مسائل ارضای محدودیت

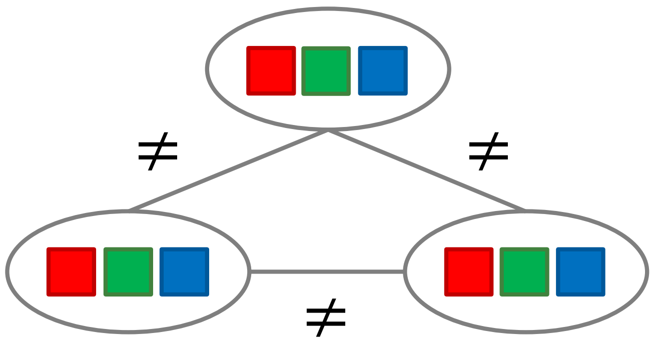
n.razavi@tabrizu.ac.ir سید ناصر رضوی
۱۳۹۷

فهرست مطالب

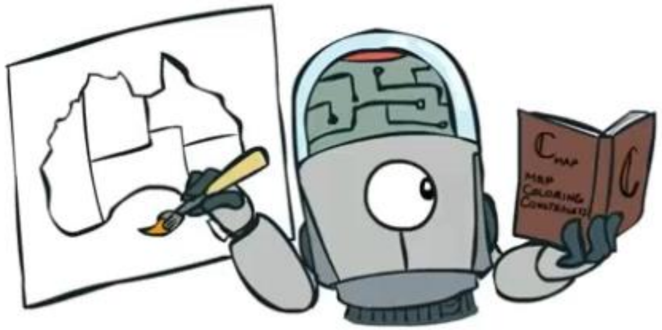
- حل کارای مسائل ارضای محدودیت
- بهره‌برداری از ساختار مسئله
- جستجوی محلی
- الگوریتم کمترین درگیری

یادآوری: مسائل ارضای محدودیت

- مسائل ارضای محدودیت.
  - متغیرها
  - دامنه‌ها
  - محدودیت‌ها
    - ضمنی و صریح
    - یکانی، دودویی و مرتبه‌های بالاتر



- اهداف.
  - یافتن یک راه‌حل
  - یافتن همه راه‌حل‌های ممکن
  - یافتن بهترین راه‌حل ممکن



الگوریتم پایه: جستجوی عقب‌گرد


```
function BACKTRACKING-SEARCH(csp) returns solution/failure
   return RECURSIVE-BACKTRACKING({}, csp)

function RECURSIVE-BACKTRAKING(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
```

بهبود عقب‌گرد

- ایده‌های همه-منظوره باعث افزایش چشمگیری در سرعت اجرا می‌شوند.

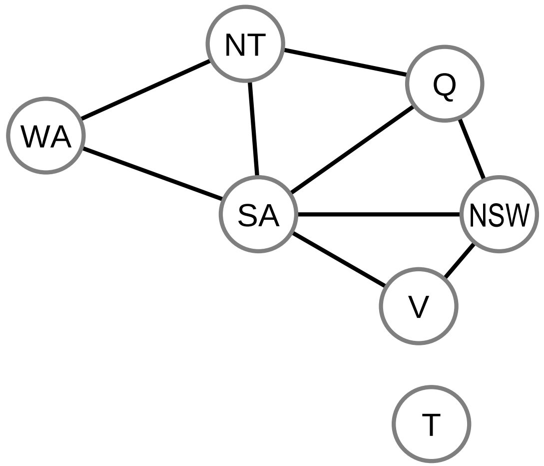
- ترتیب دهی.
  - متغیر بعدی که باید مقداردهی شود کدام است؟
  - مقدار بعدی که باید امتحان شود کدام است؟

- فیلتر کردن.
  - آیا می‌توان شکست‌های حتمی را زودتر تشخیص داد؟

- ساختار.
  - آیا می‌توان از ساختار مسئله بهره برد؟

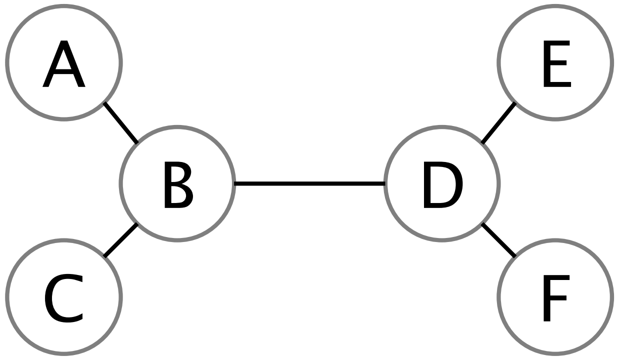
ساختار مسئله

ساختار مسئله



- حالت حدی. زیرمسائل مستقل
  - مثال: تاسمانیا و جزیره اصلی دو زیرمسئله مستقل هستند.
- زیرمسائل مستقل.
  - مؤلفه‌های همبند در گراف محدودیت.
- فرض کنید گرافی شامل n متغیر را بتوان به زیرمسائلی شامل c متغیر تجزیه کرد:
  - هزینه راه حل در بدترین حالت خطی (برحسب n) و برابر $n/c \cdot d^c$ است.
- مثال. $[n = 80, d = 2, c = 20]$
  - $۲^{۸۰}$ برابر است با ۴ میلیارد سال (با نرخ پردازش ۱۰ میلیون گره در ثانیه)
  - $(۴)(۲^{۲۰})$ برابر است با ۰/۴ ثانیه (با نرخ پردازش ۱۰ میلیون گره در ثانیه)

مسائل ارضای محدودیت با ساختار درختی

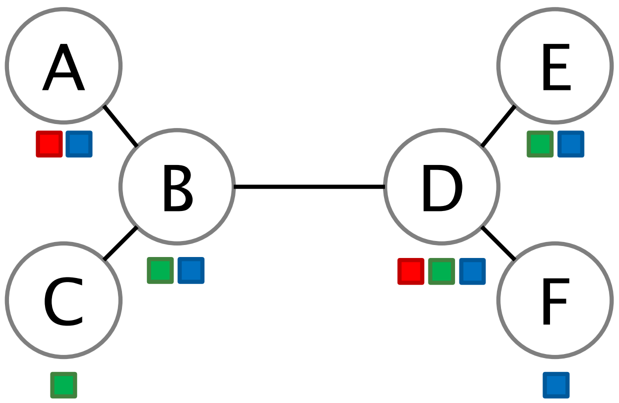


- قضیه. اگر گراف محدودیت بدون دور باشد، مسئله در زمان $O(n\ d^2)$ قابل حل است.
  - در مقایسه با زمان $O(d^n)$ برای مسائل عمومی ارضای محدودیت.

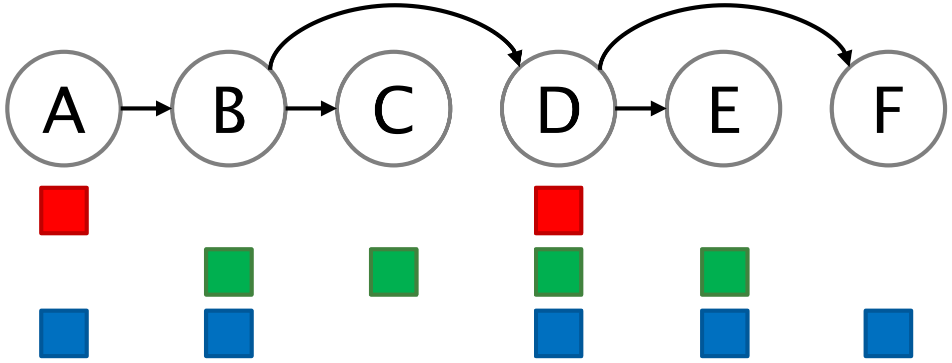
- این ویژگی در استدلال احتمالاتی نیز قابل اعمال است. [با ما باشید]

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.

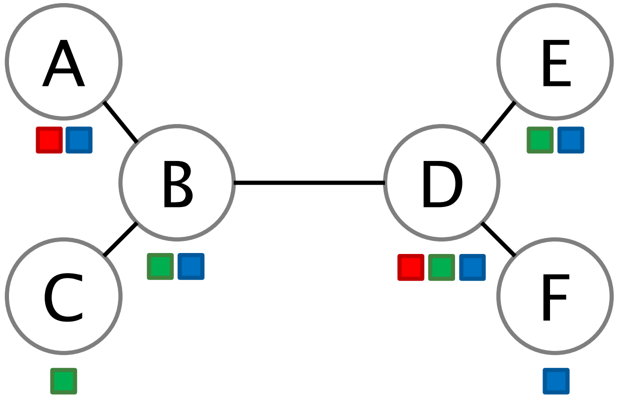


ترتیب توپولوژیکی

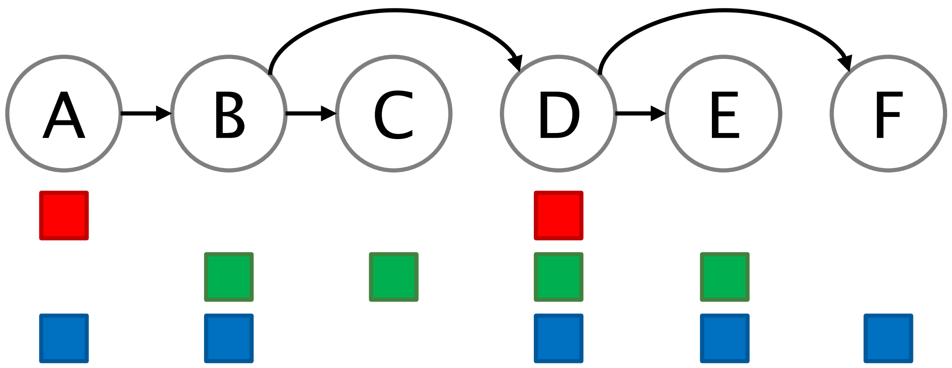


حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.



ترتیب توپولوژیکی

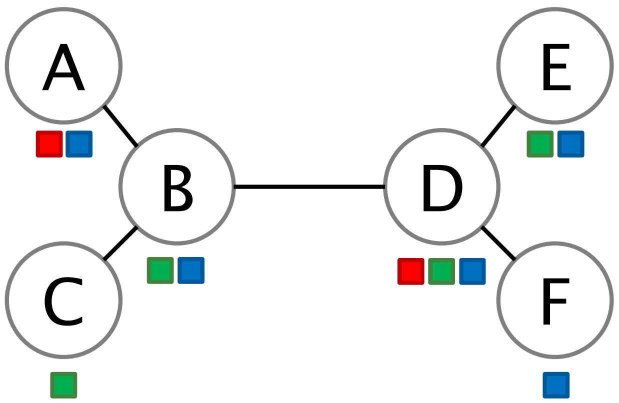


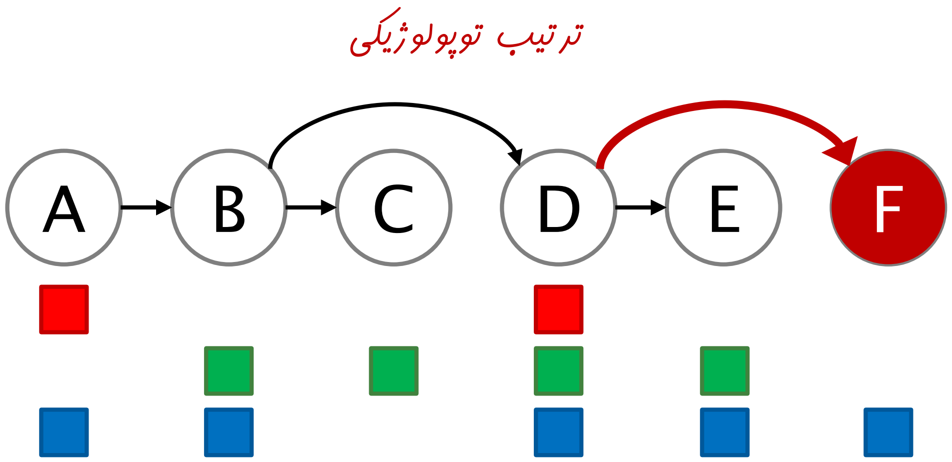
- حذف رو به عقب.

for $i = n : 2$
  REMOVE-INCONSISTENT(Parent(X_i), X_i)

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.



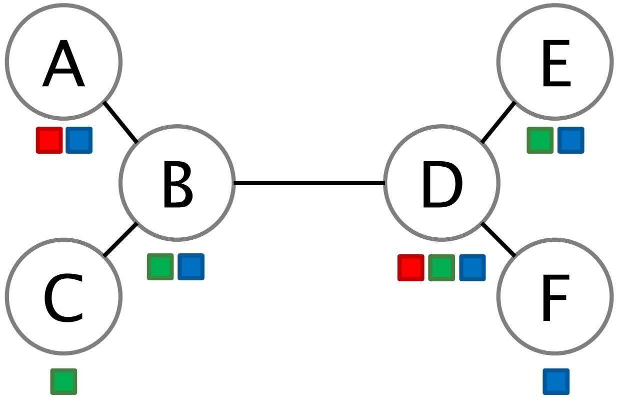


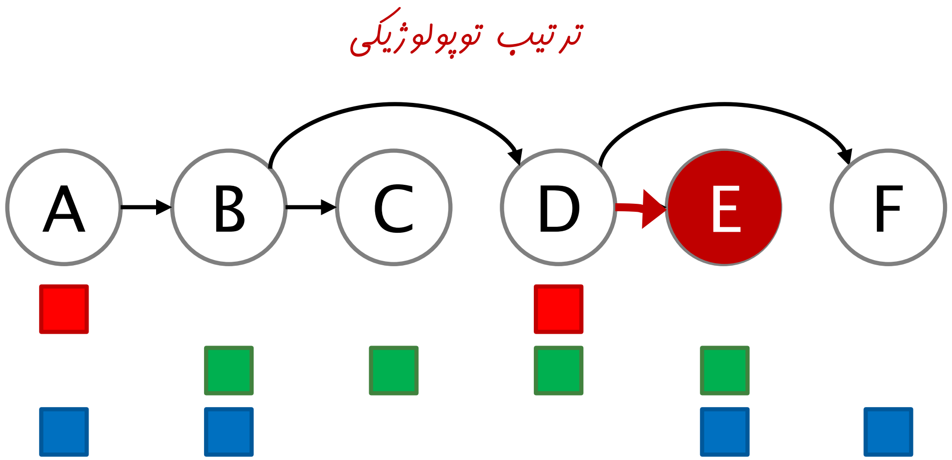
- حذف رو به عقب.

for $i = n : 2$
  REMOVE-INCONSISTENT(Parent(X_i), X_i)

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.



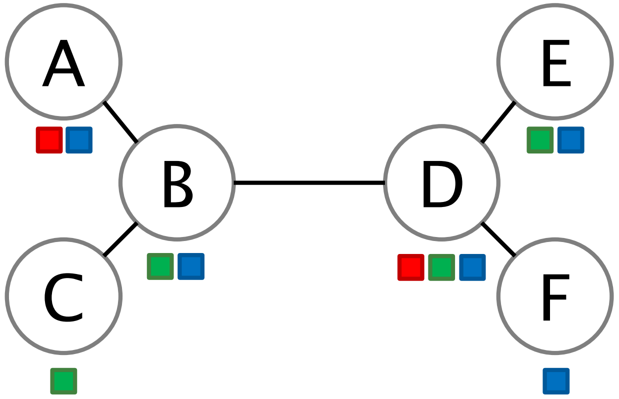


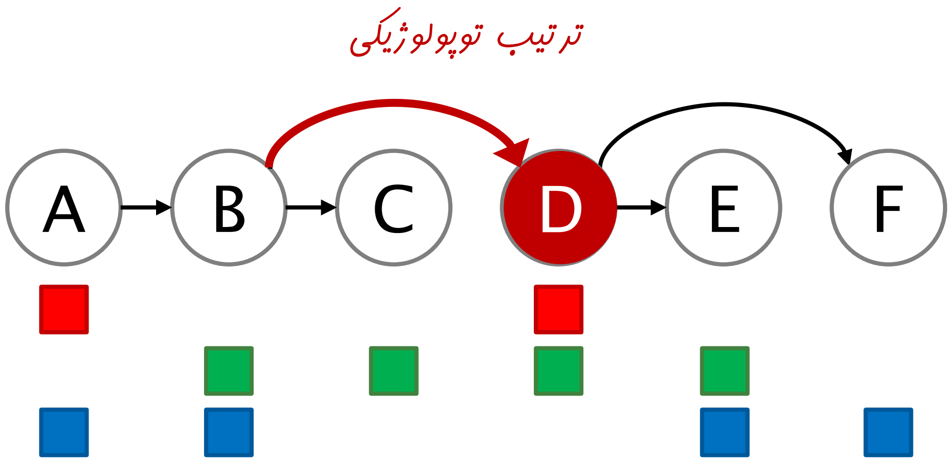
- حذف رو به عقب.

for $i = n : 2$
  REMOVE-INCONSISTENT(Parent(X_i), X_i)

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.



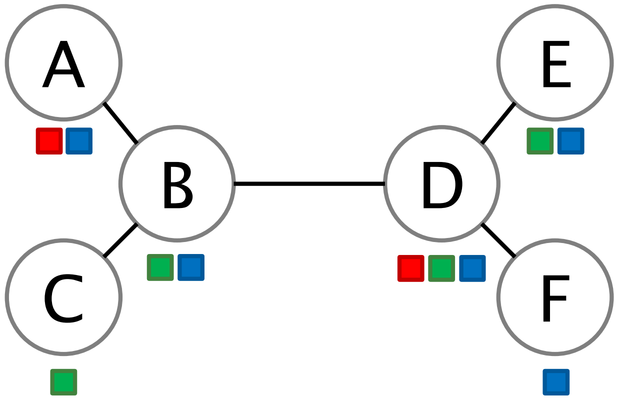


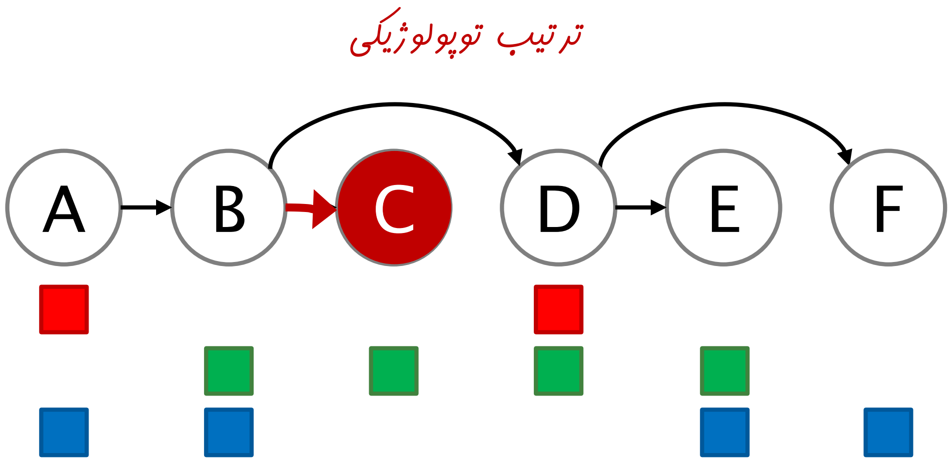
- حذف رو به عقب.

for $i = n : 2$
  REMOVE-INCONSISTENT(Parent(X_i), X_i)

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.



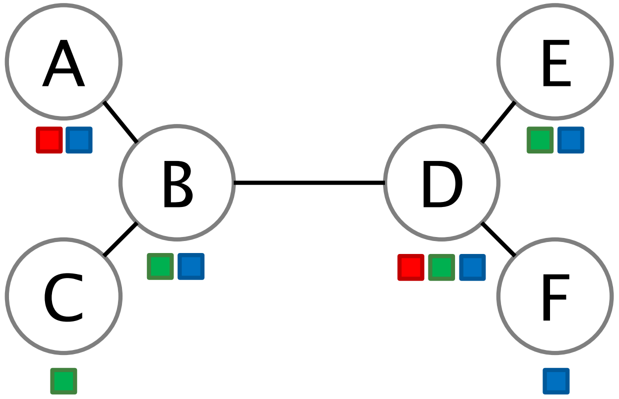


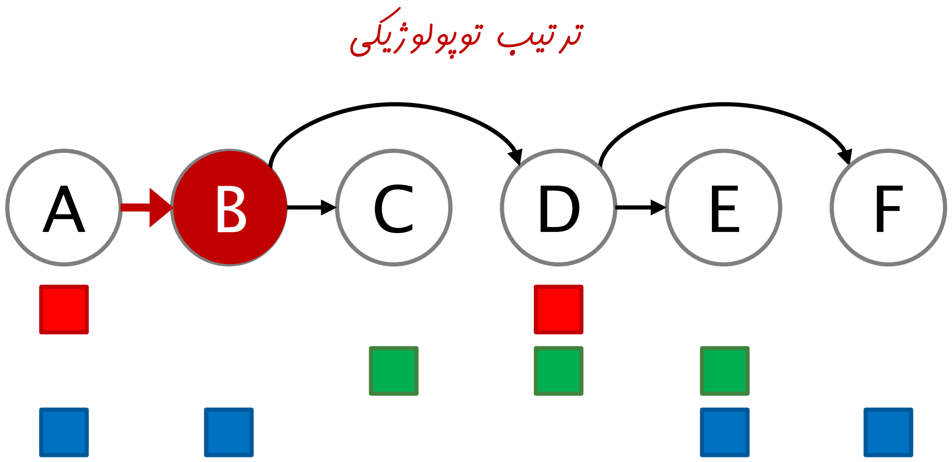
- حذف رو به عقب.

for $i = n : 2$
  REMOVE-INCONSISTENT(Parent(X_i), X_i)

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.



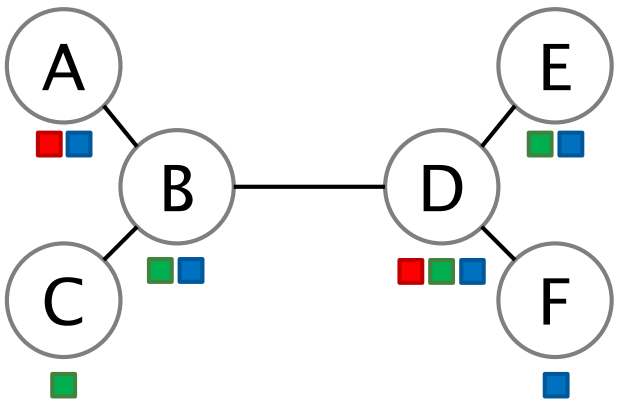


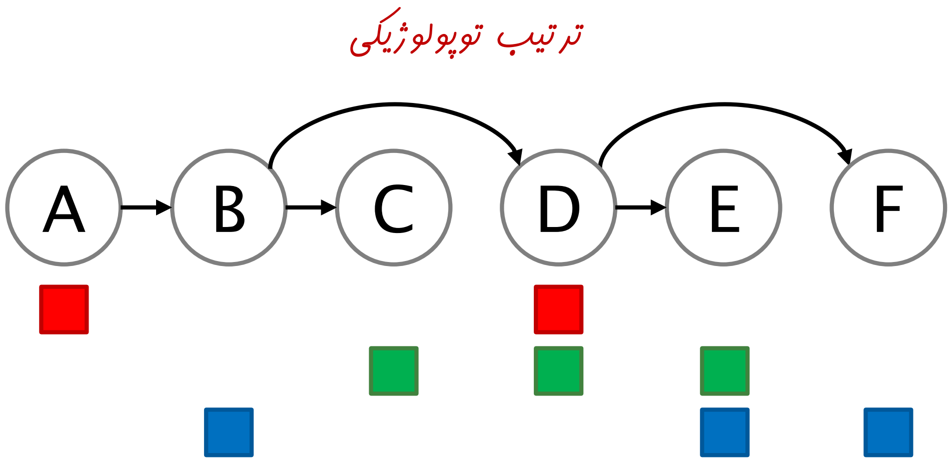
- حذف رو به عقب.

$\textbf{for } i = n : 2$
$\quad \textsc{Remove-Inconsistent}(\text{Parent}(X_i), X_i)$

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.



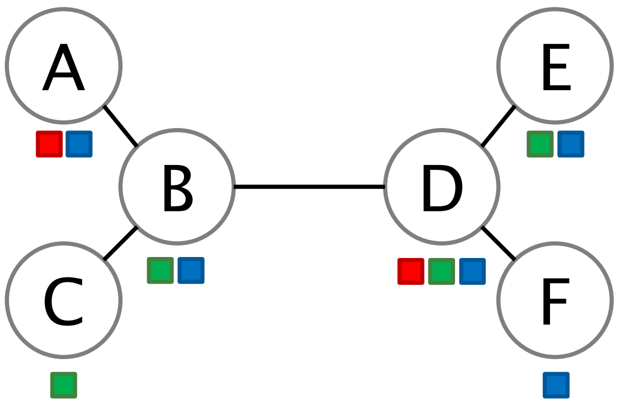


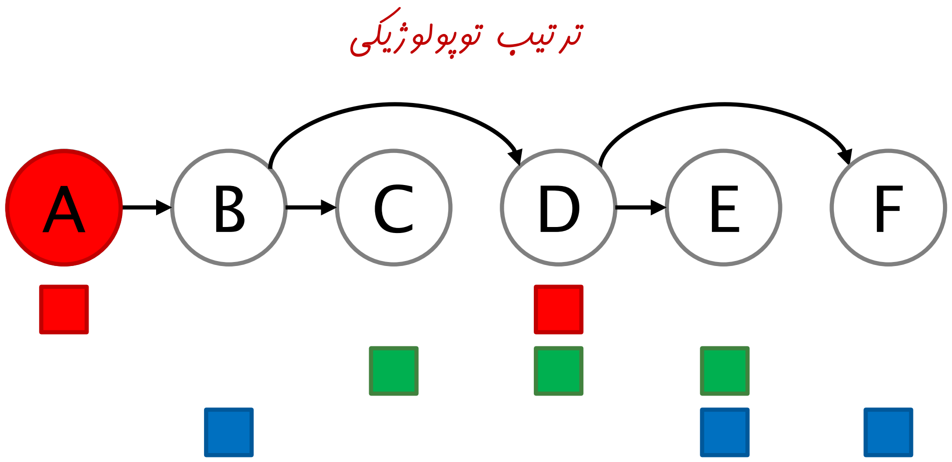
- انتساب رو به جلو.

for $i = 1 : n$
assign X_i consistently with Parent(X_i)

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.



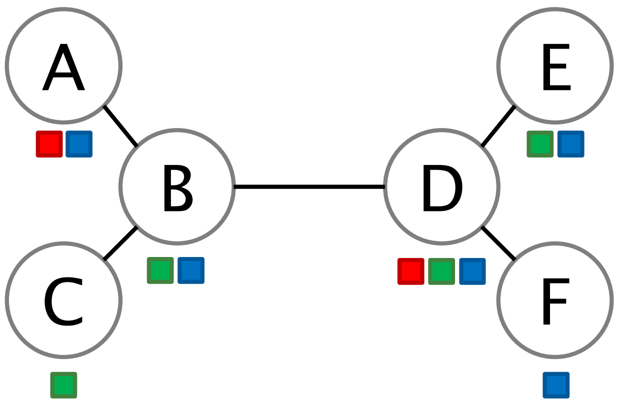


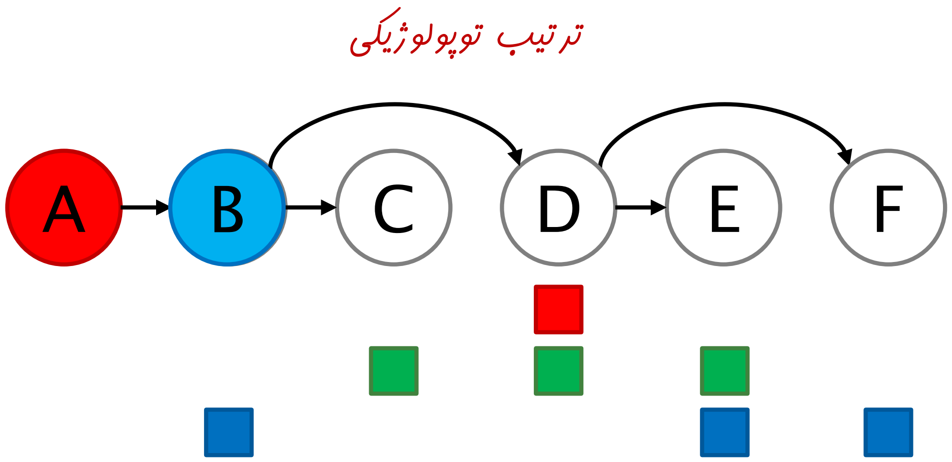
for $i \ = \ 1 : \ n$
  assign X_i consistently with Parent(X_i)

- انتساب رو به جلو.

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.



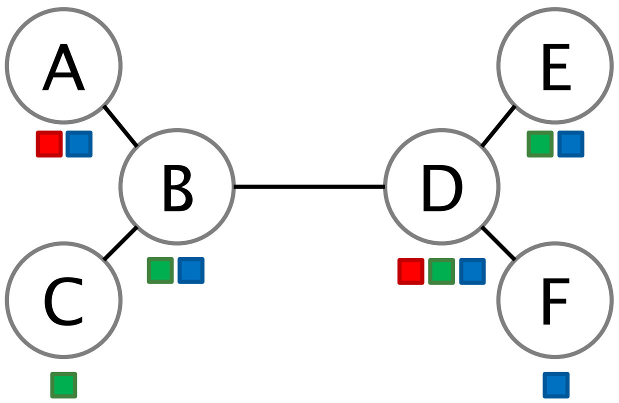


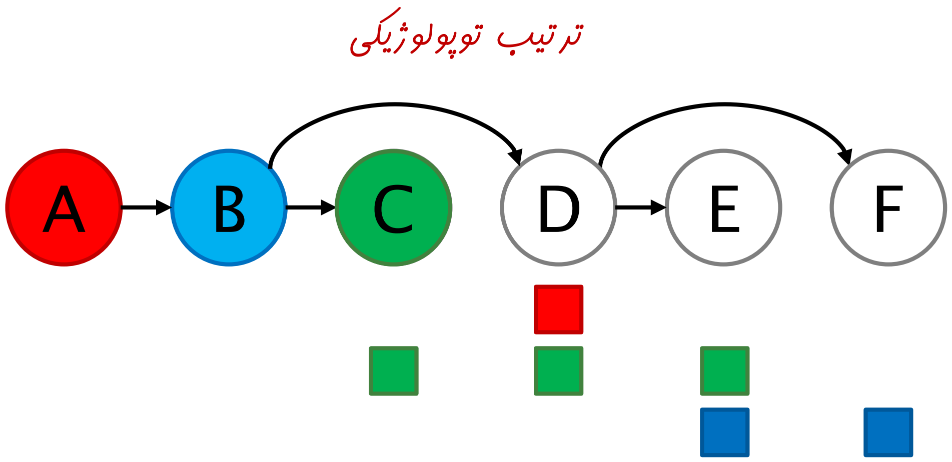
- انتساب رو به جلو.

for $i = 1 : n$
  assign X_i consistently with Parent(X_i)

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.



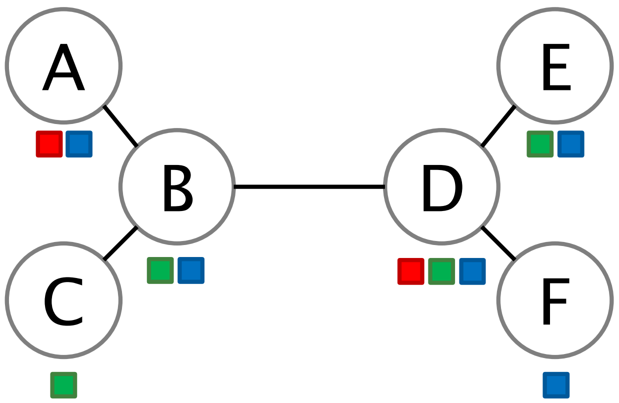


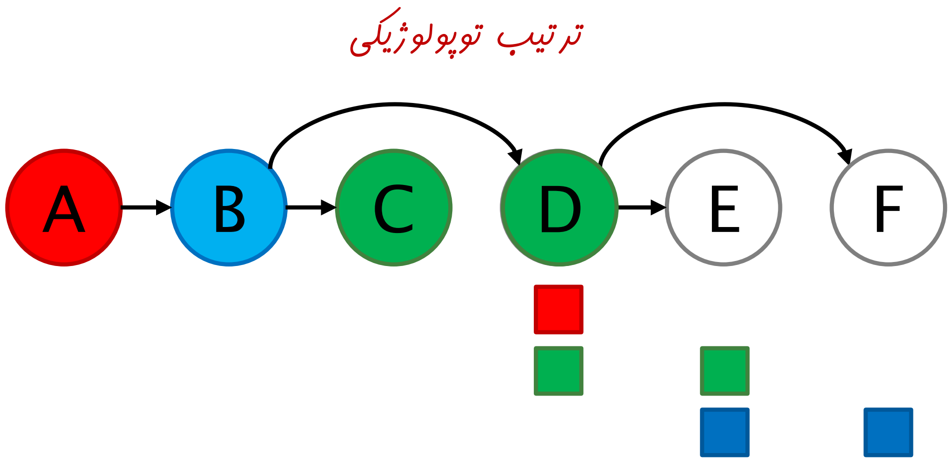
- انتساب رو به جلو.

for $i\ =\ 1:\ n$
  assign X_i consistently with Parent(X_i)

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.



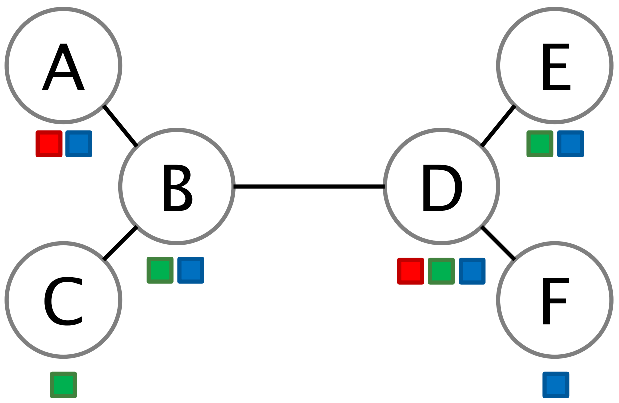


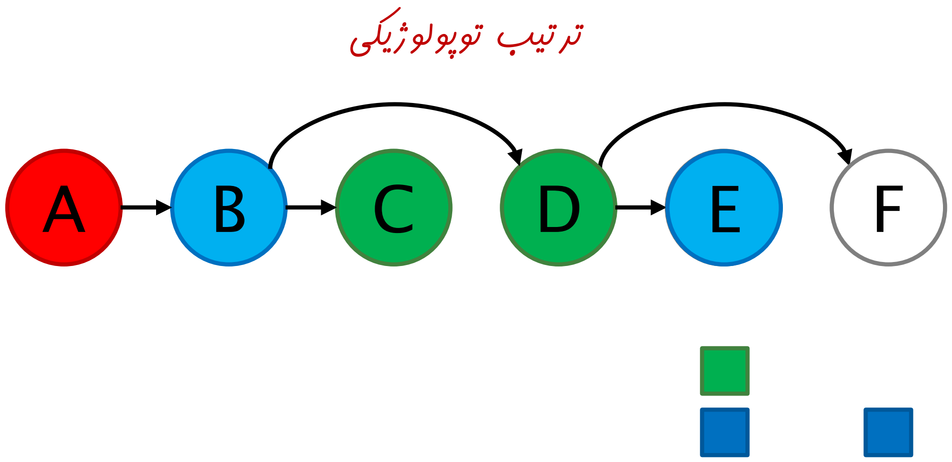
- انتساب رو به جلو.

for $i = 1 : n$
  assign X_i consistently with Parent(X_i)

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.



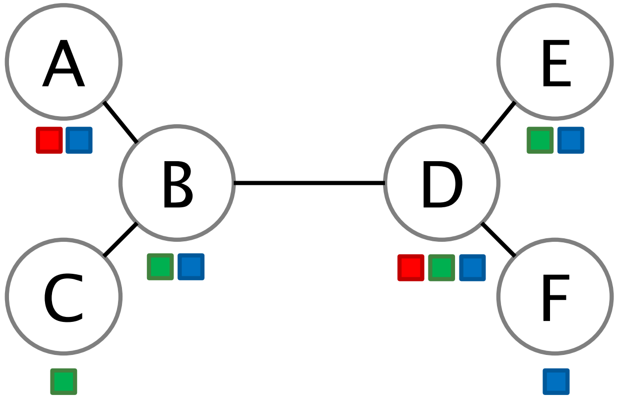


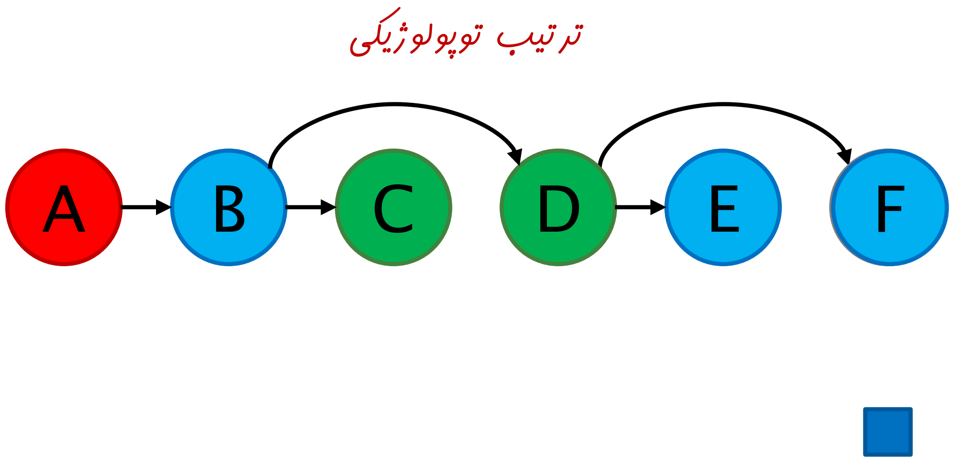
- انتساب رو به جلو.

for $i = 1 : n$
  assign X_i consistently with Parent(X_i)

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.



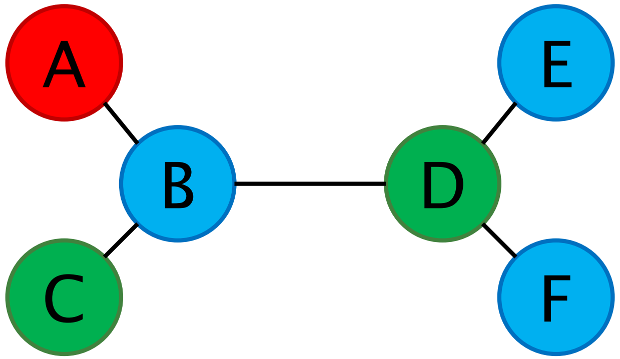


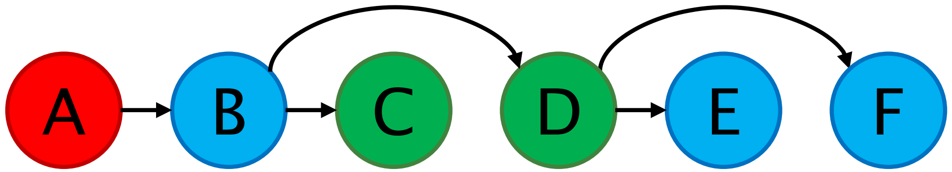
- انتساب رو به جلو.

for $i = 1 : n$
  assign X_i consistently with Parent(X_i)

حل CSP با ساختار درختی

- الگوریتم حل CSP با ساختار درختی.
  - مرتب‌سازی توپولوژیکی. انتخاب یک متغیر به عنوان ریشه و مرتب کردن متغیرها به گونه‌ای که متغیرهای پدر قبل از فرزندان خود قرار بگیرند.

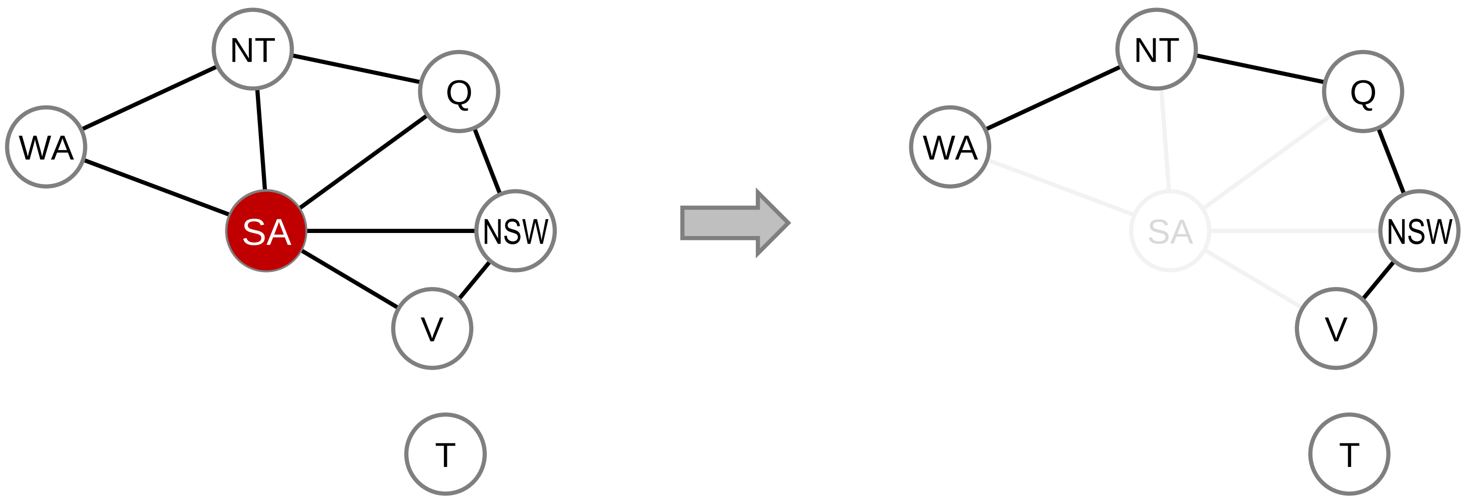




- زمان اجرا. $O(nd^2)$

بهبود ساختار

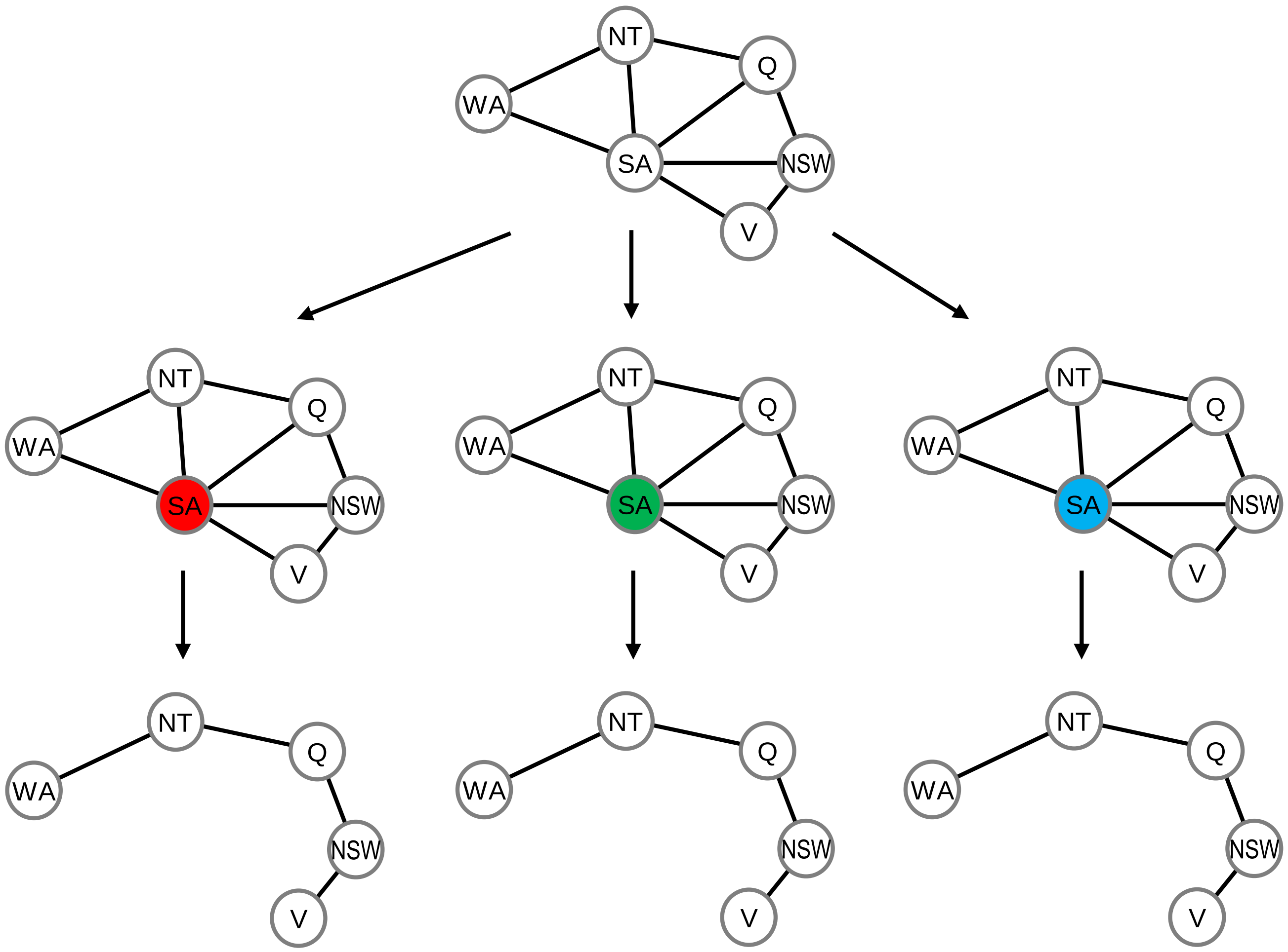
ساختارهای تقریباً درختی



- شرطی سازی. یک متغیر را مقداردهی کن و دامنه همسایه‌های آن را هرس کن.
- شرطی سازی مجموعه برشی. مقداردهی یک زیرمجموعه از متغیرها (به تمام روش‌های ممکن) به گونه‌ای که گراف محدودیت باقیمانده یک درخت باشد.
- اگر اندازه مجموعه برشی برابر با c باشد، زمان اجرا برابر با $O(d^c (n - c) d^2)$ خواهد بود.

[برای مقادیر کوچک c بسیار سریع است]

مجموعه برشی



انتخاب مجموعه برشی

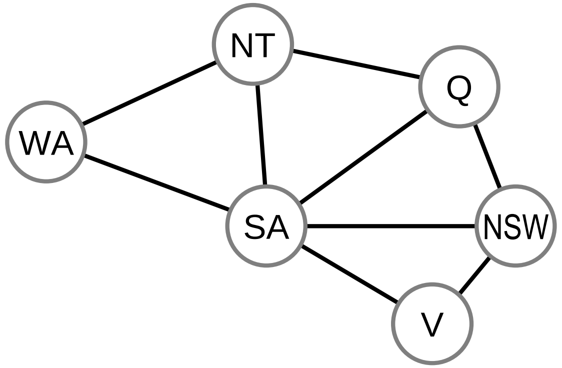
مقداردهی مجموعه برشی

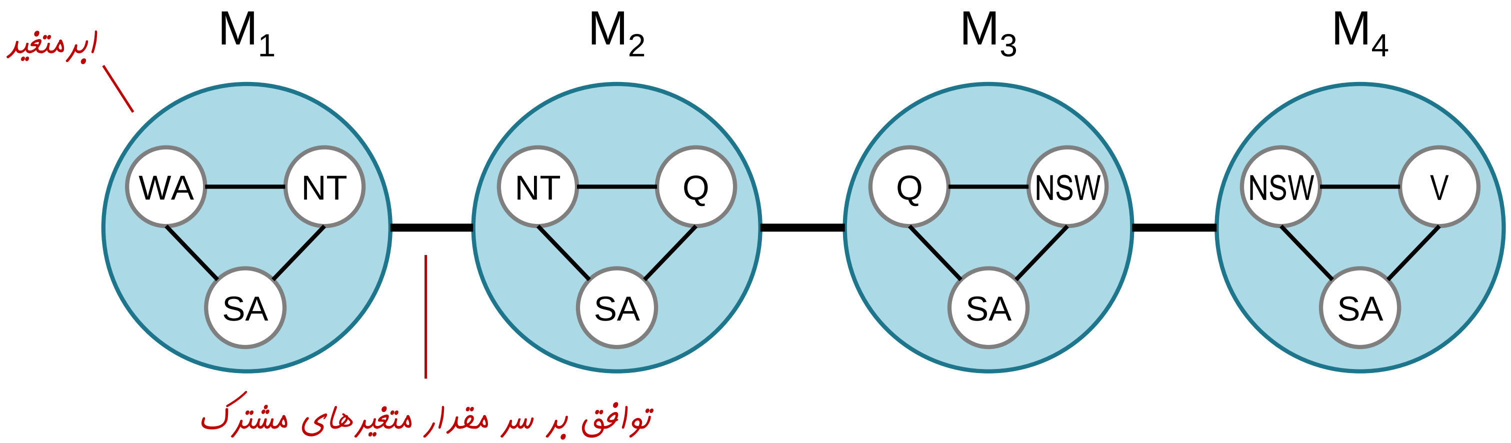
محاسبه گراف باقیمانده

حل گراف باقیمانده

تجزیه درختی

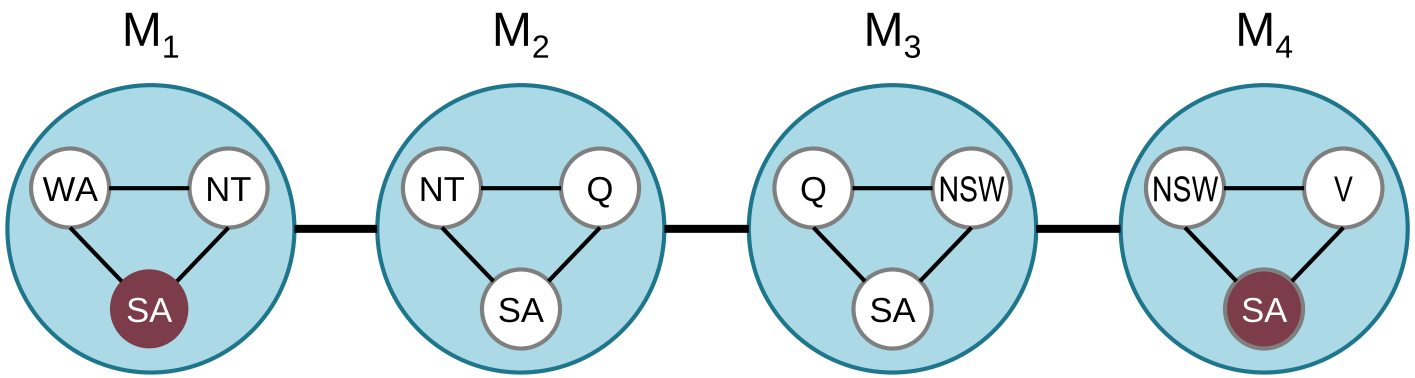
- ایده. ایجاد یک گراف از ابرمتغیرها با ساختار درختی.
  - هر ابرمتغیر دربردارنده بخشی از مسئله اولیه است.
  - برای اطمینان از سازگاری راه‌حل‌ها زیرمسائل همپوشانی دارند.





تجزیه درختی

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

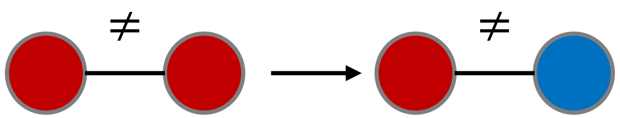


جستجوی محلی

جستجوی محلی

- فرمولەسازی حالت کامل.
  - روش‌های جستجوی محلی معمولاً با حالت‌های «کامل» کار می‌کنند.
  - حالت کامل: حالتی که در آن همه متغیرها دارای مقدار هستند.

- برای استفاده از روش‌های جستجوی محلی در حل مسائل CSP:
  - داشتن انتساب‌هایی با محدودیت‌های نقض شده، مجاز است.
  - عملگرها باید بتوانند به متغیرها مقادیر جدید بدهند.



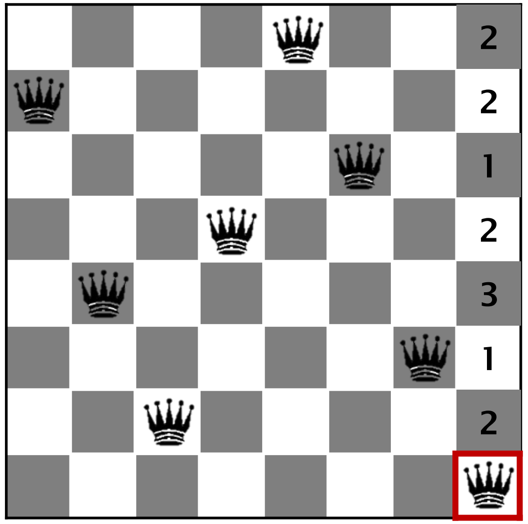
- الگوریتم.
  - انتخاب یک متغیر دارای درگیری به صورت تصادفی
  - انتخاب یک مقدار: «هیوریستیک کمترین درگیری»
    - انتخاب مقداری که کمترین محدودیت‌ها را نقض می‌کند.
    - یعنی، تپه‌نوردی با هیوریستیک «تعداد کل محدودیت‌های نقض شده»

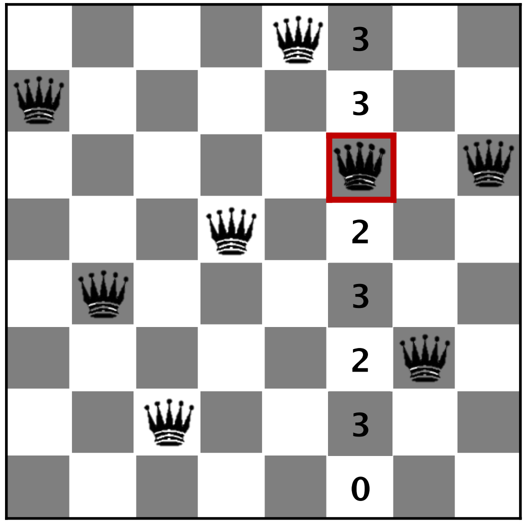
الگوریتم کمترین درگیری

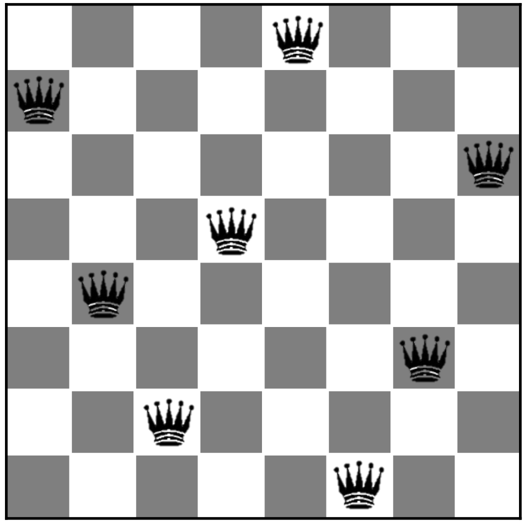

```
function MIN-CONFLICTS(csp, max_steps) returns a solution or failure
  inputs: csp, a constraint satisfaction problem
          max_steps, the number of steps allowed before giving up

  current ← an initial complete assignment for csp
  for i = 1 to max_steps do
    if current is a solution for csp then return current
    var ← a randomly chosen, conflicting variable from csp.VARIABLES
    value ← the value v for var that minimizes CONFLICTS(var, v, current, csp)
    set var = value in current
  return failure
```

الگوریتم کمترین درگیری: ۸-وزیر

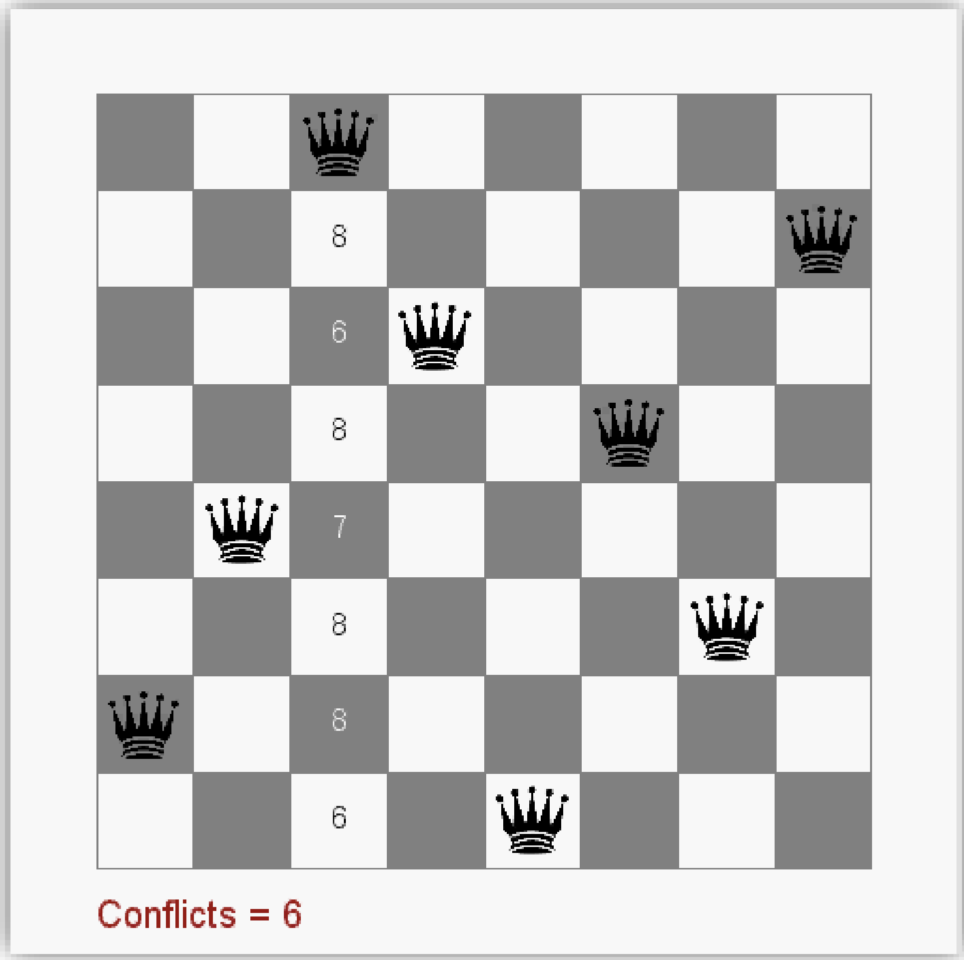
- یک راه‌حل دو مرحله‌ای برای مسئله ۸-وزیر.







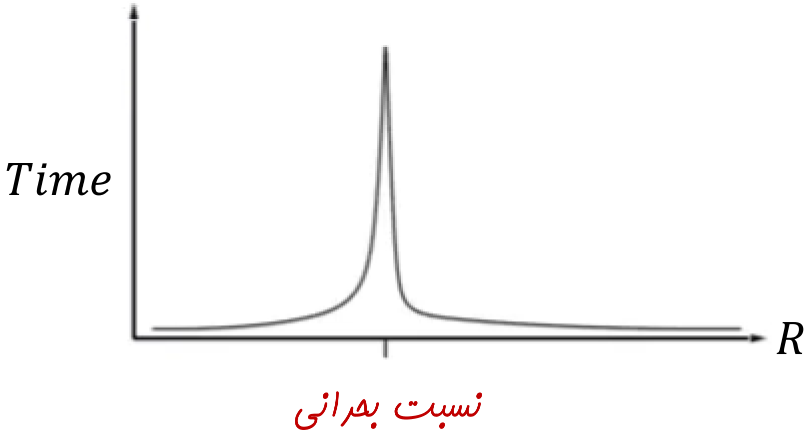
الگوریتم حداقل درگیری: اجرای نمایشی

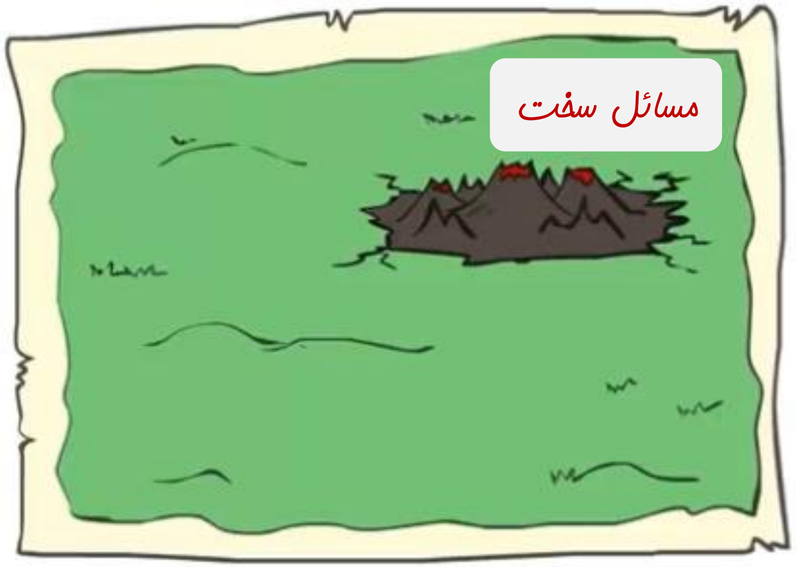


الگوریتم حداقل درگیری: کارایی

- با داشتن یک حالت اولیه تصادفی، می‌توان مسئله n-وزیر را برای مقادیر بزرگ n با احتمال بالا تقریبا در زمان ثابت حل نمود. [مثلا برای ۱۰,۰۰۰,۰۰۰ وزیر]
- به نظر می‌رسد همین موضوع برای هر مسئله دیگر با یک حالت شروع تصادفی درست باشد، به جز برای یک محدوده کوچک از مقادیر نزدیک به نسبت بحرانی.

$$R = \frac{\text{تعداد محدودیت‌ها}}{\text{تعداد متغیرها}}$$





خلاصه

- مسائل ارضای محدودیت.
  - نوع خاصی از مسائل جستجو هستند.
  - حالت‌ها: انتساب‌های جزئی
  - آزمون هدف: یک انتساب کامل و سازگار
- الگوریتم پایه. جستجوی عقب‌گرد
- افزایش سرعت.
  - ترتیب دهی (متغیرها و مقادیر)
  - فیلتر کردن (بررسی رو به جلو و سازگاری کمان)
  - بهره‌برداری از ساختار مسئله
- الگوریتم حداقل درگیری. در اغلب موارد بسیار کارا است.