Constraint Satisfaction Problems (CSPs)

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

Contents

Contents

What are Constraint Satisfaction Problem (CSPs) 什么是约束满足问题

- CSPs are mathematical problems, defined as a set of objects whose state must satisfy a number of constraints or limitations.

  CSPs是数学问题，被定义为其状态必须满足若干约束和限制的一组对象。

- CSPs represent the entities in a problem as a homogeneous collection of finite constraints over variables, which is solved by constraint satisfaction methods.

  CSPs将问题中的实体表示为对变量进行有限约束的同质集合，通过约束满足方法加以解决。

- CSPs are the research subject in both artificial intelligence and operations research,

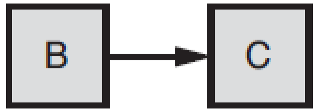
  CSPs是人工智能和运筹学共同的研究课题，

  - since the regularity in their formulation provides a common basis to analyze and solve problems of many unrelated families.

    这是因为其形式化的规则性提供了用于分析和解决许多不相关问题的共同基础。

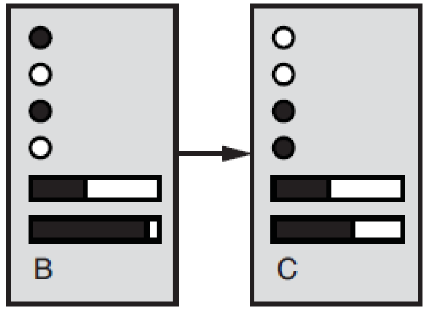
Standard Search vs. Constraint Satisfaction Problem 标准搜索与约束满足问题

- Standard search problem 标准搜索问题
  - The state is atomic or indivisible, a "black box" with no internal structure.
    其状态是原子的或不可分割的，一个没有内部结构的黑盒子。



Atomic

- Constraint satisfaction problem 约束满足问题
  - The state is a factored representation, a set of variables, each of which has a value.
    状态采用因子表示，是一系列变量，每个有相应的值。
  - Take advantage of the structure of states.
    发挥状态结构的长处。
  - Use *general-purpose* rather than *problem-specific* heuristics.
    采用一般用途而不是问题特有的启发式。



Factored

Why Study Constraint Satisfaction Problem 为什么研究约束满足问题

- CSPs often exhibit high complexity, requiring a combination of heuristics and combinatorial search methods.

  CSP通常呈现出高复杂性，需要将启发式与组合搜索方法相结合。

- Some certain forms of the CSP: 某些形式的CSP：

Boolean Satisfiability Problem (SAT)	■	布尔可满足性问题
Satisfiability Modulo Theories (SMT)	■	可满足性模理论
Answer Set Programming (ASP)	■	答案集编排

- Examples that can be modeled as a CSP: 可建模为CSP的例子：

8-queens puzzle	■	8皇后问题
Map coloring problem	■	地图着色问题
Cryptarithmetic	■	密码算术
Sudoku	■	数独

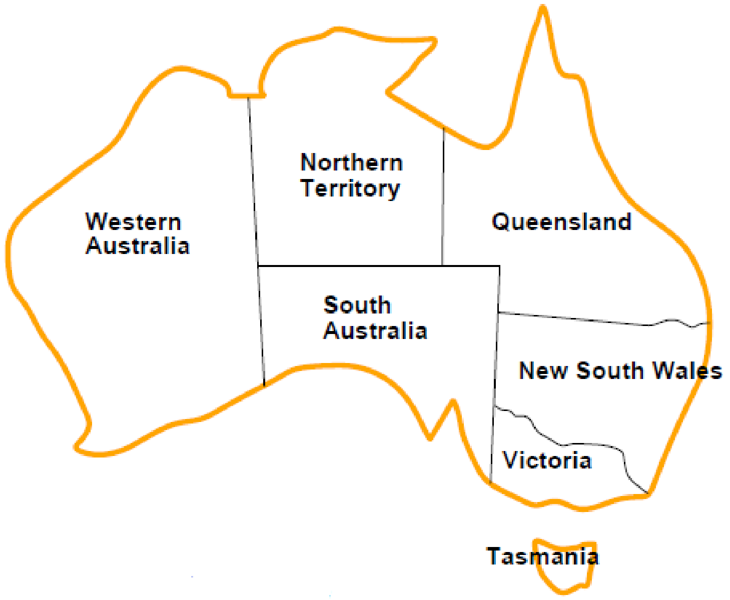
Formal Definition of CSP CSP的形式化定义

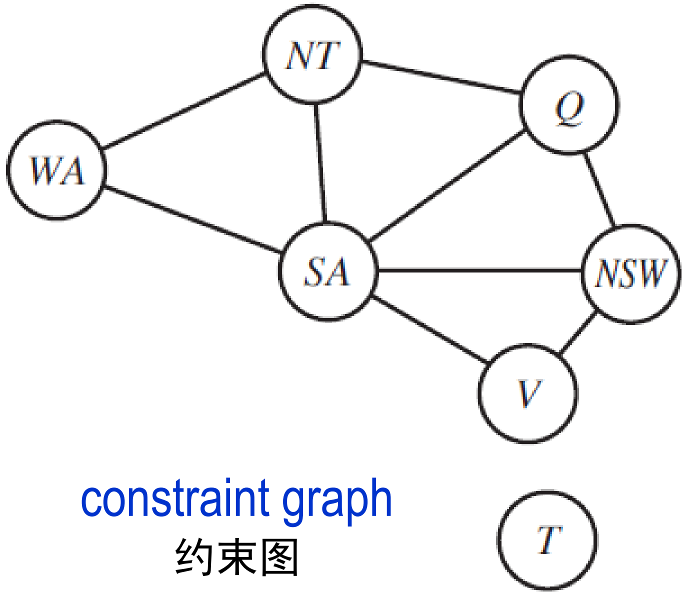
- A CPS is defined as a triple <X, D, C>,where:
  一个CSP被定义为一个三元组<X, D, C>，其中：
  - X is a set of variables, $\{X_1,..., X_n\}$
    X为变量的集合，$\{X_1,..., X_n\}$。
  - D is a set of domains, $\{D_1,..., D_n\}$. One D_i for each variable X_i ,consisting of a set of values $\{V_1,..., V_k\}$
    D为范畴的集合，$\{D_1,..., D_n\}$。每个D_1对应一个变量X_1，包含一组值$\{V_1,..., V_k\}$
  - C is a set of constrains, $\{C_1,..., C_m\}$. Each C_i consists of a pair <scope, rel> where, scope is a tuple of variables that participate in the constraint, rel is a relation that defines the values that those variables can take on.
    C为约束的集合，$\{C_1,..., C_m\}$。每个约束C_1包含一个二元组<scope, rel>, 其中，scope为参与约束的一组变量，rel是一个关系，定义这些变量可以接受的值。

Formal Definition of CSP CSP的形式化定义

- To solve a CSP, we need to define a state space and the notion of a solution.
  求解一个CSP，我们需要定义一个状态空间和解的概念。
- Each state is defined by assignment of values to variables: $\{X_i = v_i, X_j = v_j, \ldots\}$.
  每个状态都是通过对变量赋值的形式来定义的：$\{X_i = v_i, X_j = v_j, \cdots\}$。
  - Consistent assignment 一致性赋值
    a legal assignment that does not violate any constraints.
    一种不违反任何约束的合法赋值。
  - Complete assignment 完整性赋值
    every variable is assigned, and the assignment is consistent, complete.
    每个变量都被赋值，并且该赋值是一致的、完整的。
  - Partial assignment 局部性赋值
    assigns values to only some of the variables.
    仅对某些变量赋进行赋值。

Example: Map Coloring 地图着色

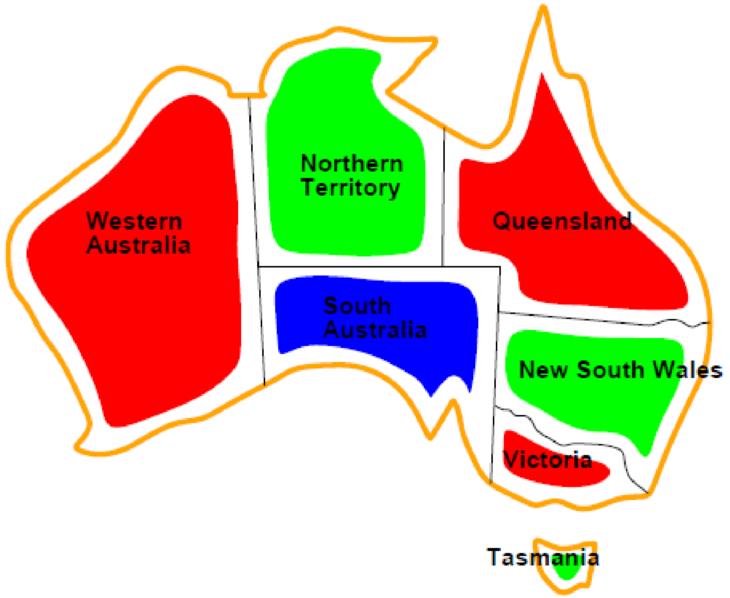


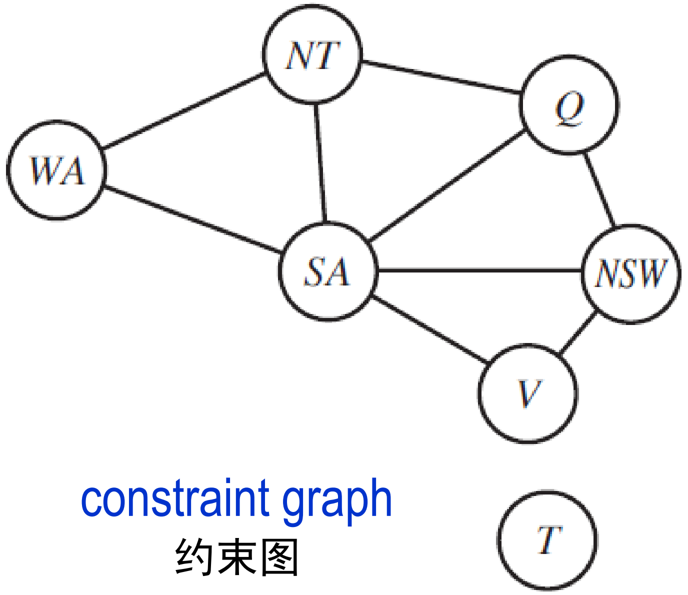


constraint graph
约束图

- Variables: 变量：
  $X = \{WA, NT, Q, NSW, V, SA, T\}$
- Domains: 范畴：
  $D_i = \{red, green, blue\}$.
- Constraints: neighboring regions should have different colors, i.e., 约束：相邻区域必须有不同的颜色，例如，
  $C = \{SA \neq WA, SA \neq NT, SA \neq Q, SA \neq NSW, SA \neq V, WA \neq NT, NT \neq Q, Q \neq NSW, NSW \neq V\}$

Example: Map Coloring 地图着色





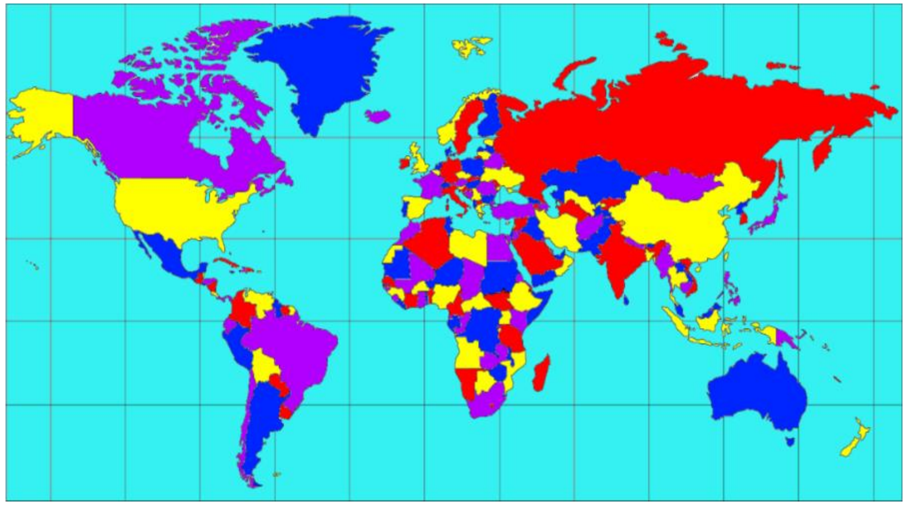
constraint graph
约束图

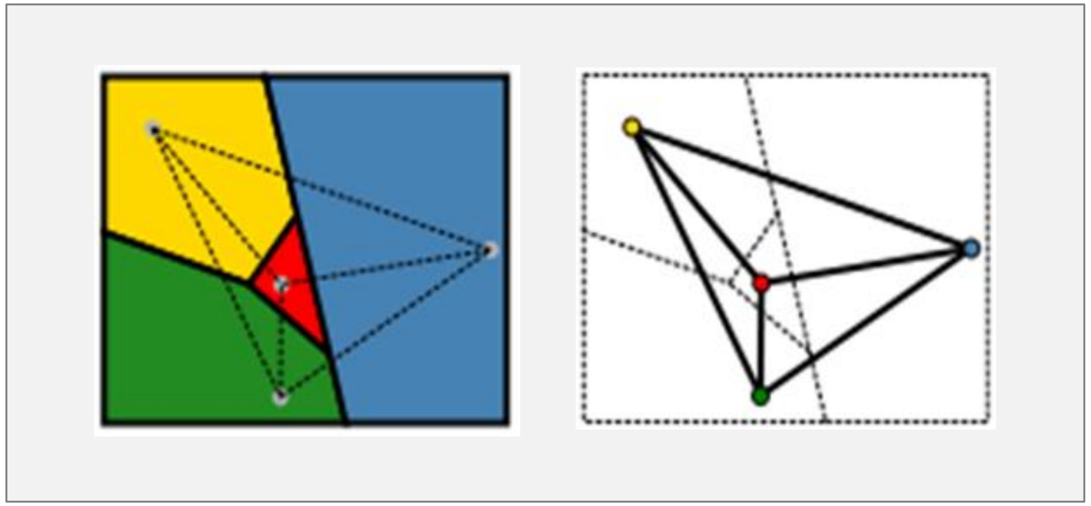
- *SA ≠ WA*, a shortcut for <*(SA, WA), SA ≠ WA*>, can be enumerated in turn as:
  SA ≠ *WA*，是<*(SA, WA), SA ≠ WA*>简写，可以有如下6种组合：
  {(red, green),(red,blue),(green, red),......,(blue,green)}
- Many possible solutions to this problem, e.g.,
  这个问题有许多种可能的解，例如：
  {(WA=red, NT=green,Q=red, NSW=green, V=red, SA=blue, T=green).}

Appendix: Four Color Theorem 四色定理

- Given any separation of a plane into contiguous regions, called a map, the regions can be colored using at most four colors so that no two adjacent regions have the same color.

  给定任意平面分割而成的相邻区域，称为地图，这些区域可以使用至多四种颜色加以着色，以致于不存在两个邻近区域具有同样的颜色。


Map of the world using just four colors
仅用四种颜色的世界地图


Precise formulation of the theorem
该定理的精确表述

Example: Job-shop scheduling 作业车间调度

- Factories have the problem of scheduling jobs, subject to various constraints.
  工厂有调度作业的问题，受各种因素制约。
- Consider the problem of scheduling the assembly of a car:
  考虑调度汽车装配的问题：
  - whole job is composed of tasks, each task is as a variable, where the value of each variable is the time that the task starts.
    整个工作由任务组成，每个任务作为一个变量，其中每个变量的值为该任务开始的时间。
  - Constraints: 约束条件：
    - one task must occur before another,
      一个任务必须在其它任务之前发生，
    - a task takes a certain amount of time to complete.
      任务需要一定的时间来完成。

Example: Job-shop scheduling 作业车间调度

- Consider a part of the car assembly, consisting of 15 tasks:
  设有一部分汽车装配问题，包含15个任务：
  - install two axles (front and back), 安装两个车轴（前、后），
  - affix all four wheels (right, left, front and back), 装上四个车轮（左、右、前、后），
  - tighten nuts for each wheel, 拧紧每个车轮的螺母，
  - affix four hubcaps, and 装上四个轮毂罩，
  - inspect the final assembly. 检查最后的组装。
- The tasks can be represented with 15 variables:
  该任务可以用15个变量表示：

$$X = \{Axle_F, Axle_B, Wheel_{RF}, Wheel_{LF}, Wheel_{RB}, Wheel_{LB}, \\ Nuts_{RF}, Nuts_{LF}, Nuts_{RB}, Nuts_{LB}, Cap_{RF}, Cap_{LF}, Cap_{RB}, Cap_{LB}, Inspect\}$$

Example: Job-shop scheduling 作业车间调度

- Next represent precedence constraints between tasks.
  下面表示任务间优先级约束条件：
  - The axles have to be in place before wheels are put on (10 minutes), so:
    在车轮安装之前（10分钟）车轴必须就位：

$$Axle_F + 10 \le Wheel_{RF}\,; \quad Axle_F + 10 \le Wheel_{LF}\,;$$
$$Axle_B + 10 \le Wheel_{RB}\,; \quad Axle_B + 10 \le Wheel_{LB}\,.$$

  - For each wheel, must affix the wheel (1 minute), then tighten nuts (2 minutes), and finally attach hubcap:
    对每个车轮，必须先装上该车轮（1分钟），再拧紧螺丝（2分钟），最后再盖上轮毂罩：

$$Wheel_{RF} + 1 \le Nuts_{RF}\,; \quad Nuts_{RF} + 2 \le Cap_{RF}\,;$$
$$Wheel_{LF} + 1 \le Nuts_{LF}\,; \quad Nuts_{LF} + 2 \le Cap_{LF}\,;$$
$$Wheel_{RB} + 1 \le Nuts_{RB}\,; \quad Nuts_{RB} + 2 \le Cap_{RB}\,;$$
$$Wheel_{LB} + 1 \le Nuts_{LB}\,; \quad Nuts_{LB} + 2 \le Cap_{LB}\,.$$

Example: Job-shop scheduling 作业车间调度

- Need a disjunctive constraint that $Axle_F$ and $Axle_B$ must not overlap in time; either one comes first or the other does:
  需要一个析取约束条件，即$Axle_F$ 和$Axle_B$ 时间上不能重叠；要么一个先做、或者另一个做。
  $$Axle_F + 10 \le Axle_B \quad \text{or} \quad Axle_B + 10 \le Axle_F$$
- Also need to assert that the inspection comes last and takes 3 minutes. For every variable except inspect we add a constraint of the form.
  还需要确保检验放在最后并且需要3分钟。除了检查每个变量，还要增加如下形式的约束：
  $$X + d_X \le Inspect$$
- Finally, suppose there is a requirement to get the whole assembly done in 30 minutes. We can achieve that by limiting the domain of all variables:
  最后，假设需要全部装配在30分钟完成。我们可以通过限制所有变量的范畴来实现：
  $$D_i = \{1, 2, 3, \ldots, 27\}$$

Why Formulate a Problem as a CSP

- ☐ CSP is a natural representation for a wide variety of problems.

  CSP是对很多种问题的一种自然表示。

- ☐ CSP-solving system is easier to solve a problem than another search technique.

  与其它搜索技术相比，CSP求解系统更容易解决问题。

- ☐ CSP solver can be faster than state-space searchers, since it can quickly eliminate large swatches of the search space.

  CSP求解系统比状态空间搜索更快，因为它可以快速排除大的搜索空间样本。

Example:

Once we chose $\{SA = blue\}$ in the Australia problem, we can conclude that none of the 5 neighboring variables can take on the value $blue$, therefore:

在澳大利亚问题中一经选择{*SA* = *blue*} ，就可以得到结论，5个相邻变量不能再选择*Blue*，因此：

$$3^5 = 243 \text{ assignments} \quad \Longrightarrow \quad 2^5 = 32, \text{ a reduction of } 87\%.$$

Why Formulate a Problem as a CSP

- In regular state-space search 常规的状态空间搜索
  - (We can only ask) is this specific state a goal? No? What about this one?
    （我们仅能查询）这个特定状态是目标吗？ 不是？那么这一个是什么？
- With CSP 采用CSP
  - Once we find out that a partial assignment is not a solution, we can immediately discard further refinements.
    一旦我们发现某个局部赋值不是解的话，我们可以迅速放弃进一步的努力。
  - And, we can see *why* the assignment is not a solution —— we see which variables violate a constraint.
    并且，我们能看到为什么该赋值不是解 —— 我们查看哪个变量违反约束。
  - So, many problems can be solved quickly when formulated as a CSP.
    故，当形式化为一个CSP的话，许多问题都可以快速得到解决。

Variations on the CSP Formalism CSP形式的变体

Domains 范畴

	Discrete 离散	Continuous 连续
Finite 有限	Map coloring problem 地图着色问题	none
Infinite 无限	Set of integers or strings 整数或字符串集合	none

Constraints 约束

	unary 1元	*binary* 2元	*n-ary* n元
Linear 线性	$\langle (SA), SA \neq green \rangle$	$SA \neq WA$	$Alldiff\ (F, T, U, W, R, O)$
Nonlinear 非线性	no algorithm exists 算法不存在		

Cryptarithmetic 算式迷

- It is a type of mathematical game consisting of a mathematical equation among unknown numbers, whose digits are represented by letters.
  是一种数学游戏，由未知数字之间的数学等式组成，这些数字被表示成字母。
  00
- The goal is to identify the value of each letter.
  其目标是识别出每个字母的值。

Cryptarithmetic is also known as:
算式迷还被称为：

Alphametics	🟥	字母算数
Verbal arithmetic	🟥	覆面算
Word addition	🟥	文字加法
Cryptarithm	🟥	隐算数

Example: Some Cryptarithmetic Problems 某些算式迷问题

- SEND + MORE = MONEY
- HALF + HALF = WHOLE
- THREE + THREE + ONE = SEVEN
- ONE + THREE + FOUR = EIGHT

- 客上天然居×4 = 居然天上客
- 海上明月×9 = 月明海上
- 车马炮×炮 = 相马炮
- 谜 + 字谜 + 数字谜 + 解数字谜 + 善解数字谜 = 巧解数字谜

Example: A Cryptarithmetic Formatted as a CSP

$$TWO + TWO = FOUR$$

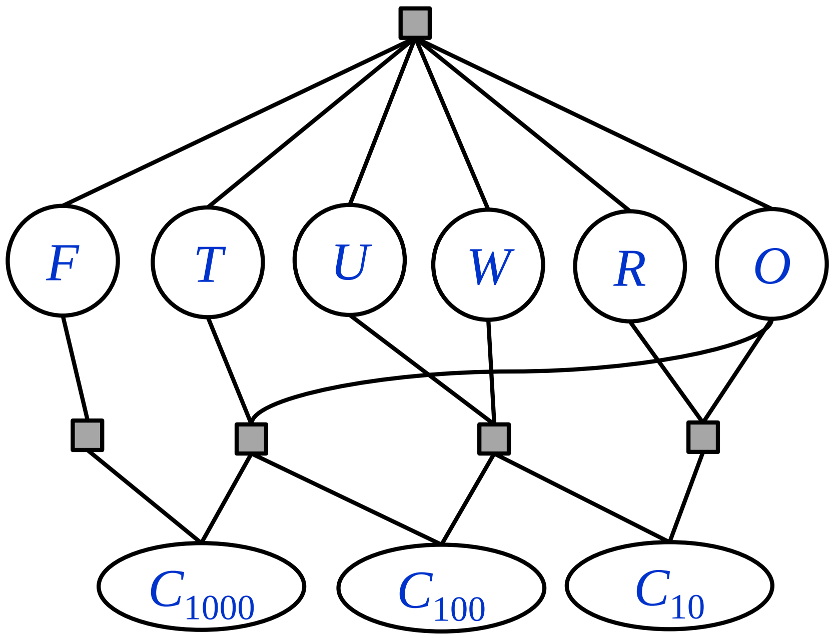
- Variables 变量： $X = \{F, T, U, W, R, O, C_{10}, C_{100}, C_{1000}\}$
- Domains 范畴：$D_i = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Global Constraints 全局约束： *Alldiff* (F, T, U, W, R, O)
- Addition constraints 附加约束：(*n-ary*)

$$O + O = R + 10 \cdot C_{10}$$
$$C_{10} + W + W = U + 10 \cdot C_{100}$$
$$C_{100} + T + T = O + 10 \cdot C_{1000}$$
$$C_{1000} = F$$

where C_{10}, C_{100}, and C_{1000} are auxiliary variables, representing the digit carried over into the tens, hundreds, or thousands column.

其中C_{10}、C_{100}、以及C_{1000} 是辅助变量，表示该数字进位到十位、百位、或者千位。

a solution 一个解：938 + 938 = 1876



Constraint hypergraph
约束超图

A Theoretical Aspect of CSPs: Decision Problems CSPs的理论方面：决策问题

- CSPs are also studied in
  - *computational complexity theory* and *finite model theory*.

  CSPs也是计算复杂性理论和有限模型论所要研究的问题。
- Most classes of CSPs that are known to be tractable are those

  大多数被认为是易处理的CSPs是这样一些问题：
  - where the hypergraph of constraints has bounded tree width, or

    约束的超图具有有界的树宽，或者
  - where the constraints exist essentially *non-unary polymorphisms* of the set of constraint relations.

    约束本质上存在约束关系集的非一元多态性。
- Every CSP can also be considered as a *conjunctive query containment problem*.

  每个CSP也可以被看作是合取查询包含问题。

Real-World CSPs 现实世界的CSPs

- Many real-world problems involve real-valued variables:
  许多现实世界的问题涉及实值变量：

Assignment problems	■	分配问题
Timetabling problems	■	时间表问题
Hardware configuration	■	硬件配置
Transportation scheduling	■	运输调度
Factory scheduling	■	工厂调度
Circuit layout	■	电路布线
Floor planning	■	楼层规划
Fault diagnosis	■	故障诊断

Resolution of CSPs CSPs的解

- CSPs on finite domains are typically solved using a form of search.
  有限范畴中的CSPs问题通常采用某种形式的搜索来解决。
- The most used techniques are variants of
  最常用的技法如下：
  - constraint propagation,
    约束传播
  - backtracking,
    回溯
  - local search.
    局部搜索。

We will discuss the three techniques in next three sections

Thank you for your attention!

PoAI