

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



هوش مصنوعی

فصل ۶

مسائل ارضای قید

Constraint Satisfaction Problems

کاظم فولادی قلعه
دانشکده مهندسی، پردیس فارابی
دانشگاه تهران

<http://courses.fouladi.ir/ai>

هوش مصنوعی

مسائل ارضای قید

۱

تعریف
مسائل
ارضای قید

مسئله‌ی ارضای قید

معرفی

مسئله‌ی ارضای قید

Constraint Satisfaction Problem (CSP)

مجموعه‌ای از **متغیرها** با **دامنه‌ی** مشخص وجود دارند.
هدف تعیین مقدار برای این متغیرهاست به گونه‌ای که یک مجموعه از **قیدها** ارضا شوند.

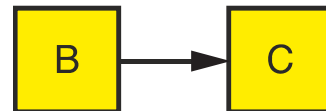
انواع زیادی از مسائل می‌توانند به صورت CSP تعریف شوند.

مسائل CSP را می‌توان توسط الگوریتم‌های عمومی جستجو حل کرد،
اما به دلیل ساختار خاص این مسائل، الگوریتم‌های کارآمدتری برای آنها وجود دارد.

مسئله‌ی ارضای قید

حالت

جستجوی استاندارد

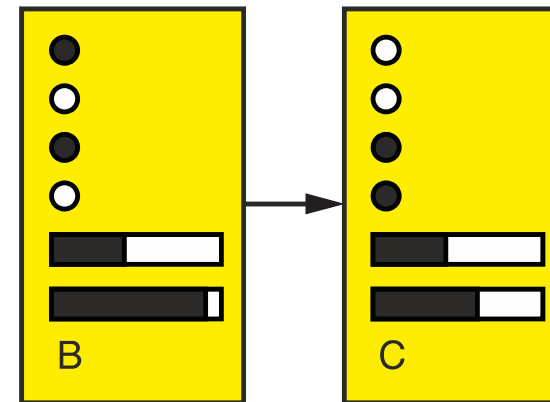


(a) Atomic

حالت: یک جعبه‌ی سیاه

آزمون هدف: تعیین هدف بودن یا هدف
نبودن یک حالت (گره): ضمنی یا صریح

مسئله‌ی ارضای قید



(b) Factored

حالت: برداری از مقادیر خصیصه‌ها

آزمون هدف: مجموعه‌ای از قیدها
برای مشخص کردن ترکیبات مجاز متغیرها

مسئله‌ی ارضای قید

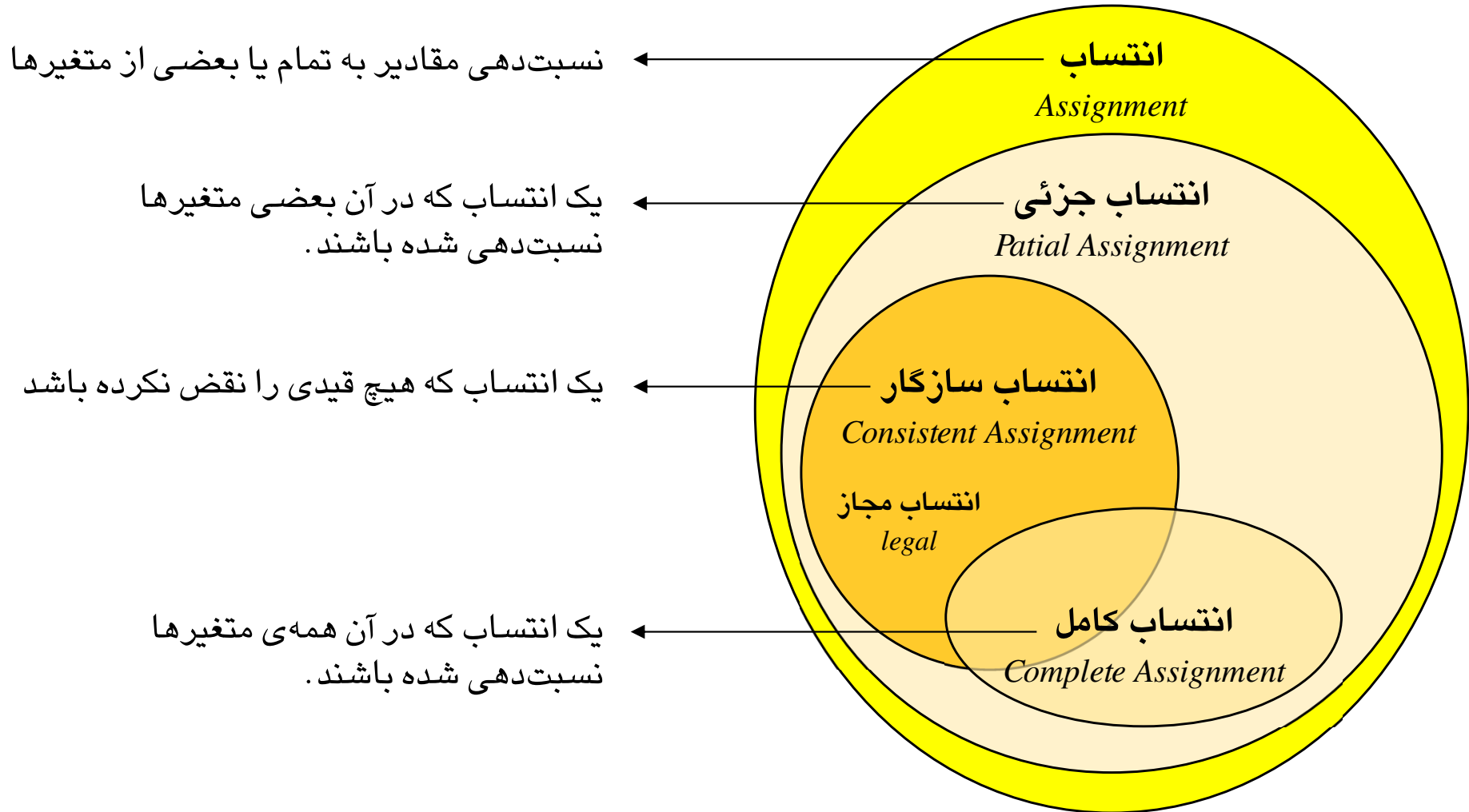
تعریف ریاضی

$\{X_1, X_2, \dots, X_n\}$	مجموعه‌ای از متغیرها	X	متغیرها Variables	مؤلفه‌های سه‌گانه‌ی تعریف مسئله‌ی «ارضای قید»
$\{D_1, D_2, \dots, D_n\}$	مجموعه‌ای از دامنه‌ها (مقادیر مجاز متغیرها) هر دامنه به صورت $\{v_1, v_2, \dots, v_k\}$	D	دامنه‌ها Domains	
مجموعه‌ای از قیدها: مشخص‌کننده‌ی مقادیر مجاز ترکیبات متغیرها هر قید به صورت رابطه‌ای روی تعدادی متغیر (نمایش لیستی رابطه / رابطه‌ی انتزاعی) قید صریح (explicit): به صورت مجموعه‌ای از مقادیر مجاز قید ضمنی (implicit): با استفاده از تابعی که ارضا شدن قیدها را بررسی می‌کند.		C	قیدها Constraints	

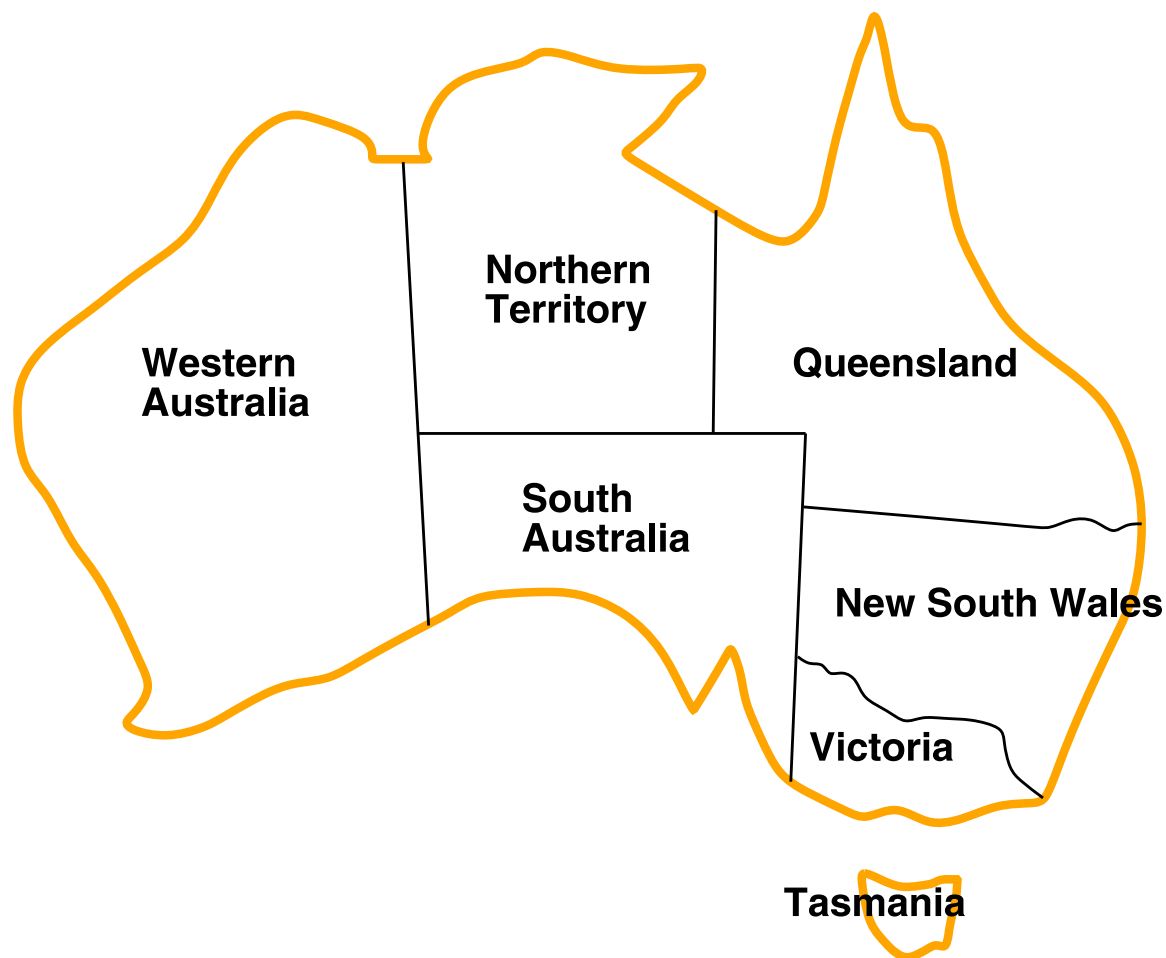
انتساب

انتساب مقدار به متغیر و انواع آن

ASSIGNMENT



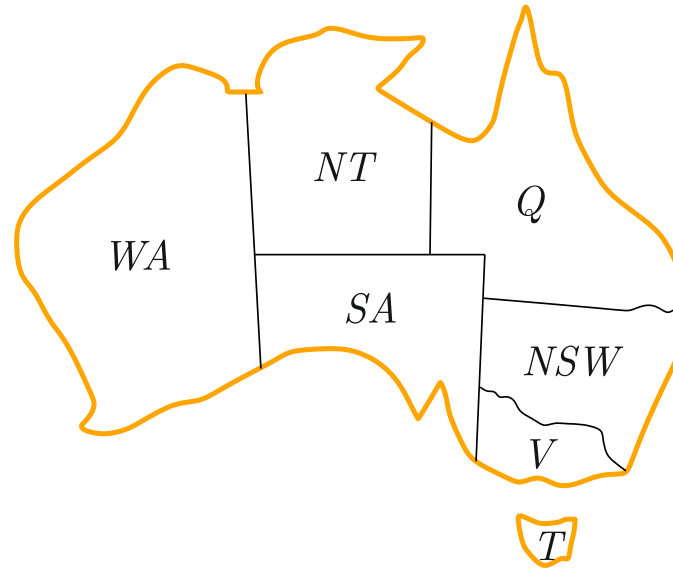
مسئله‌ی «رنگ‌آمیزی نقشه»

MAP-COLORING

انتساب رنگ به استان‌ها به طوری که استان‌های مجاور هم‌رنگ نباشند.

مسئله ارضای قید

مثال: مسئله رنگ آمیز نقشه

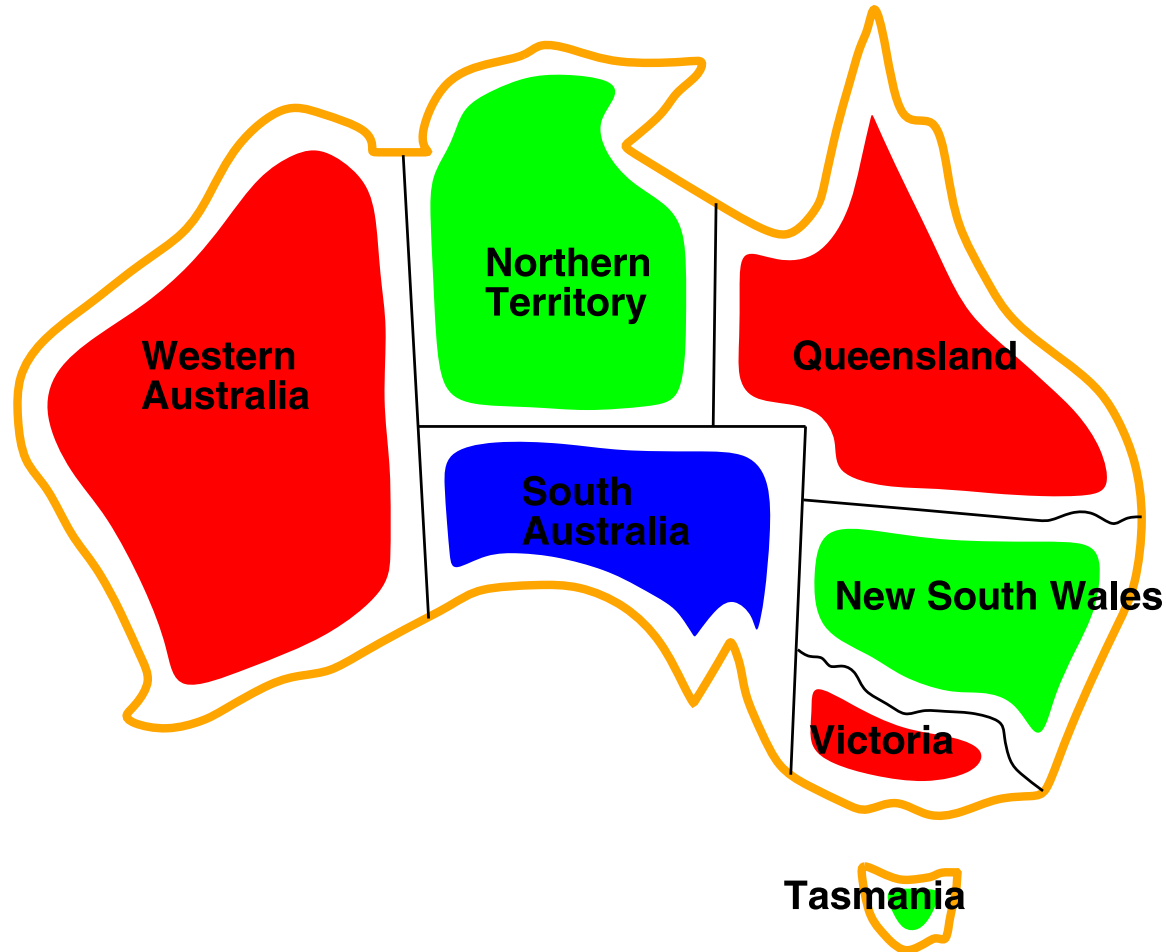


WA, NT, Q, NSW, V, SA, T	استان‌ها	X	متغیرها Variables
$D_i = \{red, blue, green\}$	رنگ‌ها	D	دامنه‌ها Domains
ناحیه‌های مجاور باید رنگ‌های متفاوت داشته باشند. e.g., $WA \neq NT, V \neq NSW, \dots$		C	قیدها Constraints

مؤلفه‌های تعریف مسئله «ارضای قید»

مسئله‌ی ارضای قید

مثال: مسئله‌ی رنگ‌آمیز نقشه: راه‌حل



راه‌حل‌ها: انتساب‌هایی هستند که همه‌ی قیدها را ارضا کنند، مانند:

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

گراف قید

CONSTRAINT GRAPH

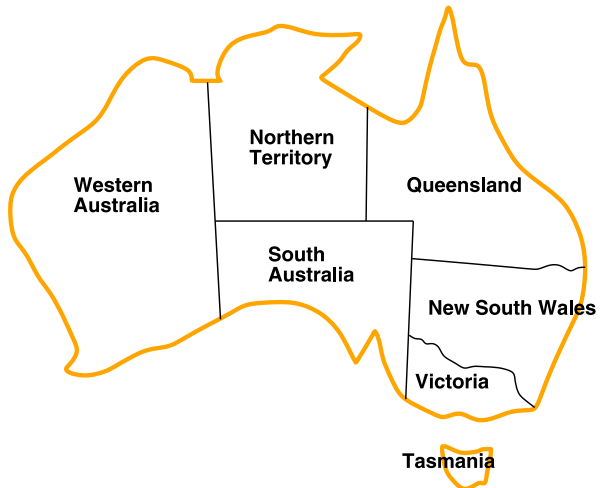
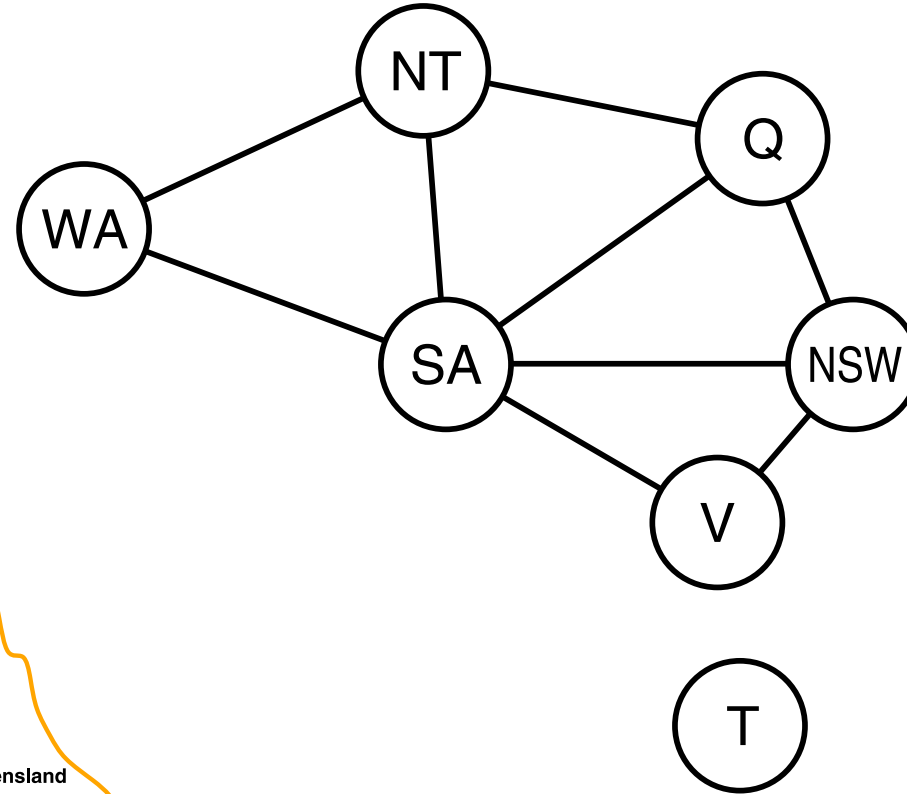
گراف قید <i>Constraint Graph</i>	
گره‌ها <i>Nodes</i>	کمان‌ها <i>Arcs</i>
نشان‌دهنده‌ی متغیرها	نشان‌دهنده‌ی قیدها

قابل استفاده برای CSP دودویی

الگوریتم‌های CSP همه‌منظوره، از ساختار گراف برای تسریع جستجو استفاده می‌کنند.

گراف قید

مثال

CONSTRAINT GRAPH

انواع «مسائل ارضای قید»

VARIETIES OF CSPs

نوع قیدها <i>Constraints Types</i>	نوع قیدها <i>Constraints Types</i>	روابط قیدها <i>Constraints Relations</i>	متغیرها / دامنه‌ها <i>Variables / Domains</i>	
ضروری <i>Mandatory</i>	خطی <i>Linear</i>	تکی <i>Unary</i>	پیوسته <i>Continuous</i>	نامتناهی <i>Infinite</i>
ترجیحی <i>Preference</i>	غیرخطی <i>Non-linear</i>	دودویی <i>Binary</i>	گسسته <i>Discrete</i>	متناهی <i>Finite</i>
		چندتایی <i>n-ary</i>		
		سراسری <i>Global</i> باهر تعداد متغیر		

انواع «مسائل ارضای قید»

CSP با دامنه‌ی متناهی و گسسته

DISCRETE FINITE-DOMAIN CSP

نوع قیدها Constraints Types	نوع قیدها Constraints Types	روابط قیدها Constraints Relations	متغیرها / دامنه‌ها Variables / Domains	
ضروری Mandatory	خطی Linear	تکی Unary	پیوسته Continuous	نامتناهی Infinite
ترجیحی Preference	غیرخطی Non-linear	دودویی Binary	گسسته Discrete	
		چندتایی n-ary		متناهی Finite
		سراسری Global با هر تعداد متغیر		

دامنه‌های با اندازه‌ی d : تعداد $O(d^n)$ انتساب کامل

مانند Boolean CSPs (شامل Boolean Satisfiability): یک مسئله‌ی NP-complete

انواع «مسائل ارضای قید»

CSP با دامنه‌ی نامتناهی و گسسته

DISCRETE INFINITE-DOMAIN CSP

نوع قیدها Constraints Types	نوع قیدها Constraints Types	روابط قیدها Constraints Relations	متغیرها / دامنه‌ها Variables / Domains	
ضروری Mandatory	خطی Linear	تکی Unary	پیوسته Continuous	نامتناهی Infinite
ترجیحی Preference	غیرخطی Non-linear	دودویی Binary	گسسته Discrete	
		چندتایی n-ary		متناهی Finite
		سراسری Global با هر تعداد متغیر		

دامنه‌هایی مانند مجموعه‌ی اعداد صحیح، رشته‌ها، ...

مانند زمان‌بندی کارها (متغیرها: روزهای شروع / پایان هر کار)

نیازمند یک زبان قید (مثلاً عباراتی چون $StartJob_1 + 5 \leq StartJob_3$)

↓
Constraint language

انواع «مسائل ارضای قید»

CSP با دامنه‌ی نامتناهی و گسسته با قیدهای خطی

DISCRETE INFINITE-DOMAIN CSP WITH LINEAR CONSTRAINTS

نوع قیدها Constraints Types		نوع قیدها Constraints Types		روابط قیدها Constraints Relations		متغیرها / دامنه‌ها Variables / Domains	
ضروری Mandatory	خطی Linear	تکی Unary	پیوسته Continuous	نامتناهی Infinite		گسسته Discrete	
ترجیحی Preference	غیرخطی Non-linear	دودویی Binary					
		چندتایی n-ary		متناهی Finite			
		سراسری Global با هر تعداد متغیر					

مسئله CSP با دامنه‌ی نامتناهی و گسسته با قیدهای خطی «قابل حل» (solvable) است.

انواع «مسائل ارضای قید»

CSP با دامنه‌ی نامتناهی و گسسته با قیدهای غیرخطی

DISCRETE INFINITE-DOMAIN CSP WITH NON-LINEAR CONSTRAINTS

نوع قیدها <i>Constraints Types</i>	نوع قیدها <i>Constraints Types</i>	روابط قیدها <i>Constraints Relations</i>	متغیرها / دامنه‌ها <i>Variables / Domains</i>	
ضروری <i>Mandatory</i>	خطی <i>Linear</i>	تکی <i>Unary</i>	پیوسته <i>Continuous</i>	نامتناهی <i>Infinite</i>
ترجیحی <i>Preference</i>	غیرخطی <i>Non-linear</i>	دودویی <i>Binary</i>	گسسته <i>Discrete</i>	
		چندتایی <i>n-ary</i>		متناهی <i>Finite</i>
		سراسری <i>Global</i> با هر تعداد متغیر		

مسئله CSP با دامنه‌ی نامتناهی و گسسته با قیدهای غیرخطی «تصمیم‌ناپذیر» (undecidable) است.

انواع «مسائل ارضای قید»

CSP با دامنه‌ی نامتناهی و پیوسته با قیدهای غیرخطی

CONTINUOUS INFINITE-DOMAIN CSP WITH LINEAR CONSTRAINTS

نوع قیدها <i>Constraints Types</i>	نوع قیدها <i>Constraints Types</i>	روابط قیدها <i>Constraints Relations</i>	متغیرها / دامنه‌ها <i>Variables / Domains</i>	
ضروری <i>Mandatory</i>	خطی <i>Linear</i>	تکی <i>Unary</i>	پیوسته <i>Continuous</i>	نامتناهی <i>Infinite</i>
ترجیحی <i>Preference</i>	غیرخطی <i>Non-linear</i>	دودویی <i>Binary</i>	گسسته <i>Discrete</i>	متناهی <i>Finite</i>
		چندتایی <i>n-ary</i>		
		سراسری <i>Global</i> با هر تعداد متغیر		

مسئله CSP با دامنه‌ی نامتناهی و پیوسته با قیدهای خطی در زمان خطی قابل حل است (با روش‌های برنامه‌ریزی خطی).

انواع «مسائل ارضای قید»

CSP با دامنه‌ی نامتناهی و پیوسته با قیدهای غیرخطی

CONTINUOUS INFINITE-DOMAIN CSP WITH NON-LINEAR CONSTRAINTS

نوع قیدها Constraints Types	نوع قیدها Constraints Types	روابط قیدها Constraints Relations	متغیرها / دامنه‌ها Variables / Domains	
ضروری Mandatory	خطی Linear	تکی Unary	پیوسته Continuous	نامتناهی Infinite
ترجیحی Preference	غیرخطی Non-linear	دودویی Binary	گسسته Discrete	
		چندتایی n-ary		متناهی Finite
		سراسری Global با هر تعداد متغیر		

مسئله CSP با دامنه‌ی نامتناهی و پیوسته با قیدهای غیرخطی «تصمیم‌ناپذیر» (undecidable) است.

انواع «مسائل ارضای قید»

CSP با قیدهای تکی

UNARY CONSTRAINTS

نوع قیدها <i>Constraints Types</i>	نوع قیدها <i>Constraints Types</i>	روابط قیدها <i>Constraints Relations</i>	متغیرها / دامنه‌ها <i>Variables / Domains</i>	
ضروری <i>Mandatory</i>	خطی <i>Linear</i>	تکی <i>Unary</i>	پیوسته <i>Continuous</i>	نامتناهی <i>Infinite</i>
ترجیحی <i>Preference</i>	غیرخطی <i>Non-linear</i>	دودویی <i>Binary</i>	گسسته <i>Discrete</i>	متناهی <i>Finite</i>
		چندتایی <i>n-ary</i>		
		سراسری <i>Global</i> با هر تعداد متغیر		

قیدهای تکی شامل یک متغیر هستند.

مانند $SA \neq green$

انواع «مسائل ارضای قید»

CSP با قیدهای دودویی

BINARY CONSTRAINTS

نوع قیدها <i>Constraints Types</i>	نوع قیدها <i>Constraints Types</i>	روابط قیدها <i>Constraints Relations</i>	متغیرها / دامنه‌ها <i>Variables / Domains</i>	
ضروری <i>Mandatory</i>	خطی <i>Linear</i>	تکی <i>Unary</i>	پیوسته <i>Continuous</i>	نامتناهی <i>Infinite</i>
ترجیحی <i>Preference</i>	غیرخطی <i>Non-linear</i>	دودویی <i>Binary</i>	گسسته <i>Discrete</i>	متناهی <i>Finite</i>
		چندتایی <i>n-ary</i>		
		سراسری <i>Global</i>		

قیدهای دودویی شامل دو متغیر هستند.

مانند $SA \neq WA$

انواع «مسائل ارضای قید»

CSP با قیدهای چندتایی

N-ARY CONSTRAINTS

نوع قیدها <i>Constraints Types</i>	نوع قیدها <i>Constraints Types</i>	روابط قیدها <i>Constraints Relations</i>	متغیرها / دامنه‌ها <i>Variables / Domains</i>	
ضروری <i>Mandatory</i>	خطی <i>Linear</i>	تکی <i>Unary</i>	پیوسته <i>Continuous</i>	نامتناهی <i>Infinite</i>
ترجیحی <i>Preference</i>	غیرخطی <i>Non-linear</i>	دودویی <i>Binary</i>	گسسته <i>Discrete</i>	متناهی <i>Finite</i>
		چندتایی <i>n-ary</i>		
		سراسری <i>Global</i> با هر تعداد متغیر		

قیدهای مرتبه بالاتر (چندتایی) شامل بیش از دو متغیر هستند (Higher-order constraints).

مانند محدودیت روی ستون‌های مسئله‌ی «حساب رمزی»

انواع «مسائل ارضای قید»

CSP با قیدهای سراسری

GLOBAL-CONSTRAINTS

نوع قیدها <i>Constraints Types</i>	نوع قیدها <i>Constraints Types</i>	روابط قیدها <i>Constraints Relations</i>	متغیرها / دامنه‌ها <i>Variables / Domains</i>	
ضروری <i>Mandatory</i>	خطی <i>Linear</i>	تکی <i>Unary</i>	پیوسته <i>Continuous</i>	نامتناهی <i>Infinite</i>
ترجیحی <i>Preference</i>	غیرخطی <i>Non-linear</i>	دودویی <i>Binary</i>	گسسته <i>Discrete</i>	متناهی <i>Finite</i>
		چندتایی <i>n-ary</i>		
		سراسری <i>Global</i> با هر تعداد متغیر		

قیدهای سراسری شامل هر تعداد دلخواهی از متغیرها هستند.

مانند قید مربوط به متفاوت بودن متغیرهای مسئله‌ی «حساب رمزی» مثل $Alldiff(F, T, U, W, R, O)$

انواع «مسائل ارضای قید»

CSP با قیدهای ترجیحی (قید نرم)

PREFERENCES (SOFT CONSTRAINT)

نوع قیدها <i>Constraints Types</i>	نوع قیدها <i>Constraints Types</i>	روابط قیدها <i>Constraints Relations</i>	متغیرها / دامنه‌ها <i>Variables / Domains</i>	
ضروری <i>Mandatory</i>	خطی <i>Linear</i>	تکی <i>Unary</i>	پیوسته <i>Continuous</i>	نامتناهی <i>Infinite</i>
ترجیحی <i>Preference</i>	غیرخطی <i>Non-linear</i>	دودویی <i>Binary</i>	گسسته <i>Discrete</i>	متناهی <i>Finite</i>
		چندتایی <i>n-ary</i>		
		سراسری <i>Global</i> با هر تعداد متغیر		

قیدهای ترجیحی (ترجیحات، اولویت‌ها، قیدهای نرم)

مثل: *red* بهتر از *green* است.

با استفاده از هزینه‌ی انتساب یک مقدار به هر متغیر قابل بیان است



مسئله‌های بهینه‌سازی قید / مقید
Constraint Optimization Problems (COP)

مسئله‌ی «حساب رمزی»

CRYPTARITHMETIC

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

انتساب رقم‌های متفاوت به حروف متفاوت برای درست شدن عبارت حسابی

مسئله‌ی ارضای قید

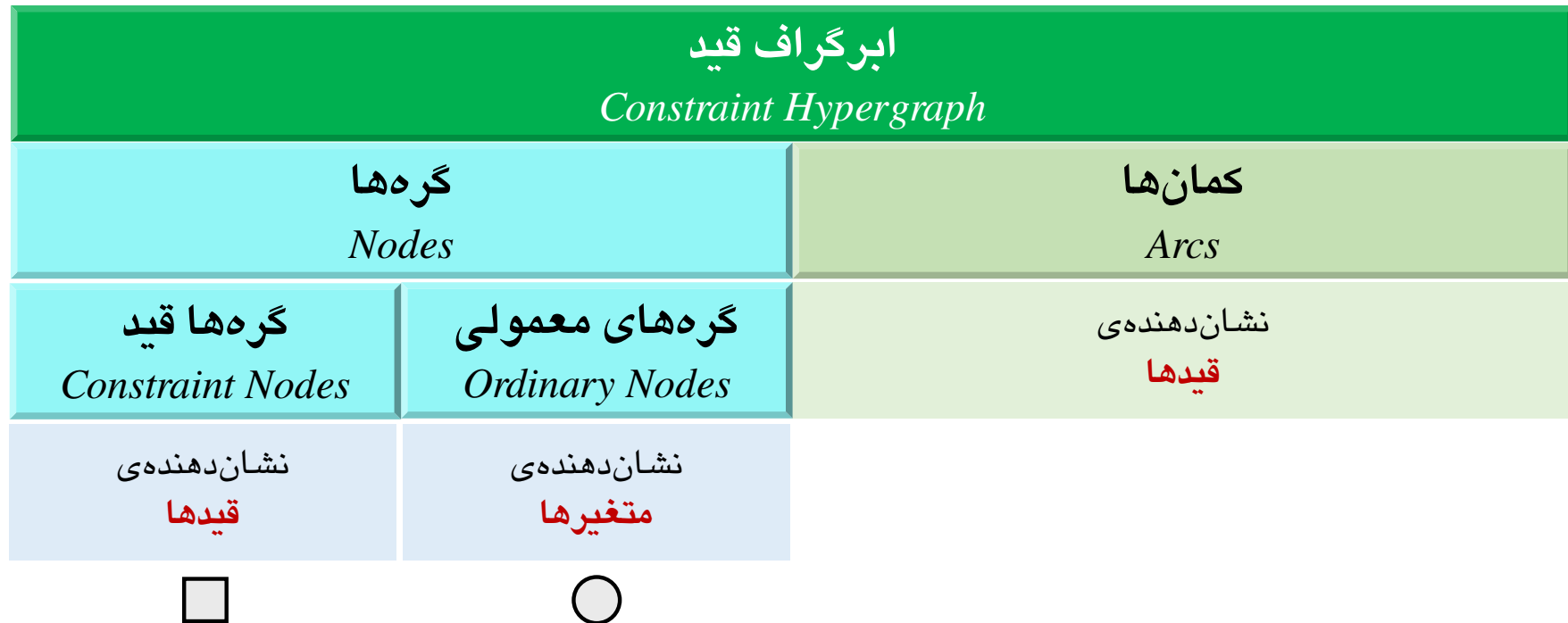
مثال: مسئله‌ی حساب رمزی

$$\begin{array}{r}
 X_3 \quad X_2 \quad X_1 \\
 \text{TWO} \\
 + \text{TWO} \\
 \hline
 \text{FOUR}
 \end{array}$$

$F, T, U, W, R, O, X_1, X_2, X_3$	حروف	X	متغیرها Variables
$D_i = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$	ارقام	D	دامنه‌ها Domains
درست بودن رابطه‌ی حسابی و متفاوت بودن رقم منتسب به همه‌ی حروف $O + O = R + 10 \cdot X_1$ $X_1 + W + W = U + 10 \cdot X_2$ $X_2 + T + T = O + 10 \cdot X_3$ $X_3 = F$ $Alldiff(F, T, U, W, R, O)$		C	قیدها Constraints

مؤلفه‌های تعریف مسئله‌ی «ارضای قید»

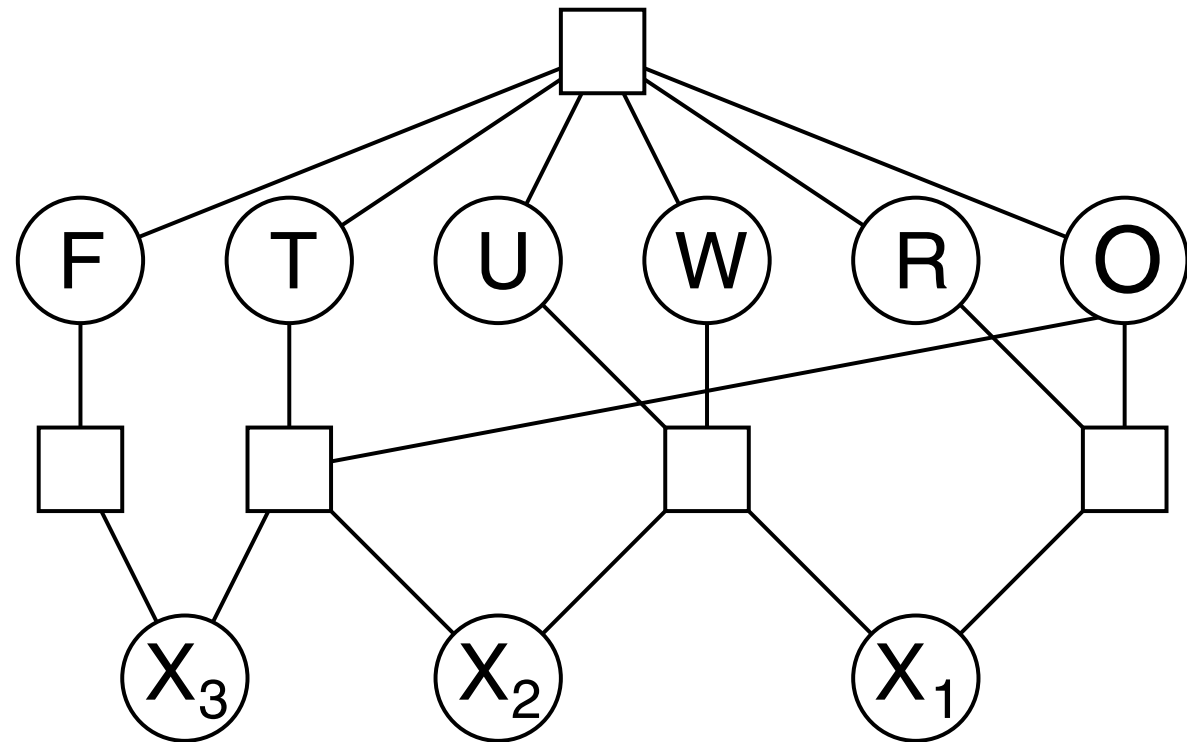
ابرگراف قید

CONSTRAINT HYPERGRAPH

قابل استفاده برای CSP چندتایی

ابرگراف قید

مثال

CONSTRAINT HYPERGRAPH

$$\begin{array}{r}
 \text{TWO} \\
 + \text{TWO} \\
 \hline
 \text{FOUR}
 \end{array}$$

کاربردهای «مسائل ارضای قید» در دنیای واقعی

مثال

مسائل جدول زمان بندی

Timetabling Problems

مثلا، کدام کلاس چه وقت و کجا برگزار شود؟

مسائل انتساب

Assignment Problems

مثلا، چه کسی به چه کلاسی درس بدهد؟

زمان بندی حمل و نقل

Transportation Scheduling

مثلا، کدام وسیله چه وقت از کجا به کجا برود؟

پیکربندی سخت افزار

Hardware Configuration

مثلا، چه ترکیبی از قطعات در کامپیوتر به کار رود؟

زمان بندی تولید کارخانه

Factory Scheduling

مثلا، کدام خط تولید در چه مدتی چه چیزی تولید کند؟

طرح ریزی کف

Floorplanning

مثلا، چه ترکیبی از موزائیک ها برای پوشش کف به کار رود؟

۲

انتشار قید: استنتاج در مسائل ارضای قید

انتشار قید: استنتاج در مسائل ارضای قید

CONSTRAINT PROPAGATION: INFERENCE IN CSPs

جستجو برای مسائل ارضای قید

امکان

جستجو

(انتخاب یک انتساب متغیر از میان امکان‌های متعدد)

+

نوعی استنتاج

(انتشار قید)

(استفاده از قیدها برای

کاهش تعداد مقادیر مجاز برای یک متغیر به صورت آبشاری)

جستجوی معمولی فضای حالت

فقط امکان

جستجوی محض

به کارگیری «انتشار قید»

همزمان با جستجو

پیش از جستجو

به عنوان گام پیش پردازش

گاهی این پیش پردازش می تواند کل مسئله را حل کند، بدون اینکه جستجویی انجام شود.

انتشار قید با اعمال «سازگاری محلی»

گراف قید	
هر متغیر	هر قید
یک گره در گراف	یک کمان در گراف

فرآیند اعمال «سازگاری محلی» در هر جزء گراف، موجب می شود مقادیر ناسازگار از سراسر گراف حذف شود.

سازگاری محلی *Local Consistency*

k -سازگاری
k-Consistency

سازگاری مسیری
Path Consistency

سازگاری کمانی
Arc Consistency

سازگاری گره‌ای
Node Consistency

انتشار قید با اعمال «سازگاری محلی»

سازگاری گره‌ای

سازگاری گره‌ای
Node Consistency

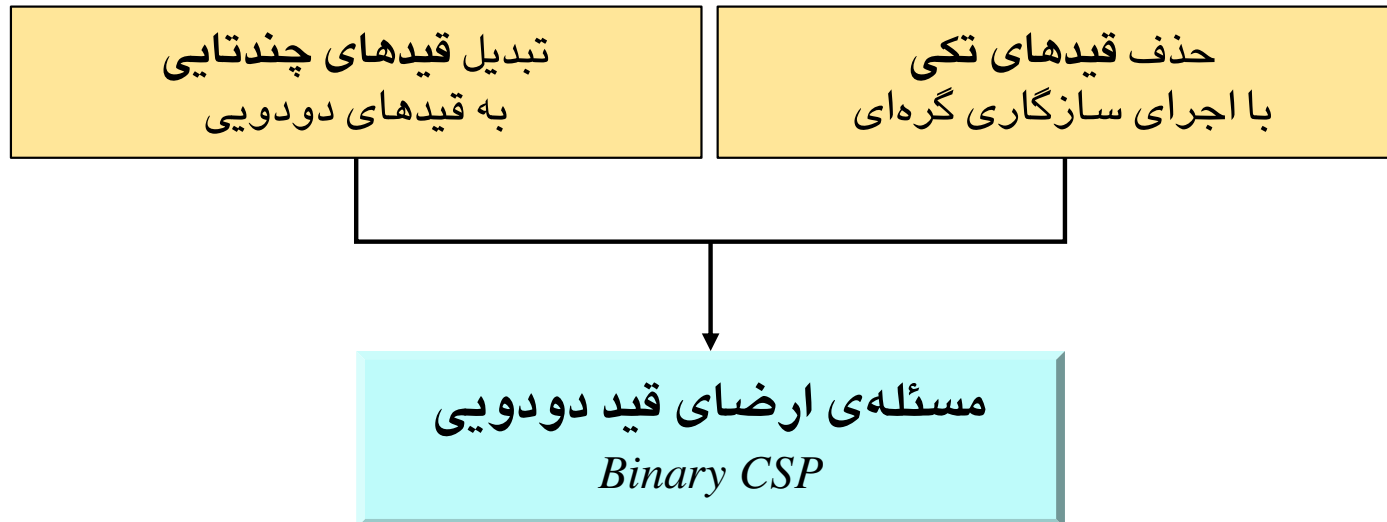
یک متغیر، سازگار گره‌ای است اگر تمام مقادیر در دامنه‌ی آن متغیر، قیده‌ای تکی آن متغیر را ارضا کند.

$$SA \neq green$$

مثال:

یک شبکه سازگار گره‌ای است اگر همه‌ی متغیرها در آن شبکه سازگار گره‌ای باشد.

تبدیل هر CSP دلخواه به یک CSP دودویی



⇐ متداول است حل‌کننده‌های CSP را به گونه‌ای تعریف کنند که تنها با قیدهای دودویی (گراف قید) کار کنند.

انتشار قید با اعمال «سازگاری محلی»

سازگاری کمانی

سازگاری کمانی

Arc Consistency

یک متغیر، سازگار کمانی است اگر
هر مقدار در دامنه‌ی آن، قیدهای دودویی آن متغیر را ارضا کند.

متغیر X_i نسبت به متغیر دیگر X_j سازگار کمانی است اگر
برای هر مقدار در دامنه‌ی فعلی D_i مقداری در دامنه‌ی D_j وجود داشته باشد که
قید دودویی بر روی کمان (X_i, X_j) را ارضا کند.

یک شبکه سازگار کمانی است اگر هر متغیر در آن شبکه نسبت به همه‌ی متغیرهای دیگر سازگار کمانی باشد.

انتشار قید با اعمال «سازگاری محلی»

سازگاری کمانی: مثال

سازگاری کمانی
Arc Consistency

یک متغیر، سازگار کمانی است اگر
هر مقدار در دامنه‌ی آن، قیدهای دودویی آن متغیر را ارضا کند.

قید $Y = X^2$ که در آن دامنه‌ی X و Y هر دو مجموعه‌ی ارقام باشد.

$$\langle (X, Y), \{(0, 0), (1, 1), (2, 4), (3, 9)\} \rangle$$

- سازگار کردن کمانی X نسبت به Y : کاهش دامنه‌ی X به $\{0, 1, 2, 3\}$
- سازگار کردن کمانی Y نسبت به X : کاهش دامنه‌ی Y به $\{0, 1, 4, 9\}$

کل شبکه‌ی CSP سازگار کمانی شد، چون همه‌ی متغیرهای آن نسبت به یکدیگر سازگار کمانی شدند.

انتشار قید با اعمال «سازگاری محلی»

سازگاری کمانی: مثال

سازگاری کمانی
Arc Consistency

یک متغیر، سازگار کمانی است اگر
هر مقدار در دامنه‌ی آن، قیدهای دودویی آن متغیر را ارضا کند.

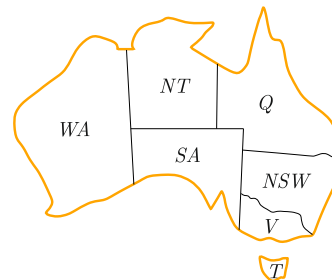
سازگاری کمانی برای مسئله‌ی «رنگ‌آمیزی نقشه‌ی استرالیا» کاری انجام نمی‌دهد، مثلاً در قید:

$$SA \neq WA$$

هر مقداری برای یک متغیر انتخاب شود، مقدار مجازی برای دیگری باقی می‌ماند.



اعمال سازگاری کمانی هیچ اثری بر دامنه‌ی هیچ متغیری نمی‌گذارد.



انتشار قید با اعمال «سازگاری محلی»

سازگاری کمانی: الگوریتم AC-3 برای سازگارسازی کمانی همه‌ی متغیرها

الگوریتم AC-3 یک صف از کمان‌ها برای بررسی ایجاد می‌کند.

این صف در ابتدا شامل همه‌ی کمان‌ها در CSP است.

سپس یک کمان دلخواه (X_i, X_j) از صف برداشته می‌شود و X_i نسبت به X_j سازگار کمانی می‌شود.

- اگر دامنه‌ی D_i تغییری نکرد $\Leftarrow$ الگوریتم فقط به سراغ کمان بعدی می‌رود.
 - اگر دامنه‌ی D_i تغییر کرد (Revise) $\Leftarrow$ تمام کمان‌های (X_k, X_i) را به صف می‌افزاییم (X_k همسایه‌ی X_i است).
 - اگر دامنه‌ی D_i به تهی کاهش یابد $\Leftarrow$ CSP راه‌حل سازگار ندارد و AC-3 بلافاصله failure برمی‌گرداند.
- بررسی‌ها ادامه می‌یابد:

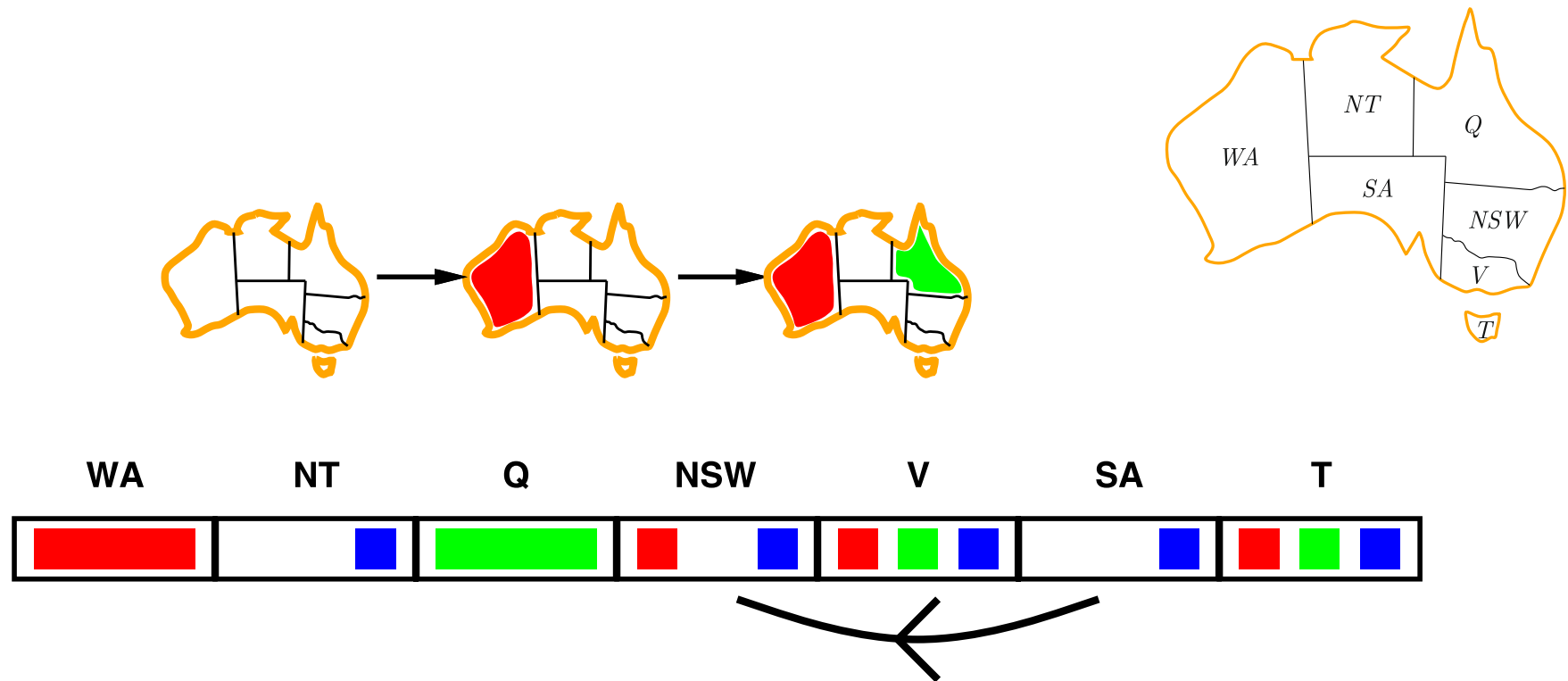
به مرور مقادیری از دامنه‌ی متغیرها حذف می‌شود تا اینکه دیگر کمانی در صف باقی نماند.

* در اینجا به یک CSP معادل با CSP اصلی می‌رسیم:

اما CSP سازگار کمانی معمولاً برای جستجو سریع‌تر است، زیرا دامنه‌ی متغیرها کوچک‌تر است.

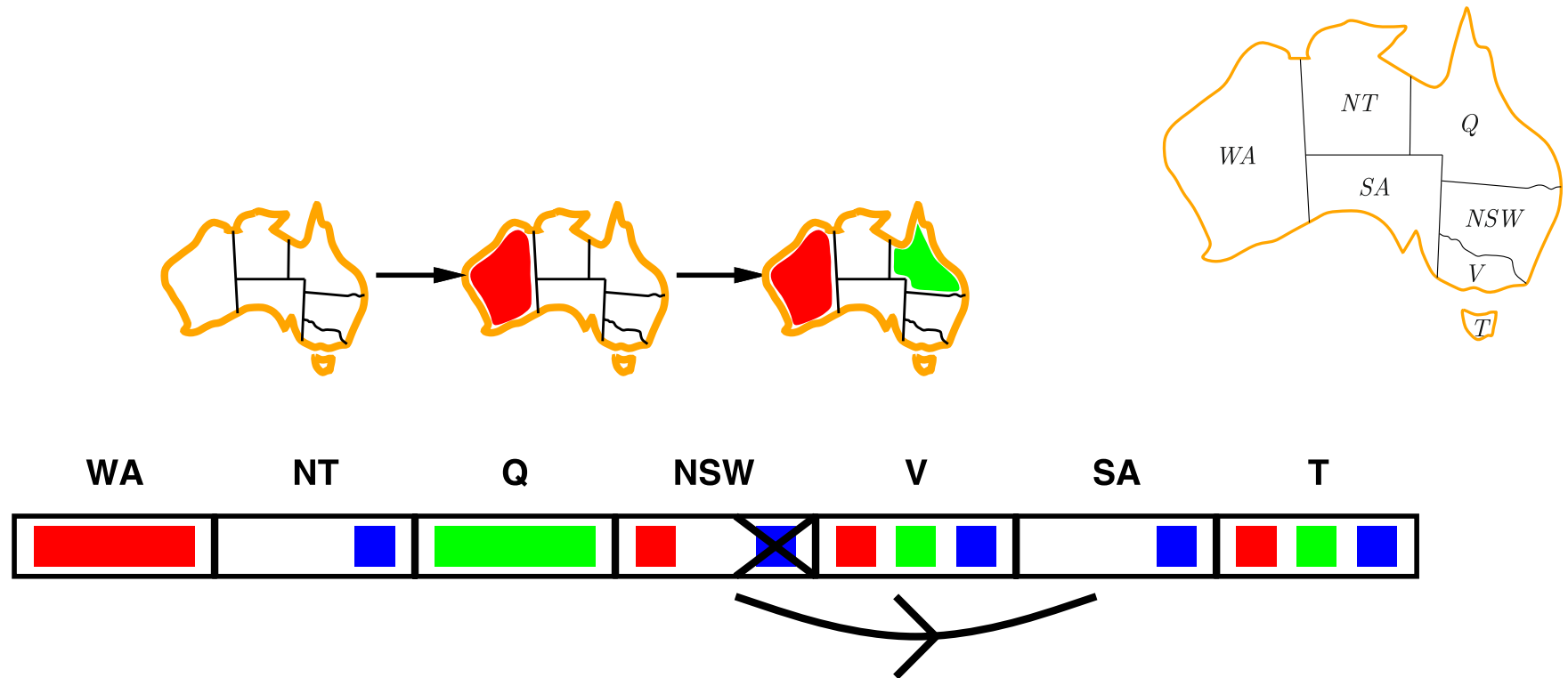
انتشار قید با اعمال «سازگاری محلی»

سازگاری کمانی: مثال (۱ از ۴)

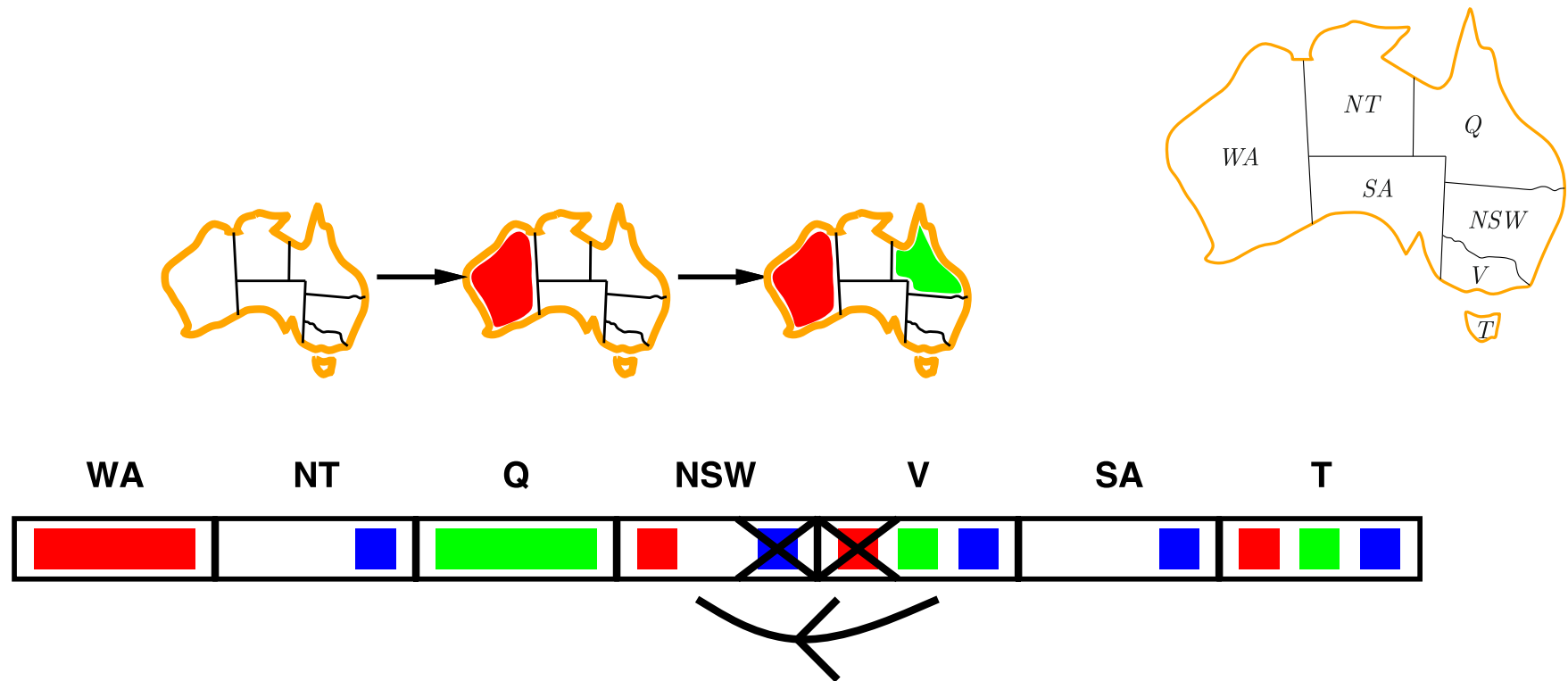


انتشار قید با اعمال «سازگاری محلی»

سازگاری کمانی: مثال (۲ از ۴)



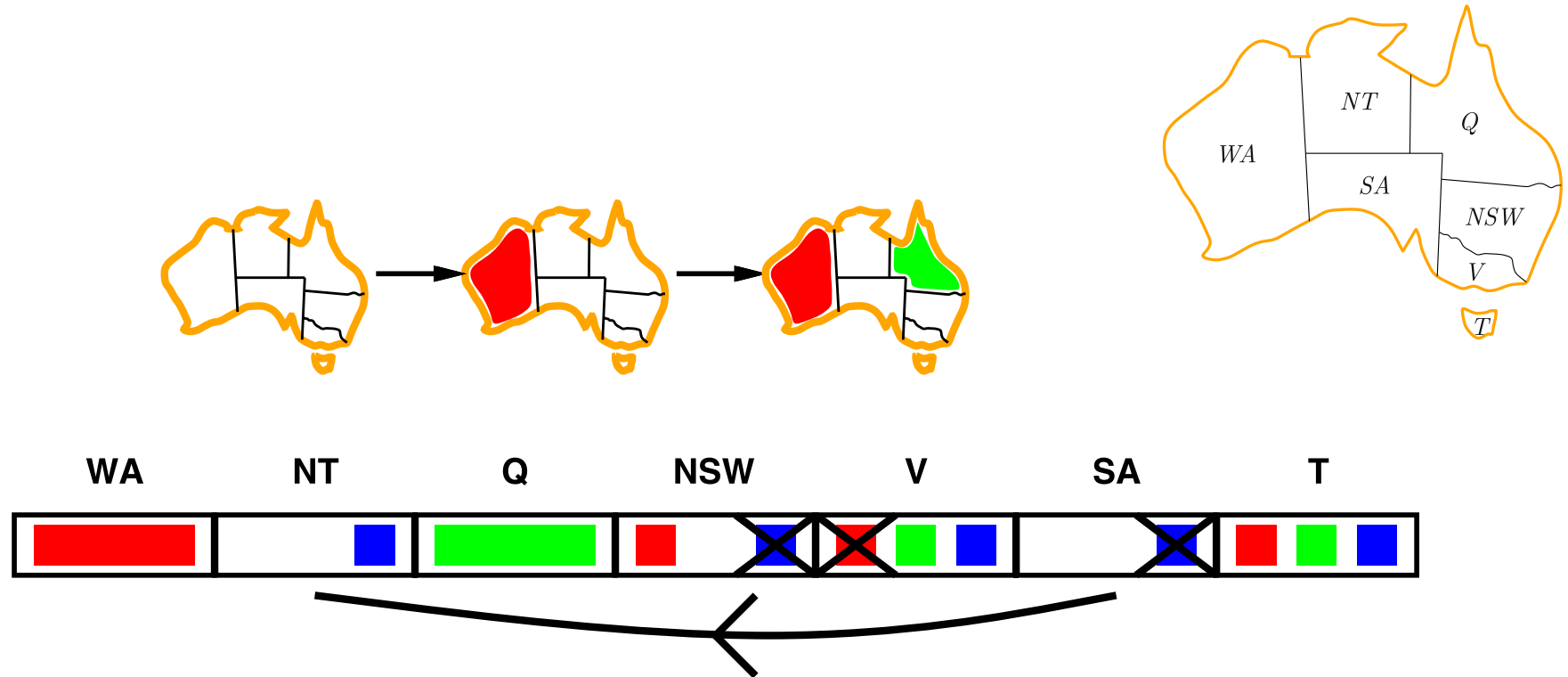
سازگاری کمانی: مثال (۳ از ۴)



اگر X مقداری را از دست بدهد، لازم است همسایه‌های X دوباره بررسی شوند.

انتشار قید با اعمال «سازگاری محلی»

سازگاری کمانی: مثال (۴ از ۴)



اگر X مقداری را از دست بدهد، لازم است همسایه‌های X دوباره بررسی شوند.

انتشار قید با اعمال «سازگاری محلی»

سازگاری کمانی: الگوریتم AC-3 برای سازگارسازی کمانی همه‌ی متغیرها: شبه‌کد

function AC-3(csp) **returns** false if an inconsistency is found and true otherwise

inputs: csp , a binary CSP with components (X, D, C)

local variables: $queue$, a queue of arcs, initially all the arcs in csp

while $queue$ is not empty **do**

$(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(queue)$

if REVISE(csp, X_i, X_j) **then**

if size of $D_i = 0$ **then return** false

for each X_k **in** $X_i.\text{NEIGHBORS} - \{X_j\}$ **do**

 add (X_k, X_i) to $queue$

return true

function REVISE(csp, X_i, X_j) **returns** true iff we revise the domain of X_i

$revised \leftarrow false$

for each x **in** D_i **do**

if no value y in D_j allows (x, y) to satisfy the constraint between X_i and X_j **then**

 delete x from D_i

$revised \leftarrow true$

return revised

انتشار قید با اعمال «سازگاری محلی»

سازگاری کمانی: الگوریتم AC-3 برای سازگارسازی کمانی همه‌ی متغیرها: تحلیل پیچیدگی زمانی

یک CSP با: $\left. \begin{array}{l} n \text{ متغیر} \\ d \text{ حداکثر اندازه‌ی دامنه‌ها} \\ c \text{ حداکثر تعداد کمان‌ها (قیدهای دودویی)} \end{array} \right\}$

* هر کمان (X_i, X_j) تنها d مرتبه می‌تواند در صف درج شود
(زیرا X_i حداکثر d مقدار برای حذف شدن دارد)

* بررسی سازگاری هر کمان می‌تواند در $O(d^2)$ انجام شود.

↓

در بدترین حالت داریم $O(c d^3)$ یعنی $O(n^2 d^3)$

الگوریتم AC-4 در بدترین حالت $O(c d^2)$ یعنی $O(n^2 d^2)$ است، اما در حالت متوسط کندتر از AC-3 است.

انتشار قید با اعمال «سازگاری محلی»

سازگاری ابرکمانی (گسترش مفهوم سازگاری کمانی برای قیدهای چندتایی)

گسترش مفهوم سازگاری کمانی برای کار با قیدهای n تایی

سازگاری ابرکمانی
Hyperarc Consistency

سازگاری کمانی تعمیم‌یافته
Generalized Arc Consistency

متغیر X_i نسبت به یک قید n تایی سازگار کمانی تعمیم‌یافته است اگر برای هر مقدار v در دامنه‌ی X_i یک چندتایی از مقادیر موجود باشد که عضوی از آن قید باشد به طوری که همه‌ی مقادیر آن از دامنه‌های متغیرهای متناظر گرفته شده باشد و مؤلفه‌ی X_i آن مساوی با v باشد.

انتشار قید با اعمال «سازگاری محلی»

سازگاری ابرکمانی (گسترش مفهوم سازگاری کمانی برای قیدهای چندتایی): مثال

گسترش مفهوم سازگاری کمانی برای کار با قیدهای n تایی

سازگاری ابرکمانی
Hyperarc Consistency

سازگاری کمانی تعمیم یافته
Generalized Arc Consistency

همه ی متغیرها، دارای دامنه ی $\{0,1,2,3\}$

متغیر X سازگار با قید $X < Y < Z$ است
اگر

2 و 3 از دامنه ی X حذف شده باشد.

(زیرا هرگاه X برابر با 2 یا 3 باشد، قید نمی تواند ارضا شود.)

انتشار قید با اعمال «سازگاری محلی»

سازگاری مسیری

سازگاری مسیری
Path Consistency

قیدهای دودویی را تنگتر می‌کند: با استفاده از قیدهای ضمنی که با نگاه کردن به مجموعه‌های سه‌تایی از متغیرها استنتاج می‌شود.

مجموعه‌ی دو متغیره‌ی $\{X_i, X_j\}$ نسبت به یک متغیر سوم X_m سازگار مسیری است اگر برای انتساب $\{X_i = a, X_j = b\}$ سازگار با $\{X_i, X_j\}$ یک انتساب برای X_m وجود داشته باشد که قیدهای روی $\{X_i, X_m\}$ و $\{X_m, X_j\}$ را ارضا کند.

(سازگاری مسیری: قابل مشاهده به صورت مسیری از X_i به X_j با X_m در میان آن.)

الگوریتم PC-2 برای سازگاری مسیری

انتشار قید با اعمال «سازگاری محلی»

سازگاری مسیری: مثال

قیدهای دودویی را تنگتر می‌کند: با استفاده از قیدهای ضمنی که با نگاه کردن به مجموعه‌های سه‌تایی از متغیرها استنتاج می‌شود.

سازگاری مسیری Path Consistency

رنگ‌آمیزی نقشه‌ی استرالیا با دو رنگ توسط سازگاری مسیری:

مجموعه‌ی $\{WA, SA\}$ را نسبت به NT سازگار مسیری می‌کنیم. با شمارش انتساب‌های سازگار این مجموعه شروع می‌کنیم: در این مورد فقط دو نمونه وجود دارد:

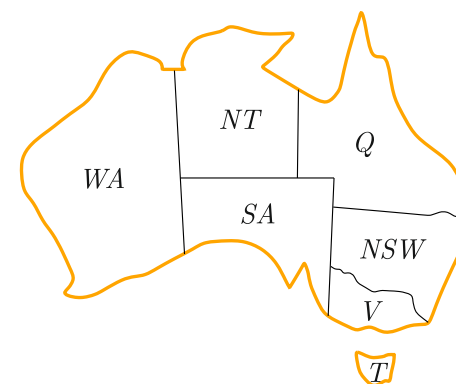
$$\{WA = red, SA = blue\}, \quad \{WA = blue, SA = red\}$$

می‌بینیم که با انتساب هر یک از این دو مورد، NT نه می‌تواند red باشد و نه $blue$.

از آنجا که هیچ گزینه‌ی معتبری برای NT وجود ندارد، هر دو انتساب حذف می‌شود و کار بدون یافتن انتساب معتبر برای $\{WA, SA\}$ خاتمه می‌یابد.



این مسئله نمی‌تواند راه‌حلی داشته باشد.



انتشار قید با اعمال «سازگاری محلی»

 k -سازگاری k -سازگاری
 k -Consistency

یک CSP، k -سازگار است اگر:
برای هر مجموعه از $k-1$ متغیر و برای هر انتساب سازگار به آن متغیرها،
همیشه یک مقدار سازگار بتواند به متغیر k -ام نسبت داده شود.

حالت‌های خاص

۱-سازگاری
1-Consistency

سازگاری گره‌ای
(با داشتن مجموعه‌ی تهی باید بتوانیم هر مجموعه از یک متغیر را سازگار کنیم)

۲-سازگاری
2-Consistency

سازگاری کمانی

۳-سازگاری
3-Consistency

سازگاری مسیری
(برای شبکه‌های قید دودویی)

انتشار قید با اعمال «سازگاری محلی»

 k -سازگاری قوی

یک CSP، k -سازگار قوی است اگر:
 k -سازگار، $k-1$ -سازگار، $k-2$ -سازگار، $\dots$ ، 1 -سازگار باشد.

k -سازگاری قوی
Strong k -Consistency

انتشار قید با اعمال «سازگاری محلی»

الگوریتم حل CSP در حالت n -سازگاری قوی (با n متغیر) **k -سازگاری قوی**
Strong k -Consistency

یک CSP، k -سازگار قوی است اگر:
 k -سازگار، $k-1$ -سازگار، $k-2$ -سازگار، $\dots$ ، 1 -سازگار باشد.

اگر یک CSP با n گره داشته باشیم که n -سازگار قوی باشد:

- ابتدا یک مقدار سازگار برای X_1 انتخاب می‌کنیم
- تضمین می‌شود که بتوانیم یک مقدار برای X_2 انتخاب کنیم (زیرا ۲-سازگار است).
- تضمین می‌شود که بتوانیم یک مقدار برای X_3 انتخاب کنیم (زیرا ۳-سازگار است).
- $\dots$

برای هر متغیر X_i تنها نیاز داریم در میان d مقدار در دامنه برای یافتن یک مقدار سازگار با

$$X_1, X_2, \dots, X_{i-1}$$

جستجو کنیم. پس

تضمین می‌شود که یک راه‌حل در زمان $O(n^2 d)$ پیدا شود.

البته الگوریتم ایجاد n -سازگاری در بدترین حالت زمان نمایی نیاز دارد!!

انتشار قید با اعمال «سازگاری محلی»

قیدهای سراسری

قیدهای سراسری
Global Constraints

قید سراسری،
قیدی شامل هر تعداد دلخواهی از متغیرها (و نه لزوماً همه‌ی آنها) است.

قیدهای سراسری به کرات در مسائل واقعی رخ می‌دهند
و می‌توانند با الگوریتم‌های خاص منظوره (که کارآمدتر از الگوریتم‌های همه‌منظوره هستند)، حل شوند.

انتشار قید با اعمال «سازگاری محلی»

قیدهای سراسری: مثال

قیدهای سراسری
Global Constraints

قید سراسری،
قیدی شامل هر تعداد دلخواهی از متغیرها (و نه لزوماً همه‌ی آنها) است.

قیدهای «همه متفاوت» *Alldiff*
(متفاوت بودن همه‌ی متغیرها)

مثل: مسئله‌ی حساب رمزی

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

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

انتشار قید با اعمال «سازگاری محلی»

قیدهای سراسری: نکته

قیدهای سراسری
Global Constraints

قید سراسری،
قیدی شامل هر تعداد دلخواهی از متغیرها (و نه لزوماً همه‌ی آنها) است.

اگر m متغیر در یک قید دخیل باشند و
اگر آنها مجموعاً n مقدار متمایز ممکن داشته باشند و
$$m > n$$

آن‌گاه این قید نمی‌تواند ارضا شود.

انتشار قید با اعمال «سازگاری محلی»

قیدهای سراسری: الگوریتم اعمال قید سراسری

قیدهای سراسری Global Constraints

قید سراسری،
قیدی شامل هر تعداد دلخواهی از متغیرها (و نه لزوماً همه‌ی آنها) است.

- ابتدا هر متغیر در قید که دامنه‌ی **تک عنصری** دارد را حذف کنید؛
- و مقادیر آن متغیر را از دامنه‌ی مابقی متغیرها حذف کنید.
- این کار را تکرار کنید تا زمانی که هنوز متغیرهای تکی وجود دارند.
- اگر در نقطه‌ای دامنه‌ی تهی تولید شد،
- یا تعداد متغیر بیشتری نسبت به مقادیر دامنه باقی ماند،
- آن‌گاه یک **ناسازگاری** کشف می‌شود.

انتشار قید با اعمال «سازگاری محلی»

قیدهای سراسری: الگوریتم اعمال قید سراسری: مثال

قیدهای سراسری
Global Constraints

قید سراسری،
قیدی شامل هر تعداد دلخواهی از متغیرها (و نه لزوماً همه‌ی آنها) است.

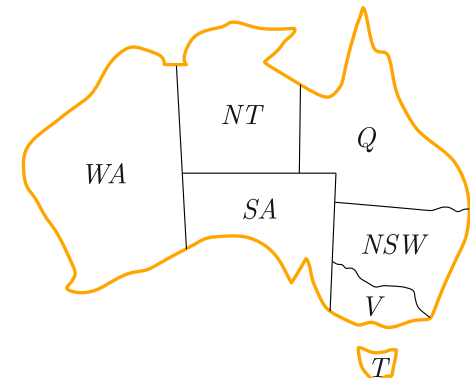
- ابتدا هر متغیر در قید که دامنه‌ی **تک عنصری** دارد را حذف کنید؛
- و مقادیر آن متغیر را از دامنه‌ی مابقی متغیرها حذف کنید.
- این کار را تکرار کنید تا زمانی که هنوز متغیرهای تکی وجود دارند.
- اگر در نقطه‌ای دامنه‌ی تهی تولید شد،
- یا تعداد متغیر بیشتری نسبت به مقادیر دامنه باقی ماند،
- آن‌گاه یک **ناسازگاری** کشف می‌شود.

مسئله‌ی رنگ‌آمیزی نقشه‌ی استرالیا با قید

$$Alldiff(SA, NT, Q)$$

یافتن ناسازگاری در انتساب $\{WA = red, NSW = red\}$

پس از اجرای AC-3 با انتساب جزئی، دامنه‌ی هر متغیر به $\{green, blue\}$ کاهش می‌یابد $\Leftarrow$ سه متغیر با دو رنگ $\Leftarrow$ قید «همه متفاوت» نقض می‌شود.



انتشار قید با اعمال «سازگاری محلی»

قیدهای مرتبه بالاتر (مثال: قید منبع، قید حداکثر)

قیدهای مرتبه بالاتر
Higher order Constraints

قید منبع *resource constraint* (مانند قید حداکثر *at most*)

مسئله‌ی زمان‌بندی

$$P_1, P_2, P_3, P_4$$

تعداد پرسنل نسبت داده شده به هر یک از چهار کار موجود

قید: در مجموع نباید بیش از ۱۰ نفر نسبت داده شود:

$$Atmost(10, P_1, P_2, P_3, P_4)$$

ناسازگاری با بررسی مجموع مقادیر می‌نیم دامنه‌های فعلی قابل تشخیص است:

- اگر همه‌ی متغیرها دامنه‌ی $\{3,4,5,6\}$ داشته باشند قید *Atmost* نمی‌تواند ارضا شود.
- اگر همه‌ی متغیرها دامنه‌ی $\{2,3,4,5,6\}$ داشته باشند و مقادیر ۵ و ۶ می‌توانند از هر دامنه حذف شوند.

(برای اعمال سازگاری: حذف مقادیر ماکزیمم از هر دامنه اگر با مقادیر می‌نیم برای دامنه‌های دیگر ناسازگار باشد)

انتشار قید با اعمال «سازگاری محلی»

انتشار کران‌ها

BOUNDS PROPAGATION

برای مسائل بزرگ منبع-محدود با مقادیر صحیح
(مانند جابجایی هزاران نفر در صدها وسیله نقلیه)

معمولاً ممکن نیست که دامنه‌ی هر متغیر به صورت یک مجموعه‌ی بزرگ از اعداد صحیح بازنمایی شود و بعد به تدریج آن مجموعه را با روش‌های بررسی سازگاری کاهش داد.

در عوض دامنه توسط **کران‌های پایین و بالا** بازنمایی می‌شوند و توسط روش **انتشار کران‌ها** مدیریت می‌شود.

انتشار قید با اعمال «سازگاری محلی»

انتشار کران‌ها: مثال

BOUNDS PROPAGATION

مسئله‌ی زمان‌بندی خط هوایی

$$F_1, F_2$$

دو پرواز با هواپیماهایی با ظرفیت ۱۶۵ نفر و ۳۸۵ نفر وجود دارد.

دامنه‌ی اولیه برای مسافران هر پرواز می‌شود:

$$D_1 = [0, 165], \quad D_2 = [0, 385]$$

با یک قید اضافی:

دو پرواز باید روی هم ۴۲۰ نفر را جابجا کنند:

$$F_1 + F_2 = 420$$

انتشار قیدهای کران، دامنه‌ها را به صورت زیر کاهش می‌دهند:

$$D_1 = [35, 165], \quad D_2 = [255, 385]$$

انتشار قید با اعمال «سازگاری محلی»

سازگاری کرانی

یک CSP سازگار کرانی است اگر
برای هر متغیر X و برای هر دو کران پایین و بالای آن
مقداری برای متغیر Y وجود داشته باشد که
قید میان X و Y را برای هر متغیر Y ارضا کند.

سازگاری کرانی
Bounds Consistency

مسئله‌ی ارضای قید

مثال: معمای سودوکو

SUDOKU PUZZLE

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

۸۱ متغیر (هر مربع یک متغیر)

X

متغیرها
Variablesدامنه‌ی مربع خالی: $\{0,1,2,3,4,5,6,7,8,9\}$
دامنه‌ی مربع پرشده: $\{\text{تک عضوی (یک رقم)}\}$

D

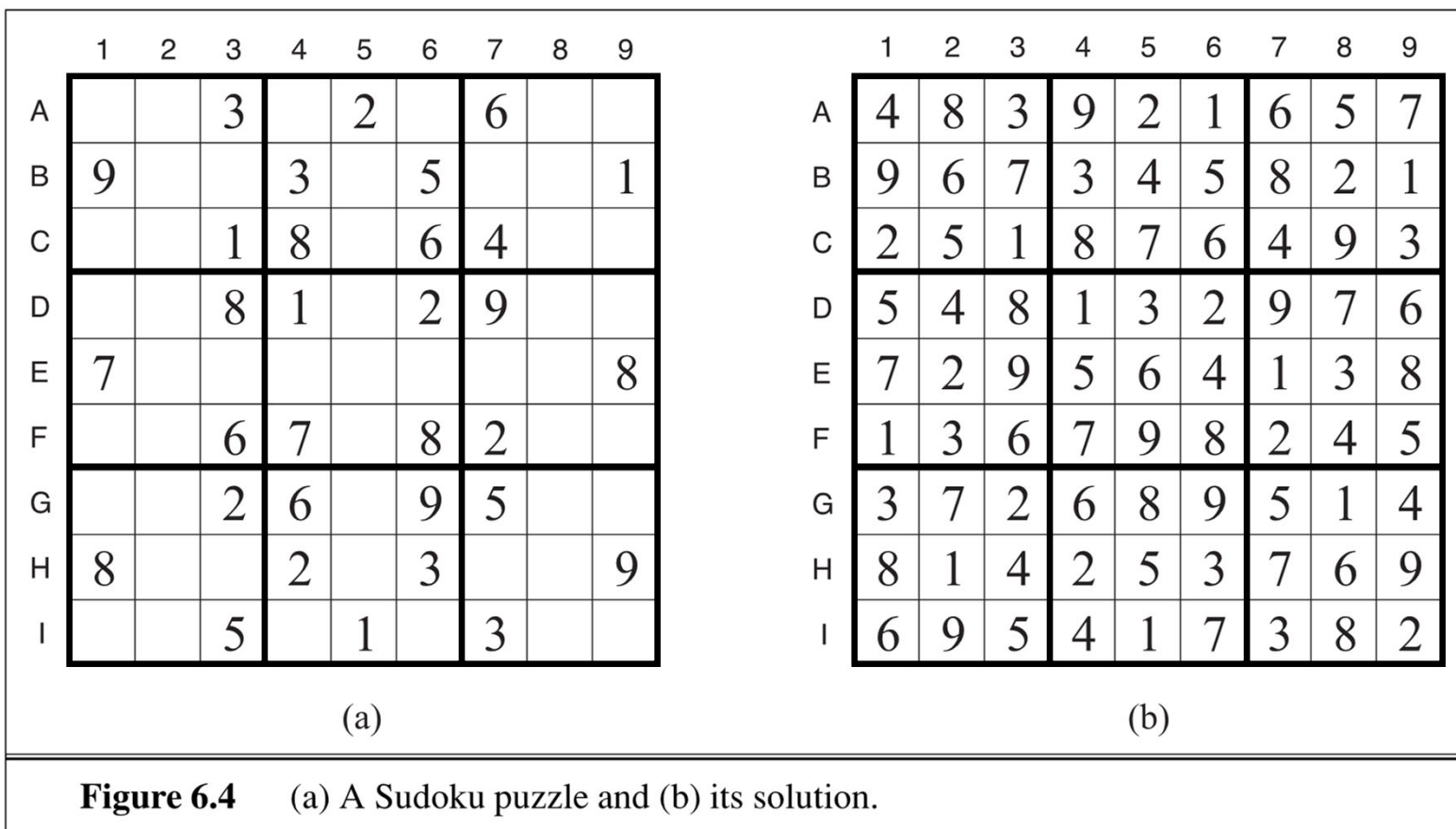
دامنه‌ها
Domains۲۷ قید *Alldiff*سطر، ستون یا چهارگوش:
یک واحد (unit)

- مقادیر ستون‌های هر سطر باید متفاوت باشند.
- مقادیر سطرهای هر ستون باید متفاوت باشند.
- مقادیر هر چهارگوش ۳ در ۳ باید متفاوت باشند.

C

قیدها
Constraints

مؤلفه‌های تعریف مسئله‌ی «ارضای قید»



Alldiff constraints: one for each row, column, and box of 9 squares.

$Alldiff(A1, A2, A3, A4, A5, A6, A7, A8, A9)$
 $Alldiff(B1, B2, B3, B4, B5, B6, B7, B8, B9)$
 $\dots$
 $Alldiff(A1, B1, C1, D1, E1, F1, G1, H1, I1)$
 $Alldiff(A2, B2, C2, D2, E2, F2, G2, H2, I2)$
 $\dots$
 $Alldiff(A1, A2, A3, B1, B2, B3, C1, C2, C3)$
 $Alldiff(A4, A5, A6, B4, B5, B6, C4, C5, C6)$
 $\dots$

۳

جستجوی
عقب‌گرد
برای
مسائل
ارضای قید

فرمول‌بندی «مسئله‌ی ارضای قید» در قالب جستجوی استاندارد

حالت‌ها States	بر اساس مقادیری که تاکنون به متغیرها نسبت داده شده‌اند
حالت آغازین Initial state	انتساب تهی $\{\}$ (هیچ متغیری نسبت داده نشده است).
کنش‌های ممکن Available actions	انتساب یک مقدار به یک متغیر نسبت‌دهی نشده (که با انتساب جاری تداخل نداشته باشد).
مدل گذر Transition model	از یک انتساب «جاری» به یک انتساب «جدید»
آزمون هدف Goal test	ضمنی (implicit): انتساب جاری کامل باشد (و همه‌ی قیدها لحاظ شده باشند).
تابع هزینه‌ی مسیر Path cost function	مسیر راه‌حل مهم نیست $\Leftarrow$ همه‌ی راه‌حل‌ها هزینه‌ی مساوی دارند.

مؤلفه‌های پنج‌گانه‌ی تعریف مسئله

این فرمول‌بندی برای تمام مسائل ارضای قید مشابه است.
هر راه‌حل با n متغیر در عمق n ظاهر می‌شود $\Leftarrow$ استفاده از جستجوی عمق-اول
مسیر راه‌حل لازم نیست $\Leftarrow$ می‌توان از فرمول‌بندی حالت-کامل استفاده کرد.
پیچیدگی زمانی بدتر از نمایی:
در عمق ℓ داریم $b = (n - \ell)d$ بنابراین $n!d^n$ برگ داریم !!! $d = \max_i D_i $

ویژگی‌ها

جستجوی استاندارد برای حل «مسائل ارضای قید»

پیاده‌سازی

درخت جستجو برای CSP شامل گره‌هایی است که هر گره یک **CSP-STATE** را نگهداری می‌کند.

datatype CSP-STATE

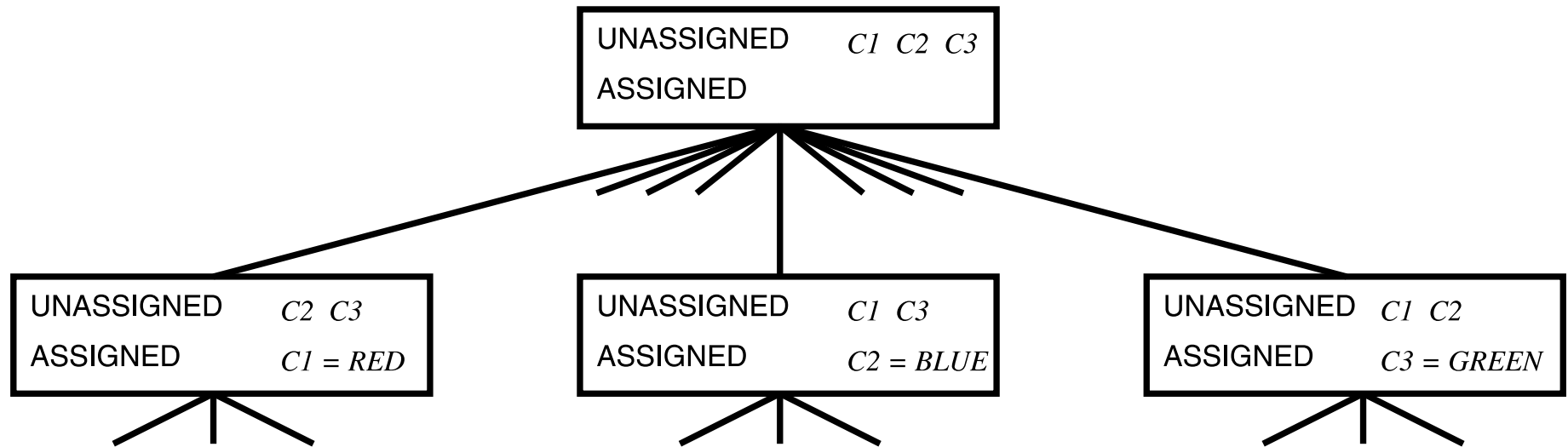
component: UNASSIGNED, a list of variables not yet assigned
ASSIGNED, a list of variables that have valued

datatype CSP-VAR

component: NAME, for i/o purposes
DOMAIN, a list of possible values
VALUE, current value (if any)

جستجوی استاندارد برای حل «مسائل ارضای قید»

پیاده‌سازی: مثال



جستجوی استاندارد برای حل «مسائل ارضای قید»

ارزیابی کارایی

۱	۲	۳	۴
حداکثر عمق فضا m	عمق درخت راه حل d	فاکتور انشعاب b	استراتژی جستجو <i>Search Strategy</i>

تعداد متغیرها:

$$m = n$$

باید تمامی متغیرها
نسبت دهی شوند:

$$d = n$$

بیشترین مقدار در
ریشه‌ی درخت:

$$b = \sum_{i=1}^n |D_i|$$

جستجوی عمق-اول
DFS

حداکثر تعداد انشعاب یک گره در درخت جستجو	<i>Branching factor</i>	فاکتور انشعاب حداکثر	b	اندازه‌گیری پیچیدگی‌های زمانی و فضایی با پارامترهای
عمق کم عمق‌ترین هدف	<i>depth of l.c. solution</i>	عمق راه حل با کم‌ترین هزینه	d	
حداکثر طول یک مسیر در گراف فضای حالت (شاید ∞)	<i>Max. depth of s.s.</i>	حداکثر عمق فضای حالت	m	

«جستجوی عقب‌گرد» برای حل مسائل ارضای قید

BACKTRACKING SEARCH FOR CSPs

جستجوی عقب‌گرد: جستجوی عمق-اول برای CSP با نسبت‌دهی تک-متغیر در هر گره

انتساب متغیرها جابجاپذیر (commutative) است یعنی

$$[WA = red \text{ then } NT = green] \equiv [NT = green \text{ then } WA = red]$$

بنابراین، لازم است انتساب تنها یک متغیر در هر گره بررسی شود.

↓

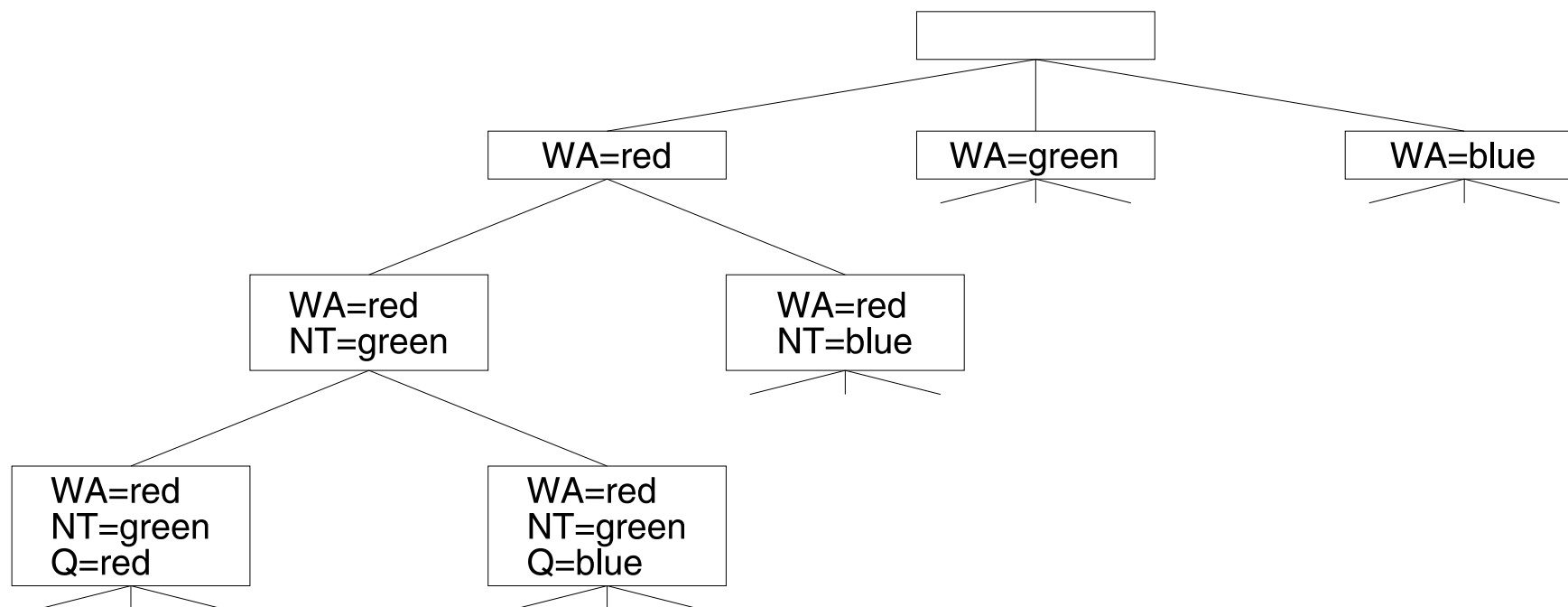
$$b = d \quad (= \max_i |D_i|)$$

برگ d^n

- الگوریتم ناآگاهانه پایه برای CSPها
- قابلیت حل مسئله‌ی n وزیر برای $n \approx 25$

«جستجوی عقب‌گرد» برای حل مسائل ارضای قید

مثال



«جستجوی عقب‌گرد» برای حل مسائل ارضای قید

شبکه‌کد

```

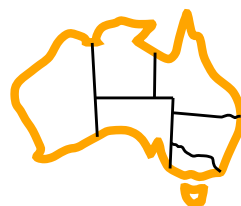
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return BACKTRACK({ }, csp)

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

```

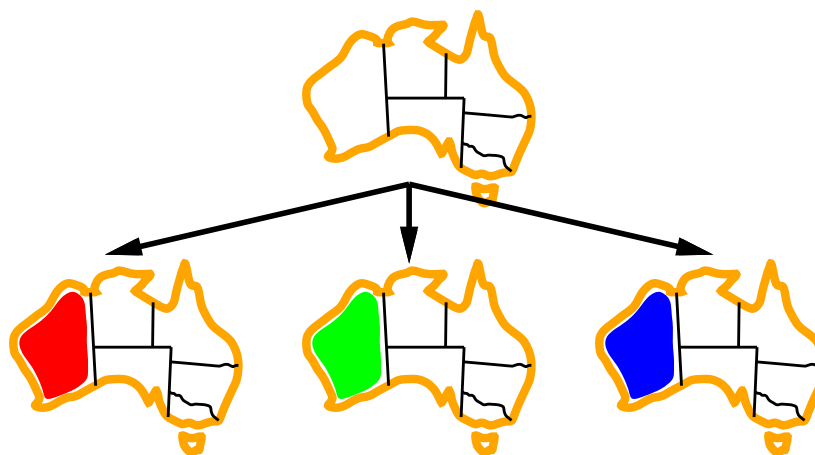
«جستجوی عقب‌گرد» برای حل مسائل ارضای قید

مثال (۱ از ۴)



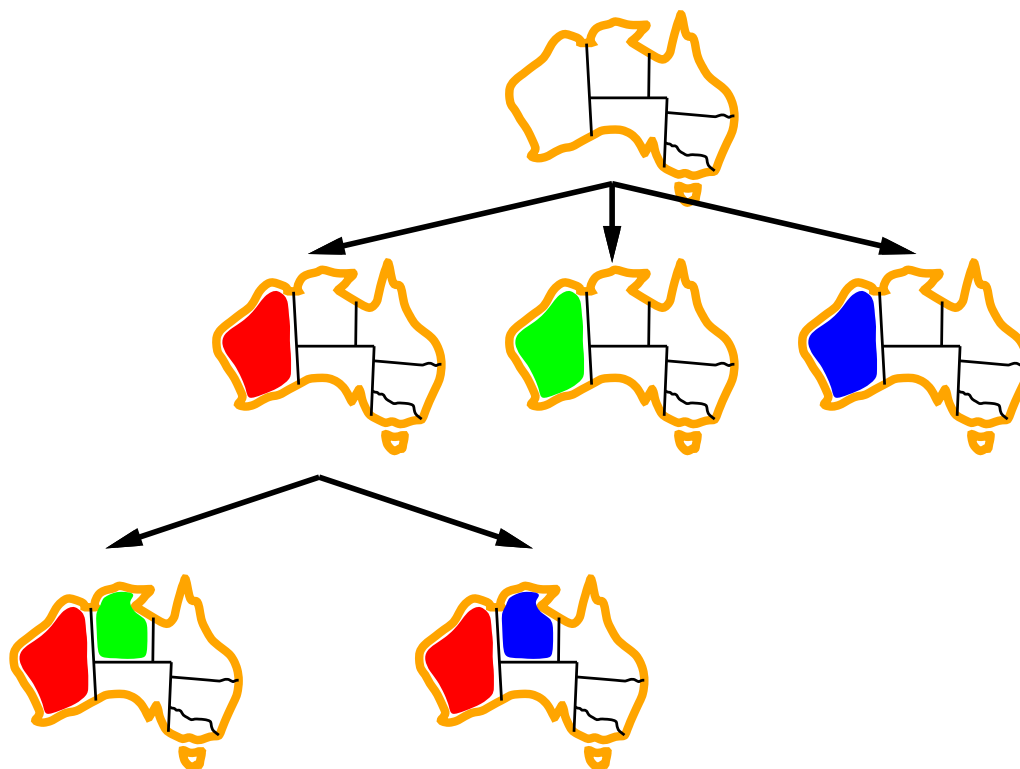
«جستجوی عقب‌گرد» برای حل مسائل ارضای قید

مثال (۲ از ۴)



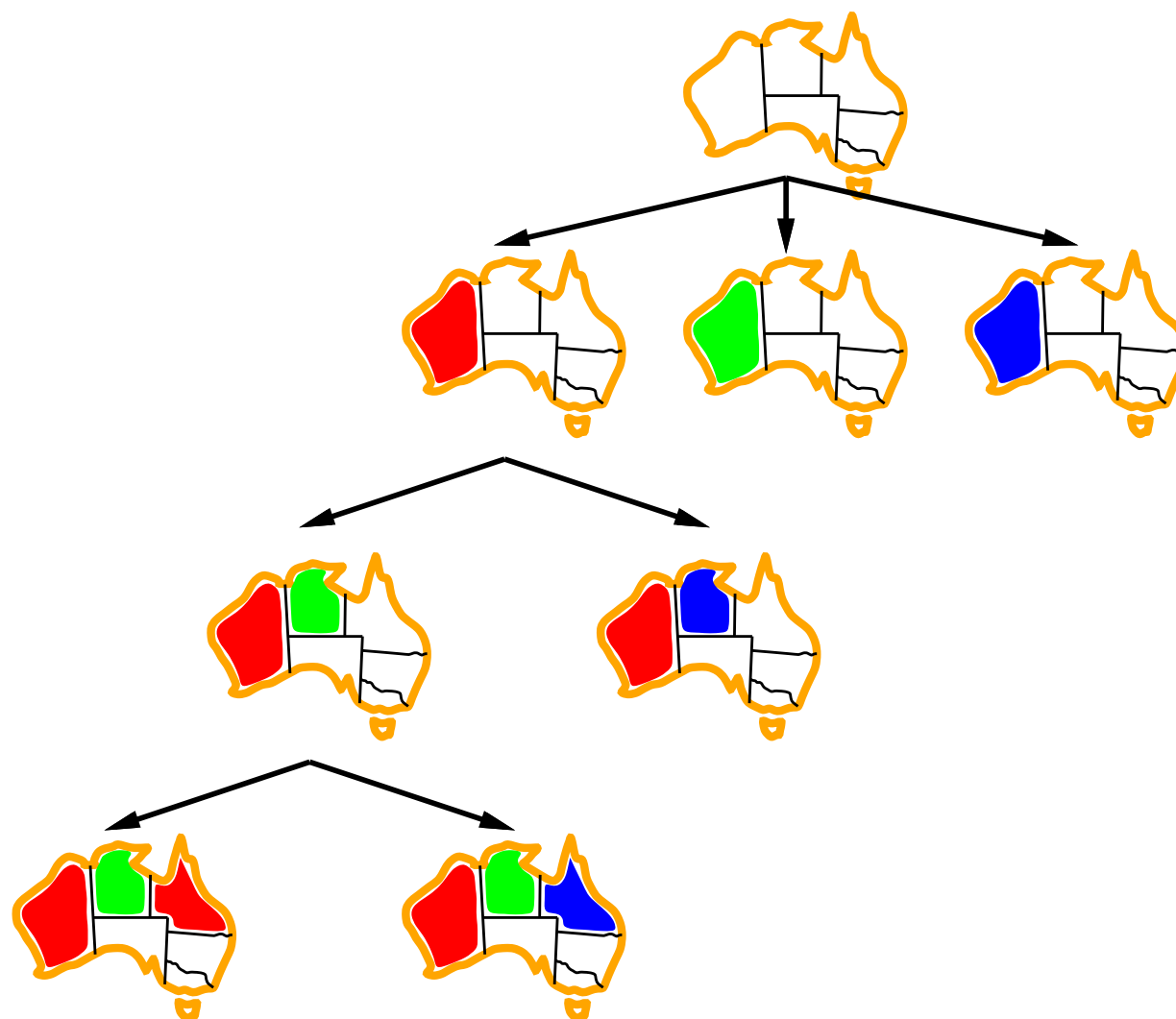
«جستجوی عقب‌گرد» برای حل مسائل ارضای قید

مثال (۳ از ۴)



«جستجوی عقب‌گرد» برای حل مسائل ارضای قید

مثال (۴ از ۴)



بهبود کارایی جستجوی عقب‌گرد

برای حل مسائل ارضای قید

CSP از مزیت همه‌منظوره بودن بهره می‌برد.
اما بهبودهای خاصی را می‌توان از طریق توابع مشخص نشده در الگوریتم کلی عقب‌گرد ایجاد کرد:

کدام متغیر باید در مرحله‌ی بعدی نسبت‌دهی شود؟

SELECT-UNASSIGNED-VARIABLE

۱

مقادیر باید به چه ترتیبی آزمایش شوند؟

ORDER-DOMAIN-VALUES

۲

چه استنتاج‌هایی باید در هر گام از جستجو انجام شود؟

INFERENCE

۳

چگونگی اجتناب از شکست مجدد، وقتی جستجو به انتسابی شامل نقض یک قید برمی‌خورد؟

۴

بهبود کارایی جستجوی عقب‌گرد

ترتیب‌دهی متغیرها و مقادیر: ترفند «حداقل مقادیر باقیمانده»

متغیری را انتخاب کنید که
دارای کمترین تعداد مقادیر مجاز است

حداقل مقادیر باقیمانده
Minimum Remaining Values (MRV)

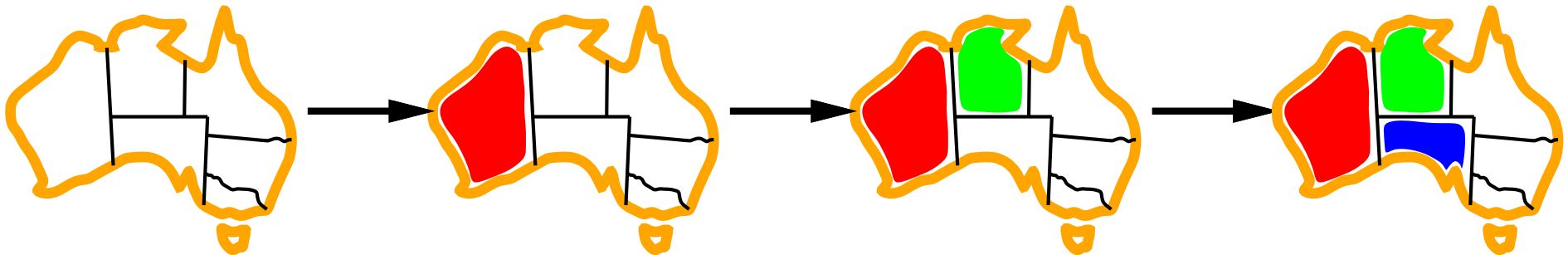
متغیر با بیشترین قید
Most Constrained Variable

بهبود کارایی جستجوی عقب‌گرد

ترتیب‌دهی متغیرها و مقادیر: ترفند «حداقل مقادیر باقیمانده»: مثال

متغیری را انتخاب کنید که
دارای کمترین تعداد مقادیر مجاز است

حداقل مقادیر باقیمانده
Minimum Remaining Values (MRV)



بهبود کارآیی جستجوی عقب‌گرد

ترتیب‌دهی متغیرها و مقادیر: ترفند «هیوریستیک درجه»

هیوریستیک درجه
Degree Heuristic

متغیری را انتخاب کنید که
بیشترین قید را بر روی متغیرهای باقیمانده ایجاد می‌کند

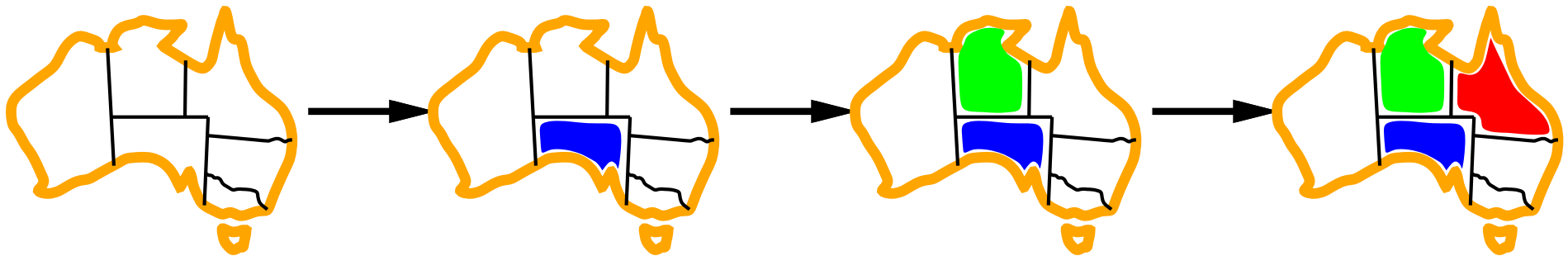
حل مشکل MRV در تعیین اولین متغیر (همه‌ی متغیرها در شروع کار معمولاً به یک اندازه قید دارند).

بهبود کارایی جستجوی عقب‌گرد

ترتیب‌دهی متغیرها و مقادیر: ترفند «هیوریستیک درجه»: مثال

هیوریستیک درجه
Degree Heuristic

متغیری را انتخاب کنید که
بیشترین قید را بر روی متغیرهای باقیمانده ایجاد می‌کند



بهبود کارآیی جستجوی عقب‌گرد

ترتیب‌دهی متغیرها و مقادیر: ترفند «مقدار با کمترین ایجاد قید»

با داشتن یک متغیر، مقداری را انتخاب کنید که
کمترین تعداد مقادیر را برای متغیرهای باقیمانده رد می‌کند.

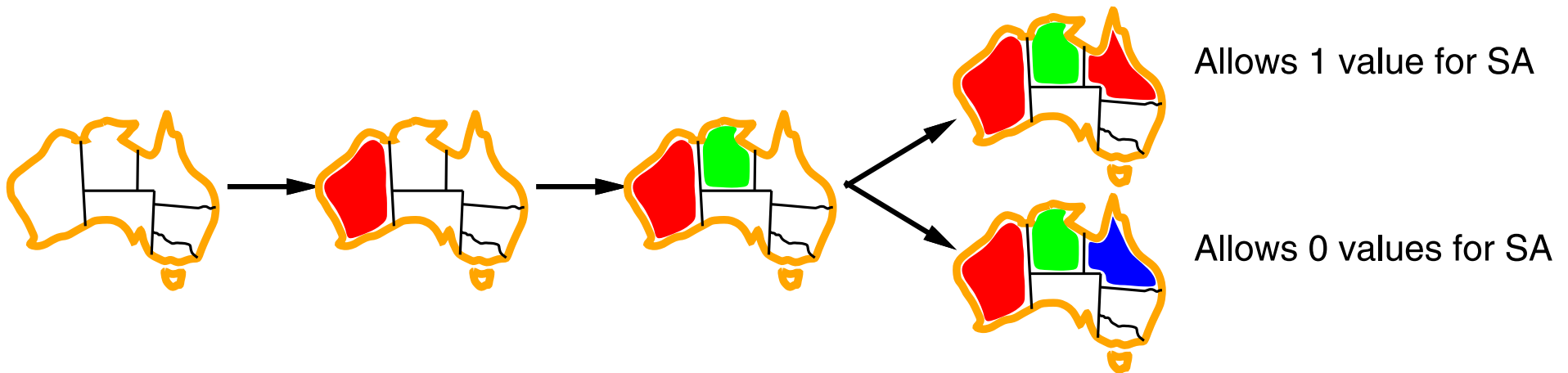
مقدار با کمترین ایجاد قید
Least Constraining Value

بهبود کارایی جستجوی عقب‌گرد

ترتیب‌دهی متغیرها و مقادیر: ترفند «مقدار با کمترین ایجاد قید»: مثال

با داشتن یک متغیر، مقداری را انتخاب کنید که کمترین تعداد مقادیر را برای متغیرهای باقیمانده رد می‌کند.

مقدار با کمترین ایجاد قید
Least Constraining Value



با ترکیب ترفندهای اشاره شده: امکان حل مسئله‌ی n وزیر برای $n \approx 1000$

بهبود کارایی جستجوی عقب‌گرد

«یک در میان» سازی جستجو و استنتاج

الگوریتم‌های استنتاج، می‌توانستند پیش از شروع جستجو، دامنه‌ی متغیرها را کاهش بدهند.

اما استنتاج در خلال جستجو می‌تواند قدرت‌مندتر باشد.

هر وقت مقداری برای یک متغیر انتخاب می‌شود،
یک فرصت مناسب برای **استنتاج دامنه‌های جدید** برای **متغیرهای همسایه** پیش می‌آید.

بهبود کارآیی جستجوی عقب‌گرد

«یک در میان» سازی جستجو و استنتاج: بررسی پیش‌رو

FORWARD CHECKING

ایده:

دنبال کردن مقادیر مجاز باقیمانده برای متغیرهای نسبت‌دهی نشده جستجو زمانی خاتمه می‌یابد که حداقل یک متغیر هیچ مقدار مجازی نداشته باشد.

بررسی پیش‌رو
Forward Checking

روش:

اجرای «سازگاری کمانی» پس از انتساب هر متغیر برای آن متغیر برای هر متغیر نسبت‌دهی نشده‌ی Y که به X متصل است، هر مقدار از دامنه‌ی Y که ناسازگار با مقدار انتخاب شده از X است، حذف می‌شود.

بهبود کارایی جستجوی عقب‌گرد

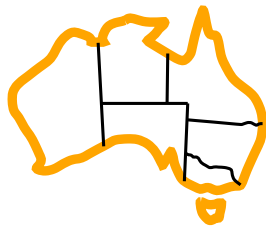
«یک در میان» سازی جستجو و استنتاج: بررسی پیشرو: مثال (۱ از ۴)

FORWARD CHECKING

ایده:

دنبال کردن مقادیر مجاز باقیمانده برای متغیرهای نسبت‌دهی نشده
جستجو زمانی خاتمه می‌یابد که حداقل یک متغیر هیچ مقدار مجازی نداشته باشد.

بررسی پیشرو
Forward Checking



WA

NT

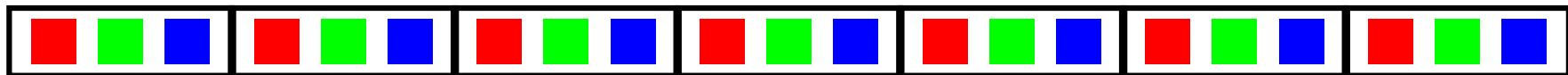
Q

NSW

V

SA

T



بهبود کارایی جستجوی عقب‌گرد

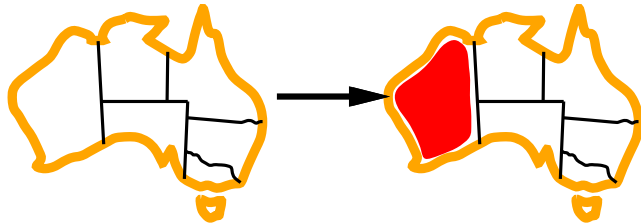
«یک در میان» سازی جستجو و استنتاج: بررسی پیش‌رو: مثال (۲ از ۴)

FORWARD CHECKING

ایده:

دنبال کردن مقادیر مجاز باقیمانده برای متغیرهای نسبت‌دهی نشده
جستجو زمانی خاتمه می‌یابد که حداقل یک متغیر هیچ مقدار مجازی نداشته باشد.

بررسی پیش‌رو
Forward Checking



WA	NT	Q	NSW	V	SA	T

بهبود کارایی جستجوی عقب‌گرد

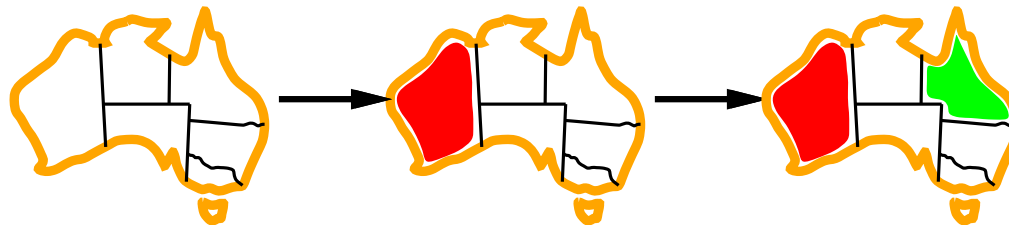
«یک در میان» سازی جستجو و استنتاج: بررسی پیش‌رو: مثال (۳ از ۴)

FORWARD CHECKING

ایده:

دنبال کردن مقادیر مجاز باقیمانده برای متغیرهای نسبت‌دهی نشده
جستجو زمانی خاتمه می‌یابد که حداقل یک متغیر هیچ مقدار مجازی نداشته باشد.

بررسی پیش‌رو
Forward Checking



WA	NT	Q	NSW	V	SA	T

بهبود کارایی جستجوی عقب‌گرد

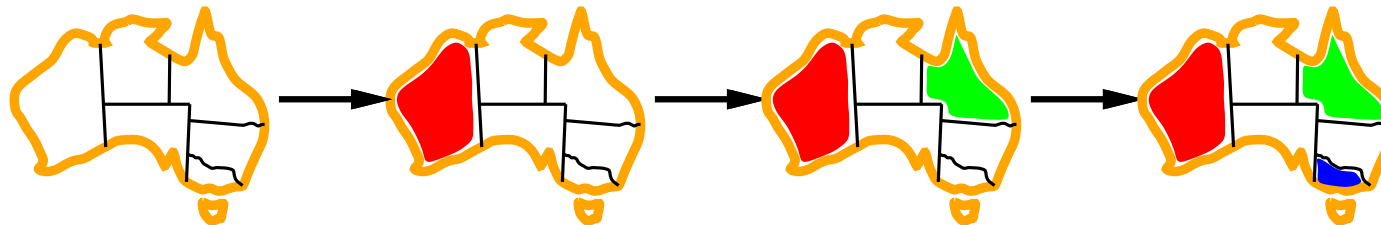
«یک در میان» سازی جستجو و استنتاج: بررسی پیش‌رو: مثال (۴ از ۴)

FORWARD CHECKING

ایده:

دنبال کردن مقادیر مجاز باقیمانده برای متغیرهای نسبت‌دهی نشده
جستجو زمانی خاتمه می‌یابد که حداقل یک متغیر هیچ مقدار مجازی نداشته باشد.

بررسی پیش‌رو
Forward Checking



WA	NT	Q	NSW	V	SA	T

	WA	NT	Q	NSW	V	SA	T
Initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After $WA=red$	Ⓡ	G B	R G B	R G B	R G B	G B	R G B
After $Q=green$	Ⓡ	B	ⓖ	R B	R G B	B	R G B
After $V=blue$	Ⓡ	B	ⓖ	R	ⓑ		R G B

Figure 6.7 The progress of a map-coloring search with forward checking. $WA = red$ is assigned first; then forward checking deletes *red* from the domains of the neighboring variables *NT* and *SA*. After $Q = green$ is assigned, *green* is deleted from the domains of *NT*, *SA*, and *NSW*. After $V = blue$ is assigned, *blue* is deleted from the domains of *NSW* and *SA*, leaving *SA* with no legal values.

بهبود کارایی جستجوی عقب‌گرد

«یک در میان» سازی جستجو و استنتاج: بررسی پیش‌رو: ویژگی‌ها

FORWARD CHECKING

ایده:

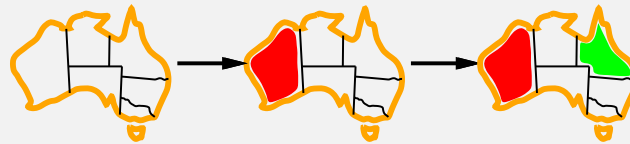
دنبال کردن مقادیر مجاز باقیمانده برای متغیرهای نسبت‌دهی نشده جستجو زمانی خاتمه می‌یابد که حداقل یک متغیر هیچ مقدار مجازی نداشته باشد.

بررسی پیش‌رو
Forward Checking

روش:

اجرای «سازگاری کمانی» پس از انتساب هر متغیر برای آن متغیر برای هر متغیر نسبت‌دهی نشده‌ی Y که به X متصل است، هر مقدار از دامنه‌ی Y که ناسازگار با مقدار انتخاب شده از X است، حذف می‌شود.

بررسی پیش‌رو بسیاری از ناسازگاری‌ها را تشخیص می‌دهد، اما نه همه‌ی آنها را. زیرا متغیر فعلی را سازگار کمانی می‌کند، اما رو به جلو نگاه نمی‌کند و سایر متغیرها را سازگار کمانی نمی‌کند.



مثال:

SA و NT هر دو نمی‌توانند آبی باشند.

WA	NT	Q	NSW	V	SA	T
red green blue	red green blue	red green blue	red green blue	red green blue	red green blue	red green blue
red red red	green blue blue	red green blue	red green blue	red green blue	green blue blue	red green blue
red red red	blue blue blue	green green green	red blue blue	red green blue	blue blue blue	red green blue

بهبود کارآیی جستجوی عقب‌گرد

«یک در میان» سازی جستجو و استنتاج: انتشار قید

CONSTRAINT PROPAGATION

انتشار قید

Constraint Propagation

ایده:

اعمال قیدها به صورت محلی

الگوریتم نگهداری سازگاری کمانی

Maintaining Arc Consistency (MAC)

روش:

اجرای «سازگاری کمانی» AC-3 پس از انتساب هر متغیر برای آن متغیر

پس از انتساب یک مقدار به متغیر Y

روال استنتاج INFERENCE، الگوریتم AC-3 را فراخوانی می‌کند
اما به جای صف همه‌ی کمان‌ها، تنها با کمان‌های (X, Y) شروع می‌کند
(برای همه‌ی متغیرهای X نسبت‌دهی نشده‌ی همسایه‌ی Y)



انتشار قید توسط AC-3 به روش معمول:

هر متغیری که دامنه‌اش به مجموعه‌ی تهی کاهش یافت، به معنی شکست AC-3 است
و بلافاصله باید عقب‌گرد صورت بگیرد.

بررسی پیش‌رو برخلاف MAC پس از تغییر در دامنه‌ی متغیرها،

قیدها را به صورت بازگشتی منتشر نمی‌کند.

۴

جستجوی
محلی
برای
مسائل
ارضای قید

جستجوی محلی برای مسائل ارضای قید

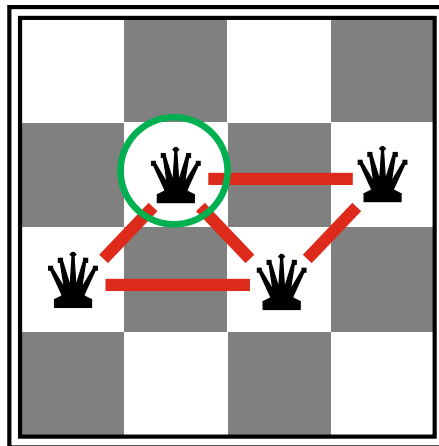
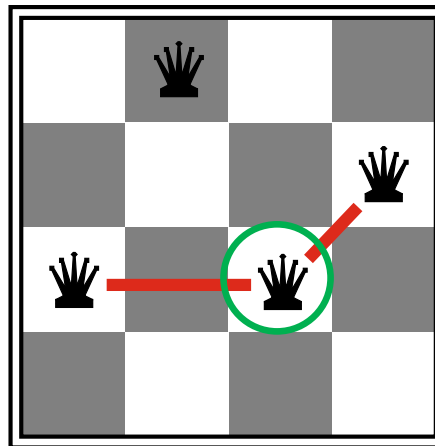
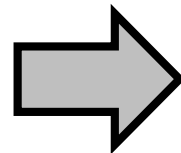
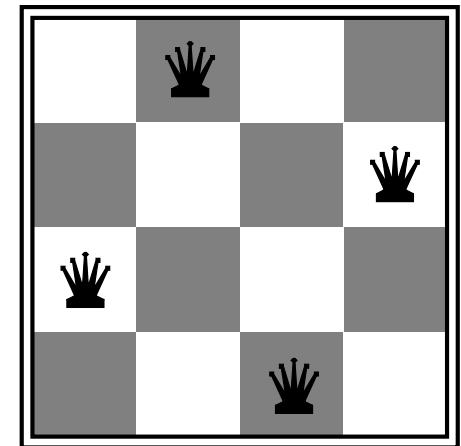
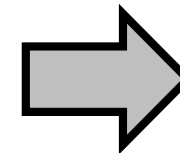
ملزومات روش‌های جستجوی محلی برای مسائل ارضای قید

انتخاب مقدار <i>Value Selection</i>	انتخاب متغیر <i>Variable Selection</i>	بازنمایی حالت‌ها <i>States Representation</i>
هیوریستیک حداقل تداخل <i>Min-Conflicts Heuristic</i>	تصادفی <i>Random</i>	اجازه دادن به وجود حالت‌هایی با قیدهای ارضاننده
انتخاب مقداری برای متغیر که کمترین تعداد قیدها را نقض می‌کند $h(n)$ = مجموع تعداد قیدهای نقض شده	انتخاب یک متغیر تداخل‌دار به صورت تصادفی	(چون جستجوهای محلی، با حالت‌های «کامل» کار می‌کنند) عملگرها، مقادیر متغیرها را نسبت‌دهی مجدد می‌کنند.

جستجوی محلی برای مسائل ارضای قید

مثال: مسئله n -وزیر

تعداد تهدیدها = مجموع تعداد قیدهای نقض شده $= h(n)$

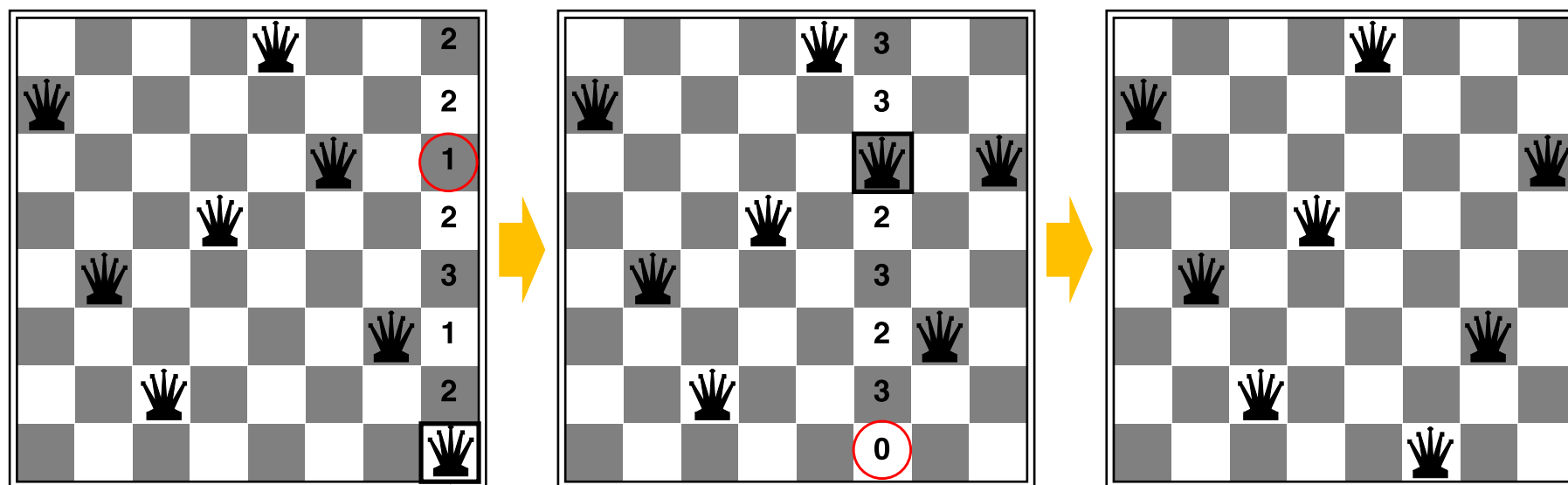
 $h = 5$  $h = 2$  $h = 0$

جستجوی محلی برای مسائل ارضای قید

مثال: مسئله n -وزیر: استفاده از هیوریستیک حداقل تداخلها

تعداد تهدیدها = مجموع تعداد قیدهای نقض شده $h(n)$

در هر مرحله، یک وزیر برای انتساب مجدد در ستون خودش انتخاب می شود.



اعداد: تعداد تداخل‌های حاصل برای انتقال وزیر آن ستون به هریک از خانه‌های آن ستون

الگوریتم جستجوی محلی با هیوریستیک «حداقل تداخلها» برای CSPها

شبکه‌کد

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 $\leftarrow$ an initial complete assignment for *csp*

for *i* = 1 to *max_steps* **do**

if *current* is a solution for *csp* **then return** *current*

var $\leftarrow$ a randomly chosen conflicted variable from *csp*.VARIABLES

value $\leftarrow$ the value *v* for *var* that minimizes CONFLICTS(*var*, *v*, *current*, *csp*)

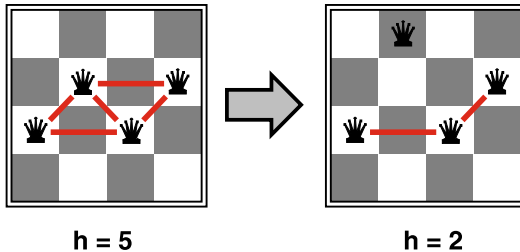
 set *var* = *value* in *current*

return *failure*

تابع CONFLICTS تعداد قیدهای نقض شده توسط یک مقدار خاص را با در نظر گرفتن سایر انتساب‌های فعلی می‌شمارد.

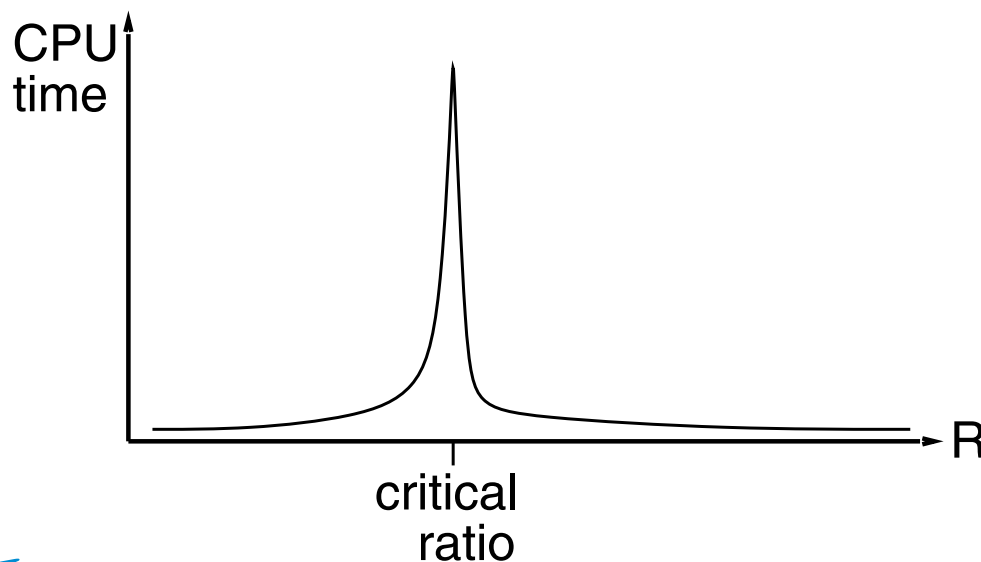
جستجوی محلی برای مسائل ارضای قید

مثال: مسئله n -وزیر: ارزیابی کارایی



با شروع از حالت آغازین تصادفی:
امکان حل مسئله n -وزیر در زمان تقریباً ثابت برای n دلخواه با احتمال بالا
(مثلاً $n = 10,000,000$)

به نظر می‌رسد این نتیجه برای هر CSP تولید شده به صورت تصادفی همین‌گونه درست باشد.
غیر از بازه‌ی کوچکی از نسبت R



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

۵

ساختار
مسئله

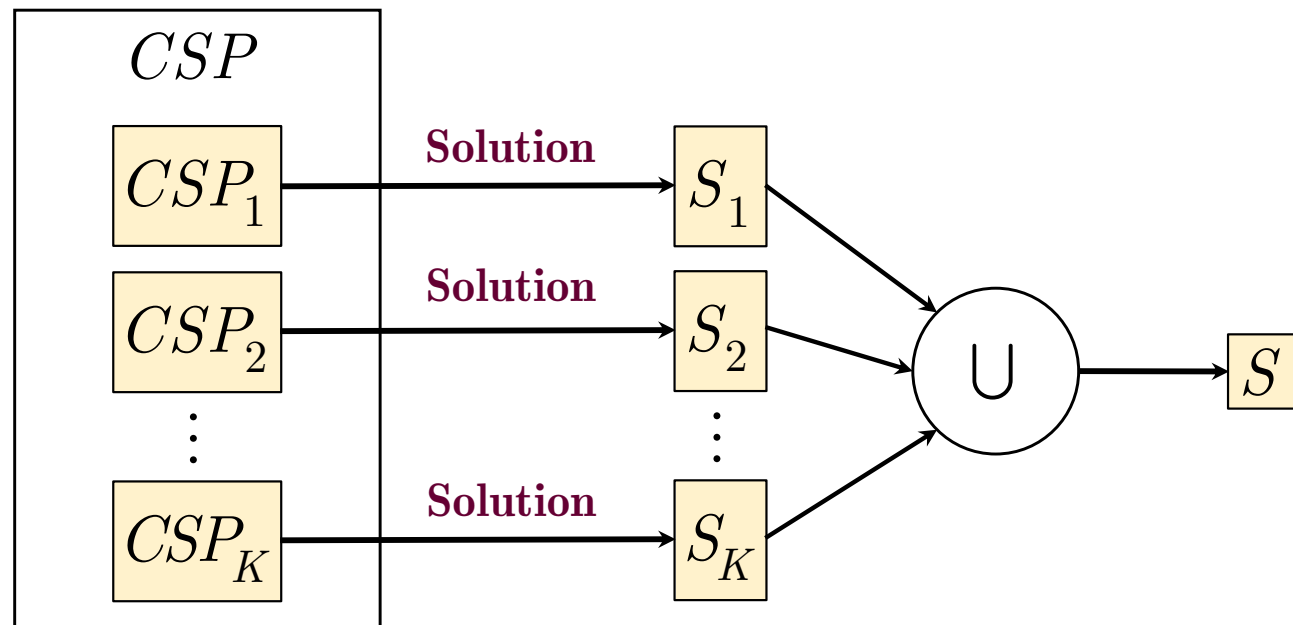
ساختار مسئله

PROBLEM STRUCTURE

استفاده‌ی مناسب از ساختار مسئله (گراف قید)
برای یافتن سریع راه‌حل‌ها

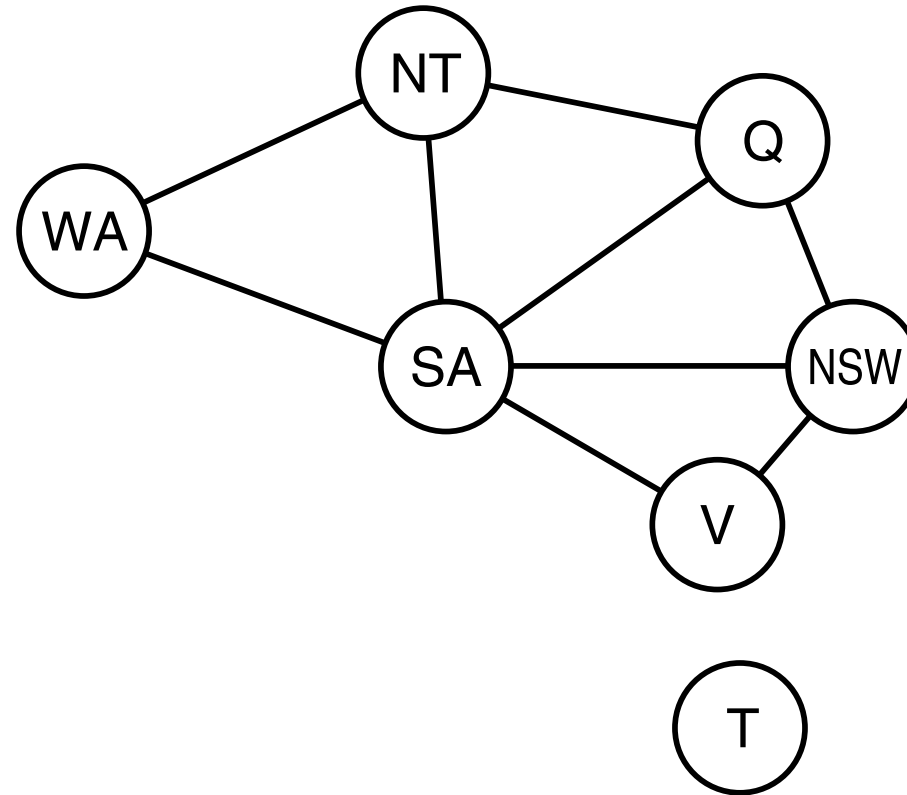
مؤلفه‌های همبند گراف قید، زیرمسئله‌های مستقل هستند
و می‌توانند به صورت جداگانه حل شوند.

K = تعداد زیرمسئله‌ها =
تعداد مؤلفه‌های همبند



ساختار مسئله

مثال



دو مؤلفه‌ی همبند = دو زیرمسئله‌ی مستقل

$$K = 2$$

ساختار مسئله

تأثیر بر پیچیدگی زمانی راه حل

زمان اجرا

$$O(n / c \cdot d^c)$$

خطی بر حسب n

به جای

$$O(d^n)$$

نمایی بر حسب n

K = تعداد زیرمسئله ها = تعداد مؤلفه های همبند

c = تعداد متغیر هر زیرمسئله

n = تعداد کل متغیرها

d = تعداد عناصر دامنه ی متغیرها

$$K = \left\lceil \frac{n}{c} \right\rceil$$

E.g., $n = 80$, $d = 2$, $c = 20$

$2^{80} = 4$ billion years at 10 million nodes/sec

$4 \cdot 2^{20} = 0.4$ seconds at 10 million nodes/sec

CSP با ساختار درختی

TREE-STRUCTURED CSPs

زمان اجرا

$$O(n d^2)$$

خطی بر حسب n

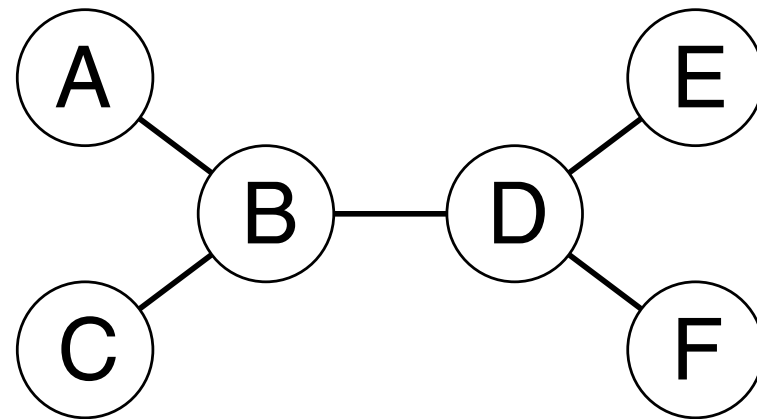
به جای

$$O(d^n)$$

نمایی بر حسب n

قضیه

اگر گراف قید، حلقه نداشته باشد،
CSP می‌تواند در زمان $O(n d^2)$ حل شود.



CSP با ساختار درختی

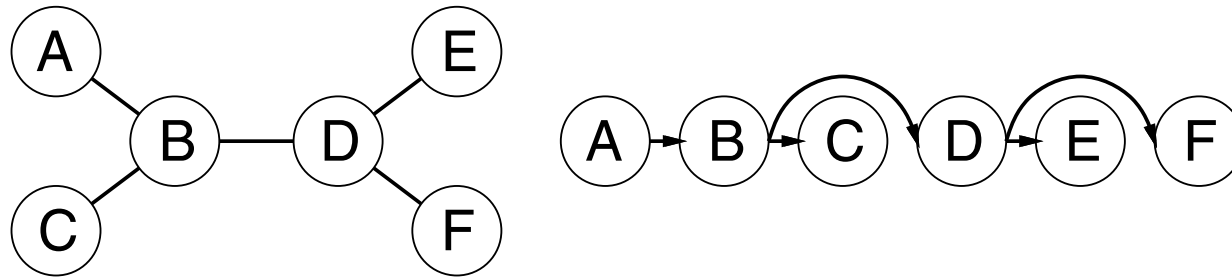
الگوریتم برای راه حل

ALGORITHM FOR TREE-STRUCTURED CSPs

یک متغیر را به عنوان ریشه انتخاب کنید،
سایر متغیرها را از ریشه تا برگها مرتب کنید
به گونه ای که والد هر گره در ترتیب قبل از آن گره بیاید.

مرتب سازی توپولوژیکی
Topological Sort

۱



for $j \leftarrow n$ down to 2
REMOVEINCONSISTENT($Parent(X_j), X_j$)

حذف ناسازگاریها
Remove Inconsistent

۲

for $j \leftarrow 1$ to n
assign X_j consistently with $Parent(X_j)$

انتساب متغیرها
Variable Assignment

۳

تبدیل به ساختار درختی

شرط‌گذاری

CONDITIONING

یک متغیر را مقداردهی اولیه کنید،
دامنه‌ی متغیرهای همسایه را هرس کنید.

شرط‌گذاری
Conditioning

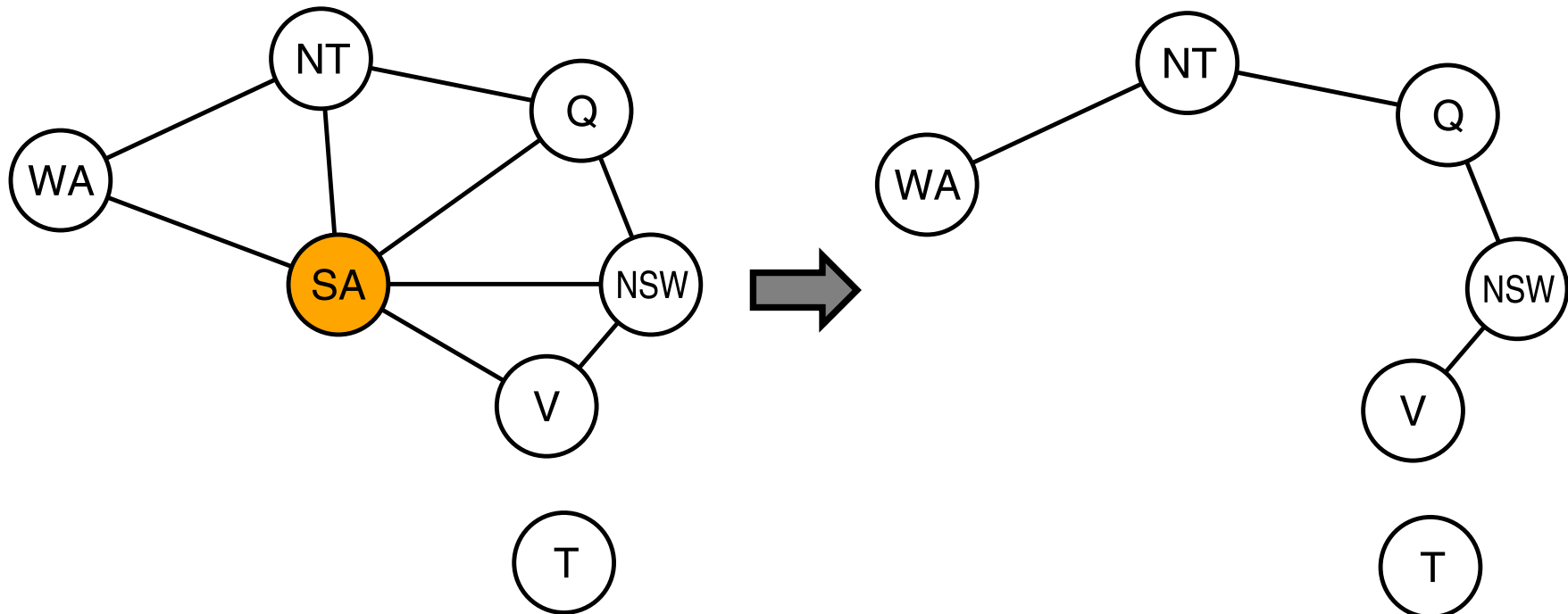
تبدیل به ساختار درختی

شرط‌گذاری: مثال

CONDITIONING

یک متغیر را مقداردهی اولیه کنید،
دامنه‌ی متغیرهای همسایه را هرس کنید.

شرط‌گذاری
Conditioning



تبدیل به ساختار درختی

شرط‌گذاری ابرگره‌ای (کاتستی)

CUTSET CONDITIONING

یک مجموعه از متغیرها را مقداردهی اولیه کنید (با همه‌ی راه‌های ممکن)،
به طوری که گراف قید حاصل، یک درخت باشد.

شرط‌گذاری کاتستی
Cutset Conditioning

تبدیل به ساختار درختی

شرط‌گذاری ابرگره‌ای (کاتستی): پیچیدگی زمانی

CUTSET CONDITIONING

یک مجموعه از متغیرها را مقداردهی اولیه کنید (با همه‌ی راه‌های ممکن)،
به طوری که گراف قید حاصل، یک درخت باشد.

شرط‌گذاری کاتستی
Cutset Conditioning

زمان اجرا

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

خطی بر حسب n

به جای

$$O(d^n)$$

نمایی بر حسب n

c = تعداد متغیر کاتست = اندازه‌ی کاتست
 n = تعداد کل متغیرها
 d = تعداد عناصر دامنه‌ی متغیرها

برای c کوچک بسیار سریع است.

تبدیل به ساختار درختی

تجزیه‌ی درختی

TREE DECOMPOSITION

مسئله را به یک درخت از زیرمسئله‌ها تبدیل می‌کند.

تجزیه‌ی درختی
Tree Decomposition

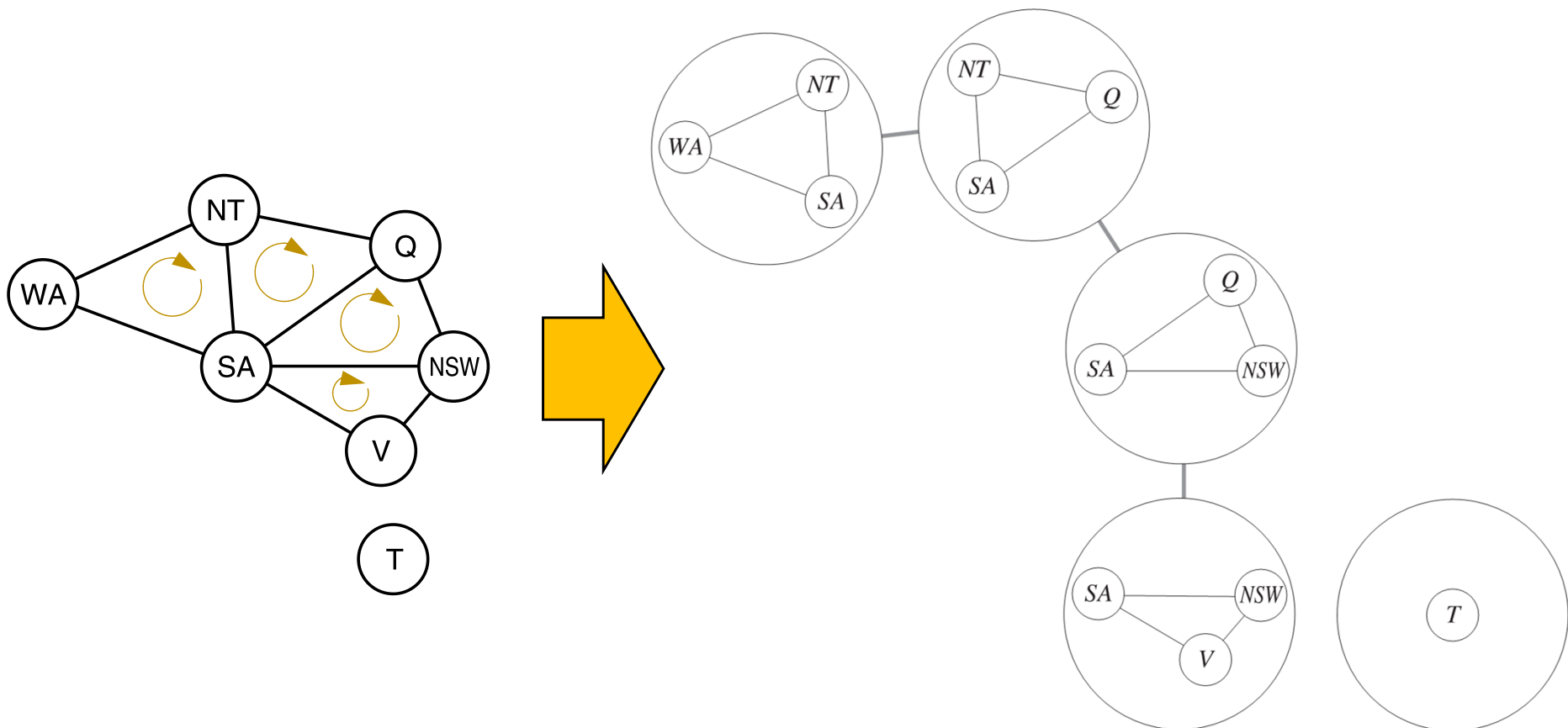
تبدیل به ساختار درختی

تجزیه‌ی درختی: مثال

TREE DECOMPOSITION

مسئله را به یک درخت از زیرمسئله‌ها تبدیل می‌کند.

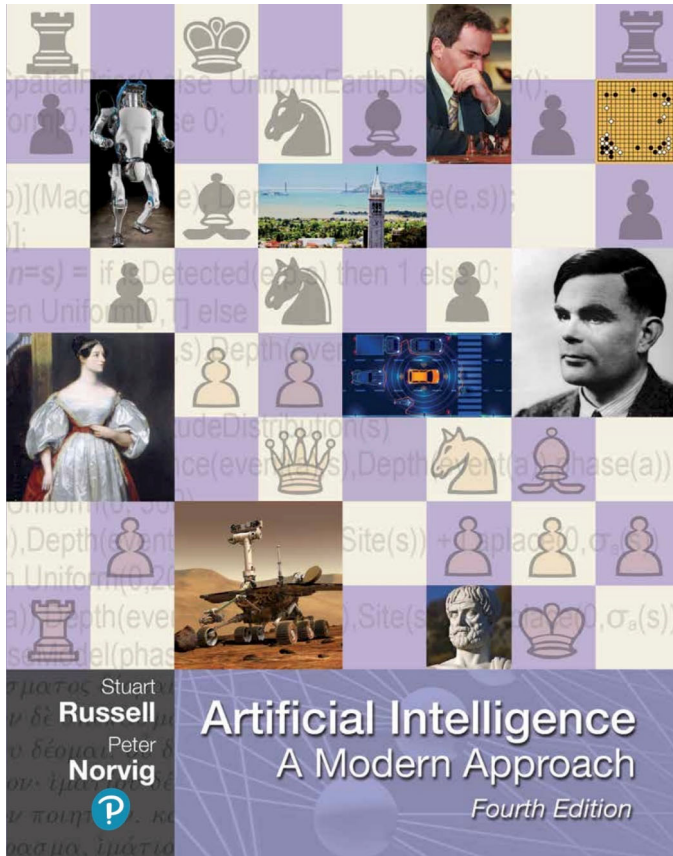
تجزیه‌ی درختی
Tree Decomposition



۶

منابع،
مطالعه،
تکلیف

منبع اصلی



Stuart Russell and Peter Norvig,
Artificial Intelligence: A Modern Approach,
 4th Edition, Prentice Hall, 2020.

Chapter 6