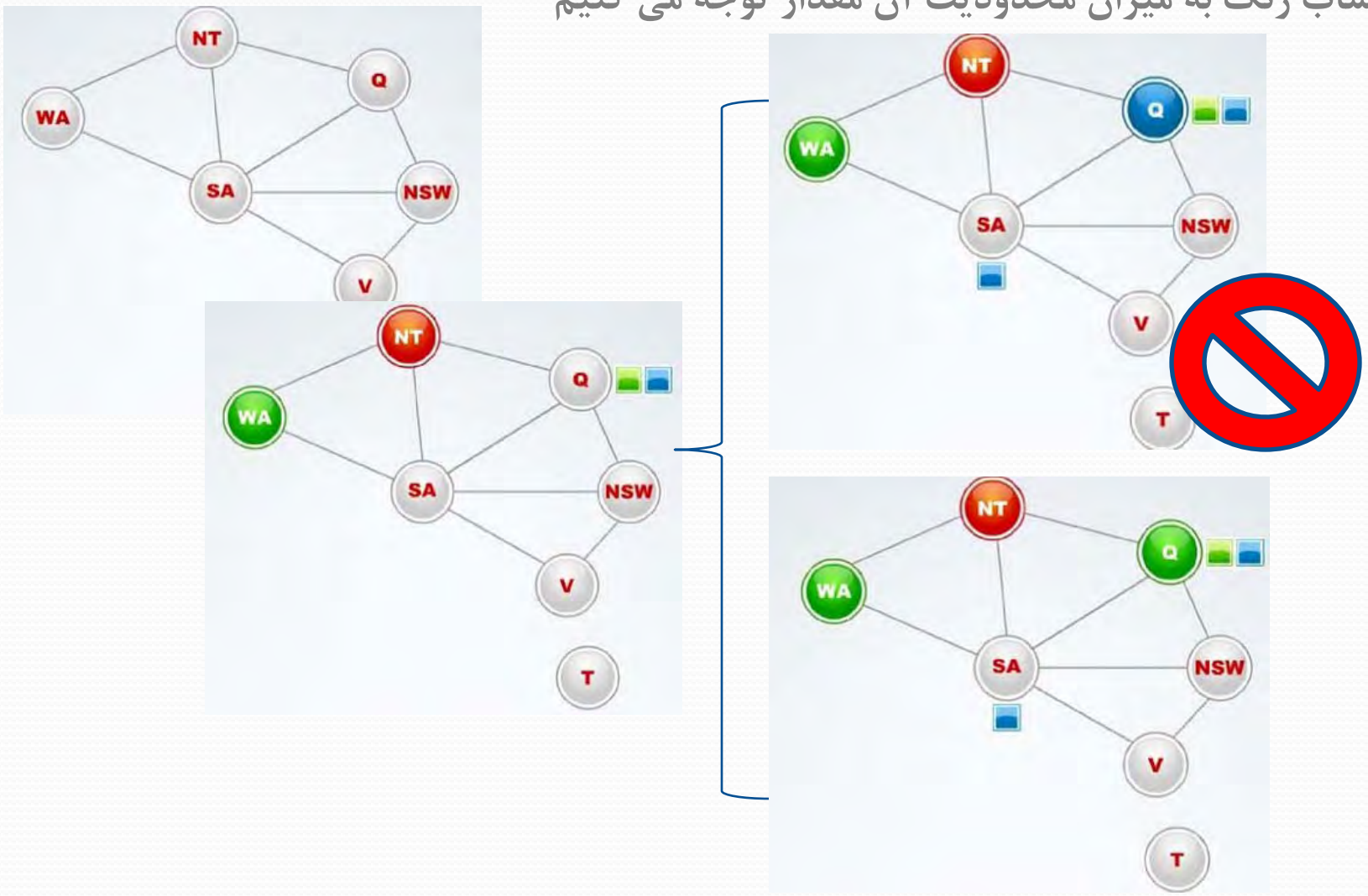
ترتیب مقدار

۱-هیوریستیک مقداری با حداقل محدودیت(همسایه دوست)

- به طور کلی، این هیوریستیک باعث ایجاد قابلیت انعطاف بیشتر درانتساب مقادیر به متغیرهای بعدی خواهد شد
- .پس از برگزیدن یک متغیر، الگوریتم باید نسبت به انتساب مقدار به آن تصمیم گیری نماید.
- در این هیوریستیک مقداری برای یک متغیر انتخاب می شود که در گراف محدودیت، باعث **ایجاد کمترین محدودیت در متغیرهای مجاور شود.**

مثال : هیوریستیک مقداری با حداقل محدودیت

در هنگام انتساب رنگ به میزان محدودیت آن مقدار توجه می کنیم

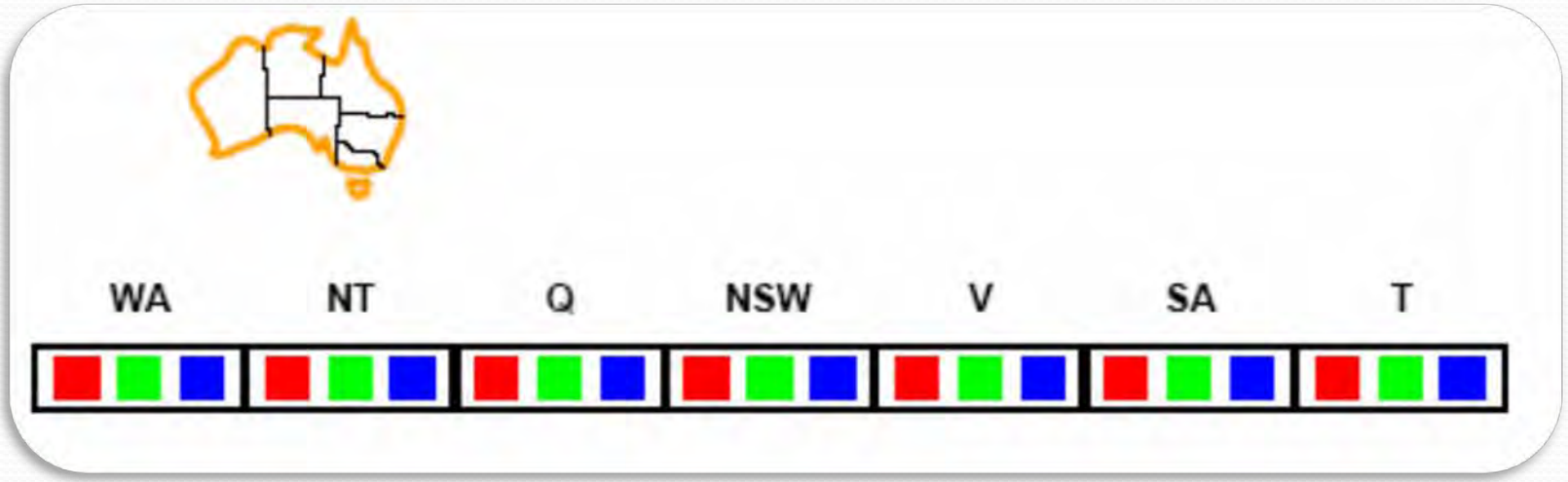


انتشار (پیش بینی)اطلاعات به کمک محدودیتها

- تا اینجا الگوریتم جستجو قادر است هر لحظه فقط محدودیتهای یک متغیر که به وسیلهٔ تابع *SELECT-UNASSIGNED-VARIABLE* انتخاب شده باشد را در نظر بگیرد.

- ولی می‌توان با در نظرگرفتن زودهنگام برخی از محدودیتها در جستجو یا حتی پیش از آغاز جستجو، تا حد زیادی فضای جستجو را کاهش دهیم.

وارسی(بررسی) پیش رو

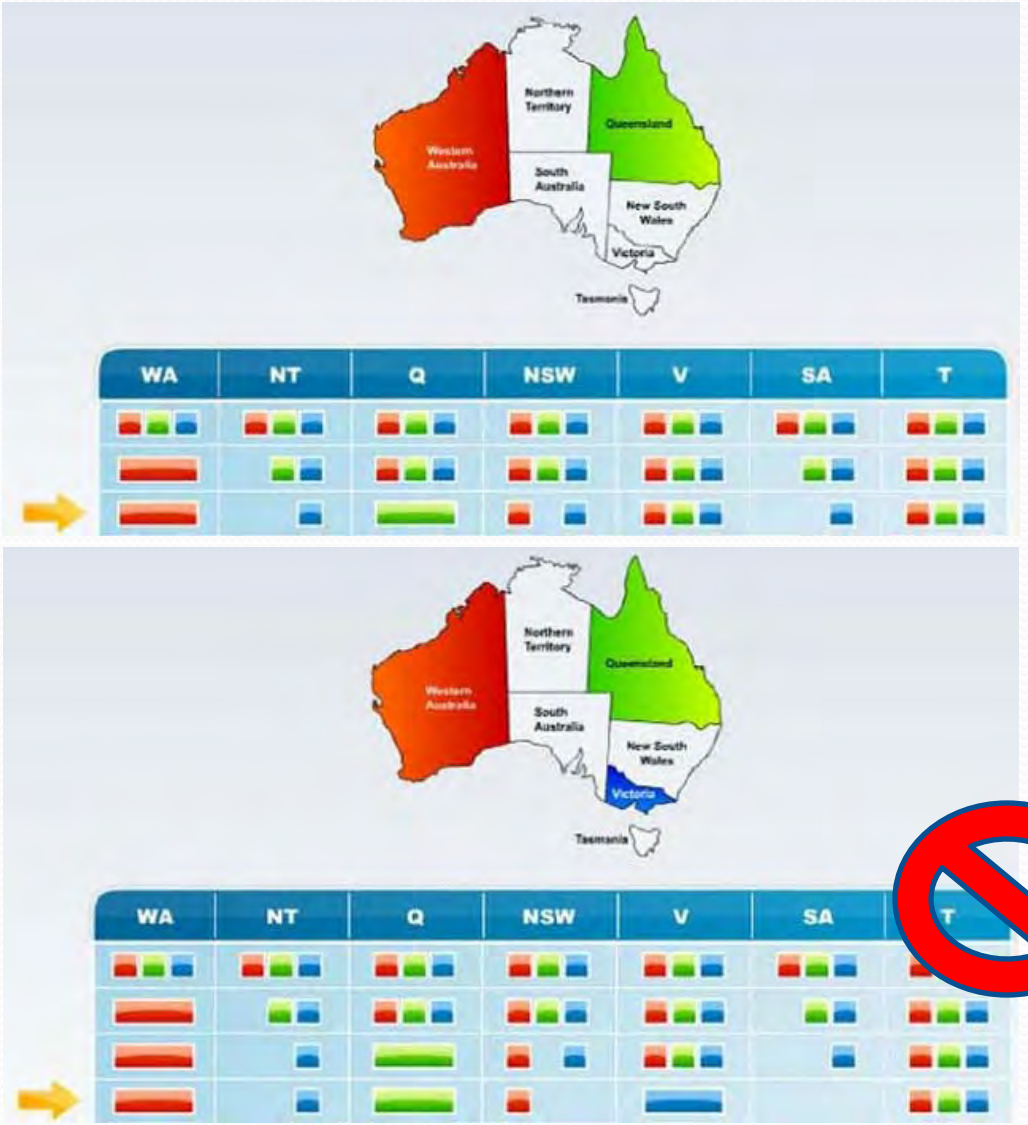


- وارسی پیش رو یکی از راه های بهتر استفاده از محدودیت ها طی عمل جستجو می باشد.

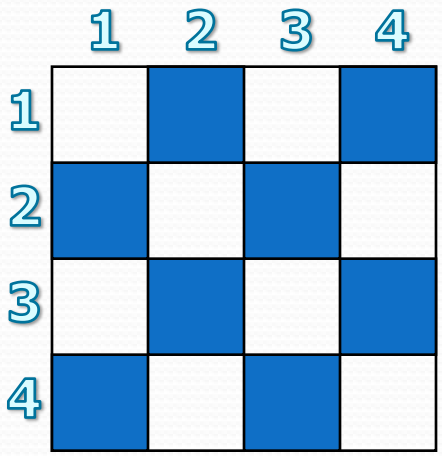
- وقتی انتساب به X صورت می گیرد، فرایند وارسی پیش رو، متغیرهای بدون انتساب مثل Y را در نظر می گیرد که از طریق یک محدودیت به X متصل است و هر مقداری را که با مقدار انتخاب شده برای X برابر است، از دامنه Y حذف می کند

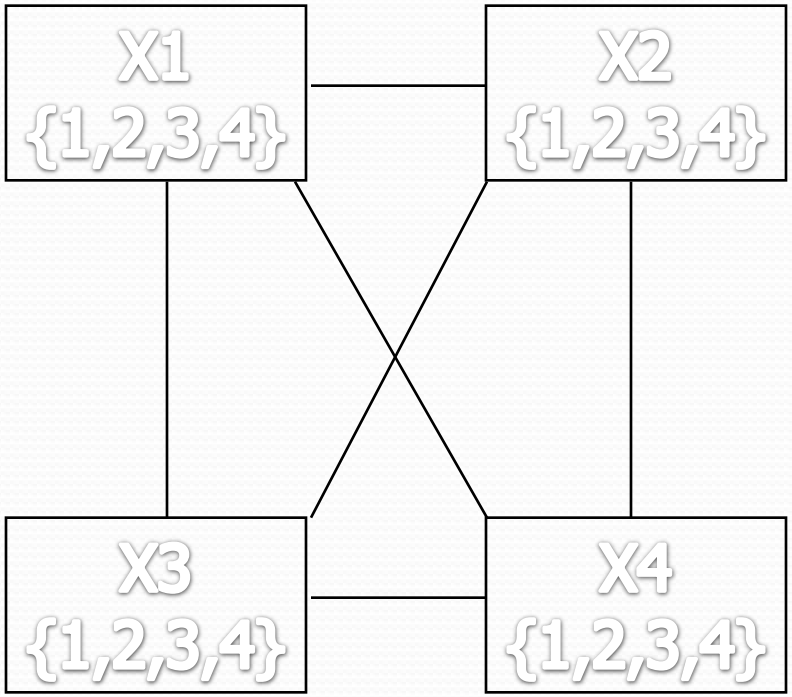
مثال وارسی پیش رو

مثال وارسی پیش رو

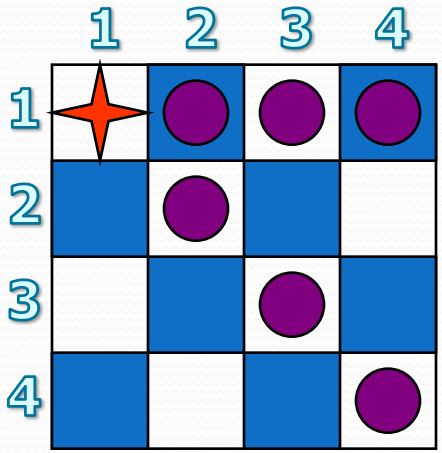


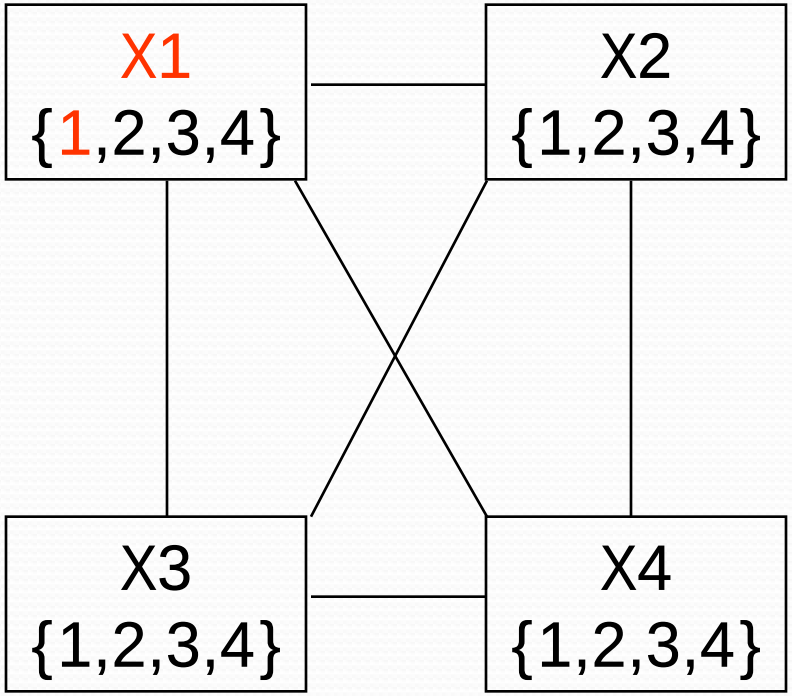
مثال: مسأله ۴-وزیر



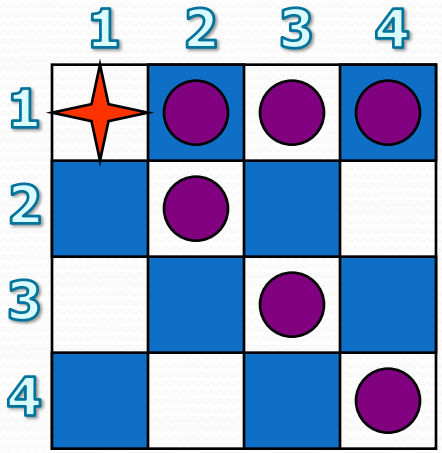


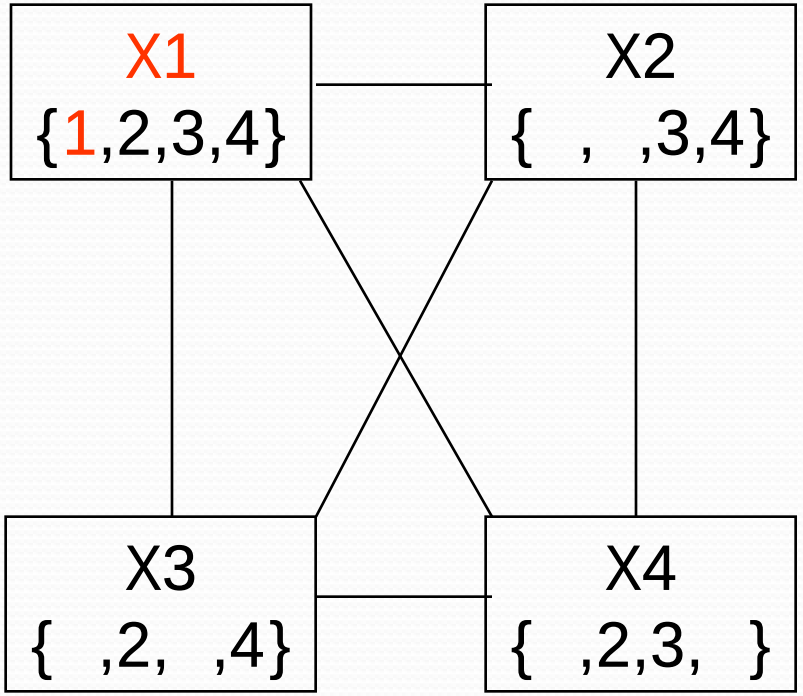
مثال: مسأله ۴-وزیر



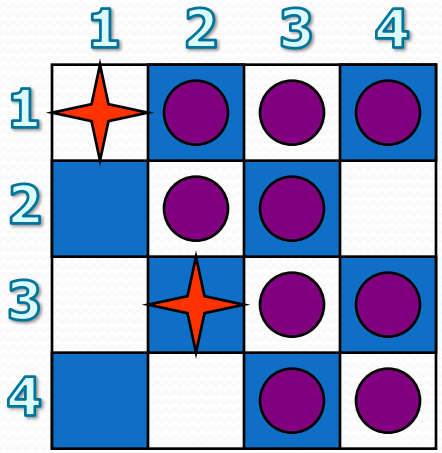


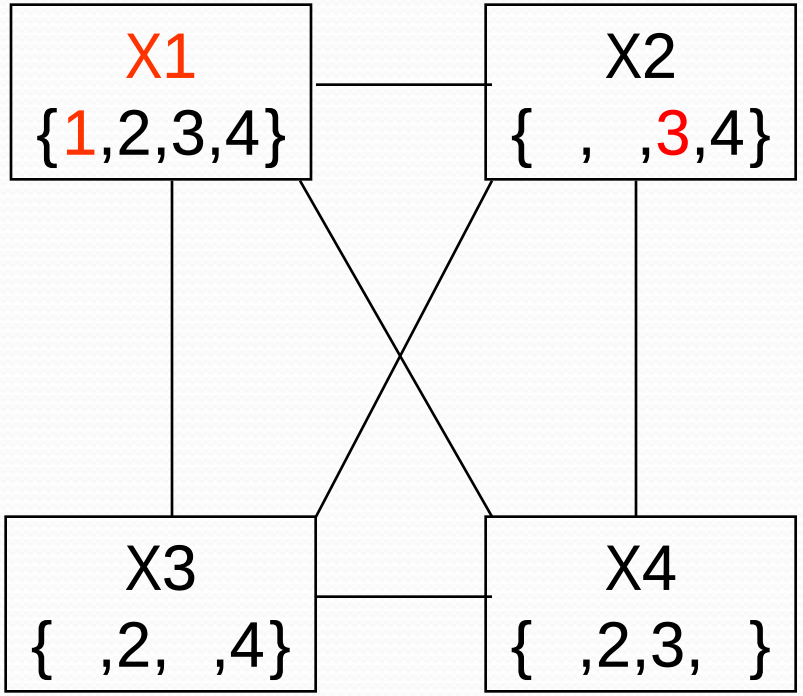
مثال: مسأله ۴-وزیر



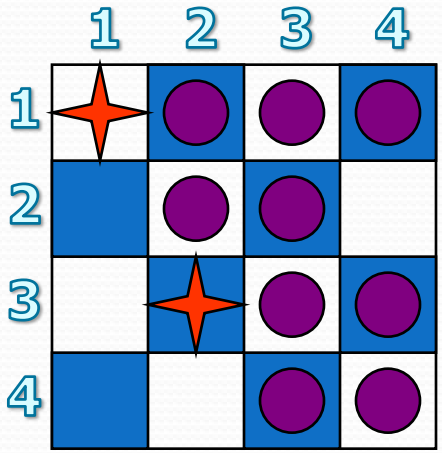


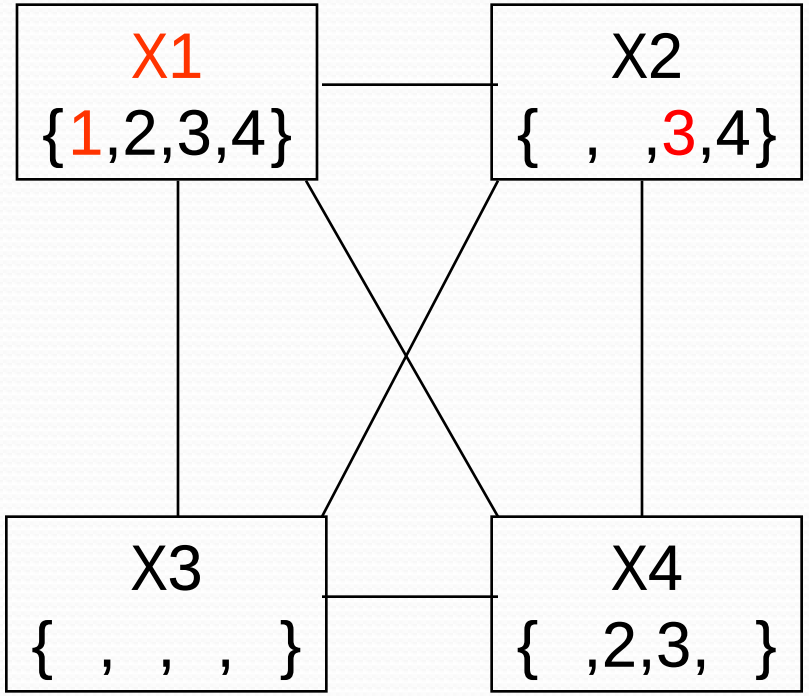
مثال: مسأله ۴-وزیر



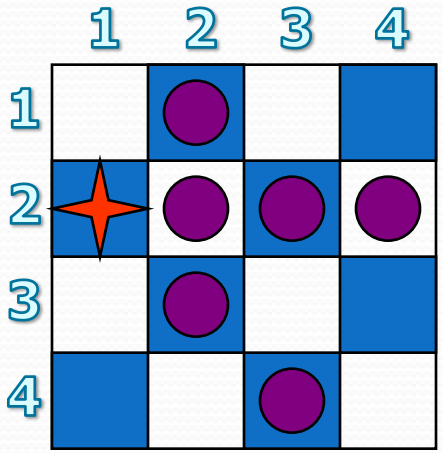


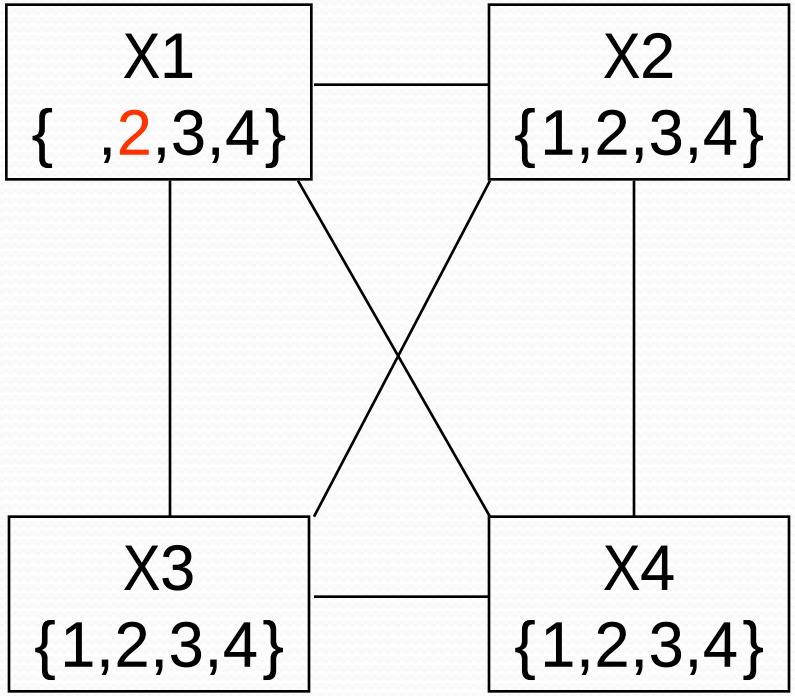
مثال: مسأله ۴-وزیر



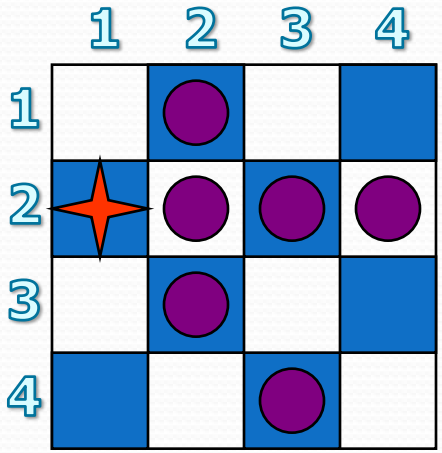


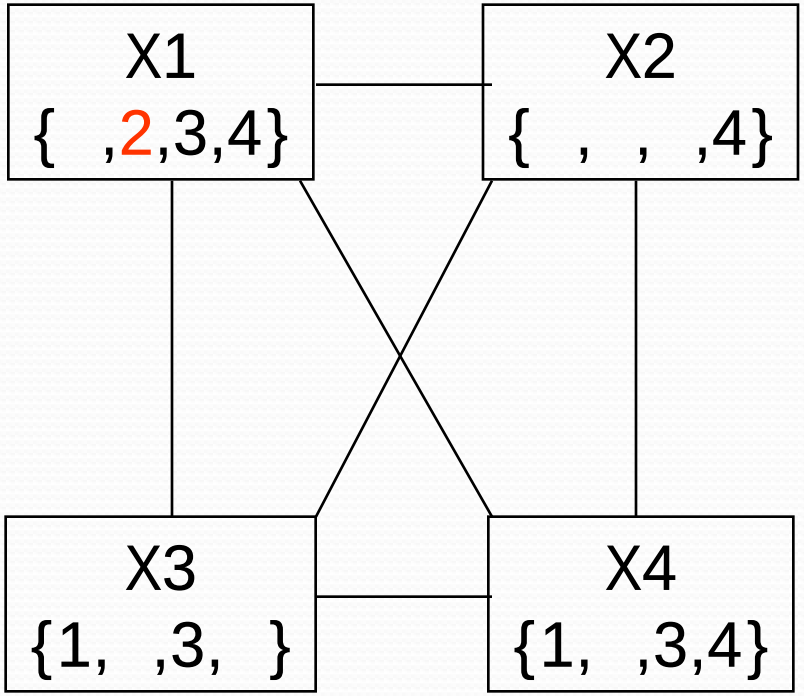
مثال: مسأله ۴-وزیر



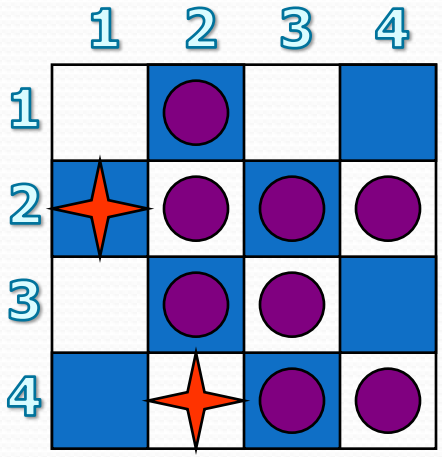


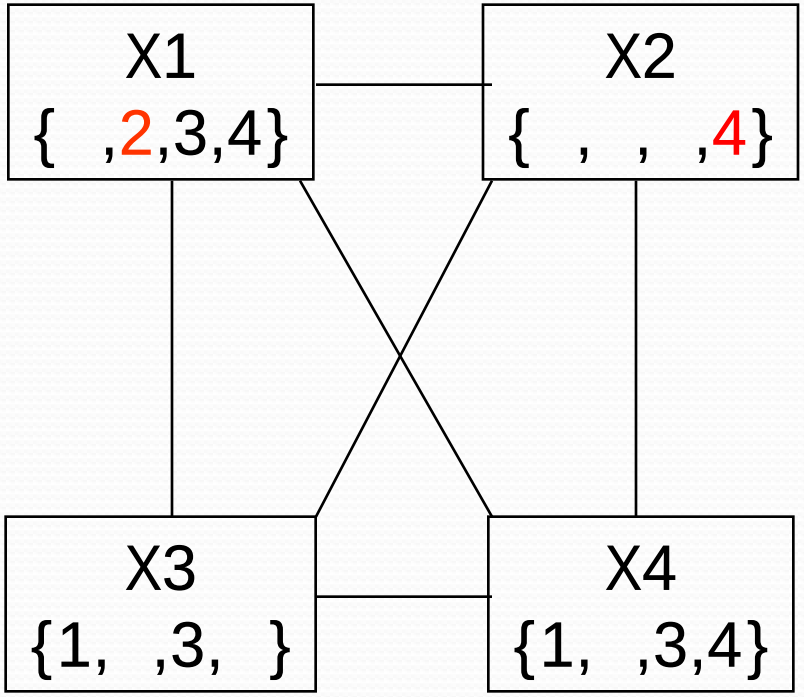
مثال: مسأله ۴-وزیر



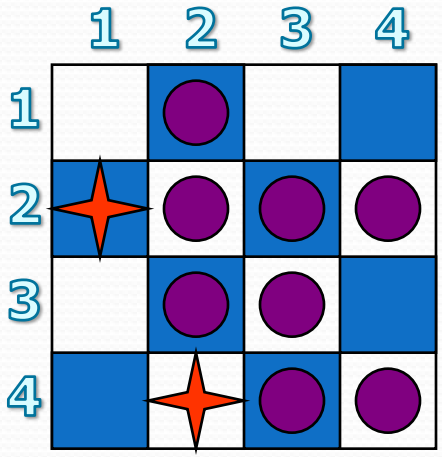


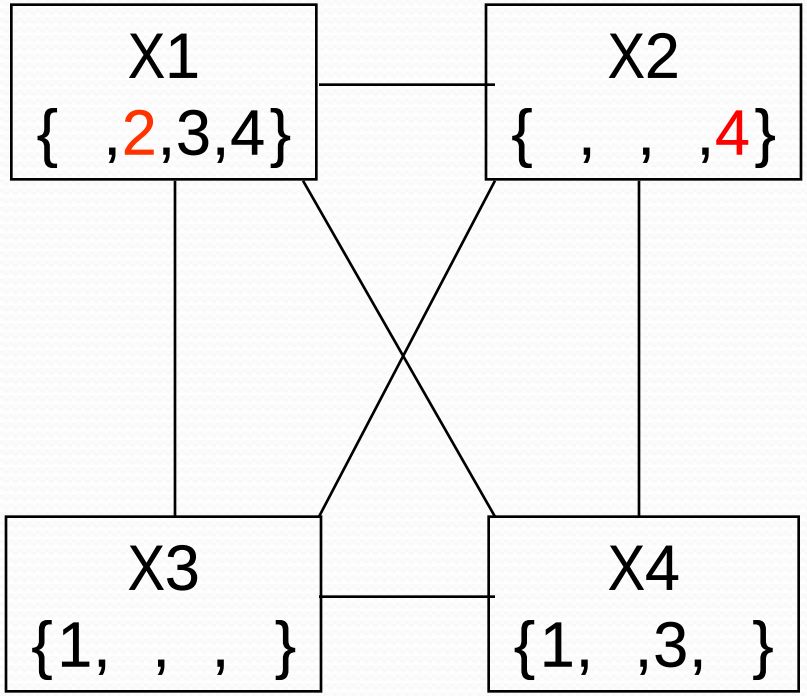
مثال: مسأله ۴-وزیر



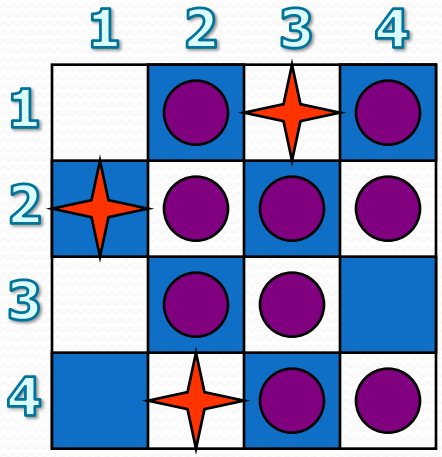


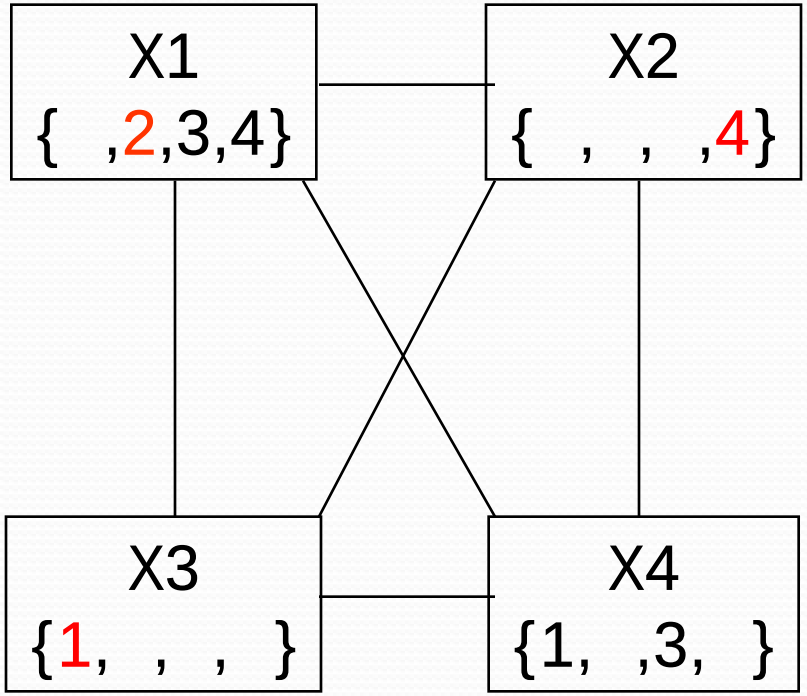
مثال: مسأله ۴-وزیر



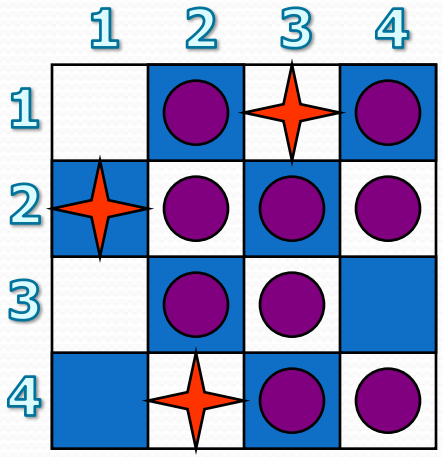


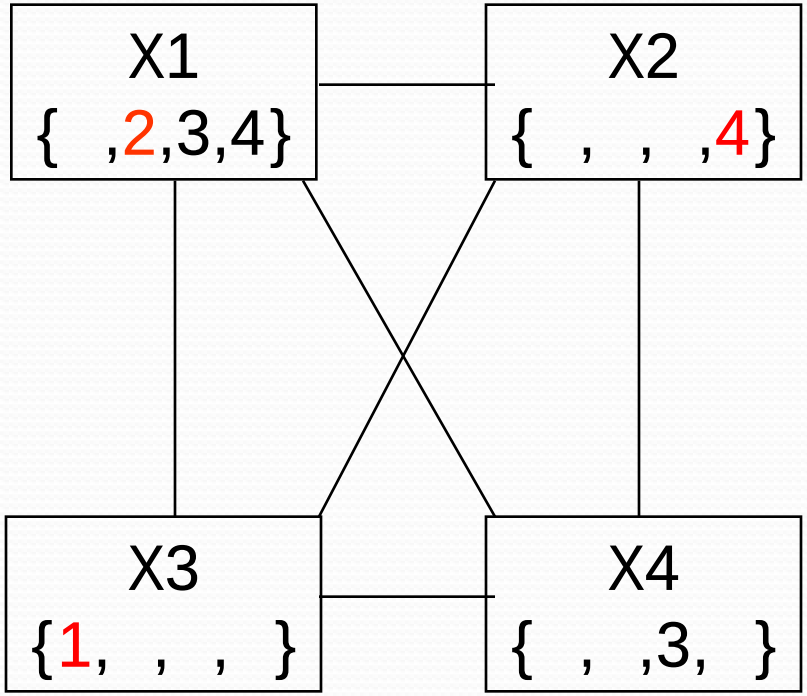
مثال: مسأله ۴-وزیر



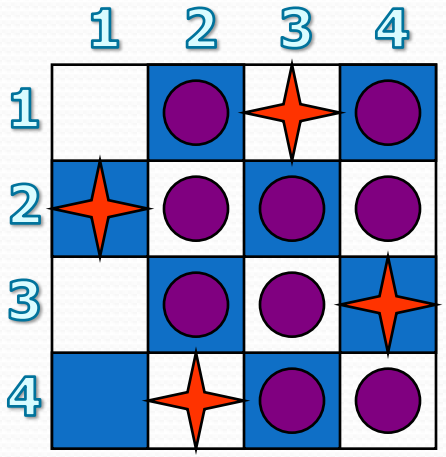


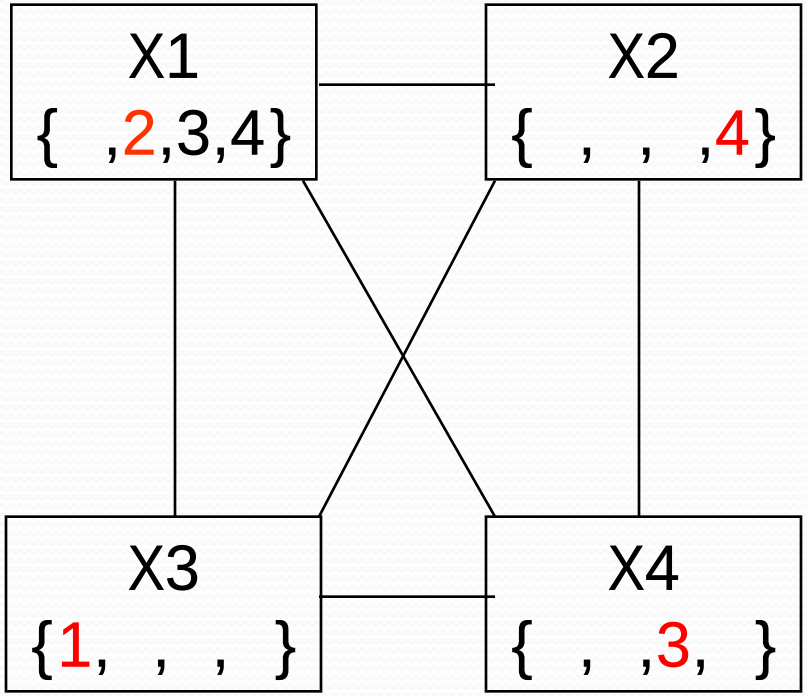
مثال: مسأله ۴-وزیر





مثال: مسأله ۴-وزیر

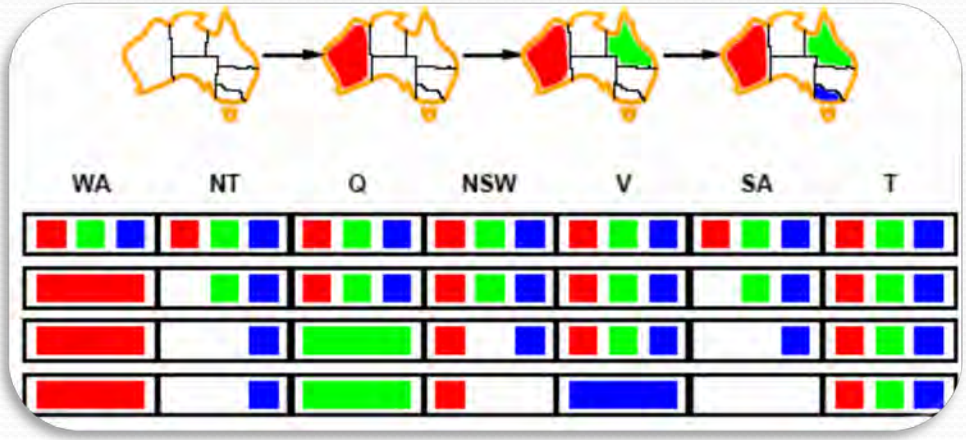




انتشار (پیش بینی)محدودیت

وارسی پیش رو فقط می تواند تعدادی از ناسازگاری ها را تشخیص دهد.

- انتشار محدودیت پیش بینی اعمال محدودیت از یک متغیر به متغیرهای دیگر است
- مثال: در ردیف سوم SA و NT می توانند آبی باشند و هنوز محدودیت لازم پیش بینی نشده و انتخاب آبی مجاز است در حالیکه همسایه اند و این انتخاب ممنوع است



روش سازگاری کمان = برای انتشار محدودیت(≠ انتشار اطلاعات)

ساده ترین شکل انتشار، هر کمان را سازگار می کند.

$X \rightarrow Y$ سازگار است اگر و فقط اگر

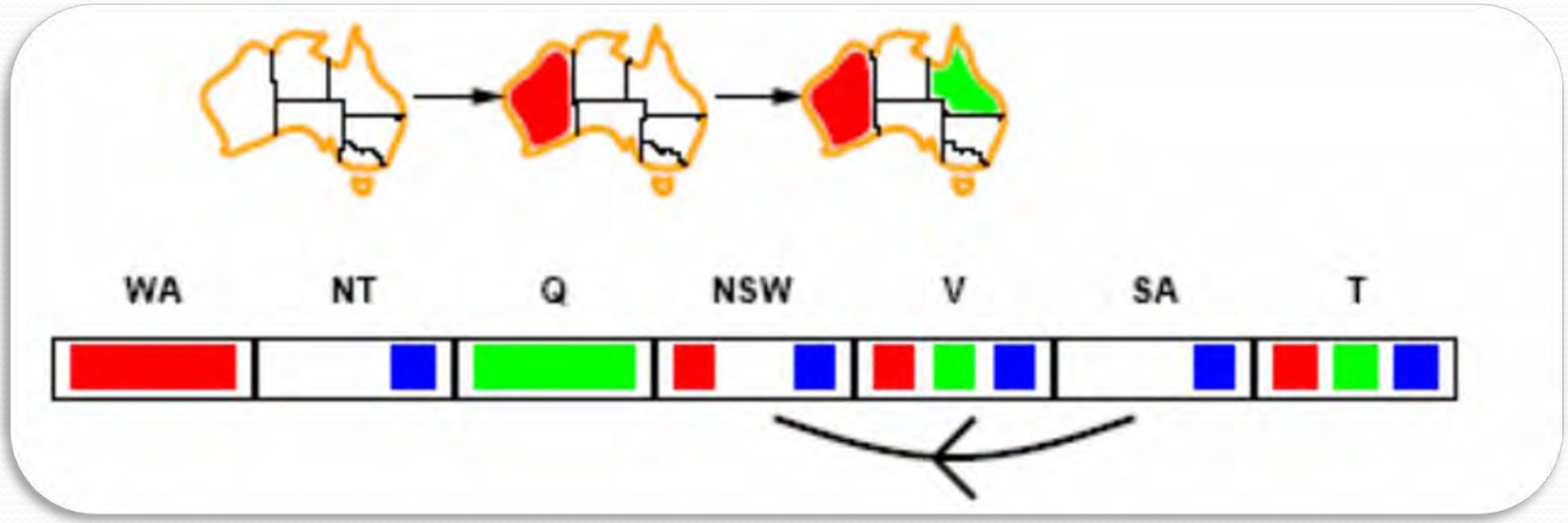
بازاء هر مقدار x برخی مقادیر مجاز y موجود باشند.

- روشی سریع در انتشارمحدودیت و به مراتب قویتر از وارسی پیش رو
- منظور از کمان همان کمان های جهت دار موجود در گراف محدودیت هستند
- زمان وارسی سازگاری کمان
  - به صورت یک مرحله پردازش، پیش از شروع روند جستجو
  - به صورت یک مرحله انتشار محدودیت پس از هر بار انتساب مقادیر در خلال عمل جستجو

به روش دوم روش حفظ سازگاری کمان یا MAC می گویند

در هر دو حالت تکرار تا هیچ ناسازگاری باقی نماند

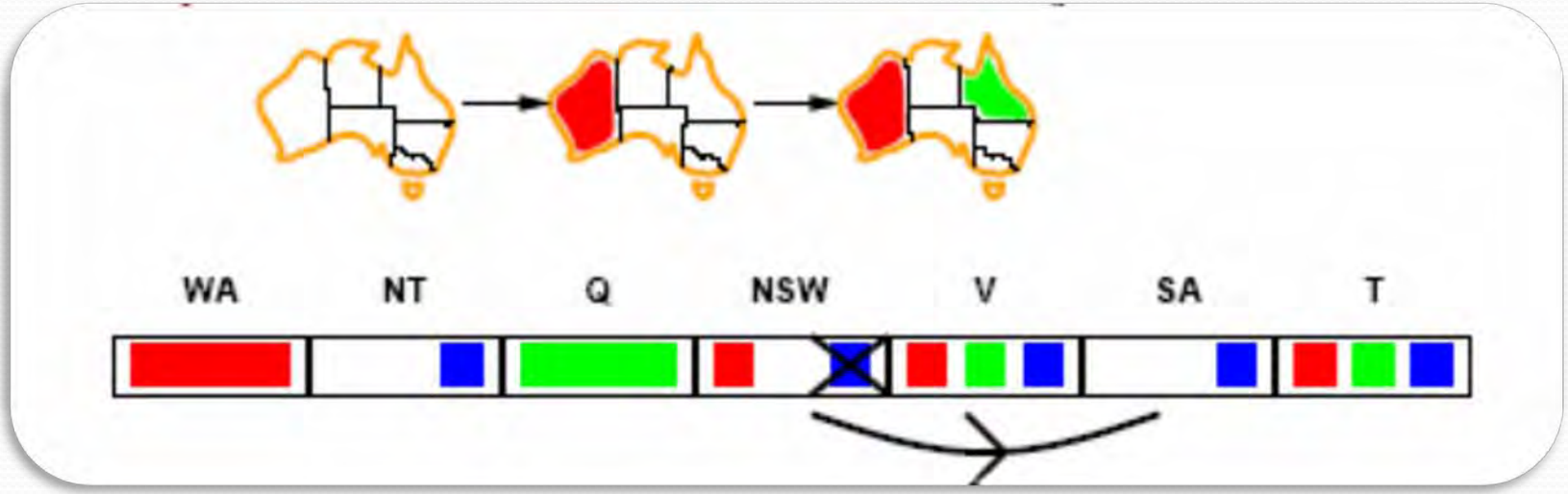
مثال: سازگاری کمان



- $SA \rightarrow NSW$ سازگار است اگر

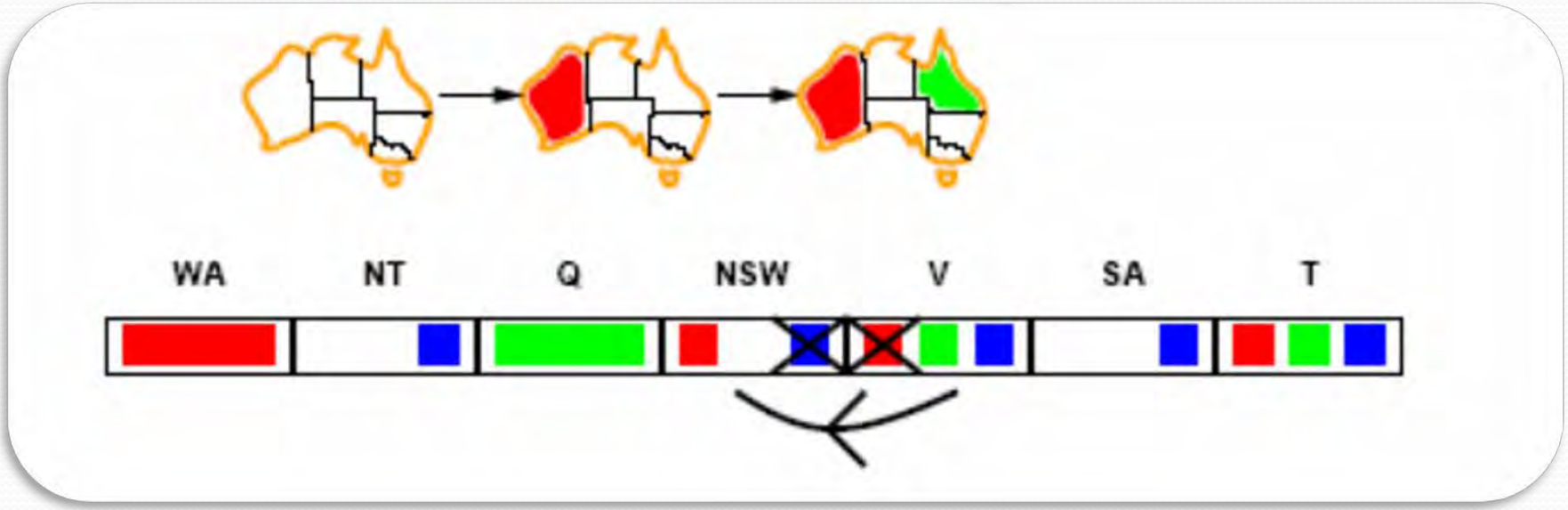
SA=blue and NSW=red

مثال: سازگاری کمان



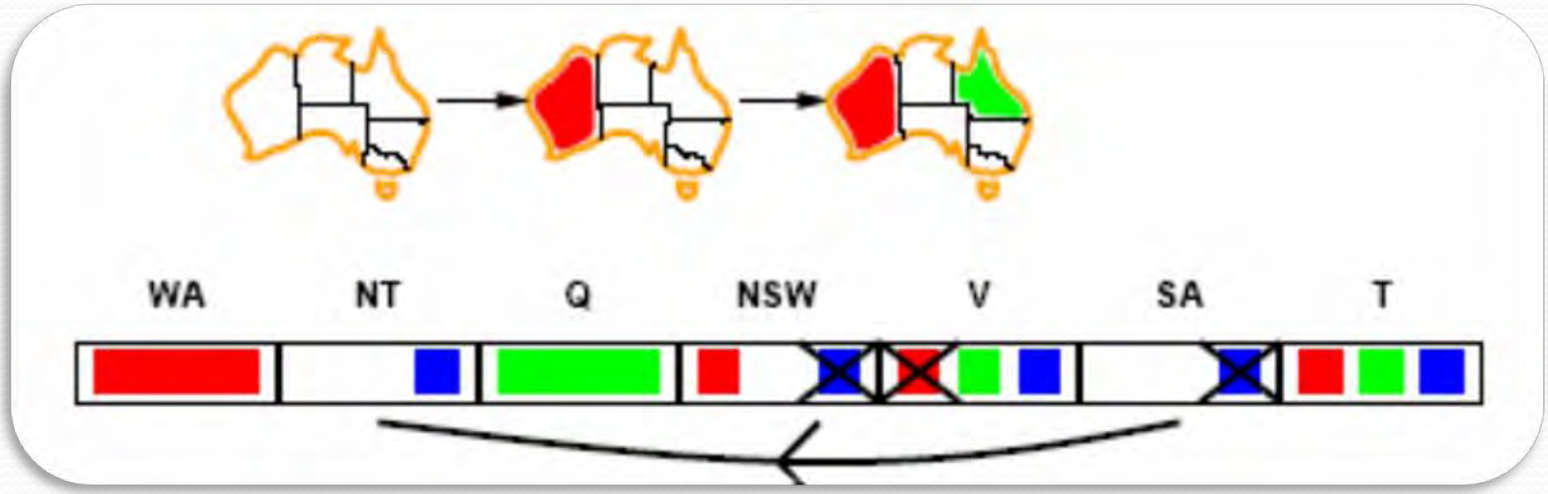
- $NSW \rightarrow SA$ سازگار است اگر
- SA=blue and NSW=red
- NSW=blue and SA=???
- کمان می تواند سازگار شود با حذف blue از NSW

مثال: سازگاری کمان



- کمان میتواند سازگار شود با حذف red از V

مثال: سازگاری کمان



- کمان می تواند سازگار شود با حذف blue از NSW
- حذف red از V
- تکرار تا هیچ ناسازگاری باقی نماند

مثال: سازگاری کمان

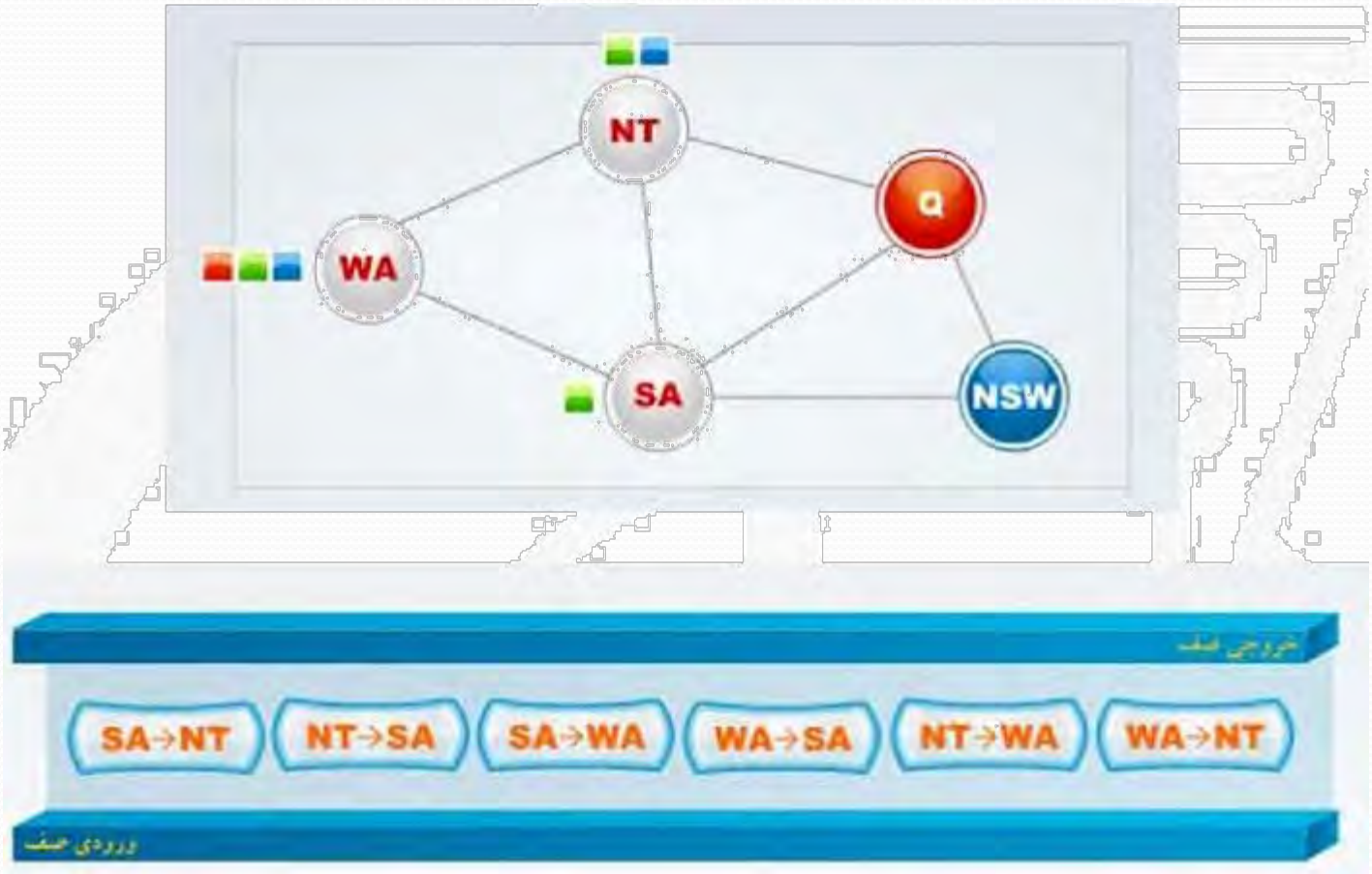
- کمان سازگار: SA→NSW, NSW→V, …

- کمان ناسازگار: NSW→SA, NT→SA, …

روش پیدا کردن ناسازگاریها با الگوریتم AC-3

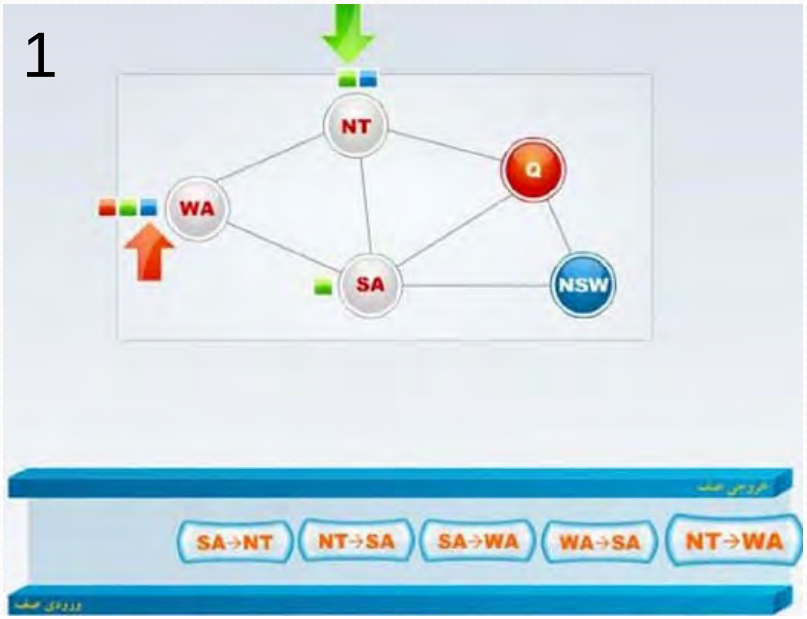
- ابتدا تمام کمان‌ها را در یک صف می‌گذاریم.
- تا صف خالی نشده، هربار اولین عنصر درون صف را بردار (A→B) و چک کن سازگار است یا نه.
- اگر سازگار نبود تمام مقادیر موجود در سر کمان (A) که باعث ناسازگاری می‌شوند را حذف کن.
- تمام کمان‌های منتهی به راس این کمان (A) را به صف اضافه کن تا دوباره بررسی شوند.

مثال : روش پیدا کردن ناسازگاریها



مثال : روش پیدا کردن ناسازگاریها

1



2

3

4

مثال : روش پیدا کردن ناسازگاریها

1

2

3

4

مثال : روش پیدا کردن ناسازگاریها

- هر کمانِ (X_i, X_j) به نوبت از لیست مورد نظر حذف و بررسی می‌شوند.
- اگر لازم باشد یک مقدار از دامنهٔ X_i حذف شود، آنگاه تمام کمانهای (X_k, X_i) که به X_i متصل هستند، باید به منظور بررسی مجدد، به صف اضافه شوند.

- بررسی میزان پیچیدگی الگوریتم سازگاری کمان به شرح زیر است:

  - یک CSP دوگانه، حداکثر دارای $O(n^2)$ کمان می‌باشد.
  - از آنجا که دامنهٔ X_i حداکثر دارای d مقدار قابل حذف است، هر کمان (X_k, X_i)، می‌تواند d مرتبه وارد لیست شود.
  - پس بررسی ناسازگاری در یک کمان، حداکثر $O(d^2)$ مرتبه می‌تواند انجام شود.
  - بنابراین، تعداد بررسیها در بدترین شرایط برابر با $O(n^2d^3)$ خواهد بود.

سازگاری مرتبه K

- سازگاری کمان قادر به مشخص کردن تمام ناسازگاری های ممکن نیست
- با روش سازگاریK، شکل های قویتری از انتشاررا می توان تعریف کرد
- یک CSP دارای سازگاری مرتبه K است، اگر برای هر مجموعه k-1 عضوی از متغیرها و برای هر انتساب سازگار به آنها، همیشه یک مقدار سازگار یافت شود ، که بتوان به هر متغیر kام انتساب داد.

سازگاری مرتبه K

- بطور مثال:
  - سازگاری۱: هر متغیر با خودش سازگار است(سازگاری گره)
  - سازگاری۲: مشابه سازگاری کمان
  - سازگاریk: بسط هر جفت از متغیرهای همجوار به سومین متغیر همسایه(سازگاری مسیر) (یک متغیر به ازای ۲ متغیر دیگر مقدارهای بدون ناسازگار داشته باشد)
- **گراف در صورتی سازگار شدید K است که:**
  - سازگارk باشد
  - همچنین سازگارk-1 و سازگارk-2 و... سازگار ۱ باشد
- در این صورت، مسأله را بدون پس گرد میتوان حل کرد
- پیچیدگی زمانی آن O(nd) است

سازگاری مرتبه K

- در ابتدا، یک مقدار سازگار را برای X_1 انتخاب می‌کنیم.
- از آنجا که گراف دارای سازگاری مرتبهٔ ۲ می‌باشد، مطمئن هستیم که می‌توان مقداری سازگار یافت که بتوان به X_2 اختصاص داد و به همین ترتیب، می‌توان مقادیری سازگار برای X_3، ...و X_n نیز یافت.
- برای هر متغیر X_i، به منظور یافتن مقداری سازگار با X_1، ...، X_{i-1} تنها نیاز به جستجو در d مقدار دامنه می‌باشد.
- تضمین می‌شود که راه‌حل مسأله حداکثر با مرتبهٔ زمانی $O(nd)$ پیدا می‌شود.

برخورد با محدودیتهای ویژه

- در مسائل حقیقی، ممکن است انواع خاصی از محدودیتها ایجاد شوند که حل آنها دیگر با استفاده از الگوریتمهای عمومی که تا به حال در مورد آنها بحث شد، ممکن نباشد. برخورد با آنها نیاز به روشهای ویژه‌ای دارد.

- **مثال:** در محدودیت Alldiff، تمامی متغیرها باید دارای مقادیر متفاوتی باشند.

- یک شکل ساده از ایجاد ناسازگاری برای این محدودیت بدین صورت است: اگر mمتغیر شامل این محدودیت باشند ولی تنها n مقدار در مجموع برای انتساب موجود باشد، دیگر این محدودیت نمی‌تواند برآورده شود. m>n

راه حل برخورد با محدودیتهای ویژه

- راه حل مسئله:
  - در ابتدا تمام متغیرهایی که دارای دامنهٔ واحد می‌باشند را حذف و مقدار مربوط به دامنهٔ آنها را نیز از دامنهٔ سایر متغیرها حذف می‌کنیم.
  - این کار را برای تمام متغیرهای تک‌مقداره تکرار می‌نماییم.
  - اگر در پایان دامنهٔ یک متغیر تهی شود یا تعداد متغیر بیشتری از تعداد مقادیر دامنه، باقی مانده باشد، ناسازگاری در مسأله تشخیص داده خواهد شد.

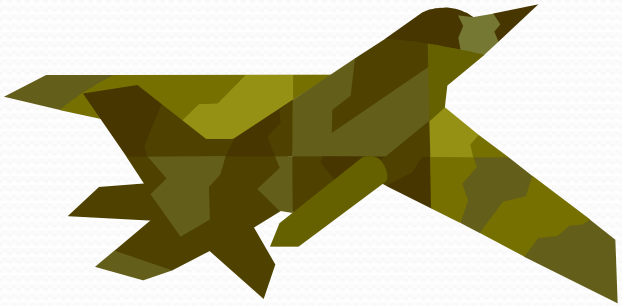
محدودیت منبع = محدودیت ماکزیمم

- مهمترین محدودیت از درجهٔ بالا، **محدودیت منبع (محدودیت بیشینه)** می‌باشد.
- **مثال:** در نظر بگیرید $PA_1,\ldots, PA_4$ به تعداد افرادی اشاره می‌کند که برای انجام چهار کار متفاوت در نظر گرفته شده‌اند.
- محدودیتی که براساس آن، تعداد افراد در مجموع نباید از ۱۰ نفر تجاوز نماید به صورت $atmost\ (10,PA_1,PA_2,PA_3,PA_4)$ نمایش داده می‌شود.
- می‌توان با بررسی مجموع حداقل مقادیر دامنه‌های فعلی، ناسازگاری را تشخیص داد.

محدودیت منبع = محدودیت ماکزیمم

- مثلاً اگر هر متغیر دارای دامنهٔ {۳،۴،۵،۶} باشد، محدودیت بیشینه، تأمین نمی‌شود.
- می‌توان برای اعمال سازگاری، مقدار بیشینه در هر دامنه را، به شرطی که با مقدار کمینه در دامنه‌های دیگر ناسازگار باشد، حذف نمود.
- اگر هر متغیری در مثال فوق، دارای دامنهٔ {۲،۳،۴،۵،۶} باشد، می‌توان به منظور رسیدن به سازگاری، مقادیر ۵ و ۶ را از دامنهٔ متغیرها حذف نمود.

مثال پرواز (محدودیت منبع = محدودیت ماکزیمم)



Flight271 ε [0,165]

Flight272 ε [0,385]

محدودیت بیشینه : مجموع مسافرین هر دو پرواز باید ۴۲۰ باشد

Atmost(420,flight271,flight272)

Flight271+ Flight272 ε [420,420]

Flight271 ε [35,165]

Flight272 ε [255,385]

پسگرد هوشمندانه (نگاه رو به عقب)

جستجوی پس گرد

- در الگوریتم *BACKTRACKING-SEARCH*، هنگامی که یکی از شاخه‌ها با شکست در جستجو مواجه می‌شود، الگوریتم به متغیر قبلی باز می‌گردد و مقداری جدید برای آن در نظر می‌گیرد.

- این روش را **پس‌گرد زمانی** می‌نامیم، چون در مقدار آخرین متغیری که مقداردهی شده است تجدید نظر می‌کند.

- یک روش پس‌گرد هوشمندانه‌تر آن است که تمام مسیر را تا رسیدن به مجموعه‌ای از متغیرها که باعث شکست شده‌اند(**مجموعهٔ تناقض**)، به عقب باز گردیم.

پسگرد هوشمندانه (نگاه رو به عقب)

- مجموعهٔ تناقض برای متغیر X عبارت است از مجموعه‌ای از متغیرهایی که قبلاً مقداردهی شده‌اند و به واسطهٔ یک محدودیت با X در ارتباطند.

- این روش (روش **پرش رو به عقب**)، مسیر پیموده‌شده را تا رسیدن به آخرین متغیری که در مجموعهٔ تناقض مقداردهی شده است، به عقب می‌پیماید.

پسگرد هوشمندانه (نگاه رو به عقب)

- وارسی پیش‌رو می‌تواند مجموعهٔ تناقض را بدون انجام کار اضافه تشکیل دهد.
- هنگامی که وارسی پیش‌رو به واسطهٔ انتساب مقدار به X، یکی از مقادیر مربوط به دامنهٔ متغیر Y را حذف کند، باید X به مجموعهٔ تناقض Y اضافه شود.
- هنگامی که آخرین مقدار دامنهٔ Y حذف شد، متغیرهای مجموعهٔ تناقض Y به مجموعهٔ تناقض X اضافه شود.
- هنگام برخورد با Y، در صورت نیاز، سریعاً تشخیص خواهیم داد که باید به کجا بازگشت.

پسگرد هوشمندانه (نگاه رو به عقب)

- *پرش رو به عقب* هنگامی رخ می‌دهد که هر مقدار متعلق به دامنه با انتساب انجام شده دارای تناقض باشد.
- وارسی پیش‌رو این حالت را تشخیص می‌دهد و از رسیدن به آن جلوگیری می‌کند.
- هر شاخه‌ای که به وسیلهٔ پرش رو به عقب هرس می‌شود، می‌تواند به وسیلهٔ انجام وارسی پیش‌رو نیز هرس شود.
- پرش رو به عقب ساده، در جستجوهایی که از روشهای قویتر بررسی سازگاری نظیر MAC استفاده می‌کنند زائد است.

MAC سازگاری با کمان است و نیازی به پسگرد هوشمندانه ندارد

پسگرد هوشمندانه (نگاه رو به عقب)

- روش محاسبهٔ مجموعه‌های تناقض: شکست نهایی در یک مسیر از الگوریتم جستجو هنگامی رخ می‌دهد که دامنهٔ یک متغیر، تهی تشخیص داده شود. آن متغیر دارای یک مجموعهٔ تناقض استاندارد می‌باشد.

- در نظر بگیرید که X_j متغیر فعلی و $conf(X_j)$ مجموعهٔ تناقض آن باشد.

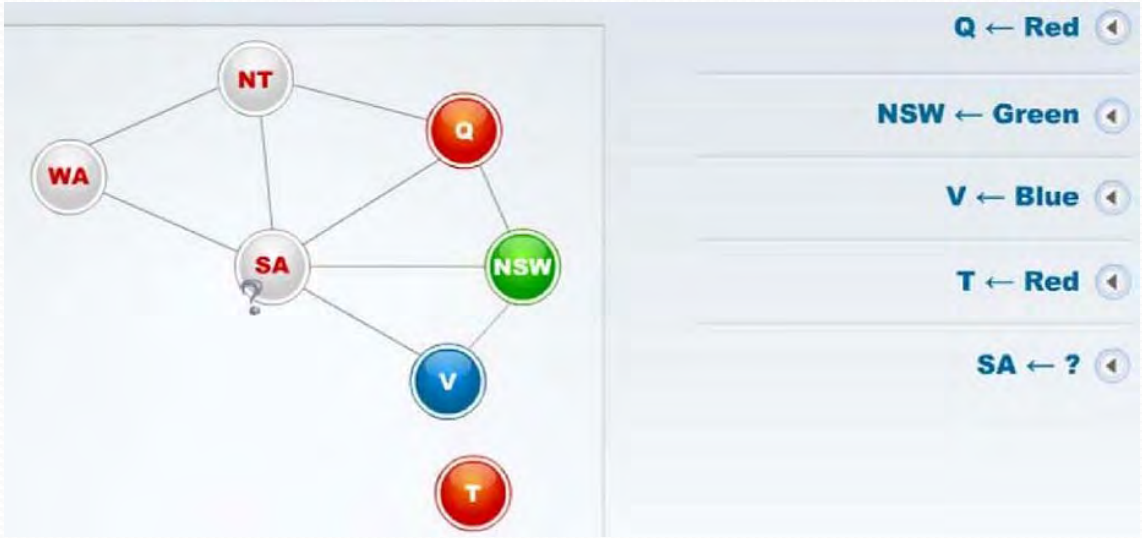
- اگر تمام مقادیر ممکن برای X_j با شکست مواجه شوند، الگوریتم به X_i برخواهد گشت که آخرین متغیر مقدار داده شده و متعلق به مجموعهٔ تناقض X_j می‌باشد و انتساب زیر را انجام می‌دهیم:

$$\text{conf}\ (X_i) \leftarrow conf\ (X_i) \cup conf\ (X_j) - \{X_i\}$$

پسگرد هوشمندانه (نگاه رو به عقب)

- الگوریتم پرش رو به عقبِ هم جهت با تناقض، اگرچه در بازگشت به عقب در درخت جستجو، مسیر درست را نشان می‌دهد، ولی عدم تکرار آن اشتباه در مسیر دیگری از درخت جستجو را تضمین نمی‌کند.

- **یادگیری محدودیت**، به واسطهٔ افزودن محدودیتهای جدیدی که به وسیلهٔ این تناقضات ایجاد می‌شوند، باعث اصلاح CSP خواهد شد.



مجموعه راس‌های مشکل‌دار (Conflict Set)

- Q, NSW, V

بازگشت به عقب تا جایی که آخرین راس در Conflict Set مقدار گرفته (Back Jumping).

یعنی بازگشت به اولین گزینه که محدودیت را روی این گره شروع کرده است

جست و جوی محلی در مسائل ارضای محدودیت

- بسیاری از CSPها را بطور کارآمد حل می کنند

الگوریتم کمترین تناقض یا ناسازگاری (Min Conflict)

- یک وضعیت اولیه‌ی تصادفی درست کنید.
- تا زمانی‌که به جواب نرسیده‌اید یا برای تعداد محدودی تکرار.
- اگر وضعیت فعلی سازگار است، به جواب رسیده‌ایم.
- یک متغیر را به صورت تصادفی انتخاب کنید و مقدارش را به چیزی تغییر دهید که کمترین تعداد ناسازگاری را ایجاد کند.

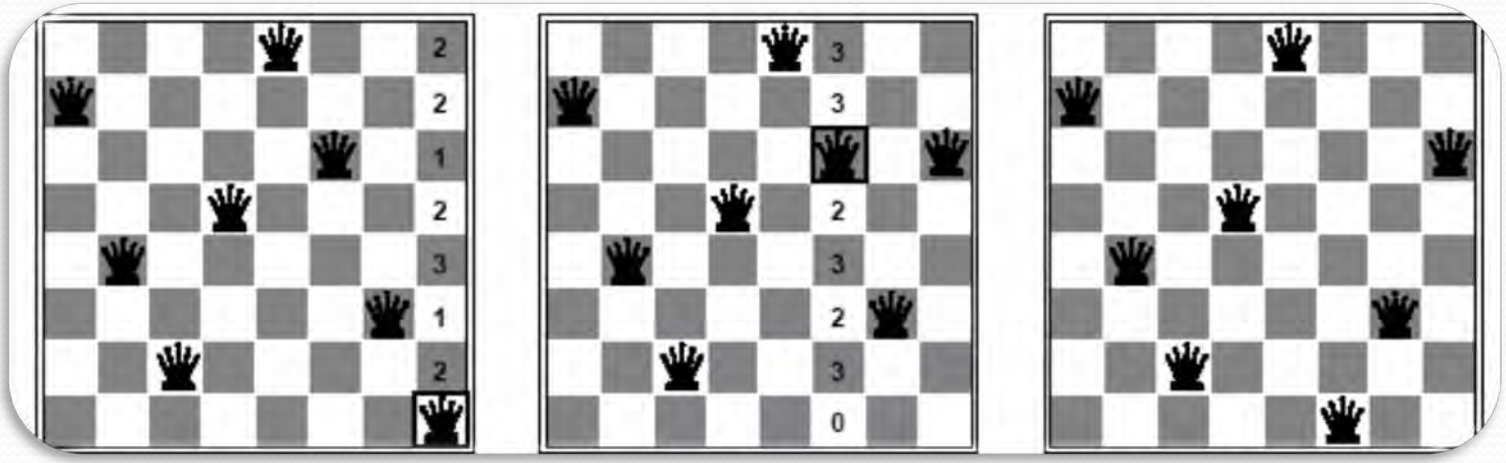
- جست و جوی محلی میتواند در صورت تغییر مسأله، تنظیمات Online را انجام دهد

مثال : جست و جوی محلی در مسائل ارضای محدودیت

- **مثال:** در مسألهٔ ۸ وزیر:
  - **حالت اولیه** قرار دادن ۸ وزیر در ۸ ستون جدول به صورت تصادفی
  - **تابع پسین** یکی از وزیرها را انتخاب می‌نماید و موقعیتش را در ستون مربوطه تغییر می‌دهد.
- امکان دیگر آن است که با هر هشت وزیر شروع نماییم و با جایگشت ردیفها، هر یک را در یک ستون قرار دهیم و پسین آنها با عوض کردن موقعیت دو وزیر با هم ایجاد شود.

مثال

راه حل دو مرحله ای برای مسأله ۸وزیر با استفاده از کمترین برخورد

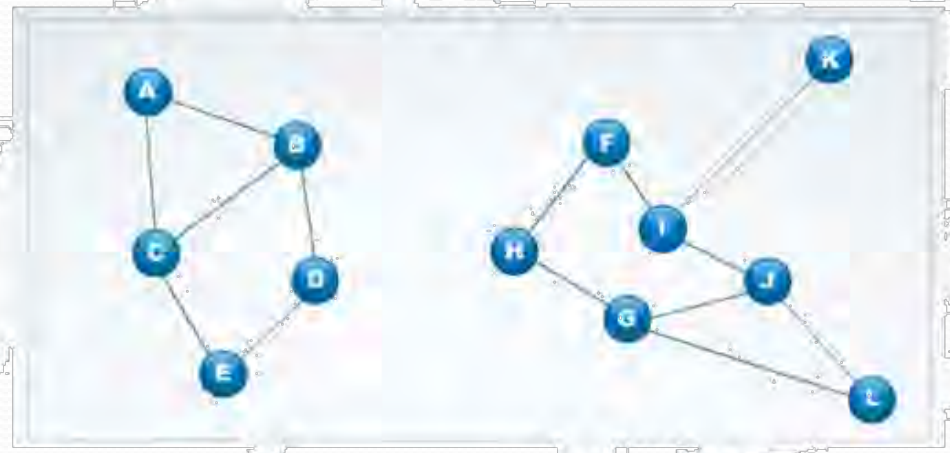


1 2 3

- در هر مرحله، یک وزیر برای انتساب مجدد در ستون خودش انتخاب میگردد
- تعداد برخوردها در هر مربع نشان داده شده است
- الگوریتم وزیر را به مربعی با کمترین برخورد انتقال میدهد، بطوریکه گره ها را بطور تصادفی میشکند

ساختار مسائل

- وجود زیر مسئله (زیر گراف)
  - هر زیر مسئله به صورت مستقل و مجزا حل شود.



n=تعداد گره ها

c=تعدادگره هر زیر مسئله

n/c=تعداد زیر مسائل

هزینه در بدترین حالت $O(n/c \cdot d^c)$=

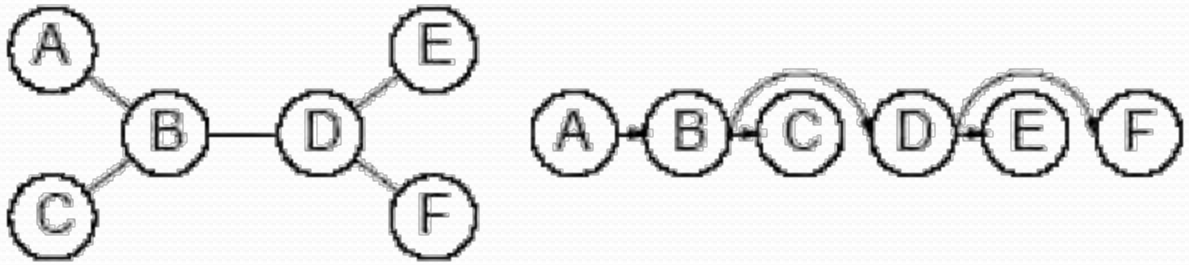
  - به جای ضرب شدن دو فضای حالت جمع می‌شوند.
- گراف حالت، درخت باشد.
  - الگوریتم

Tasmania و جزیره اصلی **زیرمسایل مستقل** هستند.

و با توجه به **مولفه های همبند** در گراف محدودیت قابل شناسایی هستند.

الگوریتم برای CSP های دارای ساختار درختی

۱. یک متغیر را به عنوان ریشه انتخاب کن و سپس متغیرها را از ریشه تا برگها به گونه ای مرتب کن که والد هر گره قبل از آن گره قرار بگیرد.



۲. بازاء j از n تا 2 عمل زیر را انجام بده

روش سازگاری کمان را به کمان xi-xj اعمال می کنیم که در آن xi والد xj است و در صورت نیاز متغیرهایی از دامنه xi را حذف می کنیم

REMOVEINCONSISTENT($Parent(X_j)$, X_j)

۳. برای j از یک تا n ، مقدار X_j را به طور سازگار با $Parent(X_j)$ تعیین کن

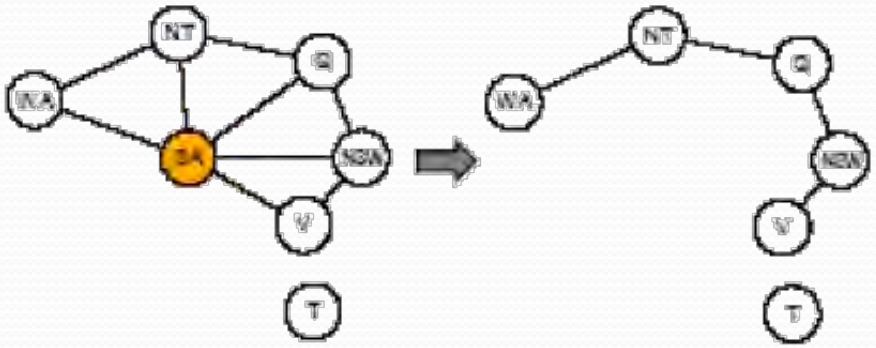
xi

پیچیدگی زمانی: $O(n \cdot d^2)$

CSP های دارای ساختار شبه درختی

روش اول مقید سازی کات ست(برشی) برای ساخت درخت

- شرطی سازی: به یک متغیر مقدار بده؛ دامنه همسایه های آن را هرس کن

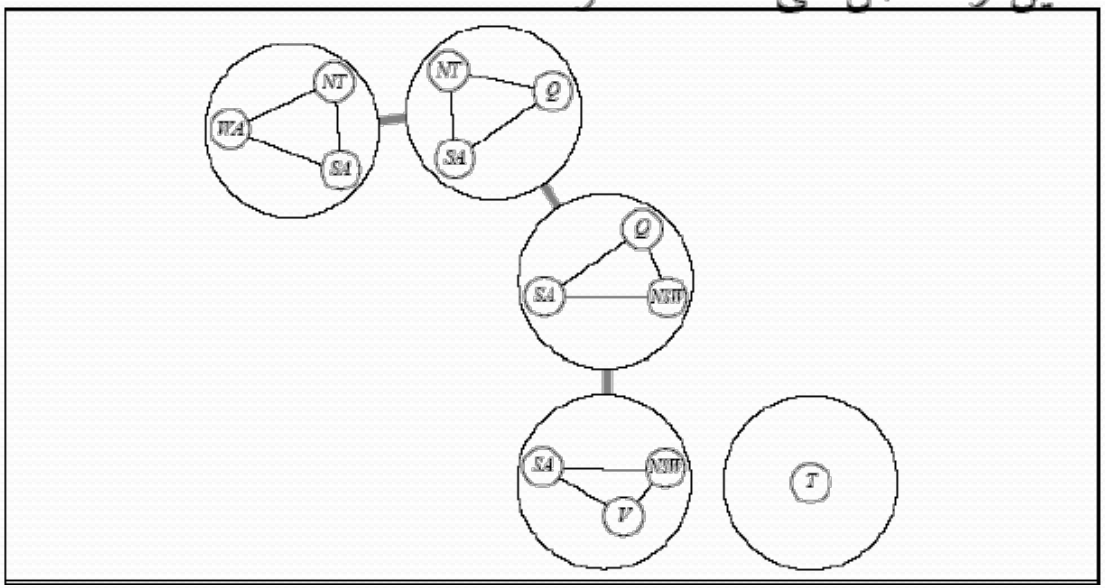


- شرط برشی: مجموعه ای از متغیرها (در تمام حالات ممکن) را مقدار دهی کن به طوریکه گراف گراف محدودیت باقیمانده یک درخت شود. **کات ست چرخه ای**

اندازه برش $= c$ ← زمان اجرا $O(d^c \cdot (n-c)\, d^2)$

رهیافت دوم: تجزیه درختی

- شرایط **تجزیه درختی**:
  - هر متغیر در مسأله اصلی باید حداقل در یک زیرمسأله ظاهر شود
  - اگر دو متغیر به وسیله محدودیتی در مسأله اصلی متصل شده باشند، آنگاه آن دو متغیربا هم(به همراه محدودیت) باید حداقل در یک زیر مسأله ظاهر شوند.
  - اگر متغیری در درخت در دو زیر مسأله ظاهر شده باشد، آنگاه باید در هر زیرمسأله در طول مسیری که زیرمسایل را متصل می کند، ظاهر شود.



رهیافت دوم: تجزیه درختی

- حل تجزیه درختی:

– هر زیر مسأله را به صورت یک mega-variable در نظر می گیریم که دامنه آن مجموعه تمام راه حل های ممکن آن زیر مسأله می باشد.

– سپس محدودیت های میان زیر مسایل را با استفاده از الگوریتم کارای درخت حل می کنیم. مثلا اگر پاسخ اولین زیر مسأله انتساب زیر باشد:

$$\{WA = red, SA = blue, NT = green\}$$

آنگاه تنها پاسخ سازگار برای زیر مسأله بعدی انتساب زیر می باشد:

$$\{SA = blue, NT = green, Q = red\}$$

پیچیدگی زمانی: $O(n \cdot d^{W+1})$

W = رض در راف دود