بسم الله الرحمن الرحیم

هوش مصنوعی

فصل ۶

مسائل ارضای قید

Constraint Satisfaction Problems

کاظم فولادی قلعه

دانشکده مهندسی، دانشکدگان 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.

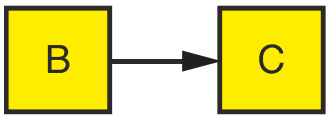
انواع زیادی از مسائل می‌توانند به صورت CSP تعریف شوند.

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

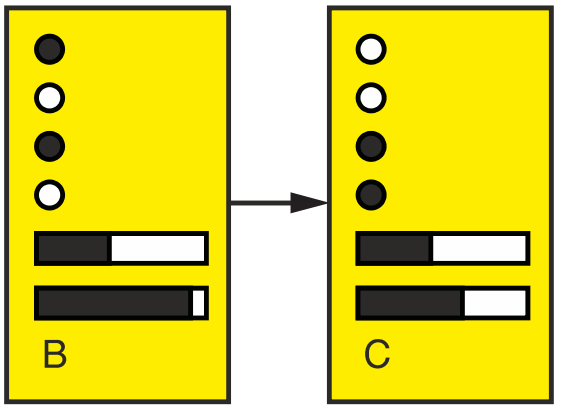


(a) Atomic

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

مؤلفه‌های سه‌گانه‌ی تعریف مسئله‌ی «ارضای قید»

X	**متغیرها** Variables	مجموعه‌ای از متغیرها $\{X_1, X_2, \ldots, X_n\}$
D	**دامنه‌ها** Domains	مجموعه‌ای از دامنه‌ها (مقادیر مجاز متغیرها) هر دامنه به صورت $\{v_1, v_2, \ldots, v_k\}$ $\{D_1, D_2, \ldots, D_n\}$
C	**قیدها** Constraints	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.

انتساب

انتساب مقدار به متغیر و انواع آن

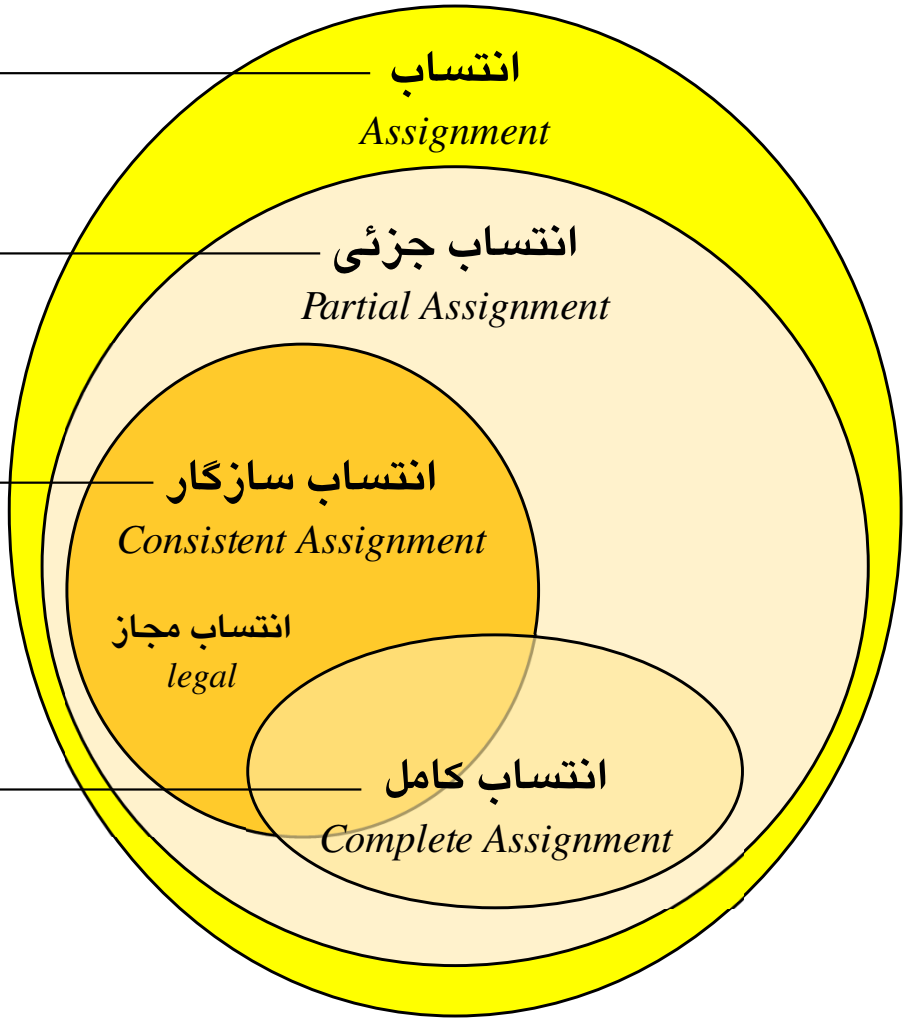
ASSIGNMENT

نسبت‌دهی مقادیر به تمام یا بعضی از متغیرها

یک انتساب که در آن بعضی متغیرها نسبت‌دهی شده باشند.

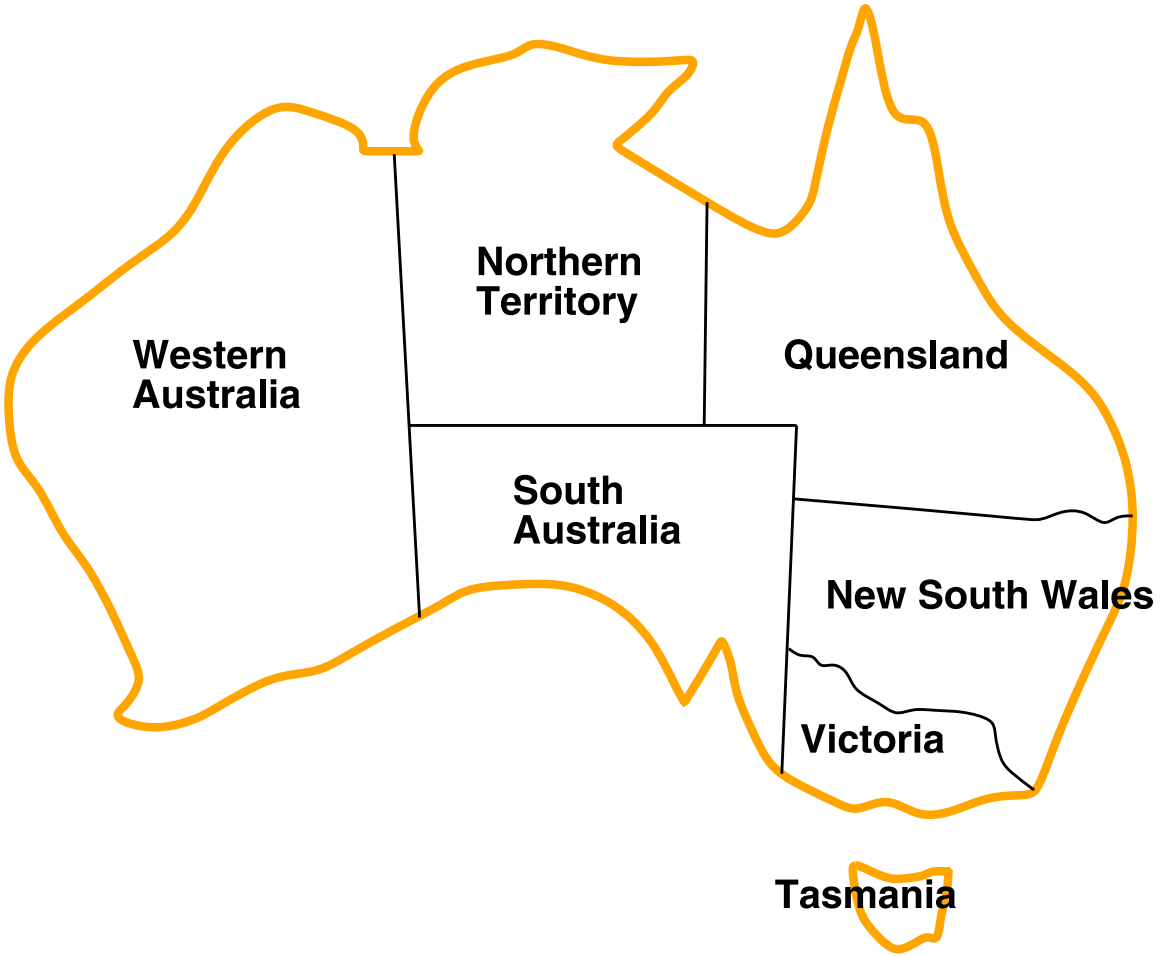
یک انتساب که هیچ قیدی را نقض نکرده باشد

یک انتساب که در آن همه‌ی متغیرها نسبت‌دهی شده باشند.



مسئله‌ی «رنگ‌آمیزی نقشه»

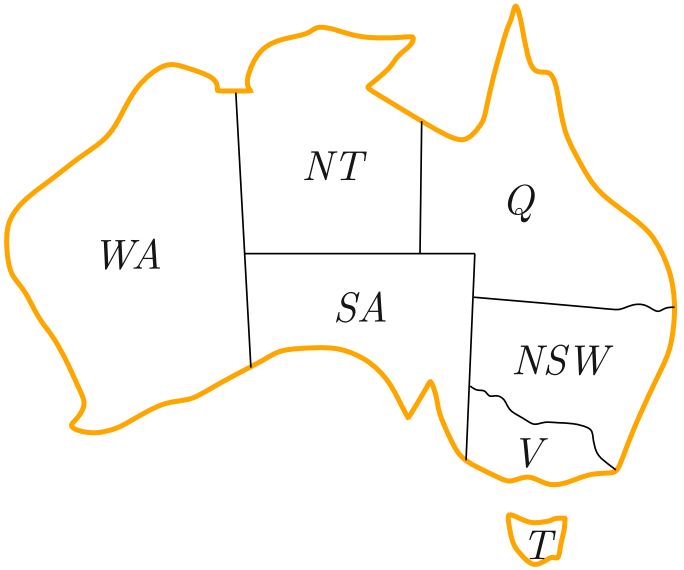
MAP-COLORING



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

مسئله‌ی ارضای قید

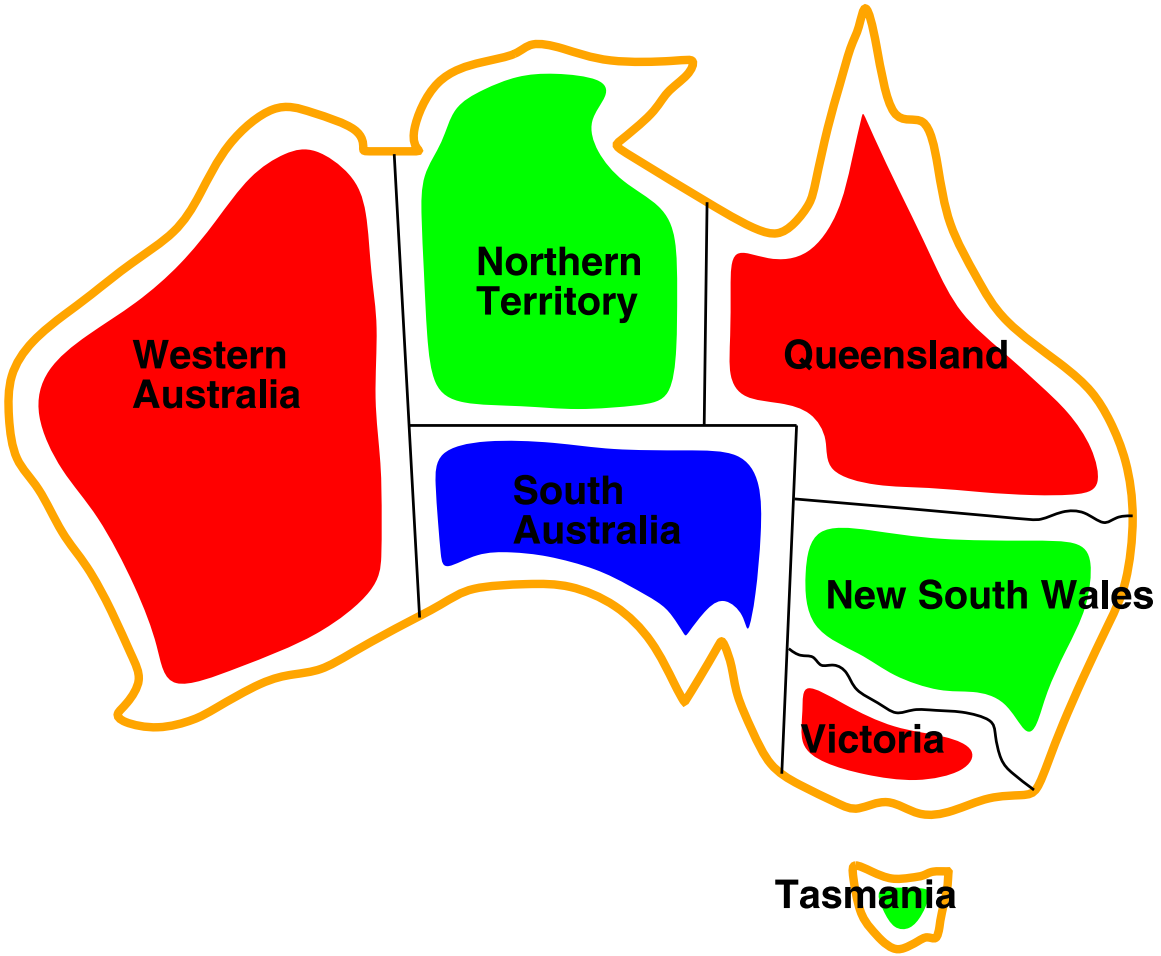
مثال: مسئله‌ی رنگ‌آمیز نقشه



مؤلفه‌های تعریف مسئله‌ی «ارضای قید»			
متغیرها Variables	X	استان‌ها	$WA,\ NT,\ Q,\ NSW,\ V,\ SA,\ T$
دامنه‌ها Domains	D	رنگ‌ها	$D_i = \{red, blue, green\}$
قیدها Constraints	C	ناحیه‌های مجاور باید رنگ‌های متفاوت داشته باشند. e.g., $WA \neq NT,\quad V \neq NSW,\quad \ldots$	

مسئله‌ی ارضای قید

مثال : مسئله‌ی رنگ‌آمیز نقشه : راه‌حل



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

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

گراف قید

CONSTRAINT GRAPH

گراف قید *Constraint Graph*	
کمان‌ها *Arcs*	**گره‌ها** *Nodes*
نشان‌دهنده‌ی **قیدها**	نشان‌دهنده‌ی **متغیرها**

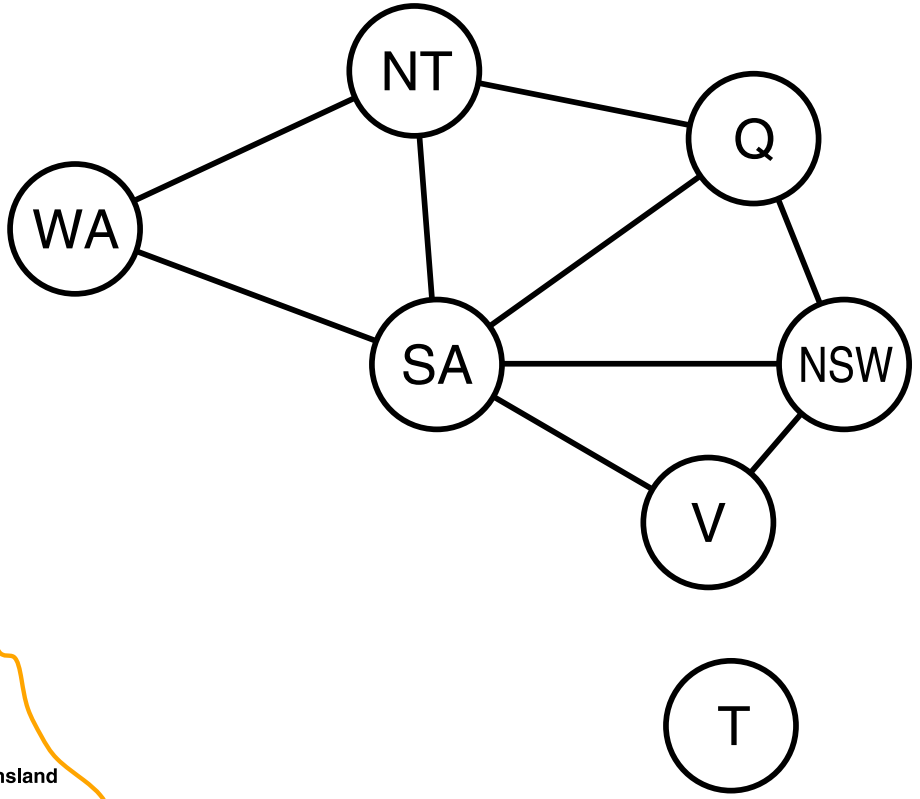
قابل استفاده برای CSP دودویی

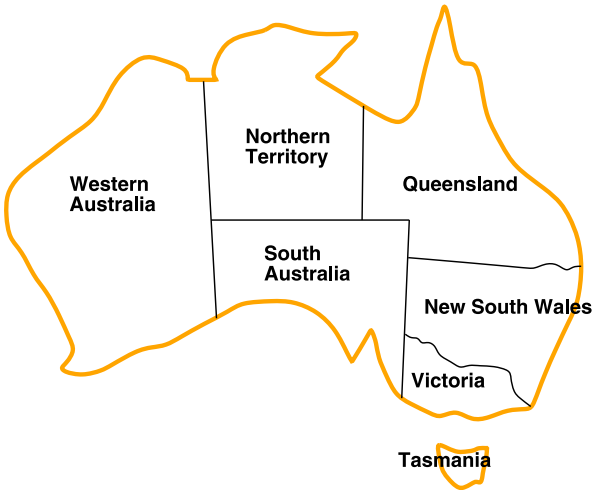
الگوریتم‌های CSP همه‌منظوره، از ساختار گراف برای تسریع جستجو استفاده می‌کنند.

گراف قید

مثال

CONSTRAINT GRAPH





انواع «مسائل ارضای قید»

VARIETIES OF CSPS

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

انواع «مسائل ارضای قید»

CSP با دامنه‌ی متناهی و گسسته

DISCRETE FINITE-DOMAIN CSP

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

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

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

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

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

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

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

قیدهای تکی شامل **یک متغیر** هستند.

مانند $SA \neq green$

انواع «مسائل ارضای قید»

CSP با قیدهای دودویی

BINARY CONSTRAINTS

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

قیدهای دودویی شامل **دو متغیر** هستند.

مانند $SA \neq WA$

انواع «مسائل ارضای قید»

CSP با قیدهای چندتایی

N-ARY CONSTRAINTS

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

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

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

```
  T W O
+ T W O
-------
F O U R
```

انتساب رقم‌های متفاوت به حروف متفاوت برای درست شدن عبارت حسابی

مسئله‌ی ارضای قید

مثال : مسئله‌ی حساب رمزی

$X_3 \quad X_2 \quad X_1$

$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$

مؤلفه‌های تعریف مسئله‌ی «ارضای قید»

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

ابرگراف قید

CONSTRAINT HYPERGRAPH

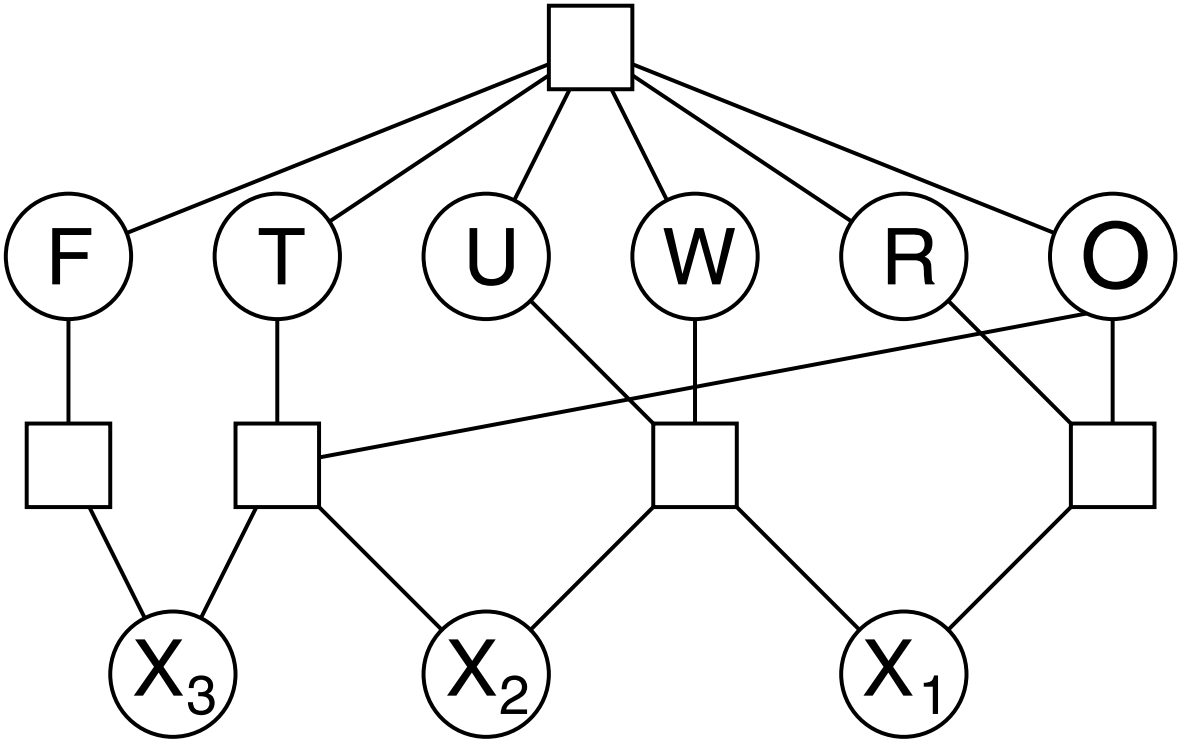
ابرگراف قید *Constraint Hypergraph*		
کمان‌ها *Arcs*	**گره‌ها** *Nodes*	
	گره‌های معمولی *Ordinary Nodes*	**گره‌ها قید** *Constraint Nodes*
نشان‌دهنده‌ی **قیدها**	نشان‌دهنده‌ی **متغیرها**	نشان‌دهنده‌ی **قیدها**
	○	□

قابل استفاده برای CSP چندتایی

ابرگراف قید

مثال

CONSTRAINT HYPERGRAPH



$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$

کاربردهای «مسائل ارضای قید» در دنیای واقعی

مثال

مسائل انتساب
Assignment Problems

مثلا، چه کسی به چه کلاسی درس بدهد؟

مسائل جدول زمان‌بندی
Timetabling Problems

مثلا، کدام کلاس چه وقت و کجا برگزار شود؟

پیکربندی سخت‌افزار
Hardware Configuration

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

زمان‌بندی حمل و نقل
Transportation Scheduling

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

طرح‌ریزی کف
Floorplanning

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

زمان‌بندی تولید کارخانه
Factory Scheduling

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

هوش مصنوعی

مسائل ارضای قید

۲

انتشار قید: استنتاج در مسائل ارضای قید

انتشار قید: استنتاج در مسائل ارضای قید

CONSTRAINT PROPAGATION: INFERENCE IN CSPS

جستجوی معمولی فضای حالت	جستجو برای مسائل ارضای قید
فقط امکان جستجوی محض	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)

به‌کارگیری «انتشار قید»

پیش از جستجو	همزمان با جستجو
به عنوان گام پیش‌پردازش	

گاهی این پیش‌پردازش می‌تواند 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 می‌شود مقادیر ناسازگار از سراسر گراف حذف شود.

سازگاری محلی *Local Consistency*			
سازگاری گره‌ای *Node Consistency*	**سازگاری کمانی** *Arc Consistency*	**سازگاری مسیری** *Path Consistency*	**k–سازگاری** *k-Consistency*

انتشار قید با اعمال «سازگاری محلی»

سازگاری گره‌ای

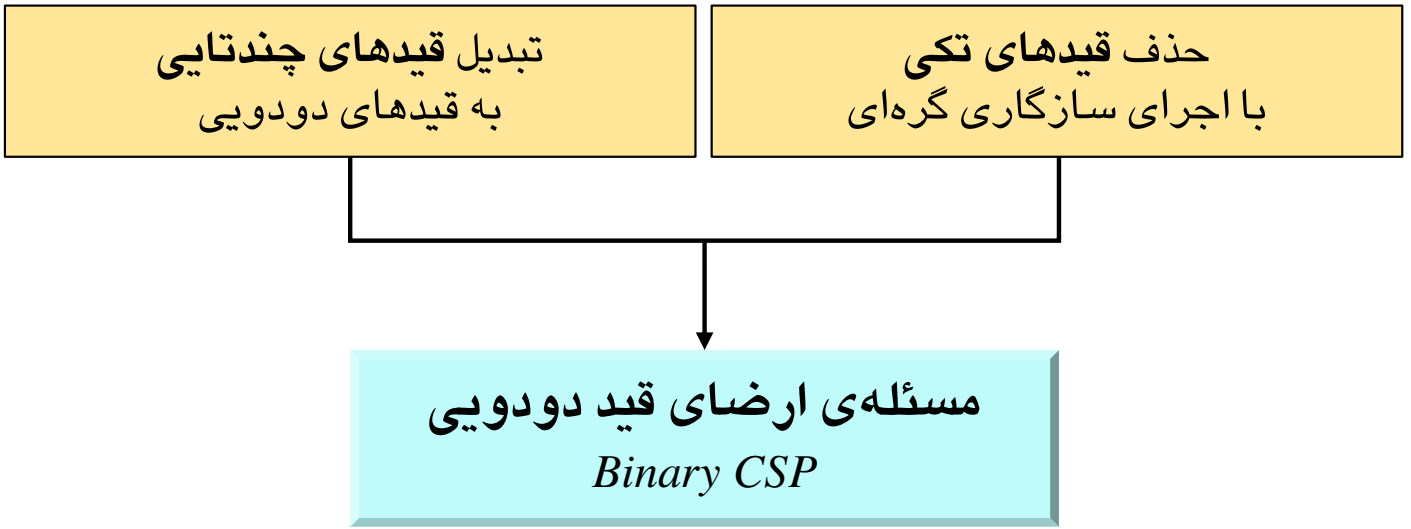
سازگاری گره‌ای
Node Consistency

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

مثال: $SA \neq green$

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.

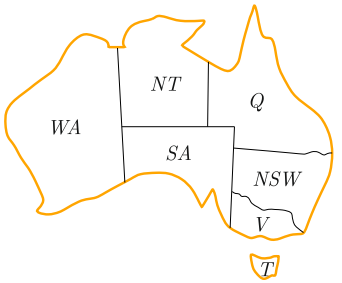
سازگاری کمانی برای مسئله‌ی «رنگ‌آمیزی نقشه‌ی استرالیا» کاری انجام نمی‌دهد، مثلاً در قید:

$$SA \neq WA$$

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

$\Downarrow$

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

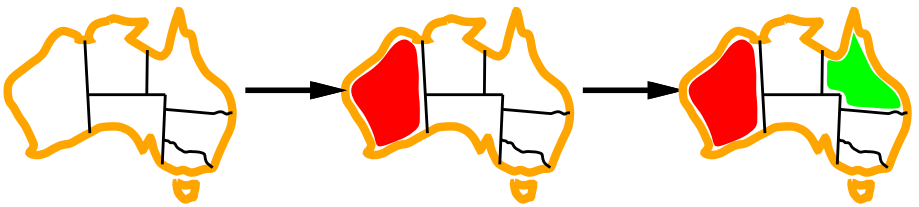
الگوریتم AC-3 **یک صف از کمان‌ها** برای بررسی ایجاد می‌کند.
این صف در ابتدا شامل همه‌ی کمان‌ها در CSP است.
سپس یک کمان دلخواه (X_i,X_j) از صف برداشته می‌شود و X_i نسبت به X_j سازگار کمانی می‌شود.

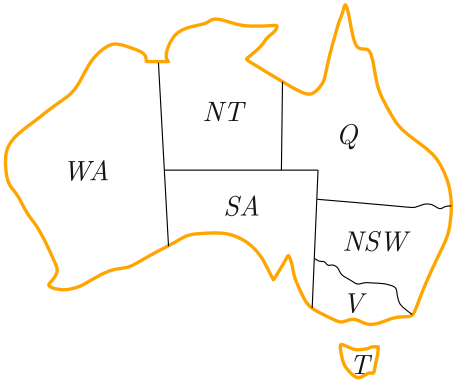
- اگر دامنه‌ی D_i تغییری نکرد ⇐ الگوریتم فقط به سراغ کمان بعدی می‌رود.
- اگر دامنه‌ی D_i تغییر کرد (Revise) ⇐ تمام کمان‌های (X_k,X_i) را به صف می‌افزاییم (X_k همسایه‌ی X_i است).
- اگر دامنه‌ی D_i به تهی کاهش یابد ⇐ CSP راه‌حل سازگار ندارد و AC-3 بلافاصله failure برمی‌گرداند.

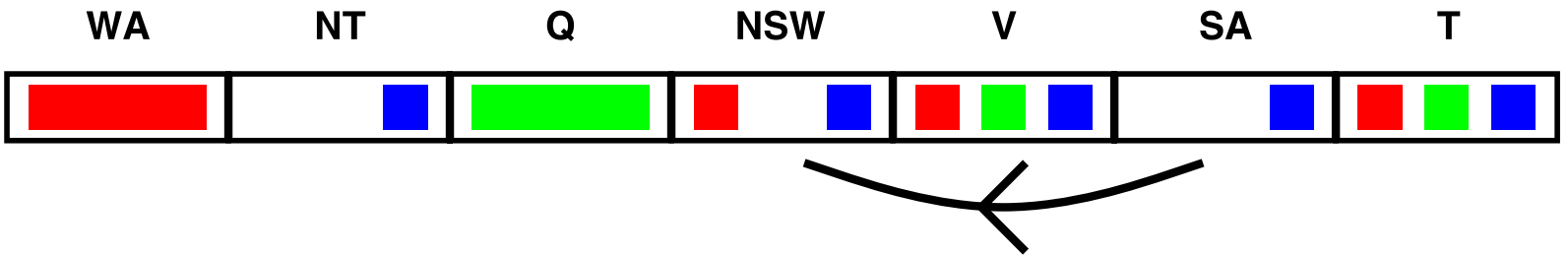
بررسی‌ها ادامه می‌یابد:
به مرور مقادیری از دامنه‌ی متغیرها حذف می‌شود تا اینکه دیگر کمانی در صف باقی نماند.
❋ در اینجا به یک CSP معادل با CSP اصلی می‌رسیم:
اما CSP سازگار کمانی معمولاً برای جستجو سریع‌تر است، زیرا دامنه‌ی متغیرها کوچک‌تر است.

انتشار قید با اعمال «سازگاری محلی»

سازگاری کمانی: مثال (۱ از ۴)

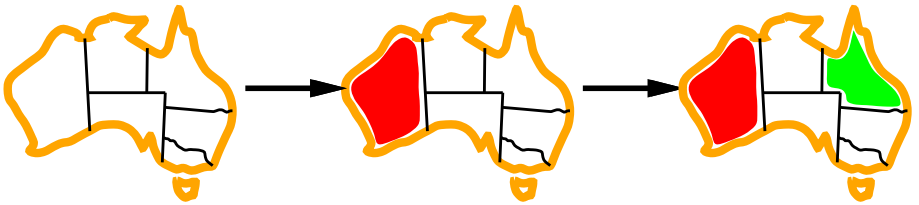


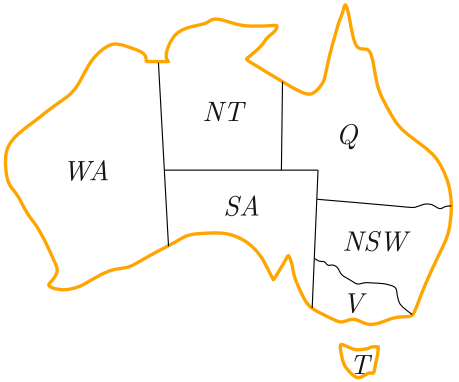


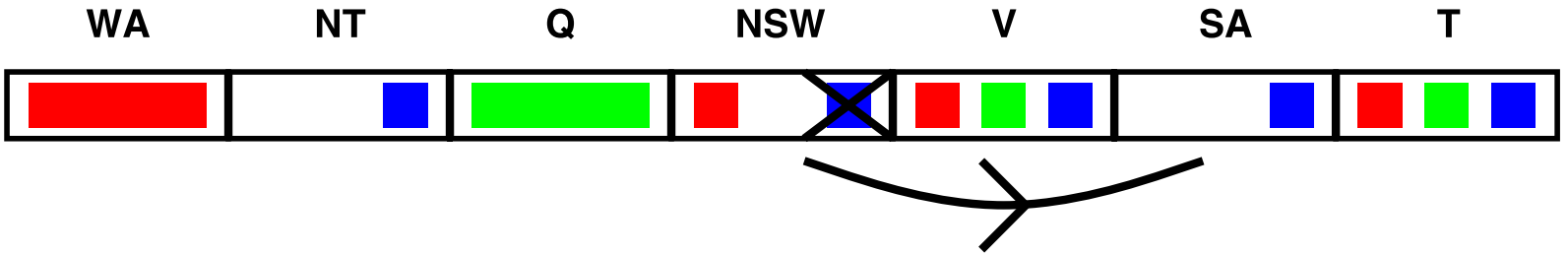


انتشار قید با اعمال «سازگاری محلی»

سازگاری کمانی: مثال (۲ از ۴)

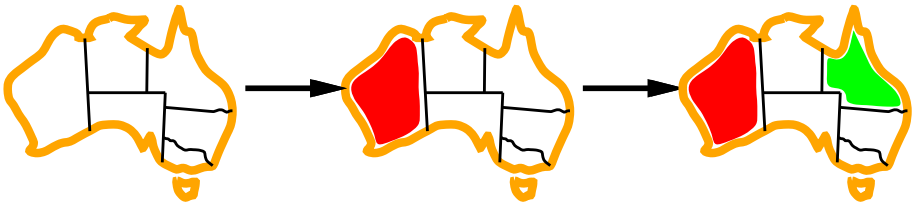


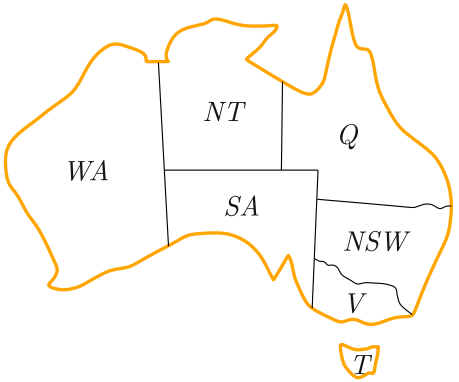


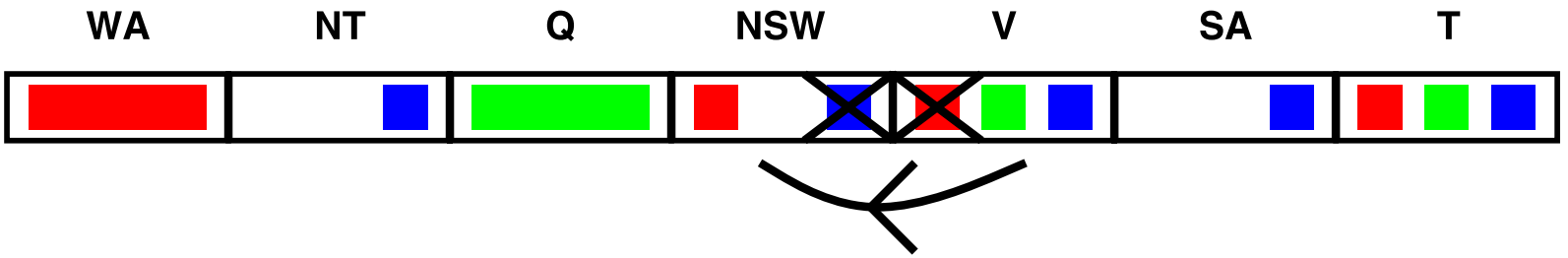


انتشار قید با اعمال «سازگاری محلی»

سازگاری کمانی: مثال (۳ از ۴)



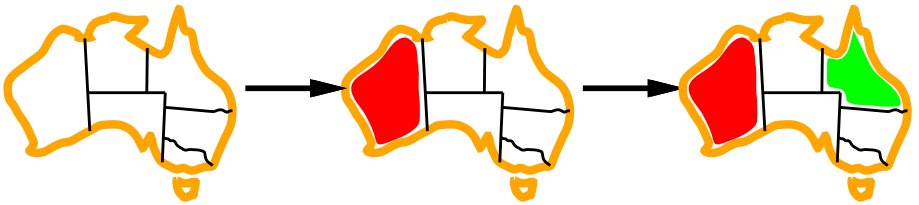


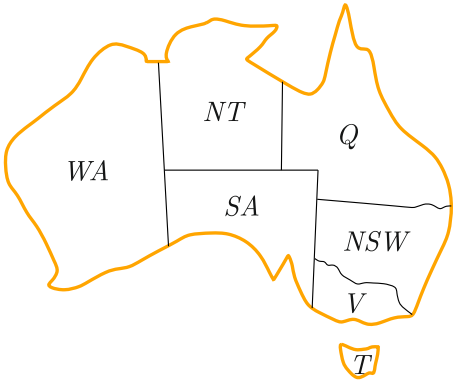


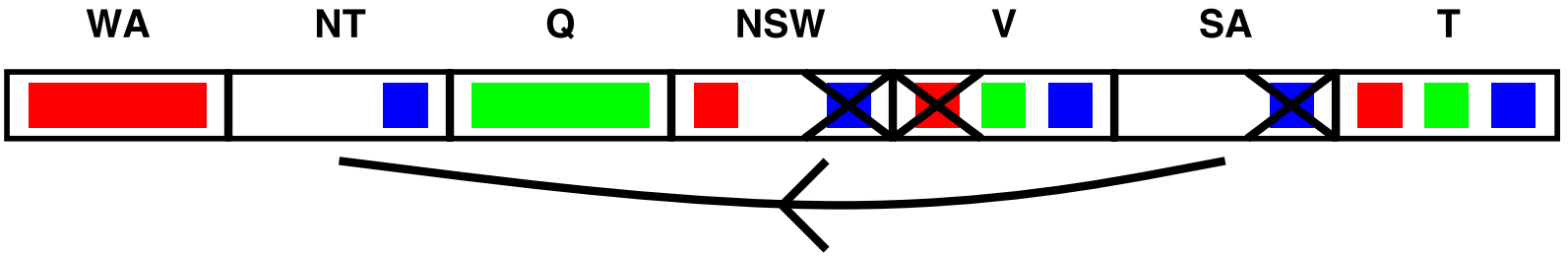
اگر X مقداری را از دست بدهد، لازم است همسایه‌های X دوباره بررسی شوند.

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: شبه کد

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**
    $(X_i,\ X_j) \leftarrow$ 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.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:
- n متغیر
- d حداکثر اندازه‌ی دامنه‌ها
- c حداکثر تعداد کمان‌ها (قیدهای دودویی)

* 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.

$$\Downarrow$$

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)

سازگاری ابرکمانی
Hyperarc Consistency

گسترش مفهوم سازگاری کمانی برای کار با قیدهای nتایی

سازگاری کمانی تعمیم‌یافته
Generalized Arc Consistency

متغیر X_i نسبت به یک قید nتایی **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

سازگاری ابرکمانی
Hyperarc Consistency

گسترش مفهوم سازگاری کمانی برای کار با قیدهای nتایی

سازگاری کمانی تعمیم‌یافته
Generalized Arc Consistency

همه‌ی متغیرها، دارای دامنه‌ی $\{0,1,2,3\}$

متغیر X سازگار با قید $X < Y < Z$ است
اگر
2 و 3 از دامنه‌ی X حذف شده باشد.

(زیرا هرگاه X برابر با 2 یا 3 باشد، قید نمی‌تواند 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 دومتغیره‌ی $\{X_i,X_j\}$ 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.

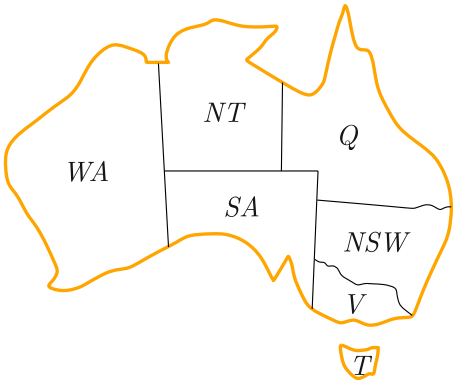
(سازگاری مسیری: قابل مشاهده به صورت مسیری از X_i به X_j با X_m در میان آن.)

الگوریتم PC-2 برای سازگاری مسیری

انتشار قید با اعمال «سازگاری محلی»

سازگاری مسیری
Path Consistency

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



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.

$\Downarrow$

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

k–سازگاری قوی
Strong k-Consistency

یک CSP، k–سازگار قوی است اگر:
k–سازگار، k-1–سازگار، k-2–سازگار، ...، ۱–سازگار باشد.

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، ...، ۱–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).
- ...

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 به‌کرات در مسائل واقعی رخ می‌دهند
و می‌توانند با الگوریتم‌های خاص‌منظوره (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 «همه‌متفاوت» $Alldiff$
(متفاوت بودن همه‌ی متغیرها)

مثل : مسئله‌ی حساب رمزی

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

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

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.

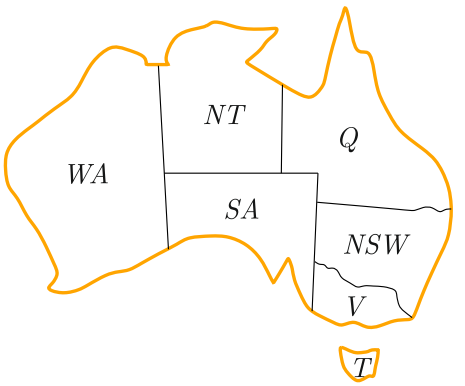
- ابتدا هر متغیر در قید که دامنه‌ی تک عنصری دارد را حذف کنید؛
  و مقادیر آن متغیر را از دامنه‌ی مابقی متغیرها حذف کنید.
- این کار را تکرار کنید تا زمانی که هنوز متغیرهای تکی وجود دارند.
- اگر در نقطه‌ای **دامنه‌ی تهی تولید شد**،
  یا **تعداد متغیر بیشتری نسبت به مقادیر دامنه باقی ماند**،
  آن‌گاه یک 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 ⇐ سه متغیر با دو رنگ ⇐ قید «همه‌متفاوت» 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 می‌نیمم 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 5 و 6 می‌توانند از هر دامنه حذف شوند.

(برای اعمال سازگاری: حذف مقادیر ماکزیمم از هر دامنه اگر با مقادیر می‌نیمم 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 وسیله‌ی نقلیه)

معمولاً ممکن نیست که دامنه‌ی هر متغیر به صورت یک مجموعه‌ی بزرگ از اعداد صحیح بازنمایی شود و بعد به‌تدریج 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

سازگاری کرانی
Bounds Consistency

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.

مسئله‌ی ارضای قید

مثال: معمای سودوکو

SUDOKU PUZZZLE

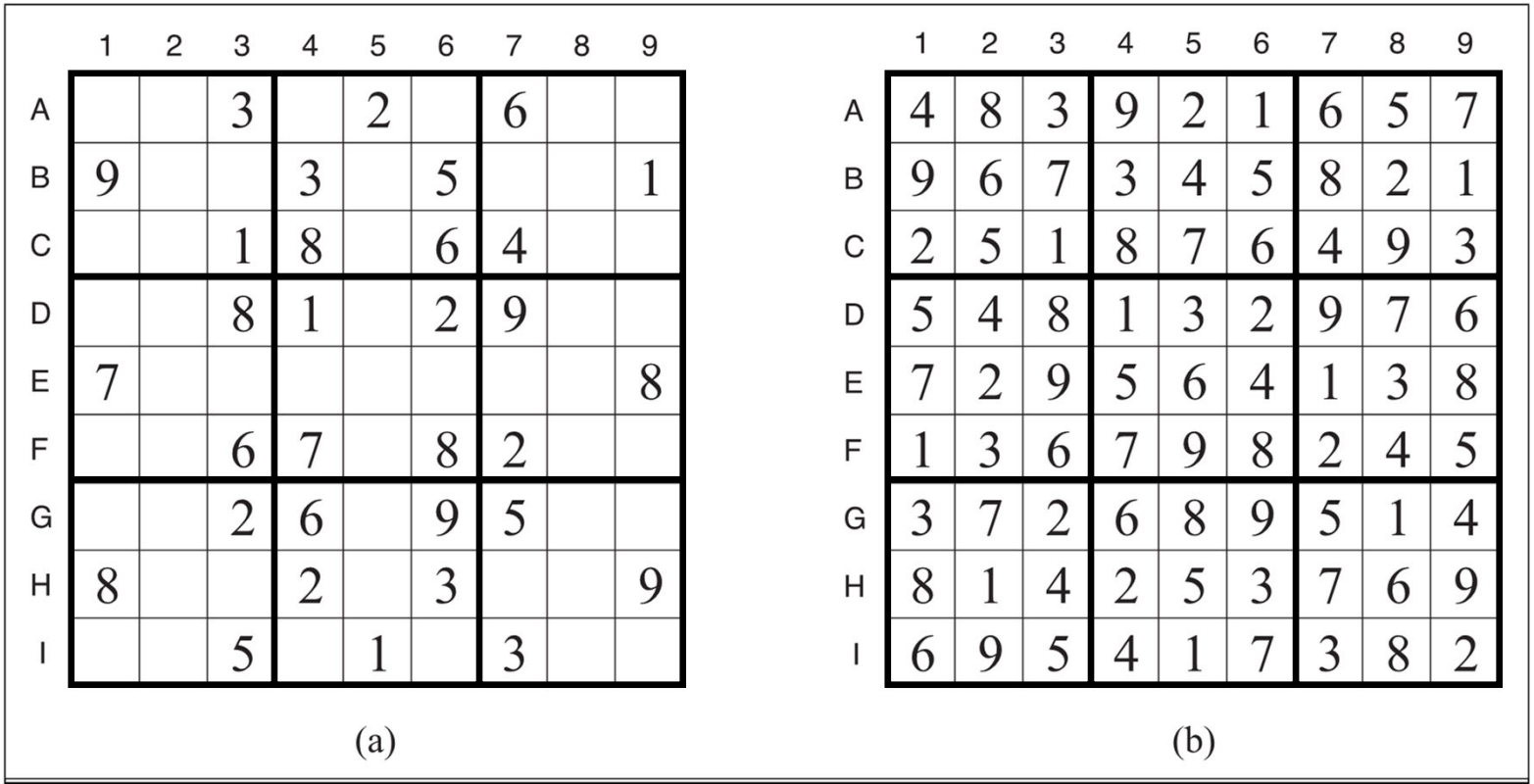
	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

	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

مؤلفه‌های تعریف مسئله‌ی «ارضای قید»

متغیرها Variables	X	۸۱ متغیر (هر مربع یک متغیر)
دامنه‌ها Domains	D	دامنه‌ی مربع خالی: $\{0,1,2,3,4,5,6,7,8,9\}$ دامنه‌ی مربع پرشده: {تک عضوی (یک رقم)}
قیدها Constraints	C	۲۷ قید *Alldiff* • مقادیر ستون‌های هر سطر باید متفاوت باشند. • مقادیر سطرهای هر ستون باید متفاوت باشند. • مقادیر هر چهارگوش ۳ در ۳ باید متفاوت باشند.

سطر، ستون یا چهارگوش: یک واحد (unit)



	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		

(a)

	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

(b)

Figure 6.4 (a) A Sudoku puzzle and (b) its solution.

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

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

هوش مصنوعی

مسائل ارضای قید

۳

جستجوی عقب‌گرد برای مسائل ارضای قید

فرمول‌بندی «مسئله‌ی ارضای قید» در قالب جستجوی استاندارد

مؤلفه‌های پنج‌گانه‌ی تعریف مسئله

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 گذار 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 ⇐ همه‌ی راه‌حل‌ها هزینه‌ی مساوی دارند.

ویژگی‌ها

- این فرمول‌بندی برای تمام مسائل ارضای قید مشابه است.
- هر راه‌حل با n متغیر در عمق n ظاهر می‌شود ⇐ استفاده از **جستجوی عمق–اول**
- مسیر راه‌حل لازم نیست ⇐ 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!!! $d = \max_i |D_i|$

جستجوی استاندارد برای حل «مسائل ارضای قید»

پیاده‌سازی

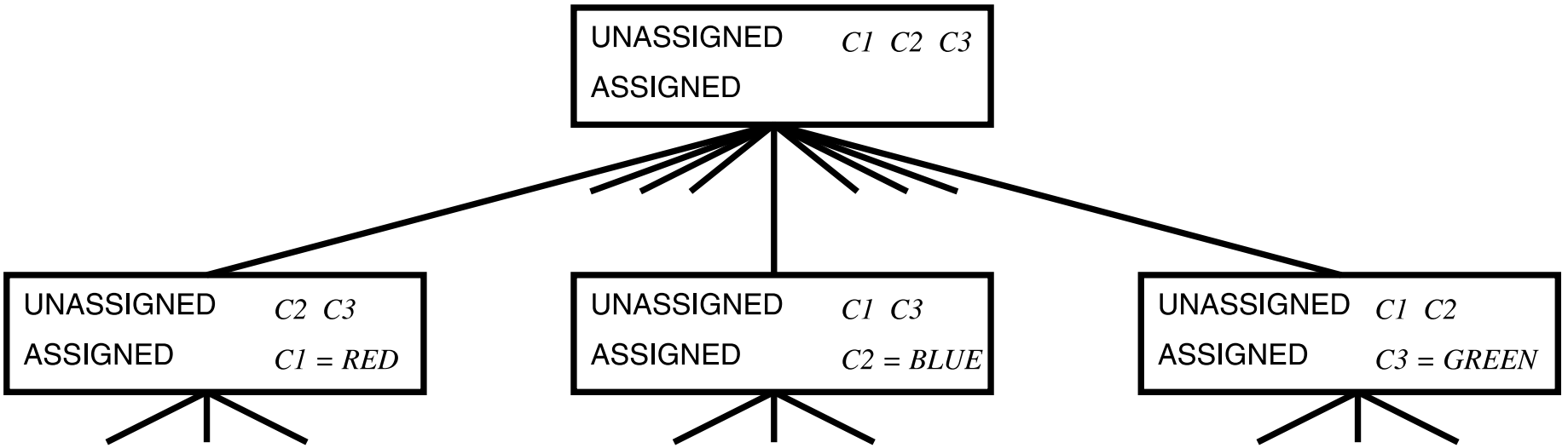
درخت جستجو برای CSP شامل گره‌هایی است که هر گره یک CSP-STATE را نگهداری می‌کند.

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 کارآیی

۴	۳	۲	۱
استراتژی جستجو *Search Strategy*	فاکتور انشعاب b	عمق درخت راه‌حل d	حداکثر عمق فضا m
جستجوی عمق–اول DFS	بیشترین مقدار در ریشه‌ی درخت: $b = \sum_{i=1}^{n} \lvert D_i \rvert$	باید همه‌ی متغیرها نسبت‌دهی شوند: $d = n$	تعداد متغیرها: $m = n$

اندازه‌گیری پیچیدگی‌های زمانی و فضایی با پارامترهای				
	b	فاکتور انشعاب حداکثر	*Branching factor*	حداکثر تعداد انشعاب یک گره در درخت جستجو
	d	عمق راه‌حل با کم‌ترین هزینه	*depth of l.c. solution*	عمق کم‌عمق‌ترین هدف
	m	حداکثر عمق فضای حالت	*Max. depth of s.s.*	حداکثر طول یک مسیر در گراف فضای حالت (شاید ∞)

«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

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

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

$$\Downarrow$$

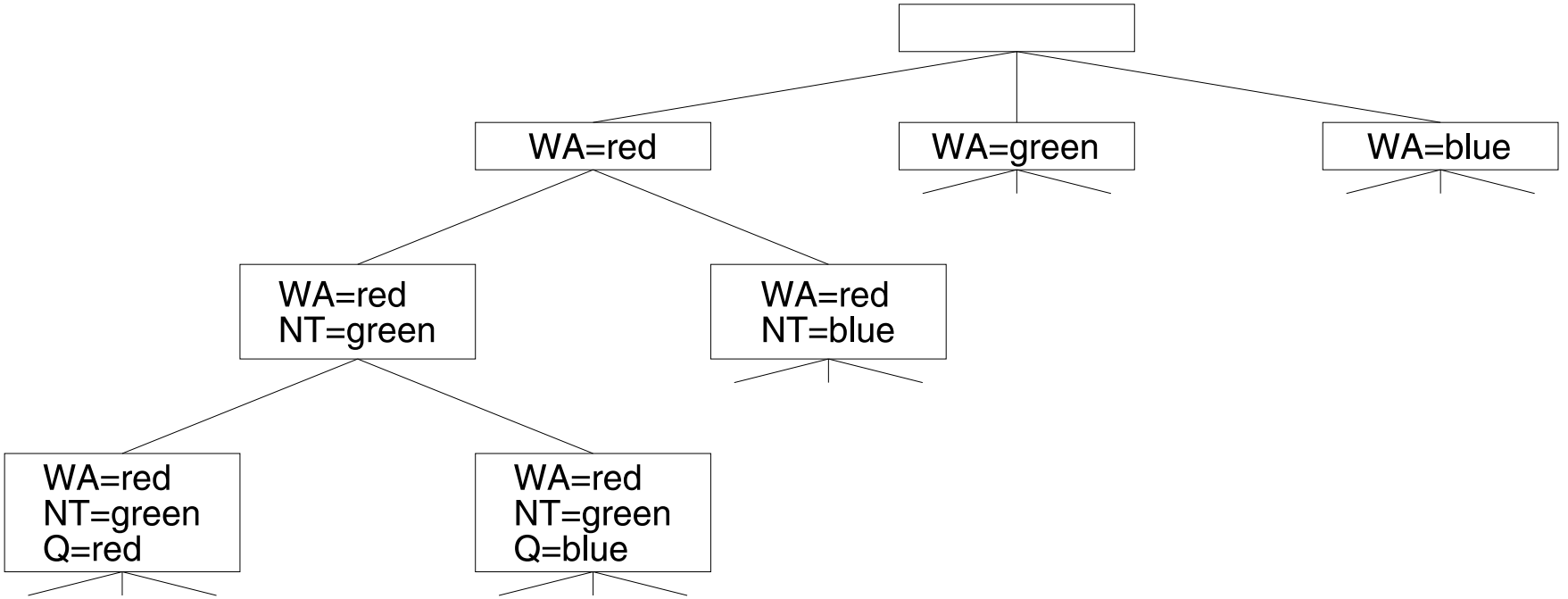
$$b = d \qquad (= \max_i \mid D_i \mid)$$

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

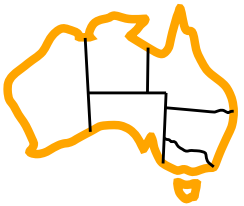
شبه‌کد

```
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
```

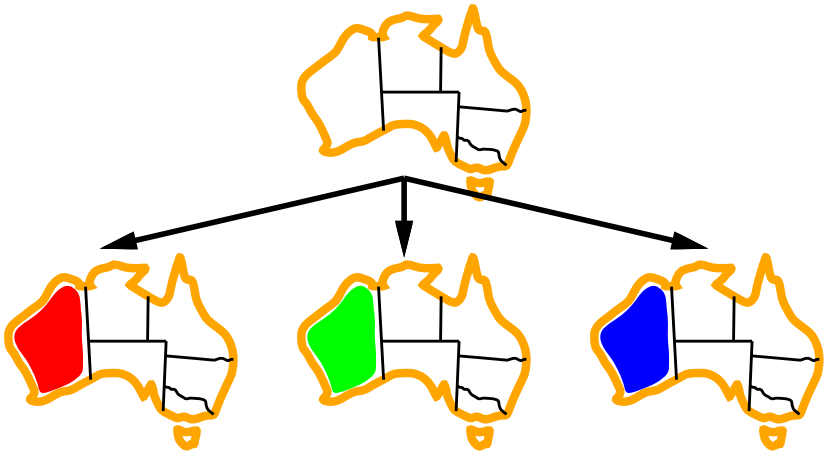
«جستجوی عقب‌گرد» برای حل مسائل ارضای قید

مثال (۱ از ۴)



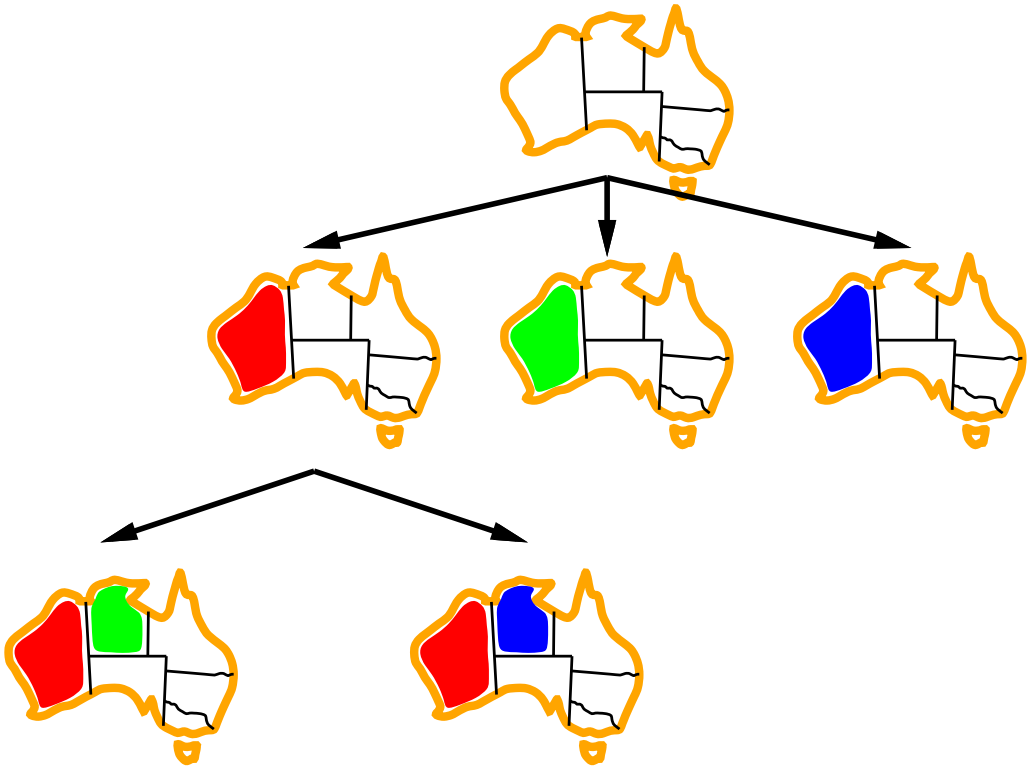
«جستجوی عقب‌گرد» برای حل مسائل ارضای قید

مثال (۲ از ۴)



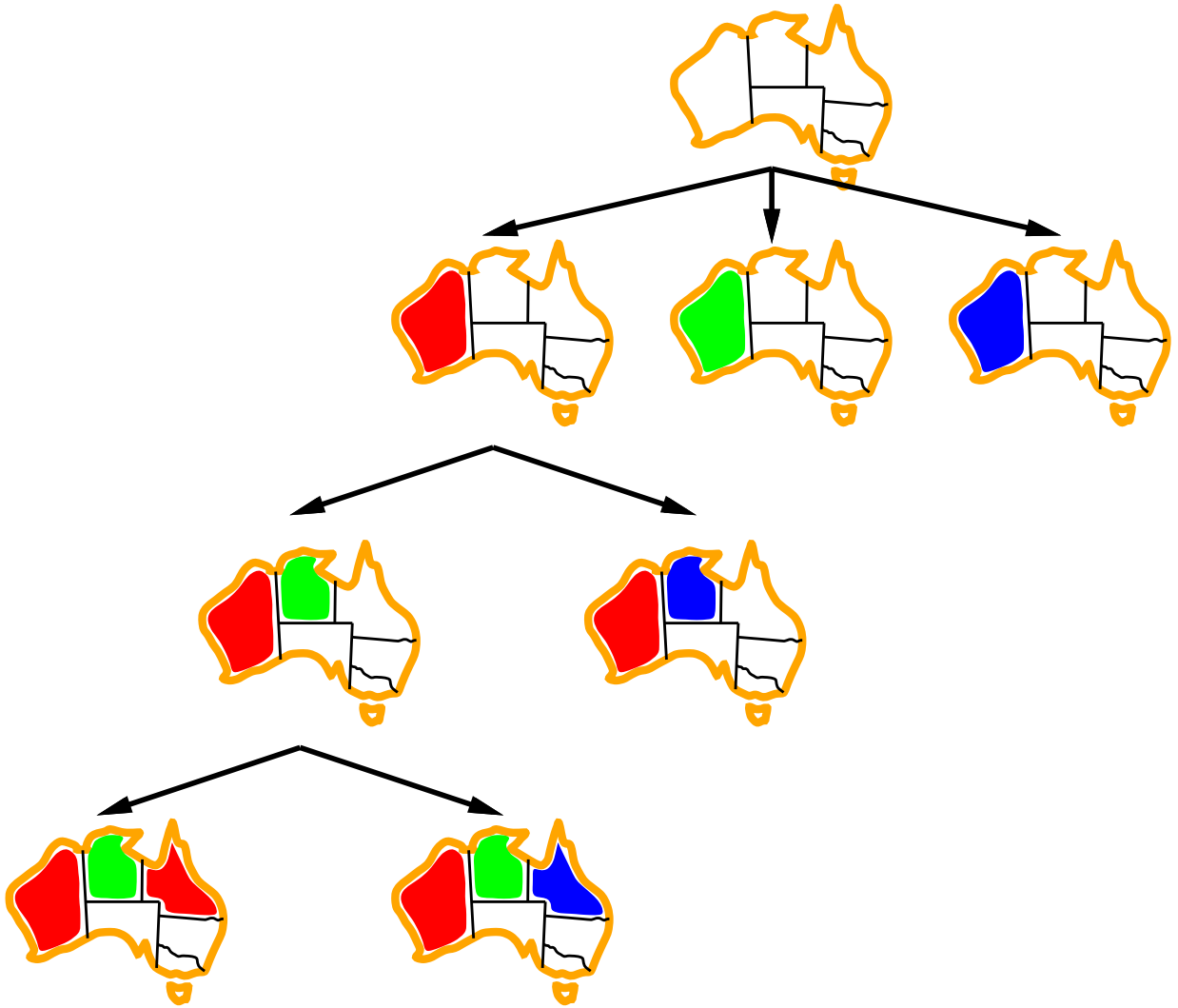
«جستجوی عقب‌گرد» برای حل مسائل ارضای قید

مثال (۳ از ۴)



«جستجوی عقب‌گرد» برای حل مسائل ارضای قید

مثال (۴ از ۴)



بهبود کارآیی 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:

۱	کدام متغیر باید در مرحله‌ی بعدی نسبت‌دهی شود؟
	SELECT-UNASSIGNED-VARIABLE
۲	مقادیر باید به چه ترتیبی آزمایش شوند؟
	ORDER-DOMAIN-VALUES
۳	چه استنتاج‌هایی باید در هر گام از جستجو انجام شود؟
	INFERENCE
۴	چگونگی اجتناب از شکست مجدد، وقتی جستجو به انتسابی شامل نقض یک قید برمی‌خورد؟

بهبود کارآیی جستجوی عقب‌گرد

ترتیب‌دهی متغیرها و مقادیر: ترفند «حداقل مقادیر باقیمانده»

حداقل مقادیر باقیمانده
Minimum Remaining Values (MRV)

متغیری را انتخاب کنید که
دارای کمترین تعداد مقادیر مجاز است

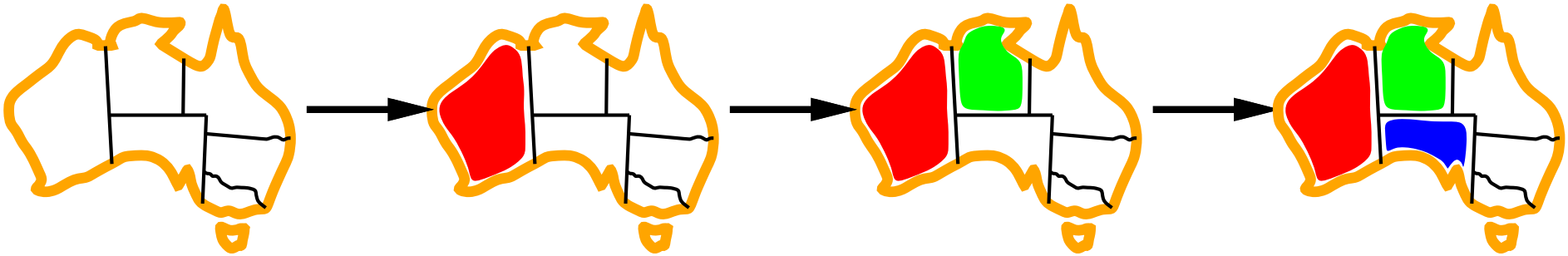
متغیر با بیشترین قید
Most Constrained Variable

بهبود کارآیی جستجوی عقب‌گرد

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

حداقل مقادیر باقیمانده
Minimum Remaining Values (MRV)

متغیری را انتخاب کنید که
دارای کمترین تعداد مقادیر مجاز است



بهبود کارآیی جستجوی عقب‌گرد

ترتیب‌دهی متغیرها و مقادیر: ترفند «هیوریستیک درجه»

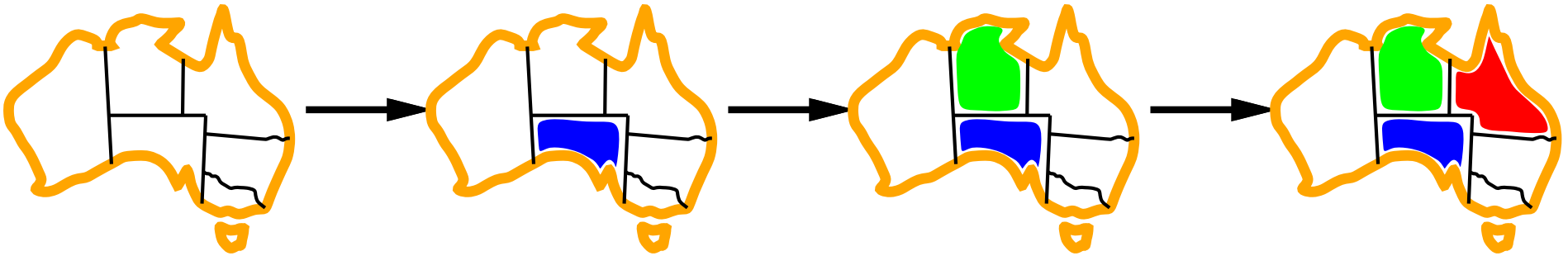
هیوریستیک درجه *Degree Heuristic*	متغیری را انتخاب کنید که بیشترین قید را بر روی متغیرهای باقیمانده ایجاد می‌کند

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

بهبود کارآیی 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»

مقدار با کمترین ایجاد قید
Least Constraining Value

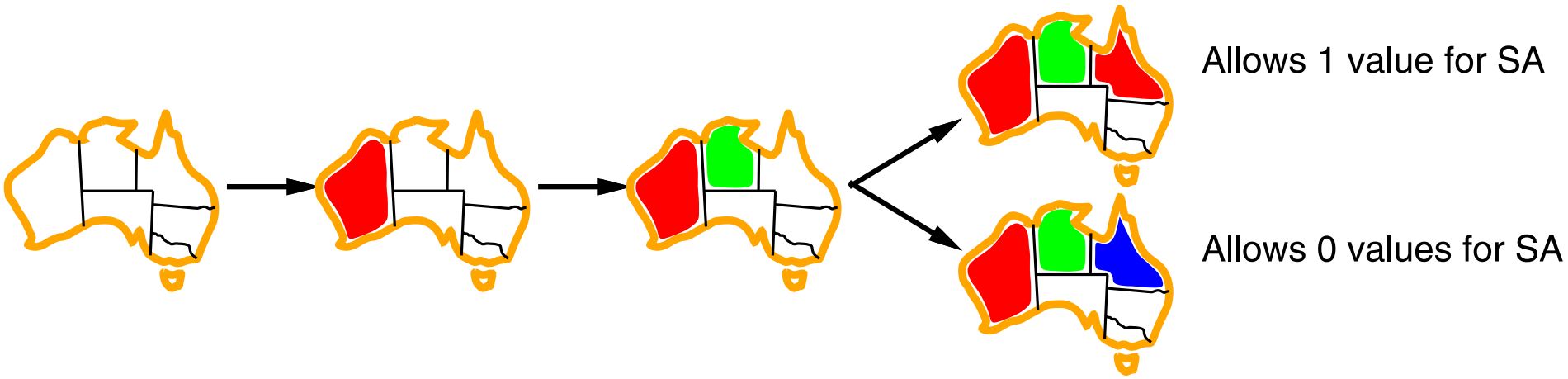
با داشتن یک متغیر، مقداری را انتخاب کنید که کمترین تعداد مقادیر را برای متغیرهای باقیمانده رد می‌کند.

بهبود کارآیی جستجوی عقب‌گرد

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

مقدار با کمترین ایجاد قید
Least Constraining Value

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



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

بهبود کارآیی 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

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.

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.

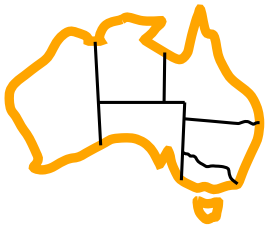
بهبود کارآیی جستجوی عقب‌گرد

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

FORWARD CHECKING

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.



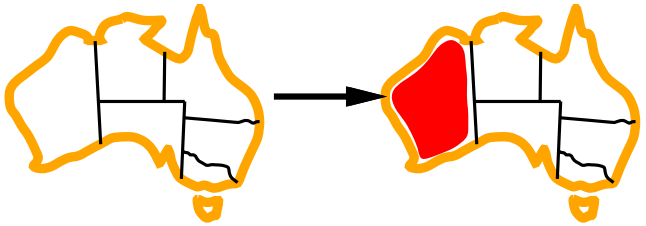
بهبود کارآیی جستجوی عقب‌گرد

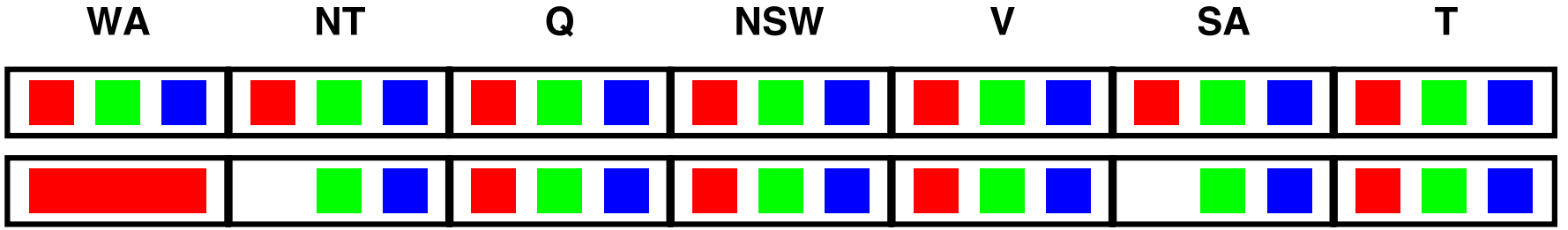
«یک در میان»سازی جستجو و استنتاج: بررسی پیش‌رو: مثال (۲ از ۴)

FORWARD CHECKING

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.





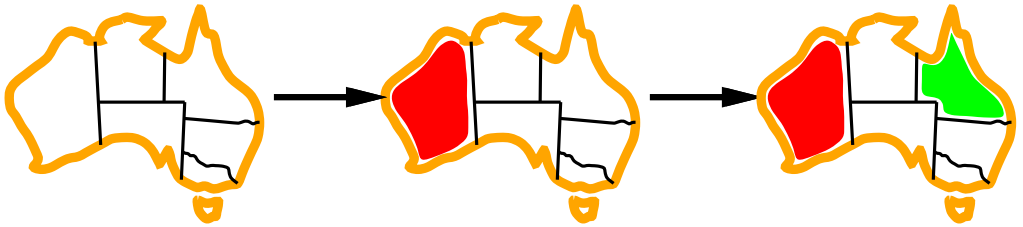
بهبود کارآیی جستجوی عقب‌گرد

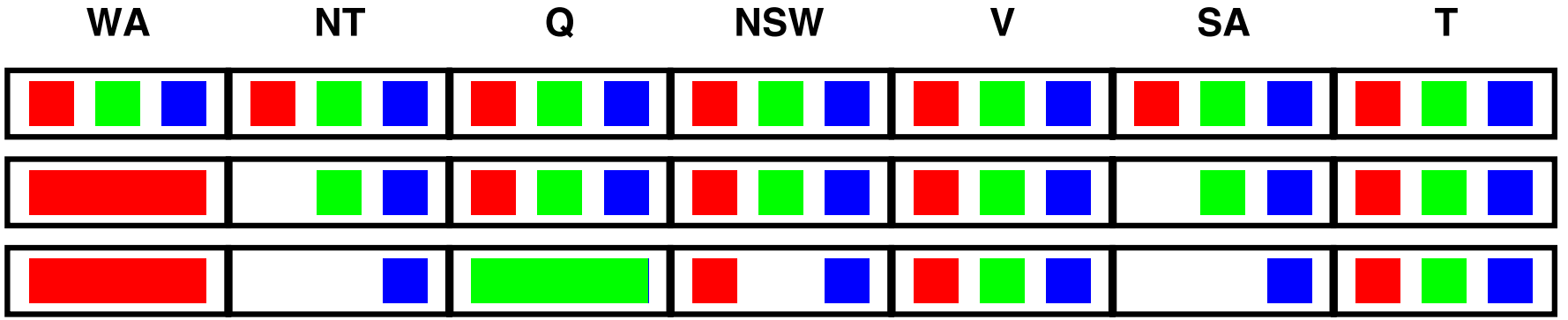
«یک در میان»سازی جستجو و استنتاج: بررسی پیش‌رو: مثال (۳ از ۴)

FORWARD CHECKING

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.





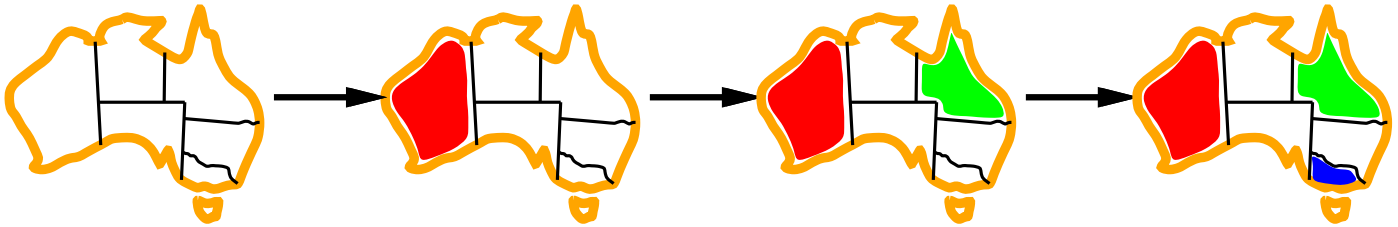
بهبود کارآیی جستجوی عقب‌گرد

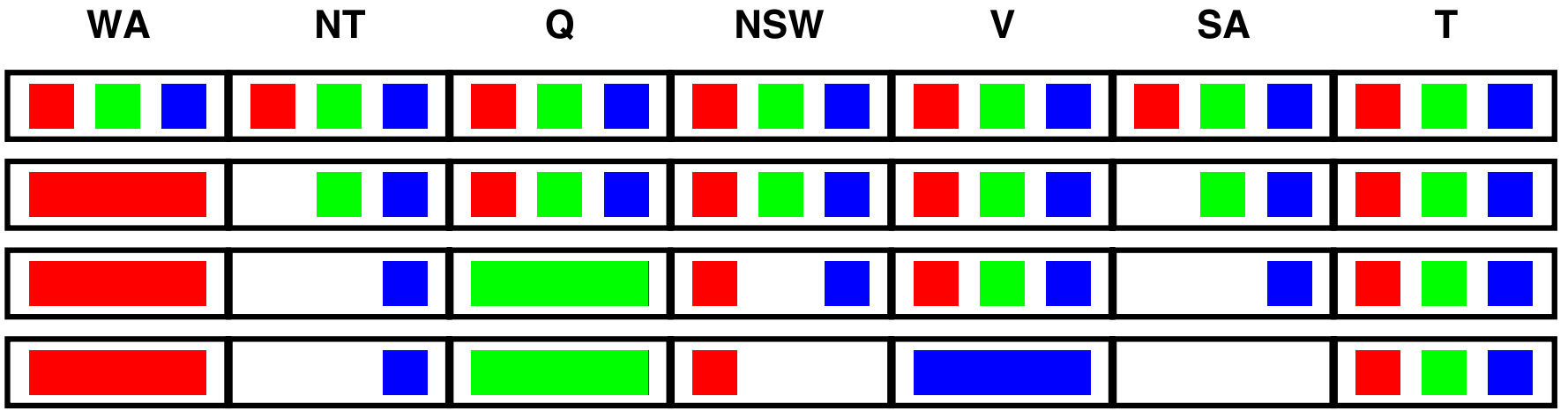
«یک در میان»سازی جستجو و استنتاج: بررسی پیش‌رو: مثال (۴ از ۴)

FORWARD CHECKING

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.





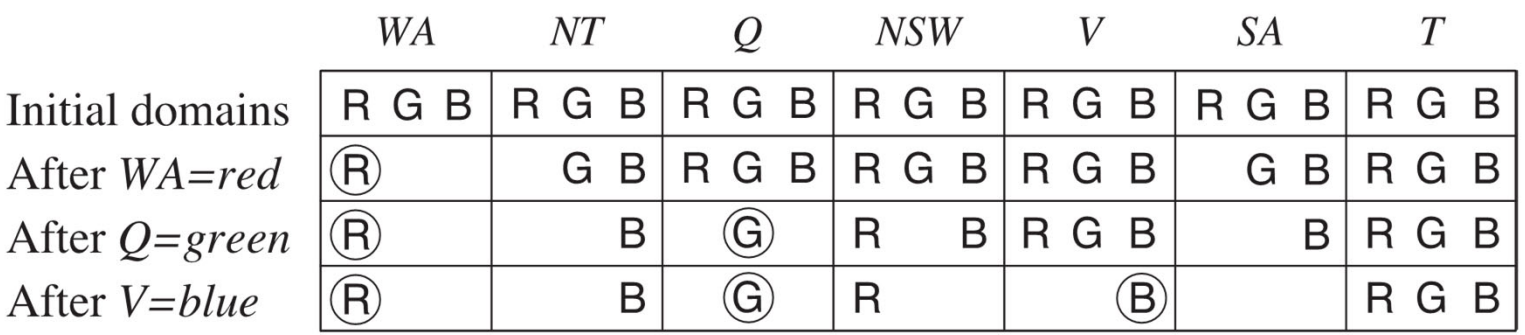


Figure 6.7 The progress of a map-coloring search with forward checking. $WA = red$ is assigned first; then forward checking deletes red from the domains of the neighboring variables NT and SA. After $Q = green$ is assigned, $green$ is deleted from the domains of NT, SA, and NSW. After $V = blue$ is assigned,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

بهبود کارآیی جستجوی عقب‌گرد

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

FORWARD CHECKING

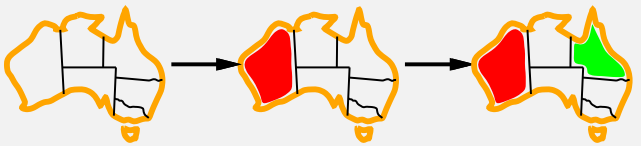
بررسی پیش‌رو
Forward Checking

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

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.

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.

هوش مصنوعی

مسائل ارضای قید

۴

جستجوی محلی برای مسائل ارضای قید

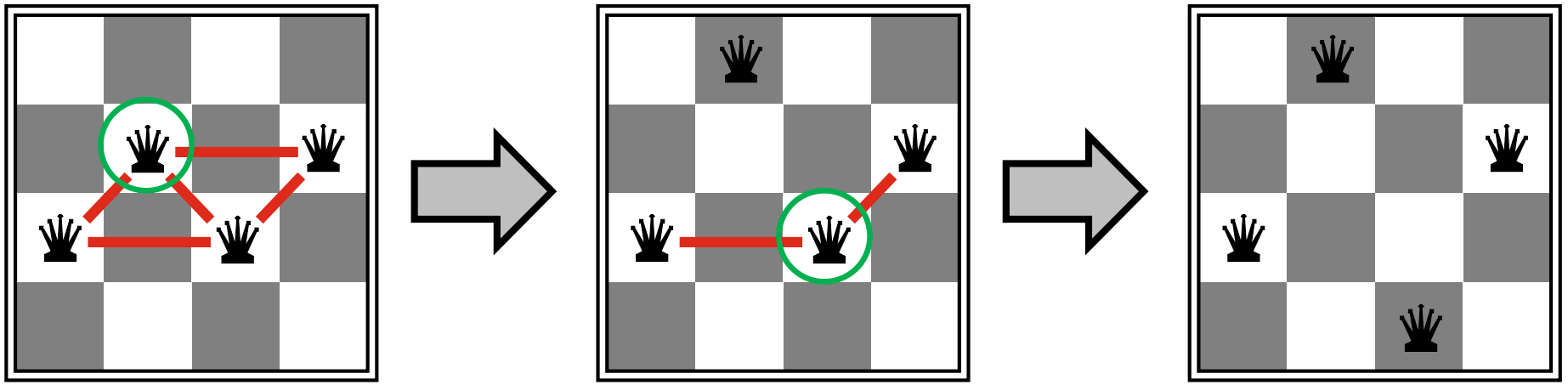
جستجوی محلی برای مسائل ارضای قید

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

جستجوی محلی برای مسائل ارضای قید

مثال: مسئله‌ی n–وزیر

تعداد تهدیدها = مجموع تعداد قیدهای نقض‌شده $= h(n)$



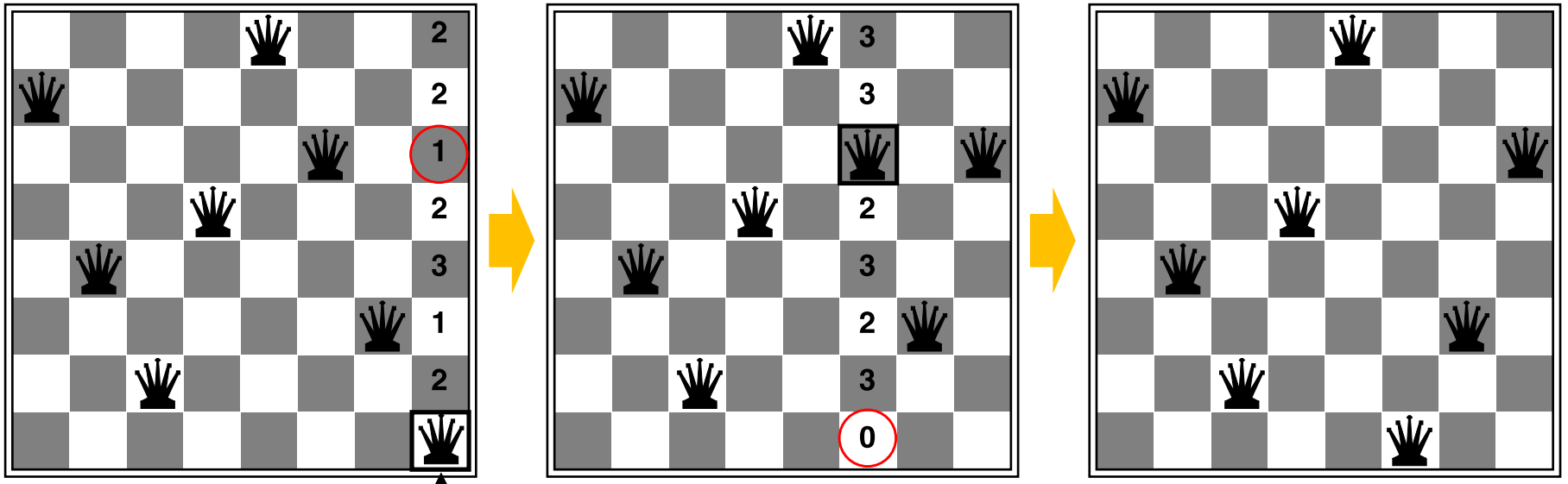
h = 5 h = 2 h = 0

جستجوی محلی برای مسائل ارضای قید

مثال: مسئله‌ی n–وزیر: استفاده از هیوریستیک حداقل تداخل‌ها

تعداد تهدیدها = مجموع تعداد قیدهای نقض‌شده $= h(n)$

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



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

الگوریتم جستجوی محلی با هیوریستیک «حداقل تداخل‌ها» برای CSPها

شبه‌کد

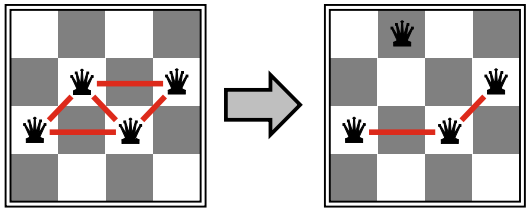
function MIN-CONFLICTS(csp, max_steps) **returns** a solution or failure
  inputs: csp, a constraint satisfaction problem
    max_steps, the number of steps allowed before giving up

  $current \leftarrow$ 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 نقض‌شده توسط یک مقدار خاص را با در نظر گرفتن سایر انتساب‌های فعلی می‌شمارد.

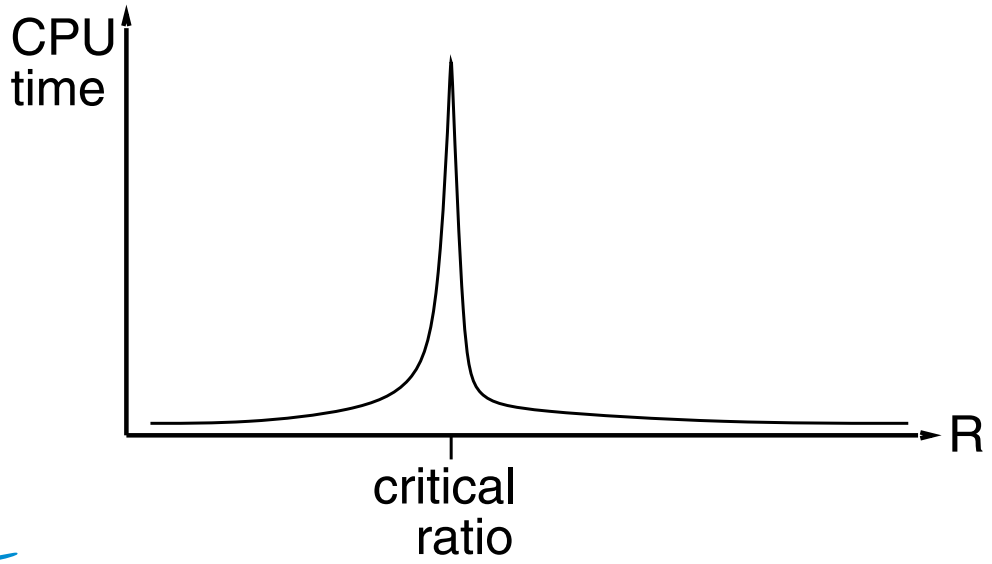
جستجوی محلی برای مسائل ارضای قید

مثال: مسئله‌ی n–وزیر: ارزیابی کارآیی



با شروع از حالت آغازین تصادفی:
امکان حل مسئله‌ی 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



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

هوش مصنوعی

مسائل ارضای قید

۵

ساختار مسئله

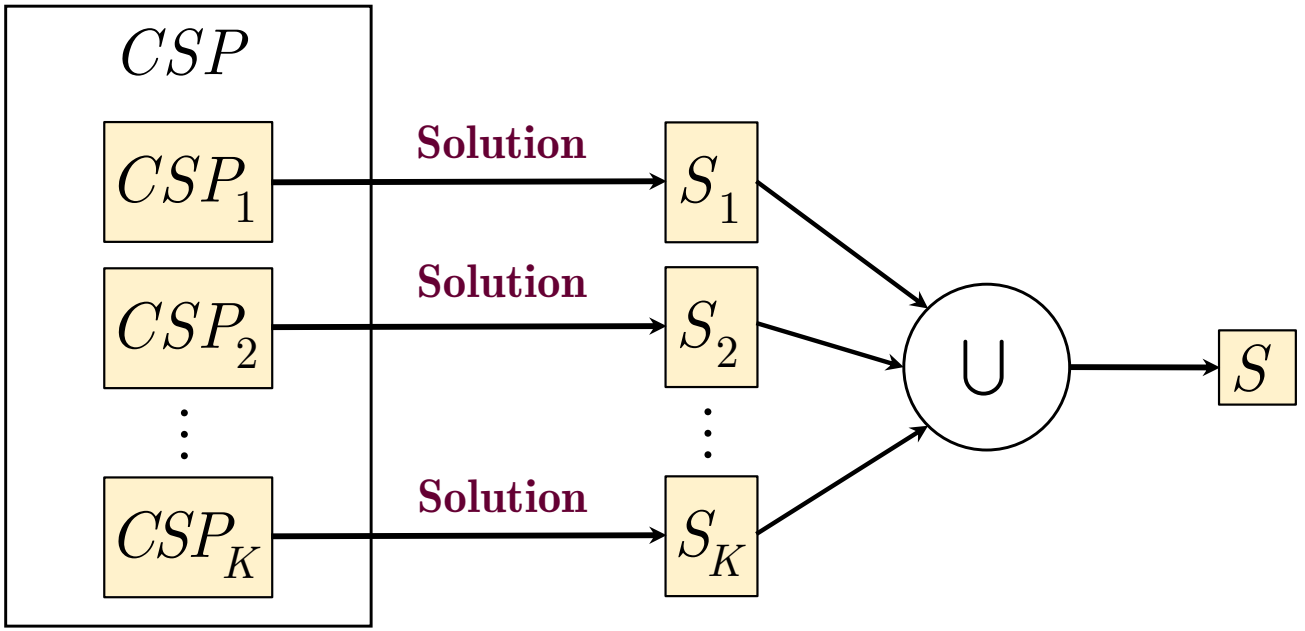
ساختار مسئله

PROBLEM STRUCTURE

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

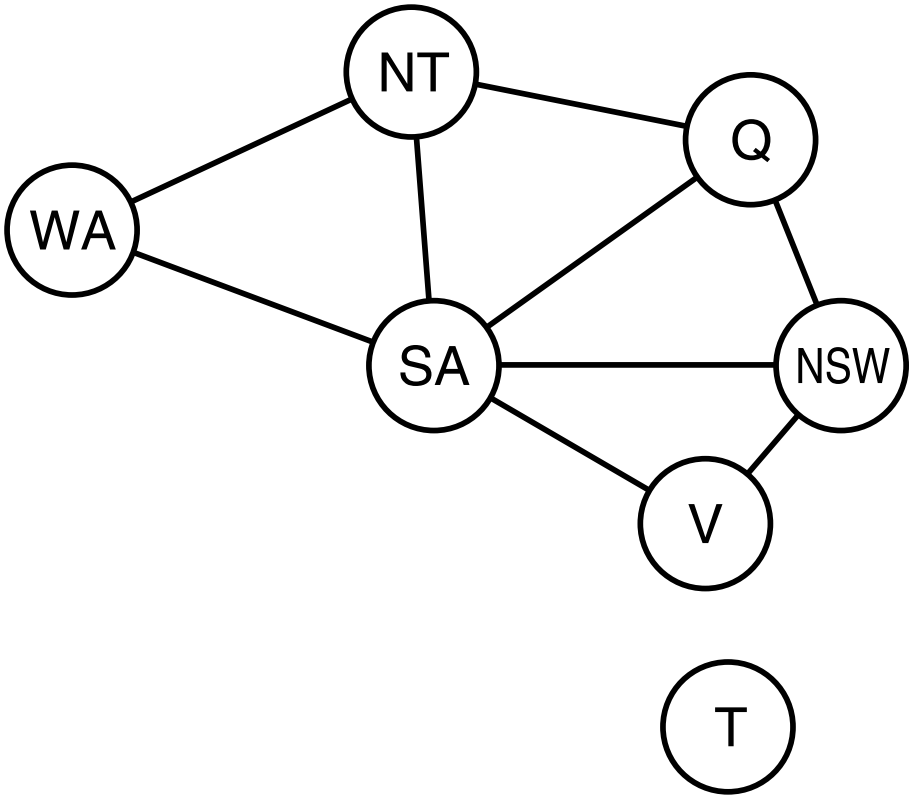
مؤلفه‌های همبند گراف قید، **زیرمسئله‌های مستقل** هستند
و می‌توانند به صورت جداگانه حل شوند.

K = تعداد زیرمسئله‌ها =
تعداد مؤلفه‌های همبند



ساختار مسئله

مثال



دو مؤلفه‌ی همبند = دو زیرمسئله‌ی مستقل

$$K = 2$$

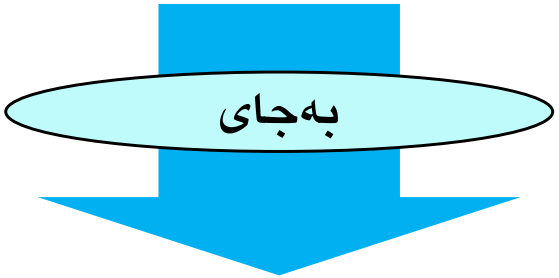
ساختار مسئله

تأثیر بر پیچیدگی زمانی راه‌حل

زمان اجرا

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

خطی بر حسب n



$$O(d^{n})$$

نمایی بر حسب n

K = تعداد زیرمسئله‌ها = تعداد مؤلفه‌های همبند
c = تعداد متغیر هر زیرمسئله
n = تعداد کل متغیرها
d = تعداد عناصر دامنه‌ی 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

قضیه

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

زمان اجرا

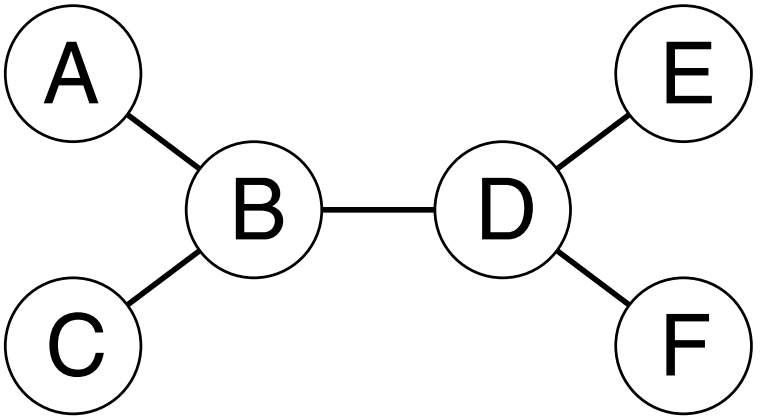
$O(n\,d^2)$

خطی بر حسب n

به‌جای

$O(d^n)$

نمایی بر حسب n



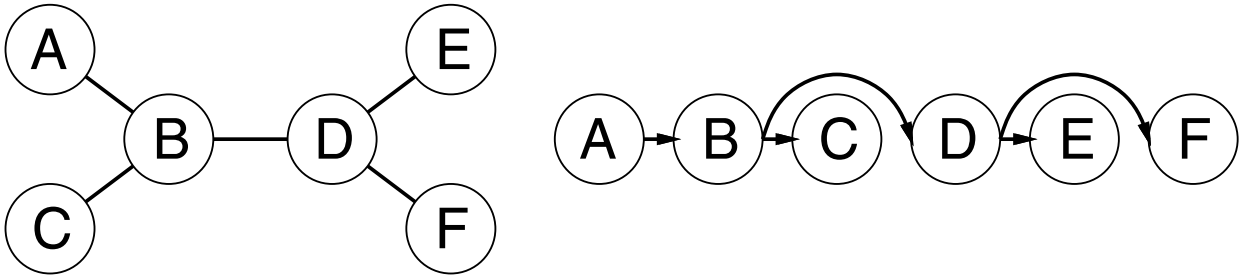
CSP با ساختار درختی

الگوریتم برای راه‌حل

ALGORITHM FOR TREE-STRUCTURED CSPS

۱ مرتب‌سازی توپولوژیکی *Topological Sort*

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



۲ حذف ناسازگاری‌ها *Remove Inconsistent*

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

۳ انتساب متغیرها *Variable Assignment*

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

تبدیل به ساختار درختی

شرط‌گذاری

CONDITIONING

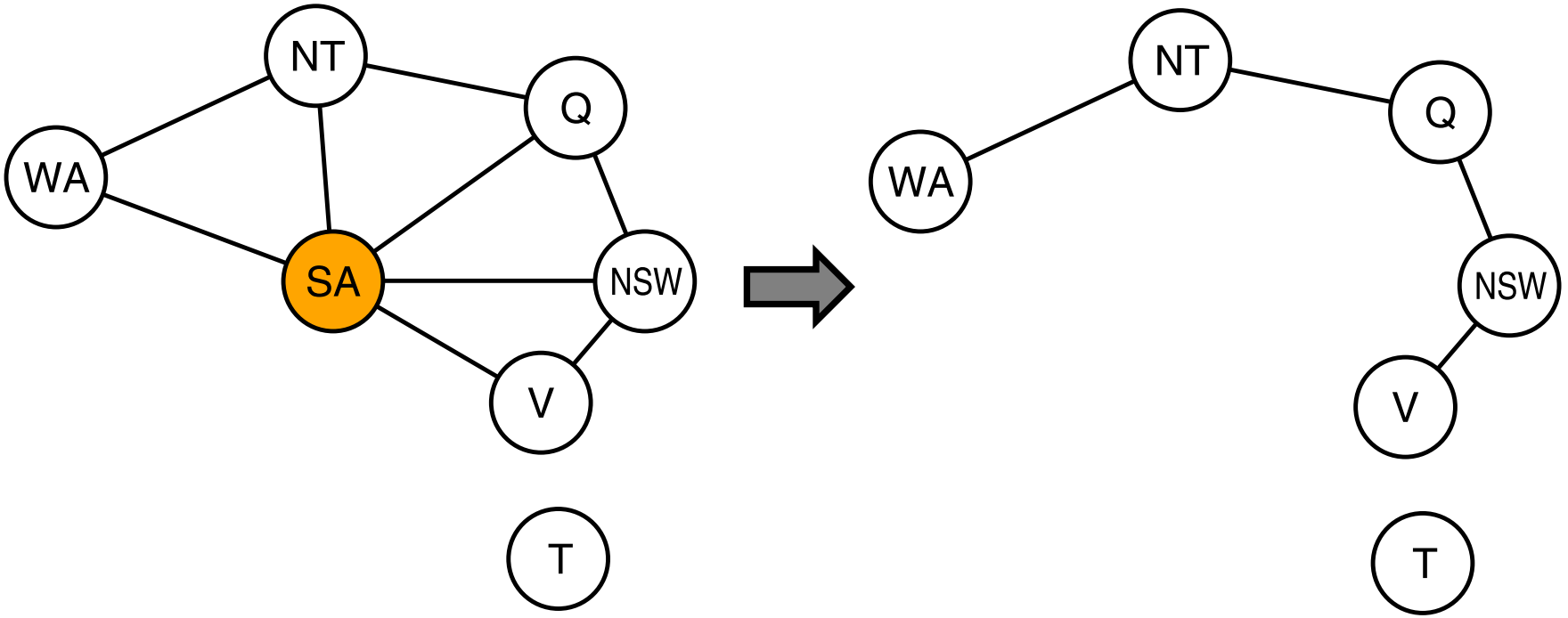
شرط‌گذاری *Conditioning*	یک **متغیر** را مقداردهی اولیه کنید، دامنه‌ی متغیرهای همسایه را هرس کنید.

تبدیل به ساختار درختی

شرط‌گذاری: مثال

CONDITIONING

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



تبدیل به ساختار درختی

شرط‌گذاری ابرگره‌ای (کات‌ستی)

CUTSET CONDITIONING

شرط‌گذاری کات‌ستی
Cutset Conditioning

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

تبدیل به ساختار درختی

شرط‌گذاری ابرگره‌ای (کات‌ستی): پیچیدگی زمانی

CUTSET CONDITIONING

شرط‌گذاری کات‌ستی
Cutset Conditioning

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

زمان اجرا

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

خطی بر حسب n

به‌جای

$$O(d^n)$$

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

تجزیه‌ی درختی
Tree Decomposition

مسئله را به یک درخت از زیرمسئله‌ها تبدیل می‌کند.

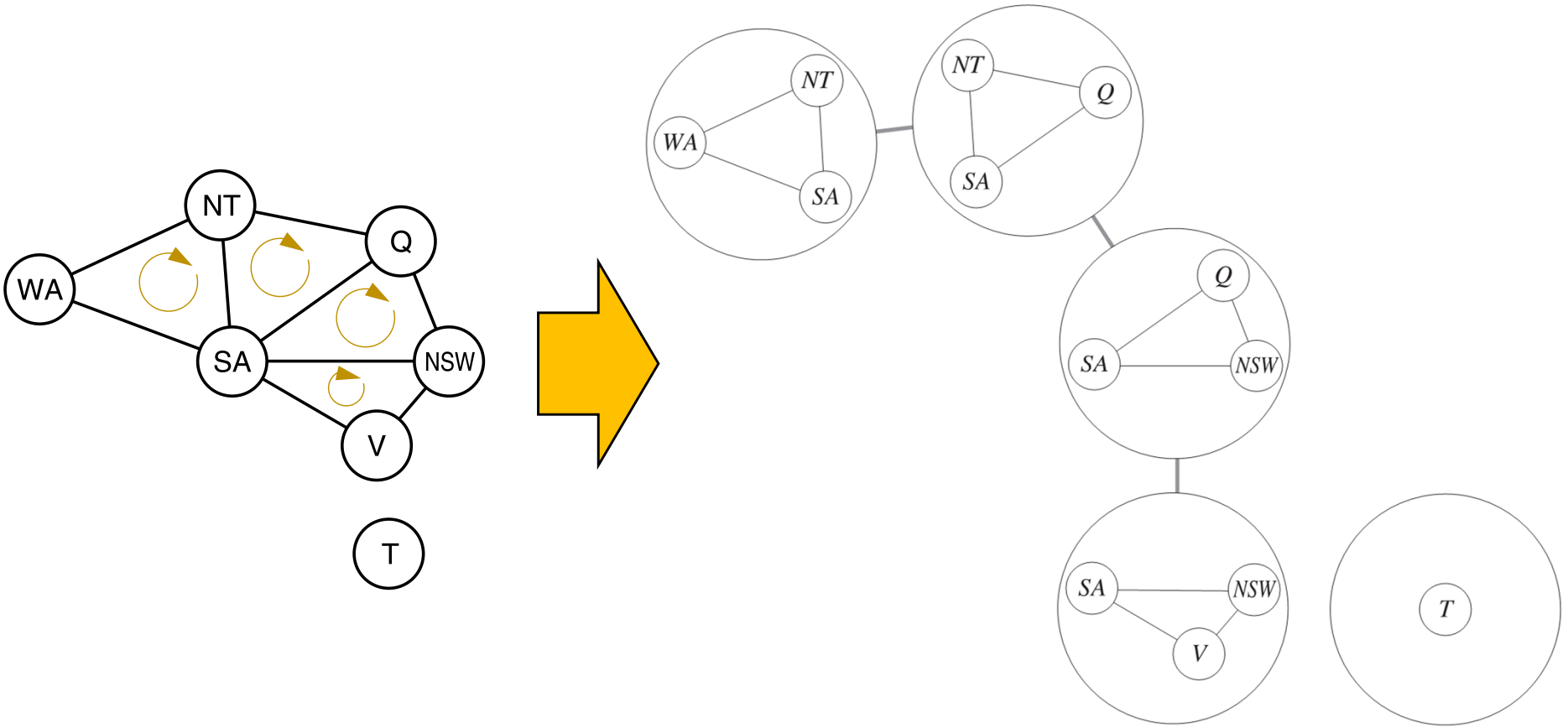
تبدیل به ساختار درختی

تجزیه‌ی درختی: مثال

TREE DECOMPOSITION

تجزیه‌ی درختی
Tree Decomposition

مسئله را به یک درخت از زیرمسئله‌ها تبدیل می‌کند.



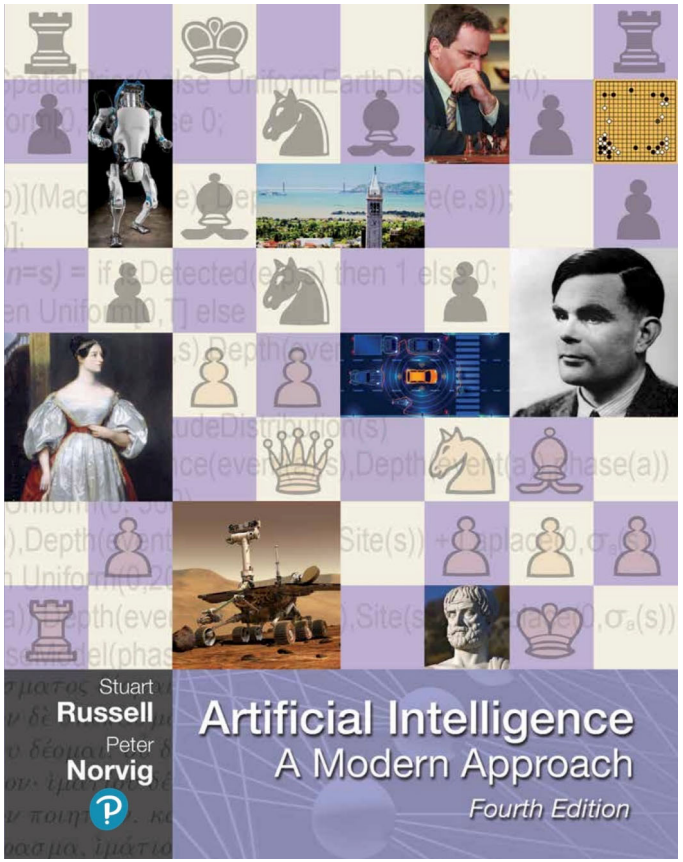
هوش مصنوعی

مسائل ارضای قید

۶

منابع

منبع اصلی



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

Chapter 6

CHAPTER 6

CONSTRAINT SATISFACTION PROBLEMS

In which we see how treating states as more than just little black boxes leads to new search methods and a deeper understanding of problem structure.

Chapters 3 and 4 explored the idea that problems can be solved by searching the state space: a graph where the nodes are states and the edges between them are actions. We saw that domain-specific heuristics could estimate the cost of reaching the goal from a given state, but that from the point of view of the search algorithm, each state is atomic, or indivisible—a black box with no internal structure. For each problem we need domain-specific code to describe the transitions between states.

In this chapter we break open the black box by using a **factored representation** for each state: a set of **variables**, each of which has a **value**. A problem is solved when each variable has a value that satisfies all the constraints on the variable. A problem described this way is called a **constraint satisfaction problem**, or **CSP**.

Constraint satisfaction problem

CSP search algorithms take advantage of the structure of states and use *general* rather than domain-specific heuristics to enable the solution of complex problems. The main idea is to eliminate large portions of the search space all at once by identifying variable/value combinations that violate the constraints. CSPs have the additional advantage that the actions and transition model can be deduced from the problem description.

6.1 Defining Constraint Satisfaction Problems

A constraint satisfaction problem consists of three components, $\mathcal{X}$, $\mathcal{D}$, and $\mathcal{C}$:

$\mathcal{X}$ is a set of variables, $\{X_1,\ldots,X_n\}$.
$\mathcal{D}$ is a set of domains, $\{D_1,\ldots,D_n\}$, one for each variable.
$\mathcal{C}$ is a set of constraints that specify allowable combinations of values.

A domain, D_i, consists of a set of allowable values, $\{v_1,\ldots,v_k\}$, for variable X_i. For example, a Boolean variable would have the domain $\{true,false\}$. Different variables can have different domains of different sizes. Each constraint $\mathcal{C}_j$ consists of a pair $\langle scope,rel\rangle$, where *scope* is a tuple of variables that participate in the constraint and *rel* is a **relation** that defines the values that those variables can take on. A relation can be represented as an explicit set of all tuples of values that satisfy the constraint, or as a function that can compute whether a tuple is a member of the relation. For example, if X_1 and X_2 both have the domain $\{1,2,3\}$, then the constraint saying that X_1 must be greater than X_2 can be written as $\langle(X_1,X_2),\{(3,1),(3,2),(2,1)\}\rangle$ or as $\langle(X_1,X_2),X_1>X_2\rangle$.

Relation