# Constrained Problems

Christos Dimitrakakis

March 18, 2024

# Outline

# Optimisation on graphs

## Discrete optimisation

- Shortest path.
- Meeting scheduling.
- Travelling salesman.
- Graph colouring.
- Bipartite matching.
- Spanning trees.

## Continuous optimisation

- Maximum flow: inequality constraints
- Minimum-cost flow: equality constraints.

# Constrained Problems

# Constrained Satisfaction Problems

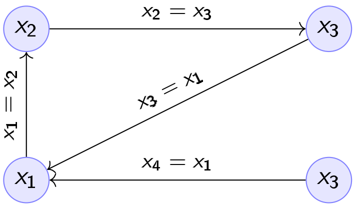
## Variables

- A set of variables $\{x_1, \ldots, x_n\} \in X$
- Each variable can take values in $x_i \in X_i$ (it's domain)

## Binary constraints

- $c_{i,j} : X_i \times X_j \to \{0, 1\}$.
- A constraint $c_{ij}(x_i)$ is violated when it has the value 1.

## Graph representation




- Goal: Find $x \in \prod_i X_i$ so that $c = 0$.

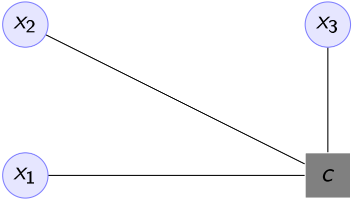
## Example: Graph colouring

# General constraints

## Example: Meeting scheduling

- Let $x_1, x_2, x_3$ be the time three people decide to go to a meeting.
- They can only meet if $x_1 = x_2$ and $x_2 = x_3$ and $x_3 = x_1$
- Instead of 3 binary constraints, use one constraint: $c = \mathbb{I}\{\neg(x_1 = x_2 = x_3)\}$.

## Graph representation




Here the constraint $c$ is linked to all variables it affects.

## Example: Sudoku

Constraints exist between (a) all numbers in a square (b) all numbers in a row (c) all numbers in a column.

# Constrained optimisation Problems

## Variables

- A set of variables $\{x_1, \ldots, x_n\}$
- Each variable can take values in $x \in X_i$, with $X \in \prod_i X_i$.

## Binary constraints

- $c_{i,j} : X_i \times X_j \to \{0, 1\}$.

## Objective function

- Maximise $u : X \to \mathbb{R}$.

## Special cases:

- $u(X) = \sum_i u_i(x_i)$
- $u(X) = \sum_{ij} u_{ij}(x_i, x_j)$

# Network Flow

- Graph $G = (N, E)$, $s, t \in N$ being the source and sink.
- Edge capacity $c : E \to \mathbb{R}_+$

## Flow $f : E \to \mathbb{R}$

The total flow from source to sink is

$$|f| = \sum_{(s,i)\in E} f_{si} = \sum_{(j,t)\in E} f_{jt}$$

## Flow constraints

The flow satisfies the following constraints:

- Capacity constraint: $f_{ij} \leq c_{ij}$
- Conservation of flows:

$$\forall n \in N \setminus \{s, t\} \sum_{i:(i,j)\in E, f_{ij}>0} f_i j = \sum_{j:(i,j)\in E, f_{ji}>0} f_j i.$$

## The maximum network flow problem

Maximise $|f|$ while satisfying the capacity and conservation constraints.

# Logic and constraints

# Logic

## Statements

- A statement $A$ may be true or false

## Unary operators

- negation: $\neg A$ is true if $A$ is false (and vice-versa).

## Binary operators

- or: $A \vee B$ ($A$ or $B$) is true if either $A$ or $B$ are true.
- and: $A \wedge B$ is true if both $A$ and $B$ are true.
- implies: $A \Rightarrow B$: is false if $A$ is true and $B$ is false.
- iff: $A \Leftrightarrow B$: is true if $A, B$ have equal truth values.

## Operator precedence

$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

# Set theory

- First, consider some universal set $\Omega$.
- A set $A$ is a collection of points $x$ in $\Omega$.
- $\{x \in \Omega : f(x)\}$: the set of points in $\Omega$ with the property that $f(x)$ is true.

## Unary operators

- $\neg A = \{x \in \Omega : x \notin A\}$.

## Binary operators

- $A \cup B$ if $\{x \in \Omega : x \in A \vee x \in B\}$ - (c.f. $A \vee B$)
- $A \cap B$ if $\{x \in \Omega : x \in A \wedge x \in B\}$ - (c.f. $A \wedge B$)

## Binary relations

- $A \subset B$ if $x \in A \Rightarrow x \in B$ - (c.f. $A \implies B$)
- $A = B$ if $x \in A \Leftrightarrow x \in B$ - (c.f. $A \Leftrightarrow B$)

# Knowledge base

- Syntax: How to construct sentences
- Semantics: What sentences mean

## Truth

- A statement $A$ is either true or false in any model $m \in \Omega$.

## Model

- $M(A)$ the set of all models where $A$ is true.

## Entailment

- $A \models B$ means that $B$ is true whenever $A$ is true.
- $A \models B$ if and only if $M(A) \subseteq M(B)$.

## Knowledge-Base

- A set of sentences that are true.

## Inference

- $KB \vdash_{\pi} A$: Algorithm $\pi$ can derive $A$ from KB.

# Propositional logic syntax

-Sentence $\rightarrow$ Atomic | Complex -Atomic $\rightarrow$ True | False | A | B | C | ...-Complex $\rightarrow$ (Sentence) | [Sentence]

- | $\neg$ Sentence (not)
- | Sentence $\wedge$ Sentence (and)
- | Sentence $\vee$ Sentence (or)
- | Sentence $\Rightarrow$ Sentence (implies)
- | Sentence $\Leftrightarrow$ Sentence (if and only if)

Precedence: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

# Set theory semantics of propositional logic

## Atoms as sets

- Let $\Omega$ be the universal set.
- Any atom $A$ is a subset of $\Omega$.
- Any model $\omega$ is an element of $\Omega$.

## For any model $\omega$:

- $\neg P$ is true iff $P$ is false in $\omega$.
- $P \wedge Q$ is true iff $P, Q$ are true in $\omega$.
- $P \vee Q$ is true iff either $P$ or $Q$ is true in $\omega$.
- $P \Rightarrow Q$ is true unless $P$ is true and $Q$ is false in $\omega$.
- $P \Leftrightarrow Q$ if $P, Q$ are both true or both false in $\omega$.
- If $A \subset B$ then, for every $\omega \in A$, $\omega \in B$.
- If $\omega \in A \cap B$ then $\omega \in A$.

# Factored state representation

## Predicates for coffee-making

- $x_c$ (machine has cup)
- $x_g$ (machine has grains)
- $x_m$ (machine is on)
- $x_w$ (machine has water)

To make coffee, $x_c \wedge x_g \wedge x_m \wedge x_w$ must be true.

# From n-ary to binary constraints

Take meeting scheduling as an example. The constraint $c = \mathbb{I}\{\neg(x_1 = x_2 = x_3)\}$ can be rewritten using the fact that $\neg(A \wedge B) = (\neg A) \vee (\neg B)$:

$$\begin{aligned} \neg(x_1 = x_2 = x_3) &= \neg(x_1 = x_2 \wedge x_2 = x_3 \wedge x_3 = x_1) \\ &= x_1 \neq x_2 \vee x_2 \neq x_3 \vee x_3 \neq x_1 . \end{aligned}$$

This leads to:

$$c = \mathbb{I}\{x_1 \neq x_2\} + \mathbb{I}\{x_2 \neq x_3\} + \mathbb{I}\{x_3 \neq x_1\} .$$

Since any constraint can be decomposed into the form

$$c = c_1 + c_2 + \cdots + c_n$$

we can always rewrite n-ary constraints as a collection of binary constraints.