On Constraint Programming for Path Planning with Uncertainty

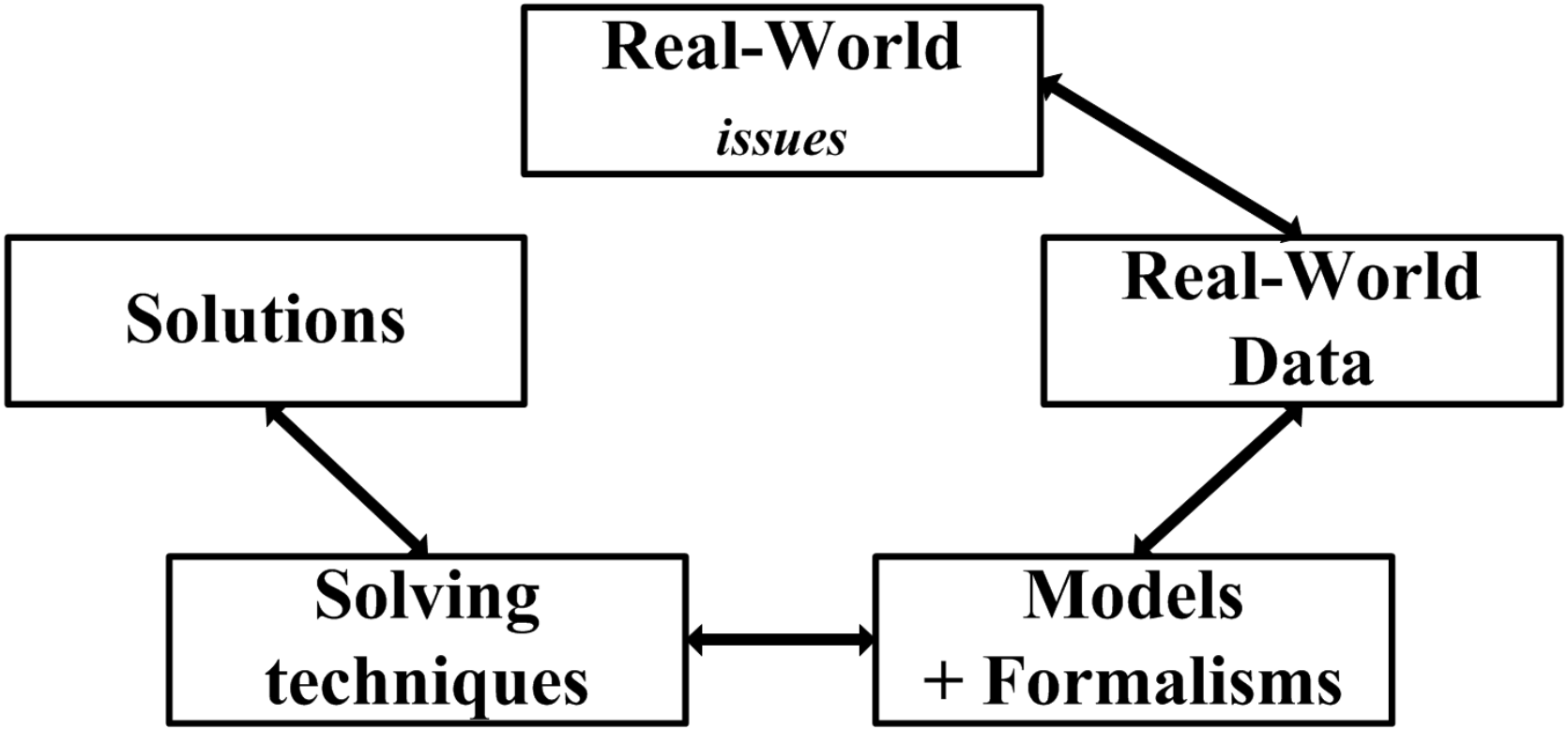
Michael Morin[1] **Anika-Pascale Papillon**[2] **Irène Abi-Zeid**[3]
François Laviolette[1] **Claude-Guy Quimper**[1]

[1]Department of Computer Science and Software Engineering
[2]Department of Mathematics and Statistics
[3]Department of Operations and Decision Systems
Université Laval, Québec, Qc, Canada
Michael.Morin.3@ulaval.ca

February 15, 2013

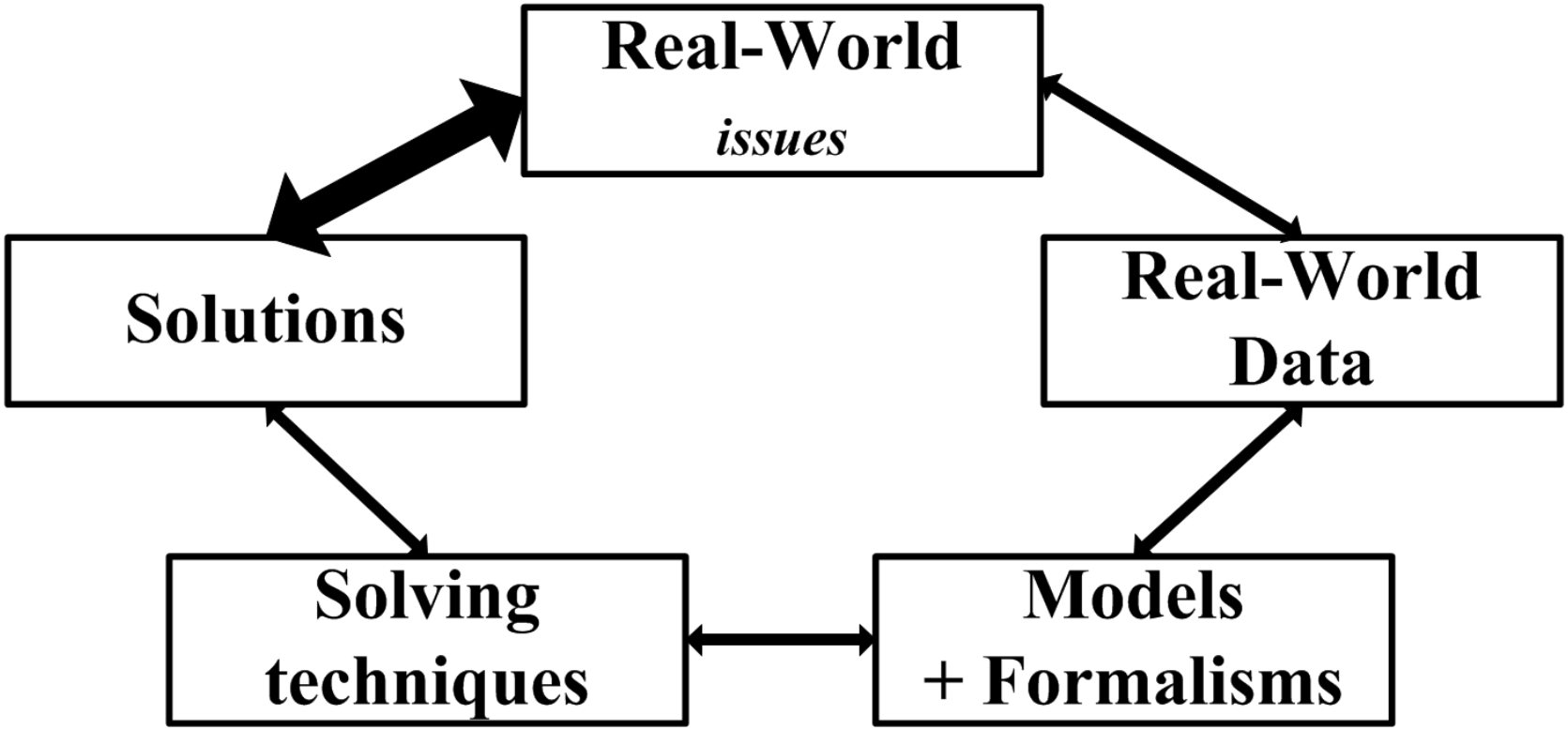
Introduction

- One of many visions of what *problem solving* is:



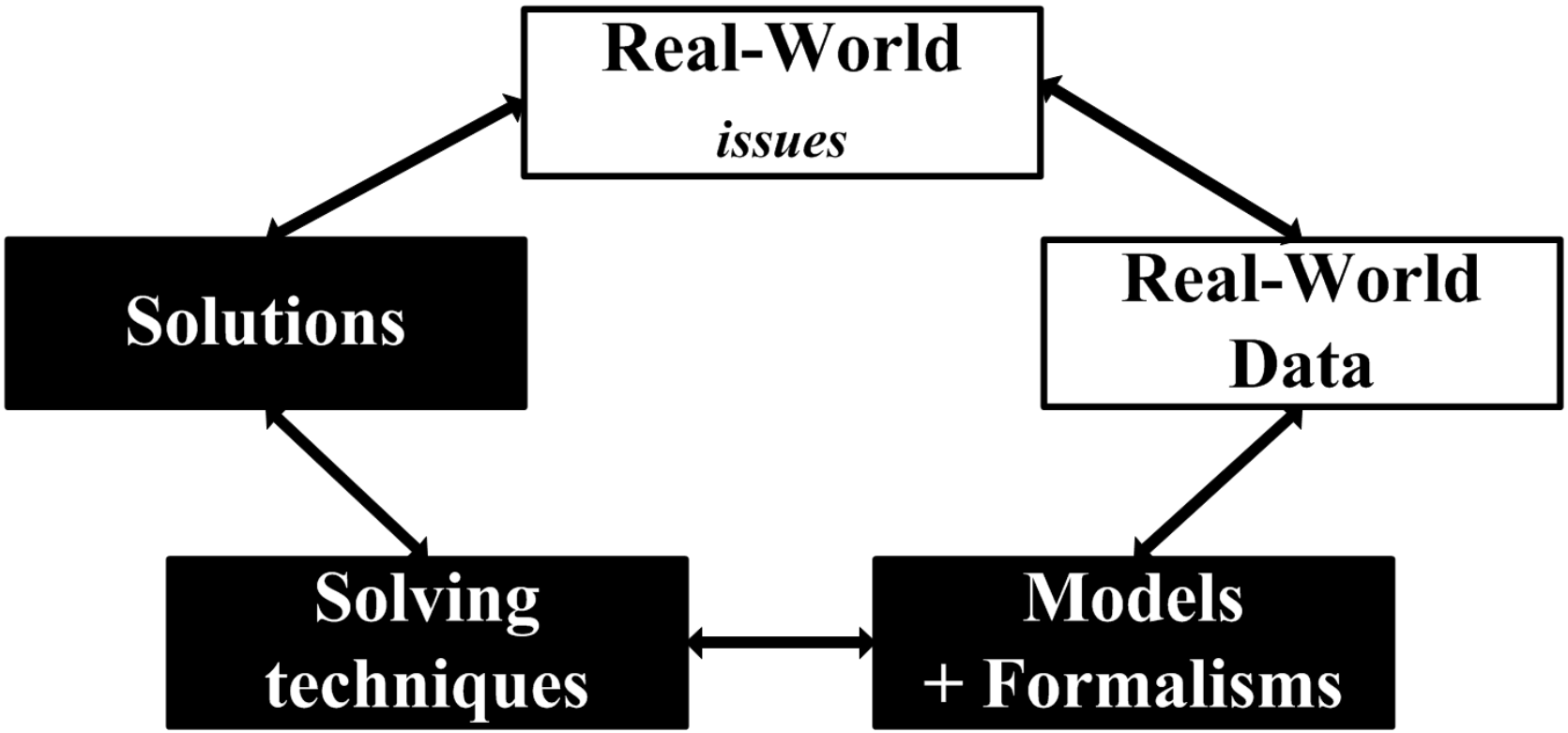
Introduction

- The most important part is the last link...



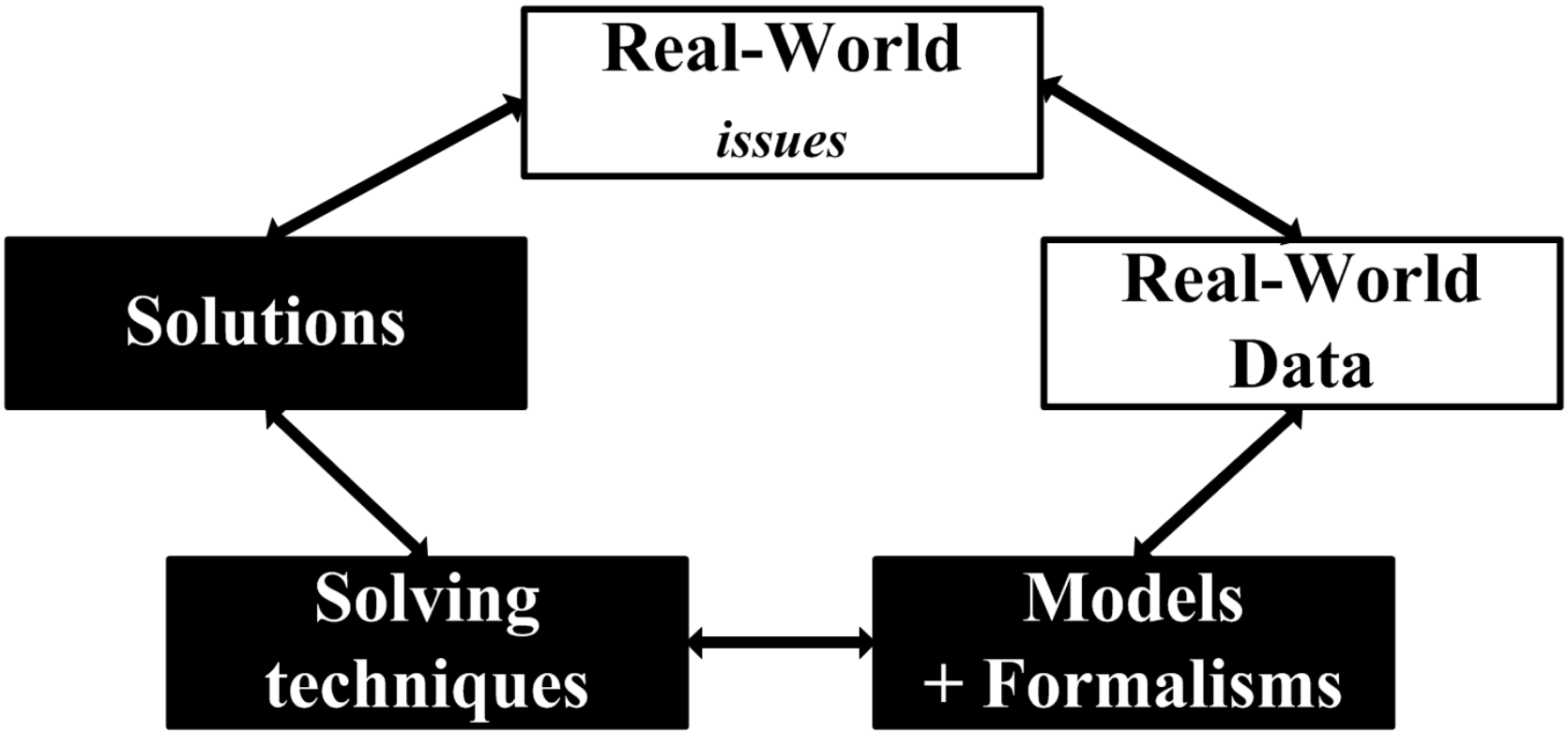
Introduction

- Fortunately for us, that link depends on everything we like... :)



Introduction

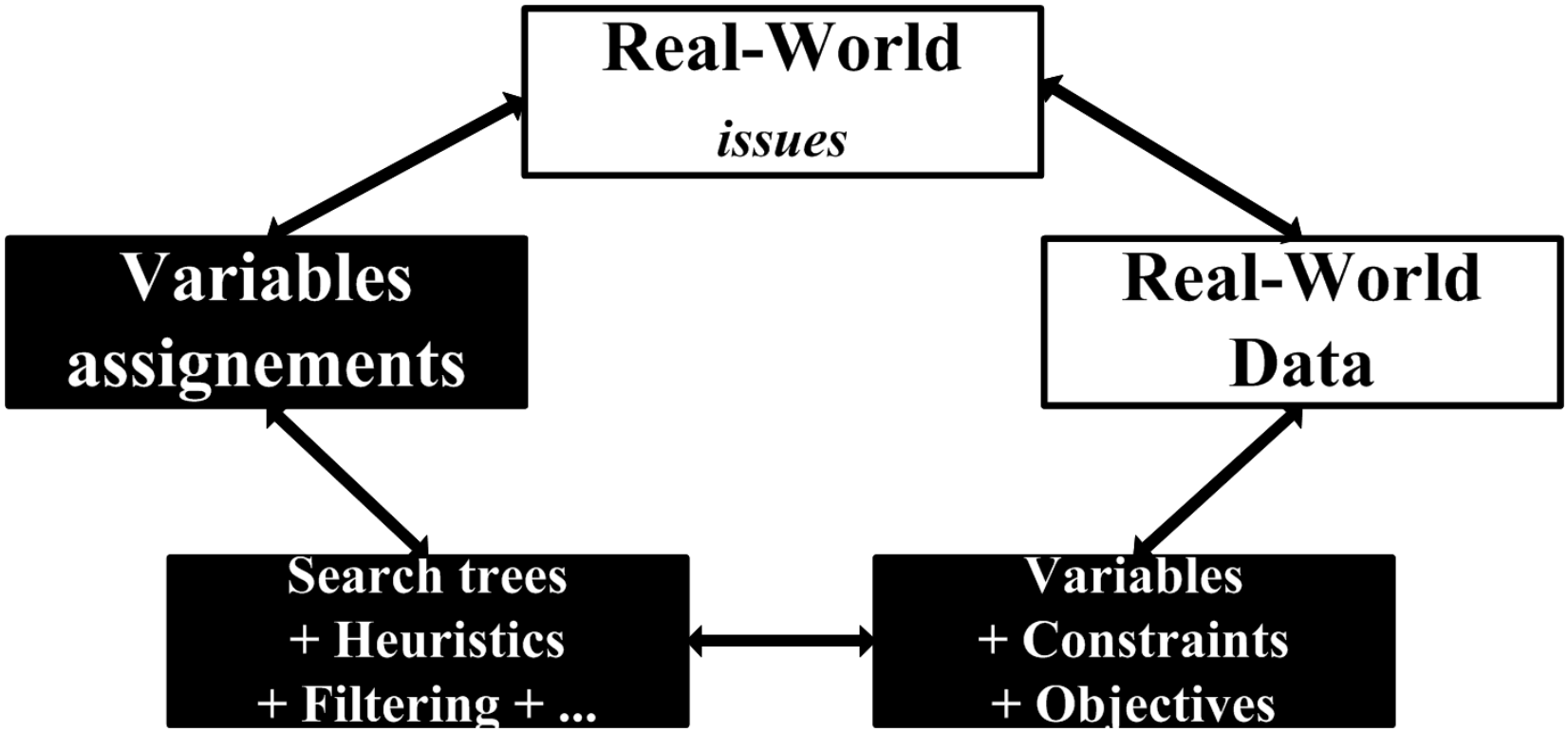
- Fortunately for us, that link depends on everything we like... :)



- Today's focus is "How to get that last link".

Introduction

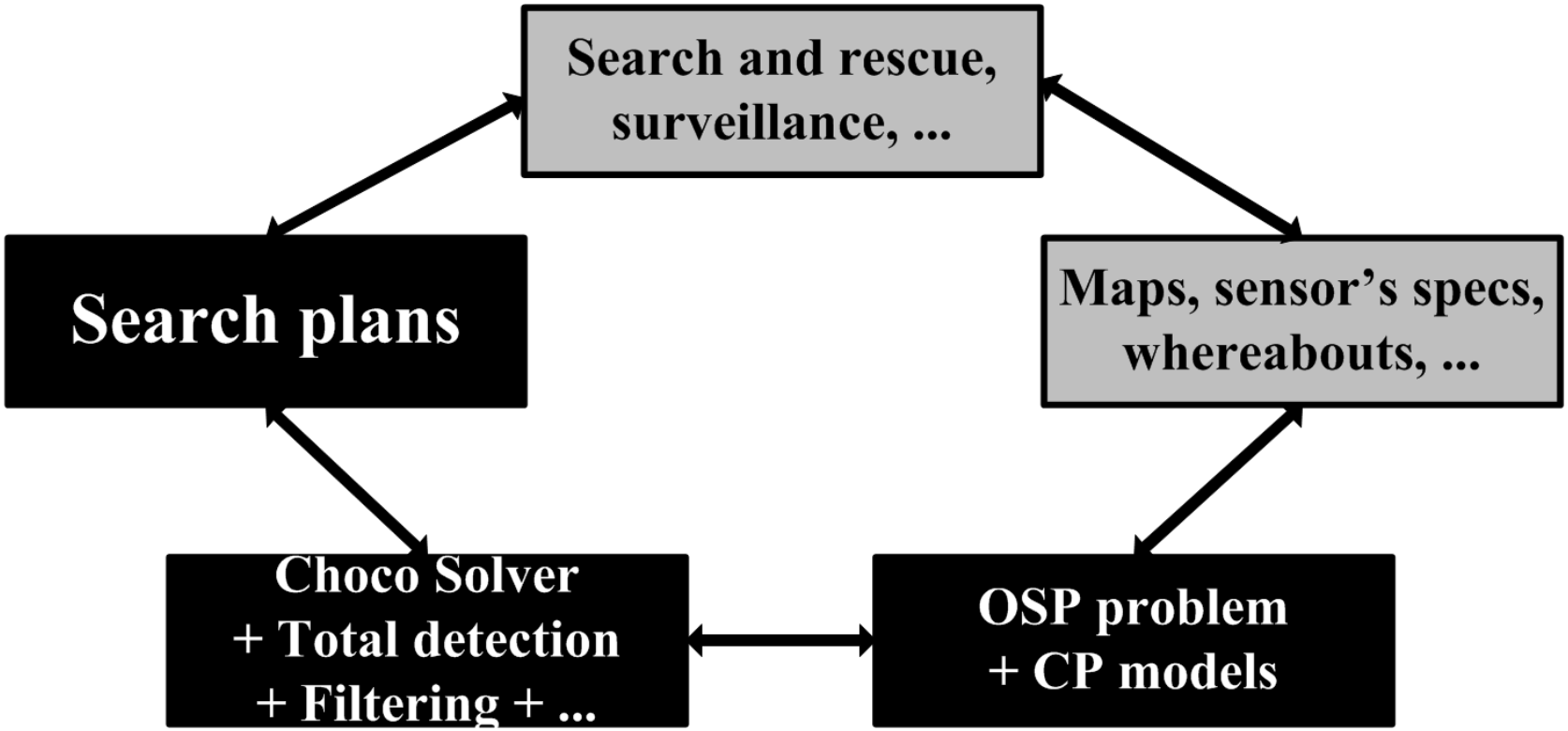
- In combinatorial optimization, one way to get that link is...



- ... *constraint programming* (CP)!

Introduction

- An application of CP to real-world issues:



OSP = Optimal search path problem

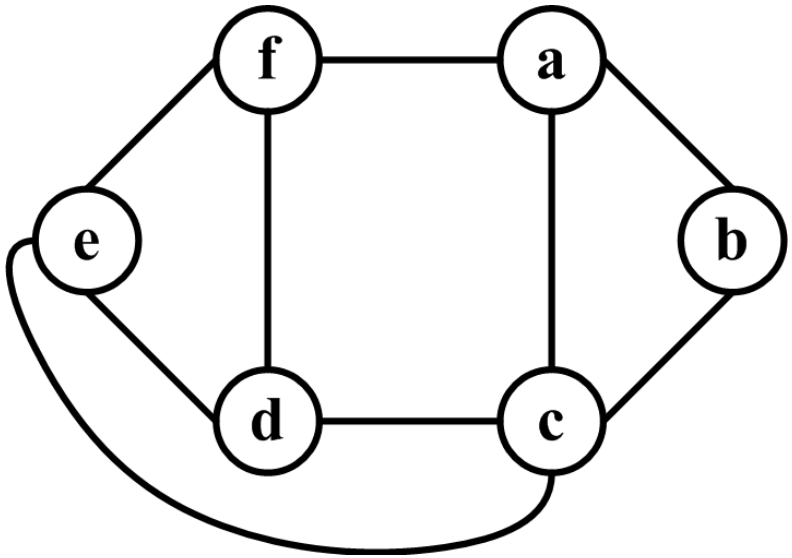
Outline

Constraint Programming

- Define the problem
- Model the problem
  - Variables, constraints(, objective function)
- Solve the problem
  - Search tree, heuristic, filtering, ...
- Obtain a solution:
  - Each variable is assign to *one* value
  - The assignment satisfies all constraints

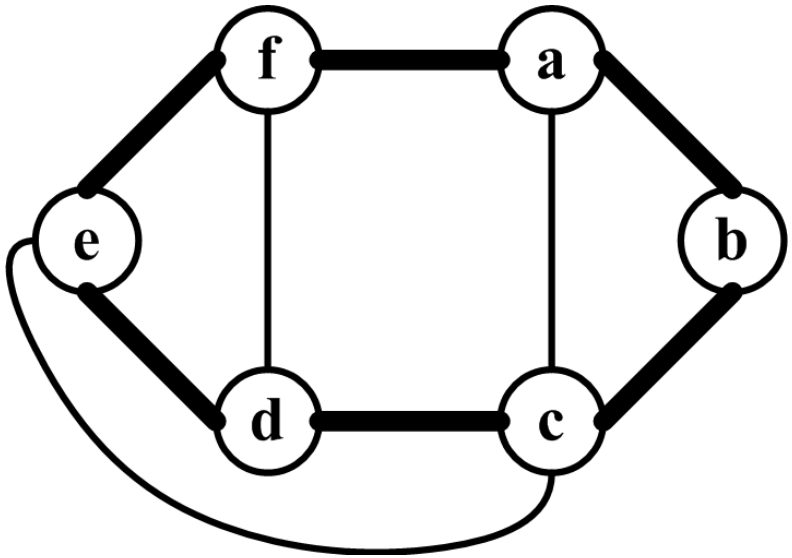
The Hamiltonian Cycle Problem

- Given a graph $G = (\mathcal{V}(G), \mathcal{E}(G))$.
- Find a cycle on G that visits each vertex exactly once.



The Hamiltonian Cycle Problem Model

- Given a graph $G = (\mathcal{V}(G), \mathcal{E}(G))$.
- Find a cycle on G that visits each vertex exactly once.



- A solution: Follow the cycle *abcdefa*.

- The variables T_i defines our position at time $1 \leq i \leq 6$.
- Their domains are $\text{dom}(T_i) = \{a, b, c, d, e, f\}$.

Variables	Domain dom T_i					
T_1	a	b	c	d	e	f
T_2	a	b	c	d	e	f
T_3	a	b	c	d	e	f
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f

- Visit each vertex exactly once:

  $AllDifferent(T_1, T_2, T_3, T_4, T_5, T_6)$.

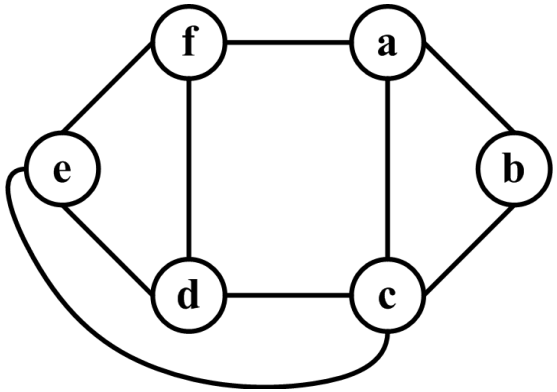
- This is a cycle:

  $$(T_i, T_{i+1}) \in \mathcal{E}(G), \quad \forall i : 1 \leq i < 6,$$
  $$(T_6, T_1) \in \mathcal{E}(G).$$

- The domains:

  $$\text{dom}(T_i) = \{a, b, c, d, e, f\}, \quad \forall i : 1 \leq i \leq 6.$$

$$\begin{aligned} \mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ (T_i, T_{i+1}) \in \mathcal{E}(G), \\ (T_6, T_1) \in \mathcal{E}(G), \\ \mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$

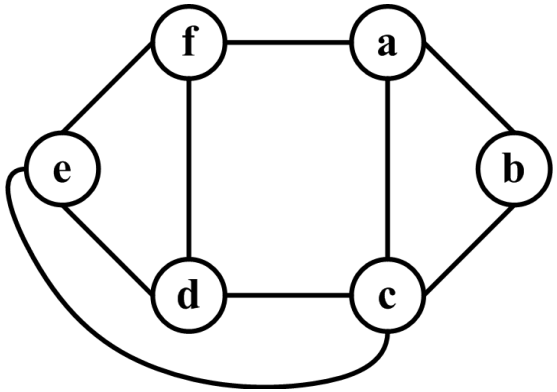


Variables	Domain dom T_i					
T_1	a	b	c	d	e	f
T_2	a	b	c	d	e	f
T_3	a	b	c	d	e	f
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f

Backtracking

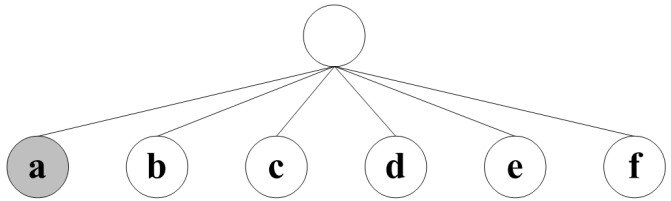
Solving the Hamiltonian Cycle Problem

$$\begin{aligned} \mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ (T_i, T_{i+1}) \in \mathcal{E}(G), \\ (T_6, T_1) \in \mathcal{E}(G), \\ \mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$



Variables	Domain dom T_i					
T_1	a	b	c	d	e	f
T_2	a	b	c	d	e	f
T_3	a	b	c	d	e	f
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f

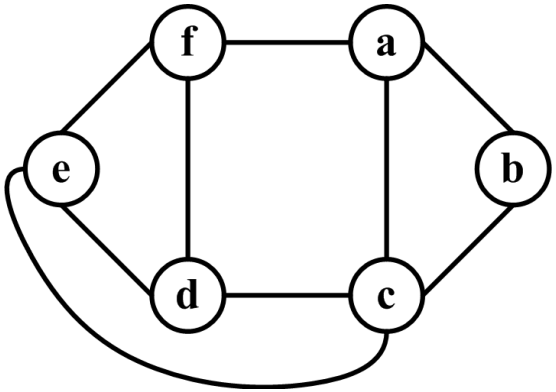
T_1



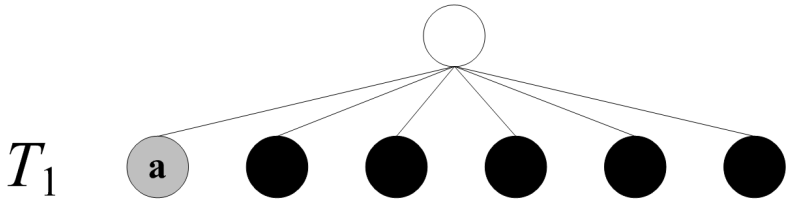
Backtracking

Solving the Hamiltonian Cycle Problem

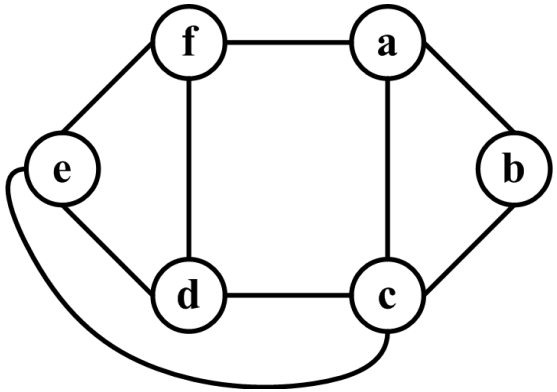
$$AllDifferent(T_1, T_2, T_3, T_4, T_5, T_6),$$
$$(T_i, T_{i+1}) \in \mathcal{E}(G),$$
$$(T_6, T_1) \in \mathcal{E}(G),$$
$$\text{dom}(T_i) = \{a, b, c, d, e, f\}.$$



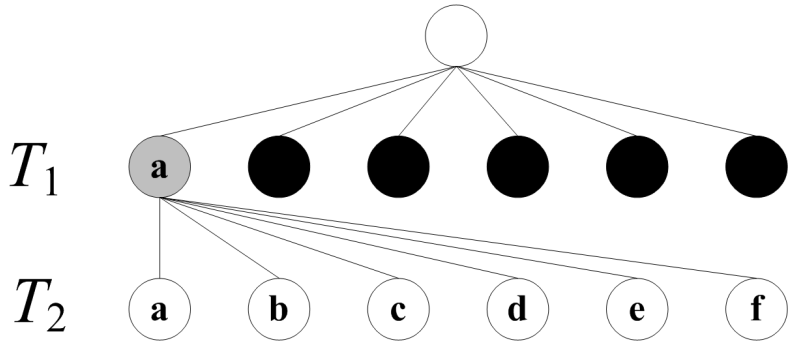
Variables	Domain dom T_i					
T_1	a					
T_2	a	b	c	d	e	f
T_3	a	b	c	d	e	f
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f



$$\begin{aligned} &\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ &(T_i, T_{i+1}) \in \mathcal{E}(G), \\ &(T_6, T_1) \in \mathcal{E}(G), \\ &\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$



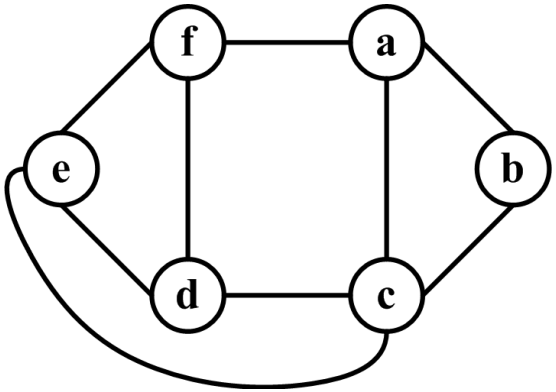
Variables	Domain dom T_i					
T_1	a					
T_2	a	b	c	d	e	f
T_3	a	b	c	d	e	f
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f



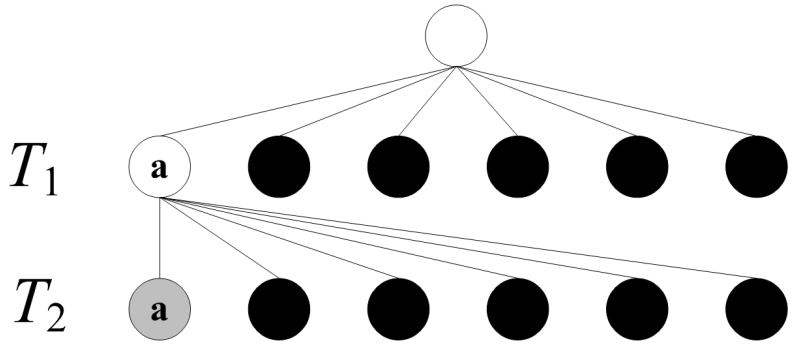
Backtracking

Solving the Hamiltonian Cycle Problem

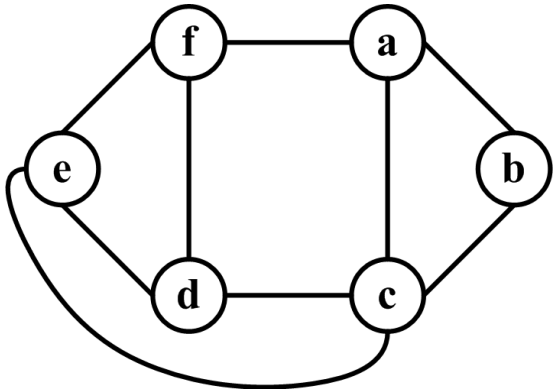
$$AllDifferent(T_1, T_2, T_3, T_4, T_5, T_6),$$
$$(T_i, T_{i+1}) \in \mathcal{E}(G),$$
$$(T_6, T_1) \in \mathcal{E}(G),$$
$$\text{dom}(T_i) = \{a, b, c, d, e, f\}.$$



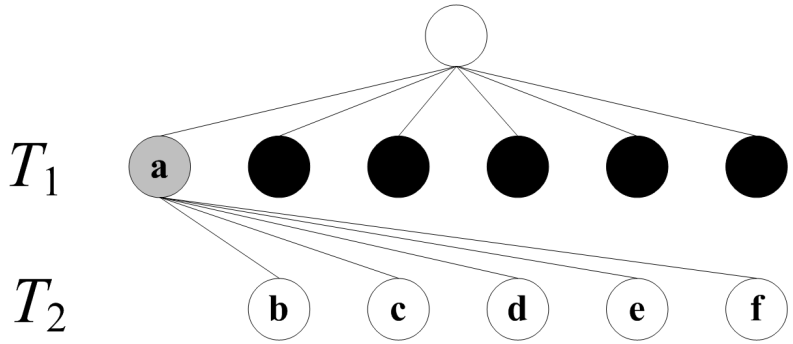
Variables	Domain dom T_i					
T_1	a					
T_2	a					
T_3	a	b	c	d	e	f
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f



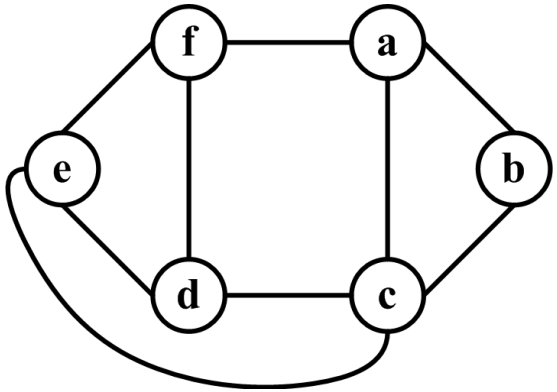
$$\begin{aligned} &\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ &(T_i, T_{i+1}) \in \mathcal{E}(G), \\ &(T_6, T_1) \in \mathcal{E}(G), \\ &\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$



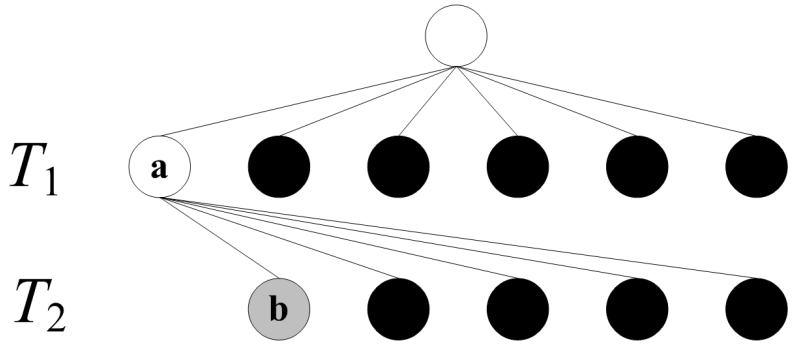
Variables	Domain dom T_i					
T_1	a					
T_2		b	c	d	e	f
T_3	a	b	c	d	e	f
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f



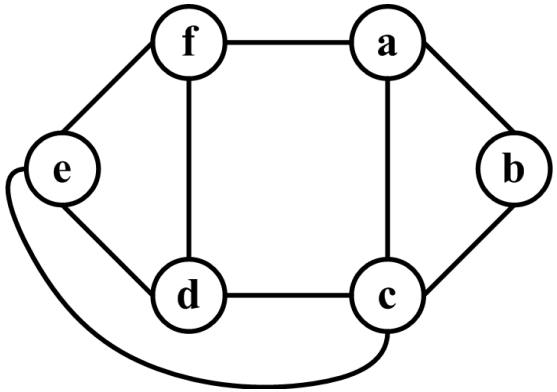
$$\begin{aligned} &\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ &(T_i, T_{i+1}) \in \mathcal{E}(G), \\ &(T_6, T_1) \in \mathcal{E}(G), \\ &\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$



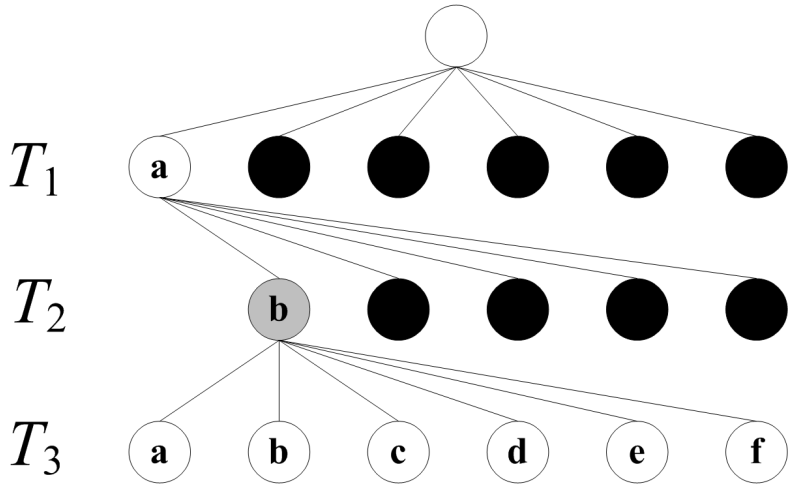
Variables	Domain dom T_i					
T_1	a					
T_2		b				
T_3	a	b	c	d	e	f
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f



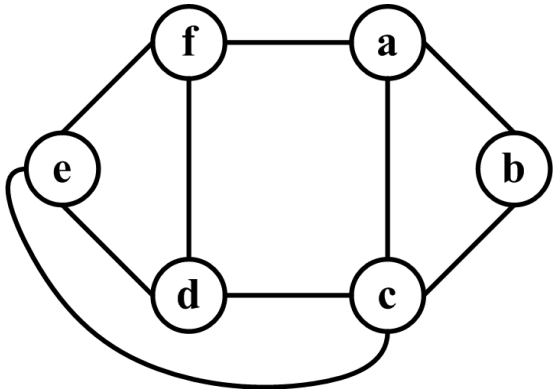
$$\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6),$$
$$(T_i, T_{i+1}) \in \mathcal{E}(G),$$
$$(T_6, T_1) \in \mathcal{E}(G),$$
$$\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}.$$



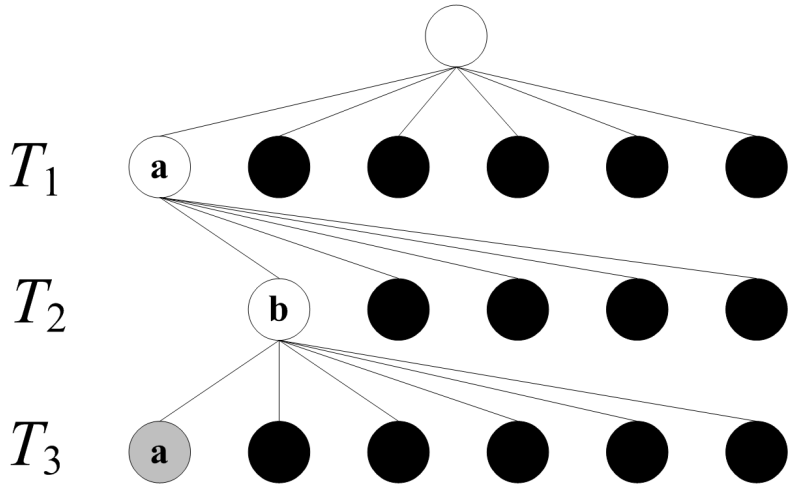
Variables	Domain dom T_i					
T_1	a					
T_2		b				
T_3	a	b	c	d	e	f
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f



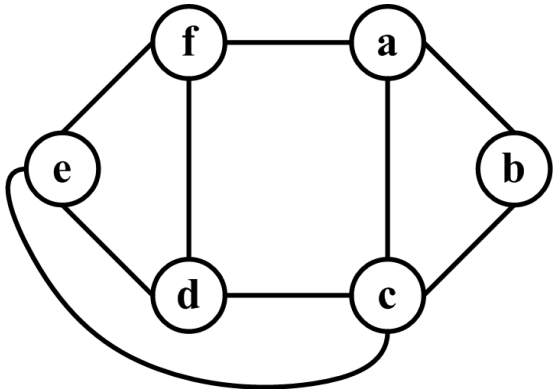
$$\begin{aligned} &\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ &(T_i, T_{i+1}) \in \mathcal{E}(G), \\ &(T_6, T_1) \in \mathcal{E}(G), \\ &\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$



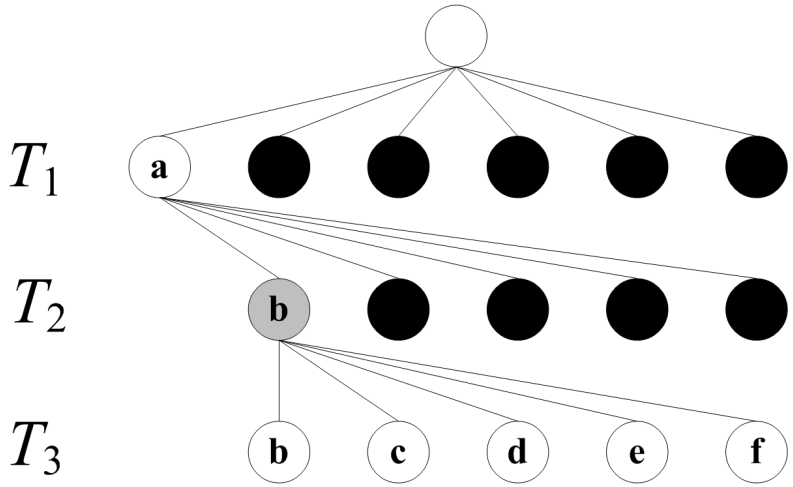
Variables	Domain dom T_i					
T_1	a					
T_2		b				
T_3	a					
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f



$$\begin{aligned} &\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ &(T_i, T_{i+1}) \in \mathcal{E}(G), \\ &(T_6, T_1) \in \mathcal{E}(G), \\ &\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$



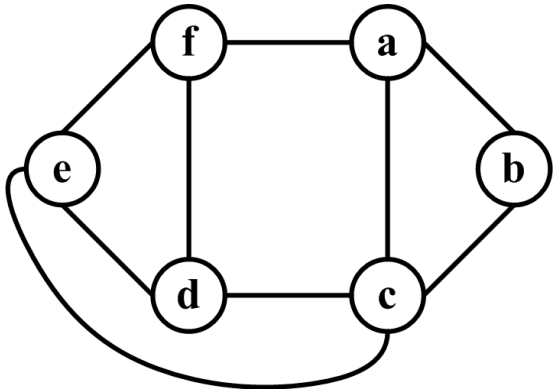
Variables	Domain dom T_i					
T_1	a					
T_2		b				
T_3		b	c	d	e	f
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f



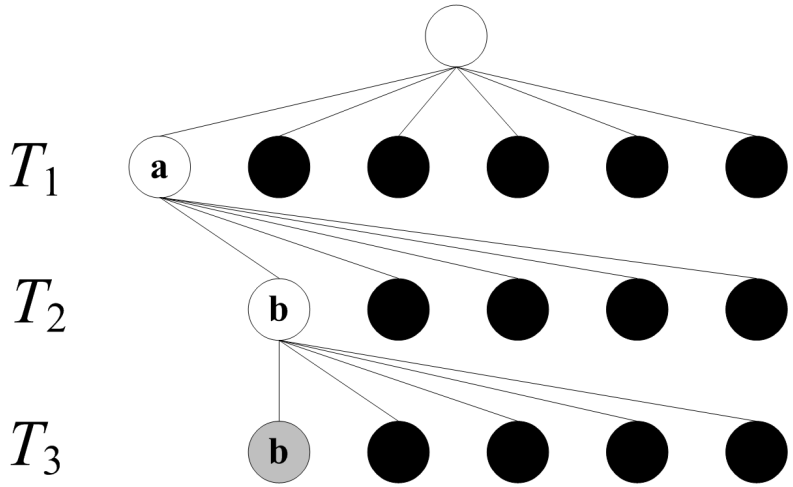
Backtracking

Solving the Hamiltonian Cycle Problem

$$AllDifferent(T_1, T_2, T_3, T_4, T_5, T_6),$$
$$(T_i, T_{i+1}) \in \mathcal{E}(G),$$
$$(T_6, T_1) \in \mathcal{E}(G),$$
$$\text{dom}(T_i) = \{a, b, c, d, e, f\}.$$



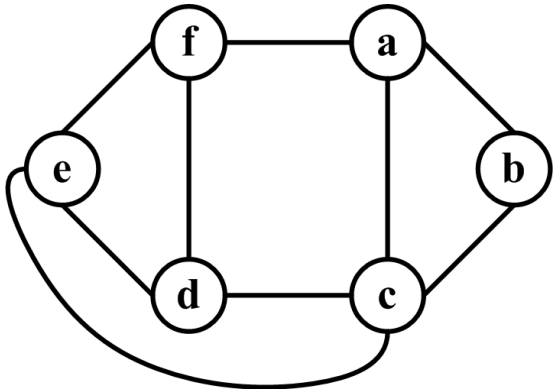
Variables	Domain dom T_i					
T_1	a					
T_2		b				
T_3		b				
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f



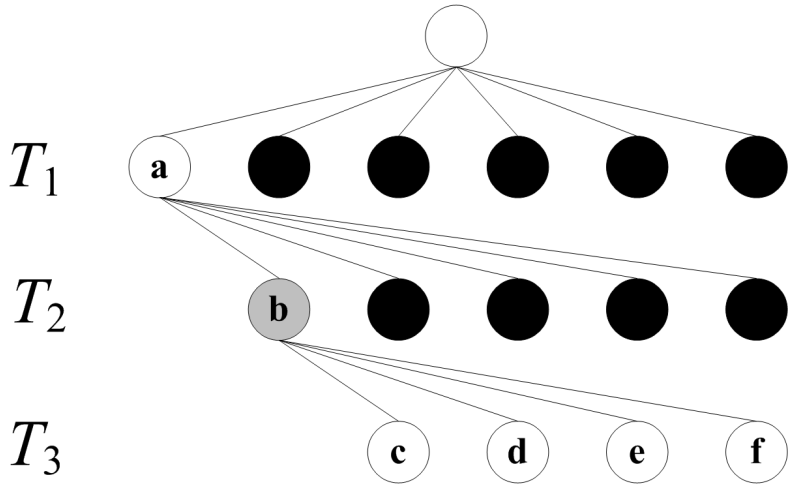
Backtracking

Solving the Hamiltonian Cycle Problem

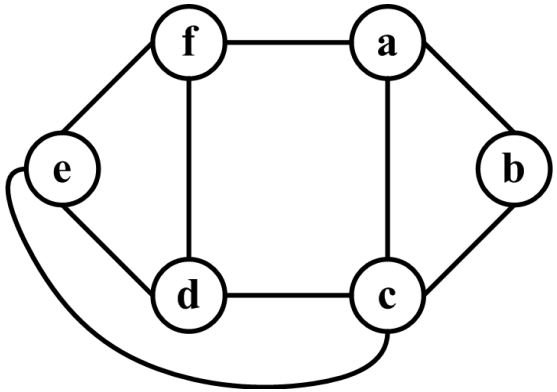
$$\begin{aligned} &\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ &(T_i, T_{i+1}) \in \mathcal{E}(G), \\ &(T_6, T_1) \in \mathcal{E}(G), \\ &\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$



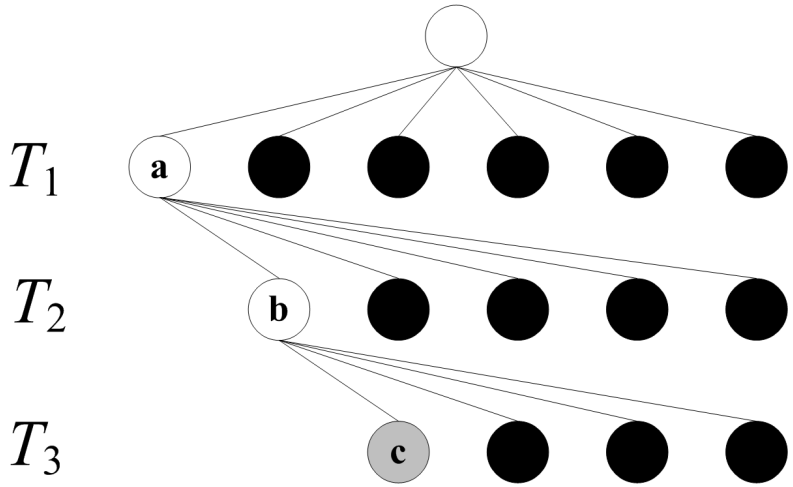
Variables	Domain dom T_i					
T_1	a					
T_2		b				
T_3			c	d	e	f
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f



$$\begin{aligned} &\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ &(T_i, T_{i+1}) \in \mathcal{E}(G), \\ &(T_6, T_1) \in \mathcal{E}(G), \\ &\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$



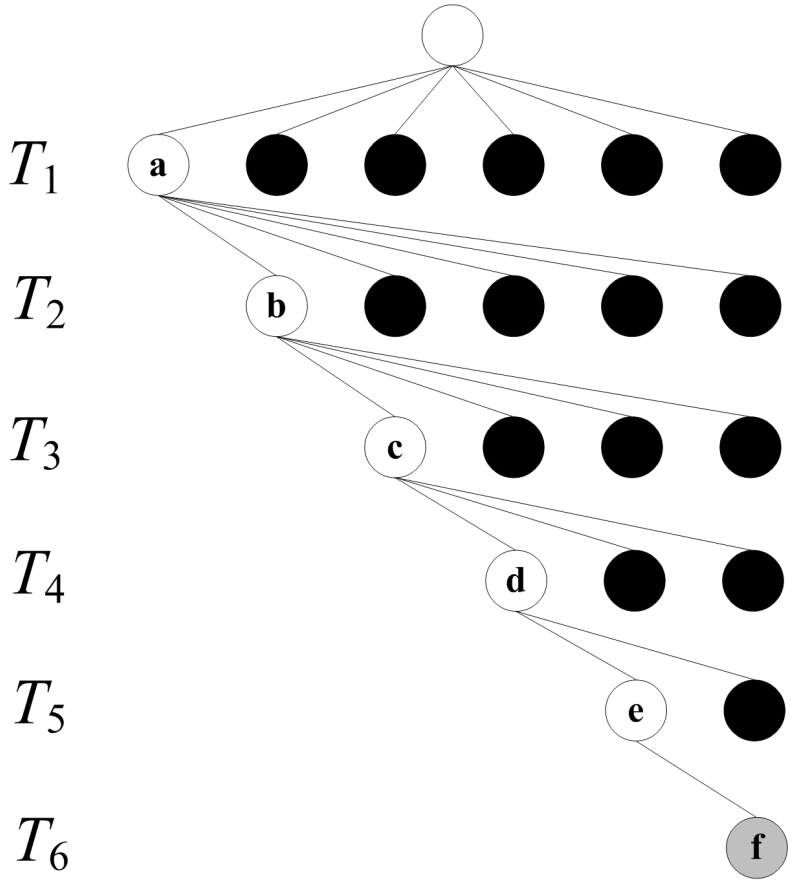
Variables	Domain dom T_i					
T_1	a					
T_2		b				
T_3			c			
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f



The Solution and its Search Tree

Solving the Hamiltonian Cycle Problem

Variables	Domain dom T_i
T_1	a
T_2	b
T_3	c
T_4	d
T_5	e
T_6	f



All Search Trees Are Not Created Equals

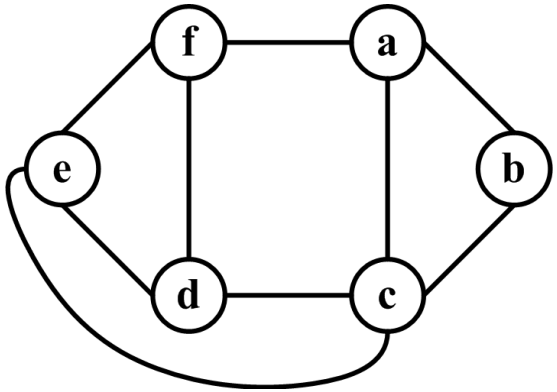
Solving the Hamiltonian Cycle Problem

- Performance metrics
  - Total number of backtrackings
  - Total solving time
- Possible enhancements
  - Variable selection heuristics
  - Value selection heuristics
  - Filtering

Domain filtering

Solving the Hamiltonian Cycle Problem

$$\begin{aligned} &\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ &(T_i, T_{i+1}) \in \mathcal{E}(G), \\ &(T_6, T_1) \in \mathcal{E}(G), \\ &\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$

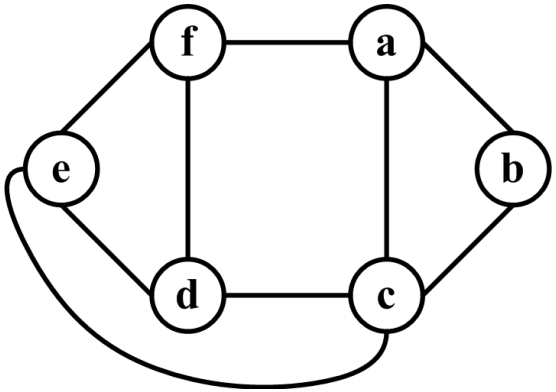


Variables	Domain dom T_i					
T_1	a	b	c	d	e	f
T_2	a	b	c	d	e	f
T_3	a	b	c	d	e	f
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f

Domain filtering

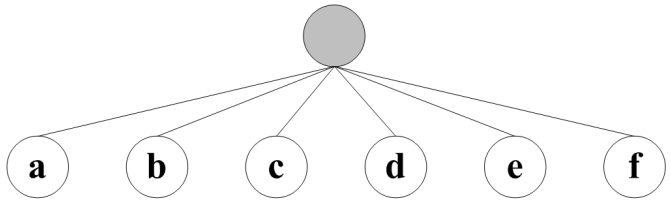
Solving the Hamiltonian Cycle Problem

$$\begin{aligned} &\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ &(T_i, T_{i+1}) \in \mathcal{E}(G), \\ &(T_6, T_1) \in \mathcal{E}(G), \\ &\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$



Variables	Domain dom T_i					
T_1	a	b	c	d	e	f
T_2	a	b	c	d	e	f
T_3	a	b	c	d	e	f
T_4	a	b	c	d	e	f
T_5	a	b	c	d	e	f
T_6	a	b	c	d	e	f

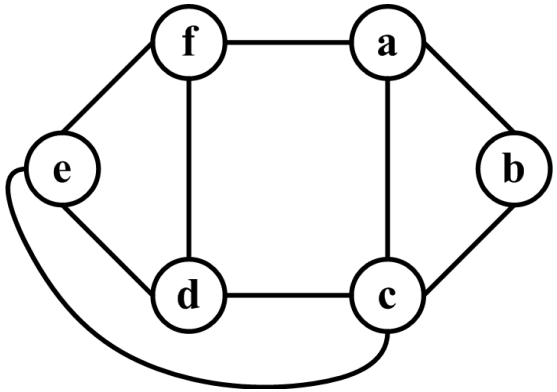
T_1



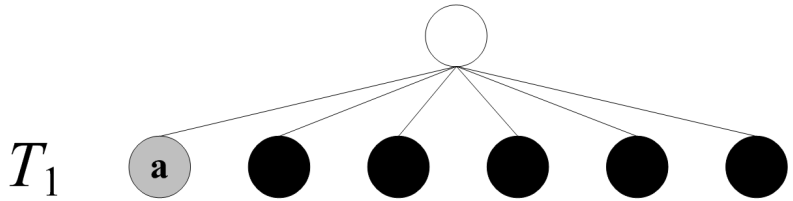
Domain filtering

Solving the Hamiltonian Cycle Problem

$$\begin{aligned} &\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ &(T_i, T_{i+1}) \in \mathcal{E}(G), \\ &(T_6, T_1) \in \mathcal{E}(G), \\ &\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$



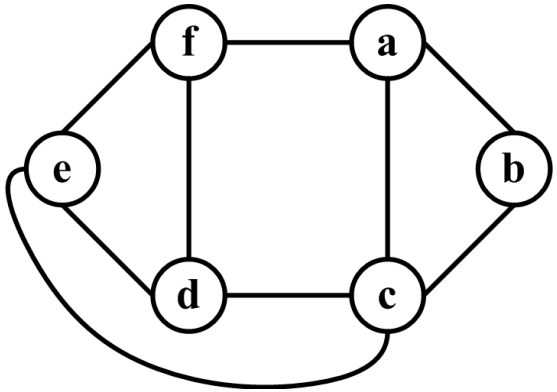
Variables	Domain dom T_i					
T_1	a					
T_2	~~a~~	b	c	~~d~~	~~e~~	f
T_3	~~a~~	b	c	d	e	f
T_4	~~a~~	b	c	d	e	f
T_5	~~a~~	b	c	d	e	f
T_6	~~a~~	b	c	d	e	f



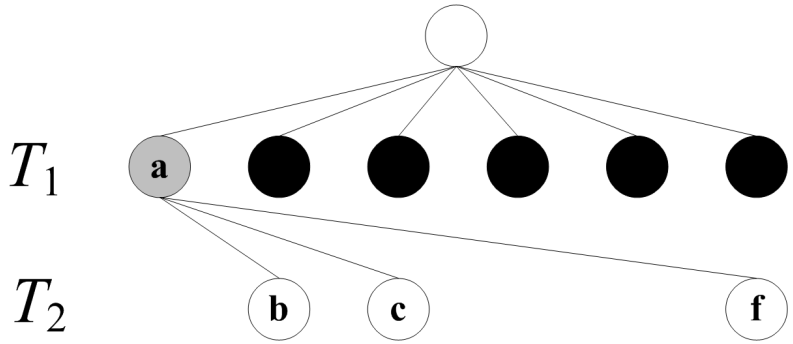
Domain filtering

Solving the Hamiltonian Cycle Problem

$$\begin{aligned} &\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ &(T_i, T_{i+1}) \in \mathcal{E}(G), \\ &(T_6, T_1) \in \mathcal{E}(G), \\ &\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$



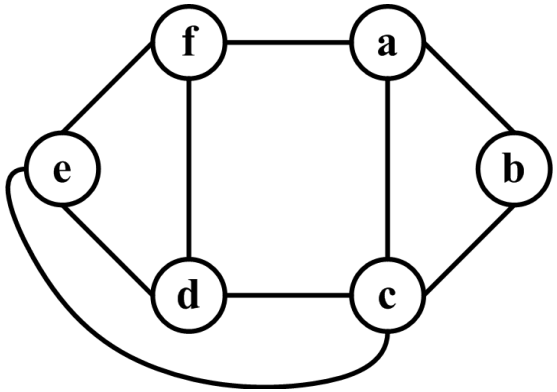
Variables	Domain dom T_i				
T_1	a				
T_2	b	c			f
T_3	b	c	d	e	f
T_4	b	c	d	e	f
T_5	b	c	d	e	f
T_6	b	c	d	e	f



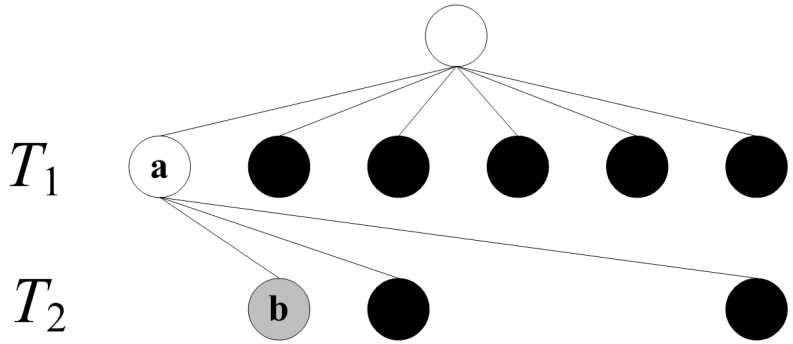
Domain filtering

Solving the Hamiltonian Cycle Problem

$$AllDifferent(T_1, T_2, T_3, T_4, T_5, T_6),$$
$$(T_i, T_{i+1}) \in \mathcal{E}(G),$$
$$(T_6, T_1) \in \mathcal{E}(G),$$
$$\text{dom}(T_i) = \{a, b, c, d, e, f\}.$$



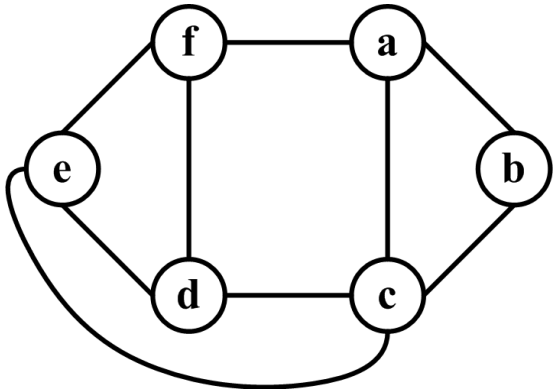
Variables	Domain dom T_i				
T_1	a				
T_2		b			
T_3		~~b~~	c	~~d~~	~~e~~	~~f~~
T_4		~~b~~	~~c~~	d	e	~~f~~
T_5		~~b~~	~~c~~	d	e	f
T_6		~~b~~	~~c~~	~~d~~	~~e~~	f



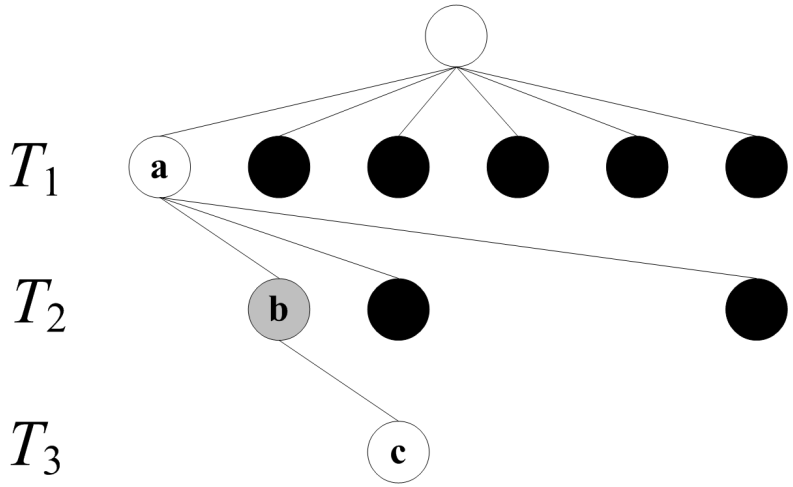
Domain filtering

Solving the Hamiltonian Cycle Problem

$$AllDifferent(T_1, T_2, T_3, T_4, T_5, T_6),$$
$$(T_i, T_{i+1}) \in \mathcal{E}(G),$$
$$(T_6, T_1) \in \mathcal{E}(G),$$
$$\text{dom}(T_i) = \{a, b, c, d, e, f\}.$$



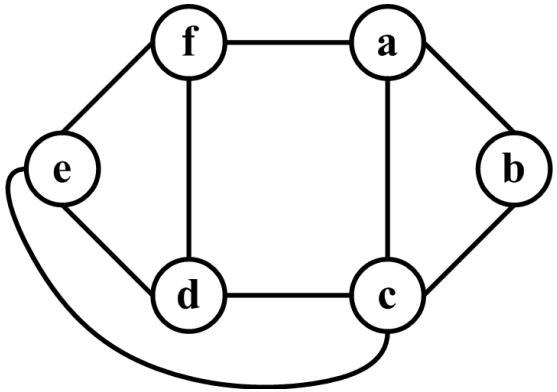
Variables	Domain dom T_i
T_1	a
T_2	b
T_3	c
T_4	d e
T_5	d e f
T_6	f



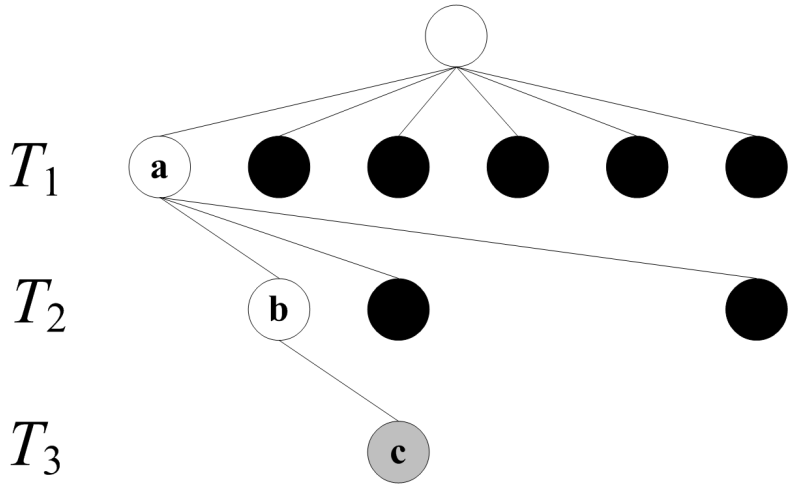
Domain filtering

Solving the Hamiltonian Cycle Problem

$$\begin{aligned} &\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6), \\ &(T_i, T_{i+1}) \in \mathcal{E}(G), \\ &(T_6, T_1) \in \mathcal{E}(G), \\ &\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}. \end{aligned}$$



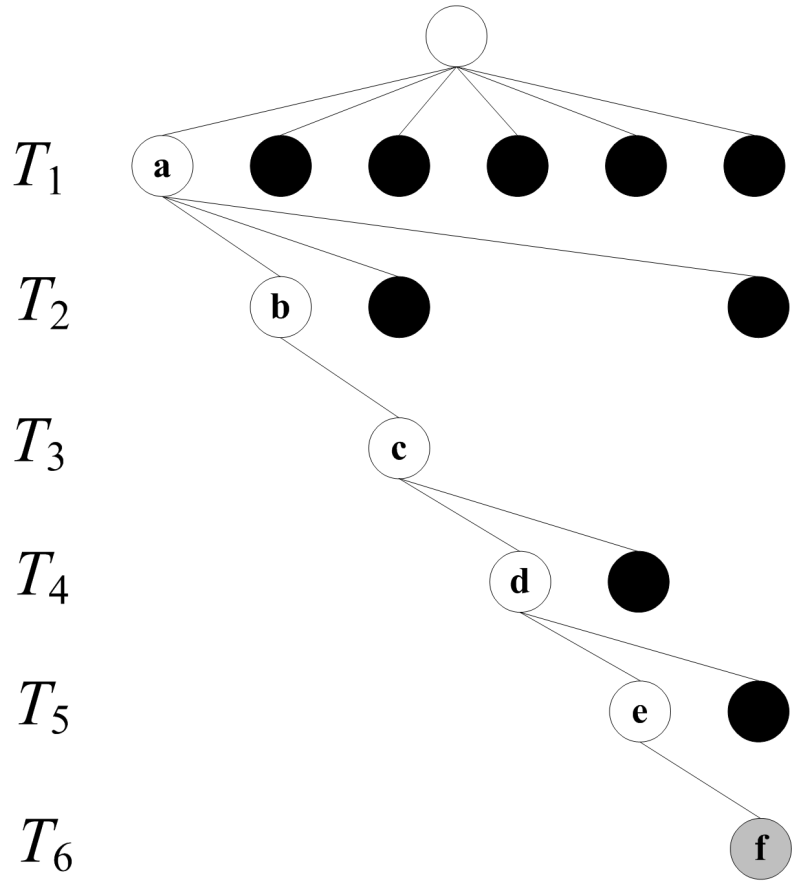
Variables	Domain dom T_i
T_1	a
T_2	b
T_3	c
T_4	d e
T_5	d e f
T_6	f



The Solution and its Search Tree

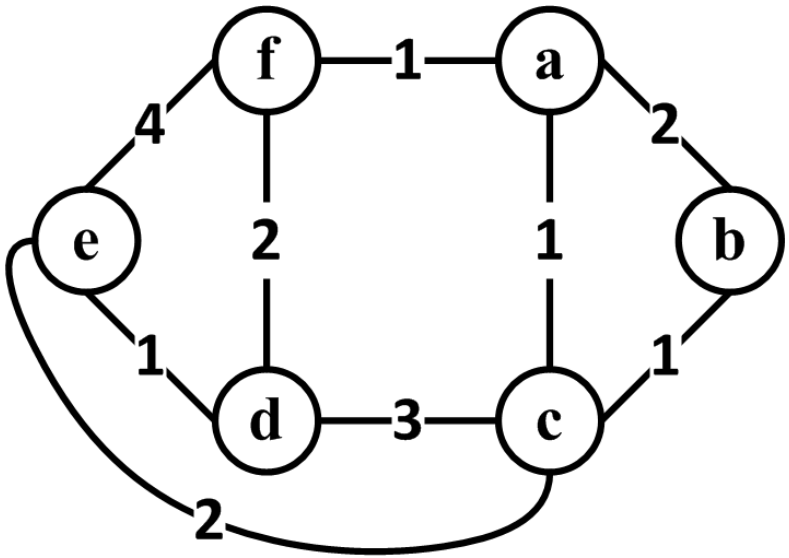
Solving the Hamiltonian Cycle Problem

Variables	Domain dom T_i
T_1	a
T_2	b
T_3	c
T_4	d
T_5	e
T_6	f



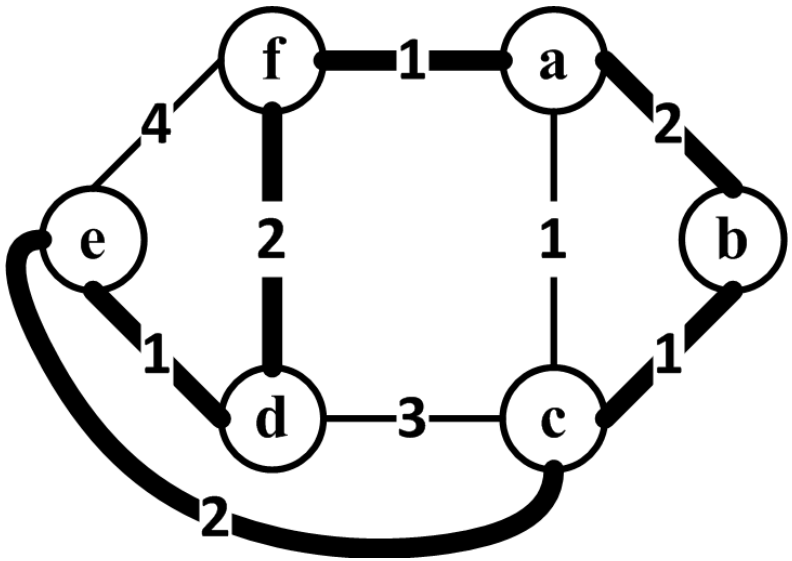
The Travelling Salesman Problem

- Given a graph $G = (\mathcal{V}(G), \mathcal{E}(G))$, and a cost function $c(x, y)$ on the edges of G.
- Find an hamiltonian cycle of minimal cost on G.



The Travelling Salesman Problem Model

- Given a graph $G = (\mathcal{V}(G), \mathcal{E}(G))$, and a cost function $c(x, y)$ on the edges of G.
- Find an hamiltonian cycle of minimal cost on G.



- A solution: Follow the cycle *abcedfa*.

The Constraints

The Travelling Salesman Problem Model

- Minimize the sum of the cost of each edge

  $\min C,$

  where $C = c[T_6, T_1] + \sum_{i=1}^{5} c[T_i, T_{t+1}]$.
- Visit each vertex exactly once:

  $\mathit{AllDifferent}(T_1, T_2, T_3, T_4, T_5, T_6).$

- This is a cycle:

  $$(T_i, T_{i+1}) \in \mathcal{E}(G), \qquad \forall i : 1 \leq i < 6,$$
  $$(T_6, T_1) \in \mathcal{E}(G).$$

- The domains:

  $$\mathrm{dom}(T_i) = \{a, b, c, d, e, f\}, \qquad \forall i : 1 \leq i \leq 6.$$

The Constraints

Solving he Travelling Salesman Problem

- Use the same concepts of search trees:
  - An objective function?
  - It's just more variables and constraints...
- However, the bound on the objective function is of capital importance.
- The stronger the filtering, the best is the performance.
  - A tight upper bound filters the domain of the decision variables.

The Optimal Search Path Problem

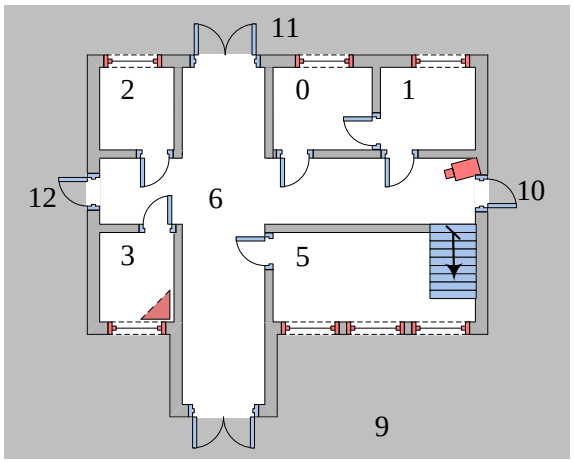
Goals

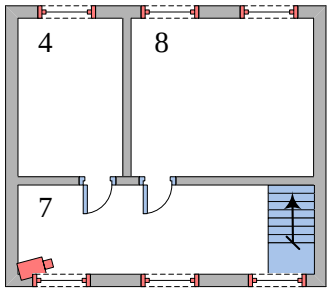
- Find a path that maximizes the probability of locating a survivor, a robber, an object, etc.
- Uncertain object detectability and location
- Markovian motion model
- Search theory (Stone [2004])
- $\mathcal{NP}$-hard problem ([Trummel and Weisinger, 1986])

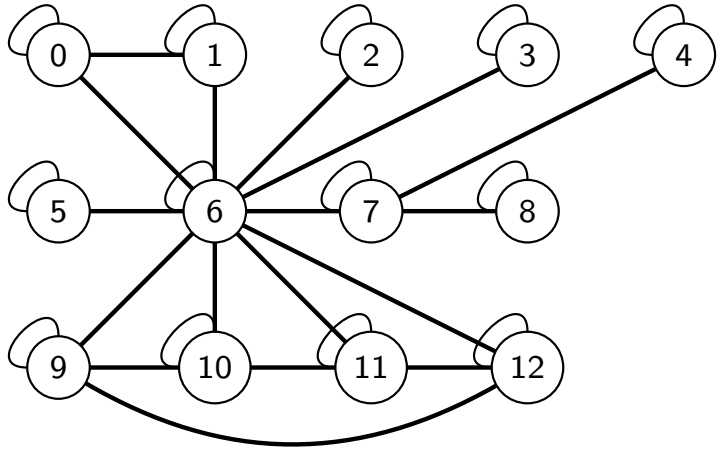
The OSP Problem

Definitions

- $G_A = (\mathcal{V}(G_A), \mathcal{E}(G_A))$ where $\mathcal{V}(G_A)$ is a set of discrete regions.



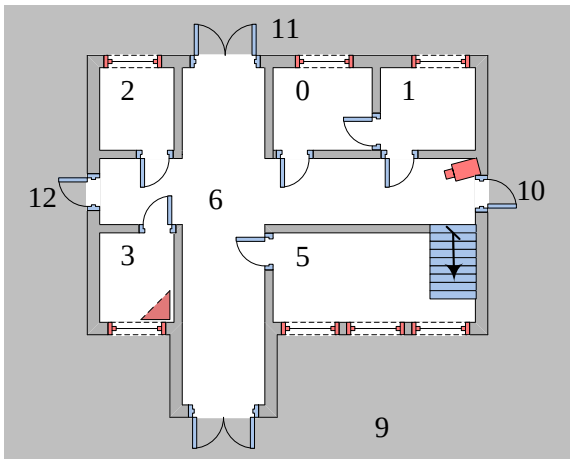


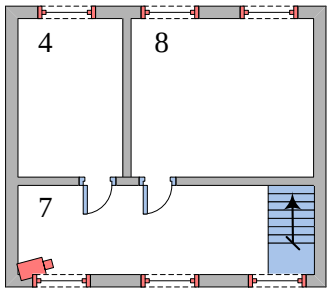


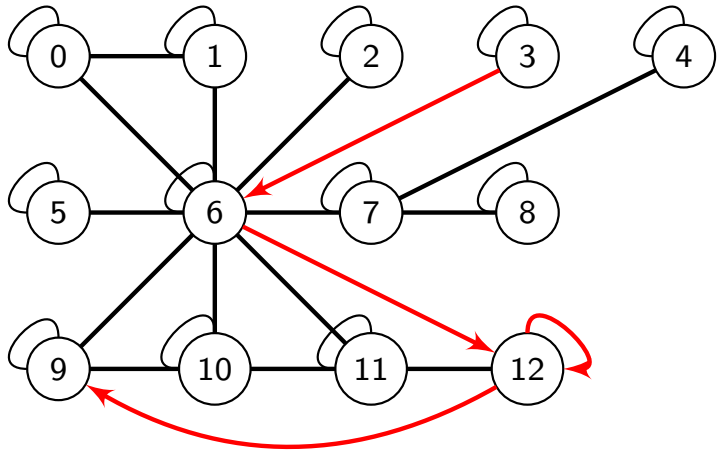
The OSP Problem

Definitions

- $\mathcal{T} = \{1, \ldots, T\}$ is the set of time steps available to search G_A.
- $y_t \in \mathcal{V}(G_A)$ is the searcher's location at time $t \in \mathcal{T}$.
  - When $y_t = r \in \mathcal{V}(G_A)$, the vertex r is searched at time t.
- $P = [y_0, y_1, \ldots, y_T]$ is the search path (plan).
  - $y_0 \in \mathcal{V}(G_A)$ is the searcher's starting location.
  - For all $t \in \mathcal{T}$, $(y_{t-1}, y_t) \in \mathcal{E}(G_A)$.

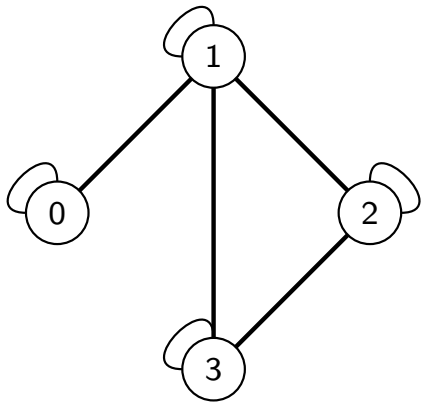






- The object's movements are independent of the searcher's actions.
- **M** is the Markovian motion model matrix.

$$\mathbf{M} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{5} & \frac{2}{5} & \frac{1}{5} & \frac{1}{5} \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{5} & \frac{2}{5} & \frac{1}{5} \end{pmatrix}.$$



Blue terms are a priori known probabilities.

The OSP Problem

Definitions

- The initial probability of containment distribution: poc_1.
- The local probability of success ($\forall t \in \mathcal{T}$):

$$\overbrace{pos_t(r)}^{\text{Prob. of success}} = \overbrace{\underbrace{poc_t(r)}_{\text{Prob. of containment}} \times \underbrace{pod(r)}_{\text{Prob. of detection}}}^{\text{Prob. of success}}.$$

- The probability of detection (conditional to the presence of the object):

$$\begin{aligned} pod(r) &\in (0, 1], && \text{if } y_t = r; \\ pod(r) &= 0, && \text{otherwise.} \end{aligned}$$

- The local probability of containment ($\forall t \in \{2, \ldots, T\}$):

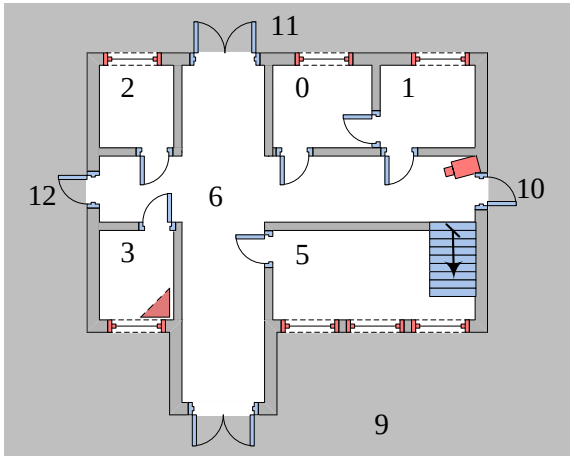
$$poc_t(r) = \sum_{s \in \mathcal{V}(G_A)} \mathbf{M}(s, r) \left[poc_{t-1}(s) - pos_{t-1}(s) \right].$$

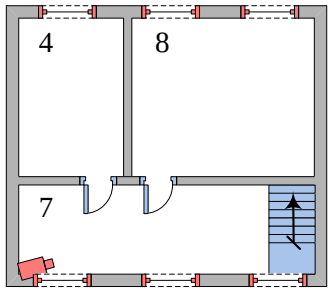
Find an optimal search plan $P = [y_0, y_1, \ldots, y_T]$ maximizing the cumulative overall probability of success (COS) defined as:

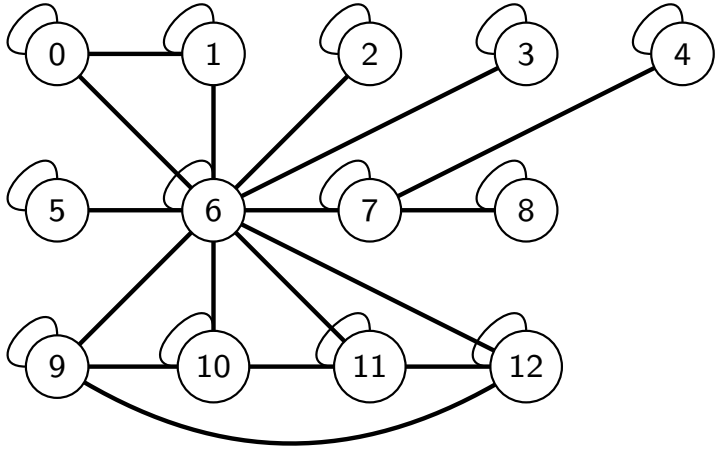
$$COS(P) = \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{V}(G_A)} pos_t(r).$$

The OSP Problem

Example

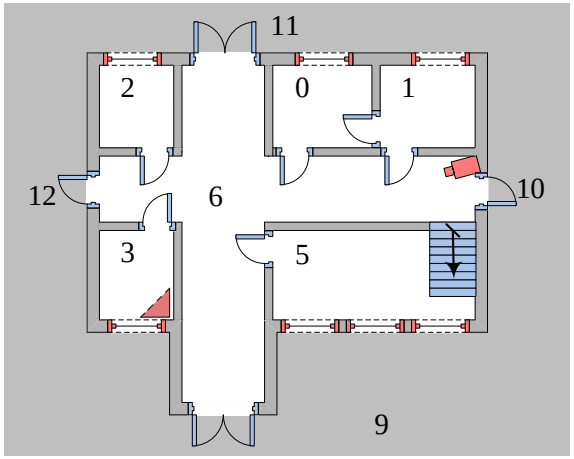


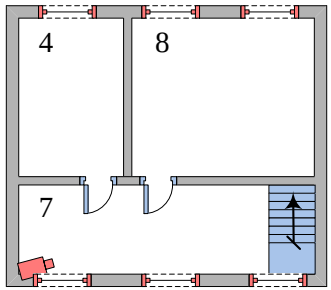


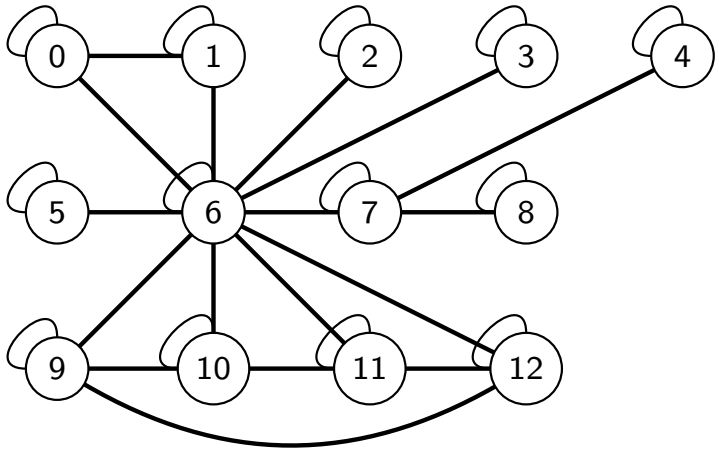


The OSP Problem

Example



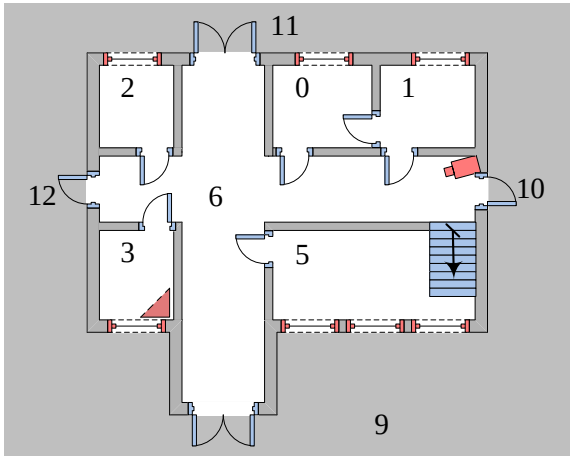


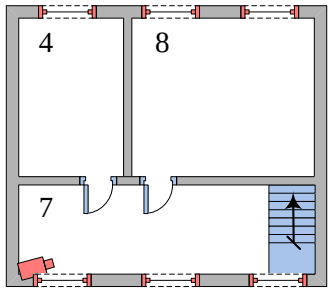


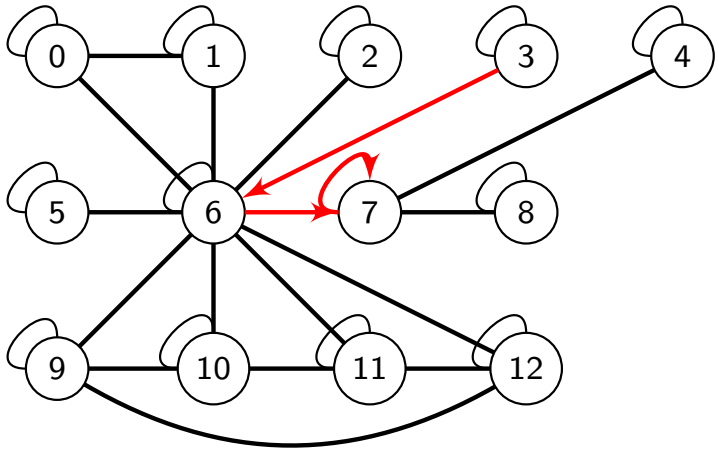
- Let $T = 5$, $y_0 = 3$, $poc_1(4) = 1.0$, $pod(y_t) = 0.9$ $(\forall t \in \mathcal{T})$, and assume a uniform Markovian motion model between accessible vertices.

The OSP Problem

Example



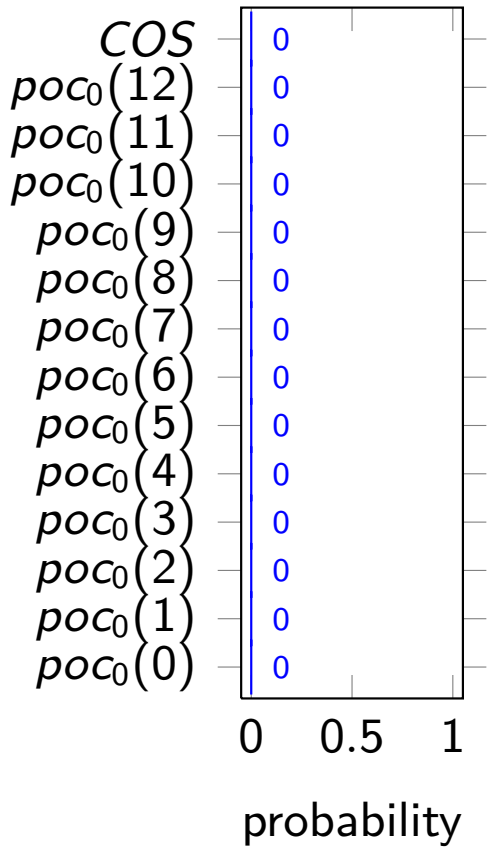




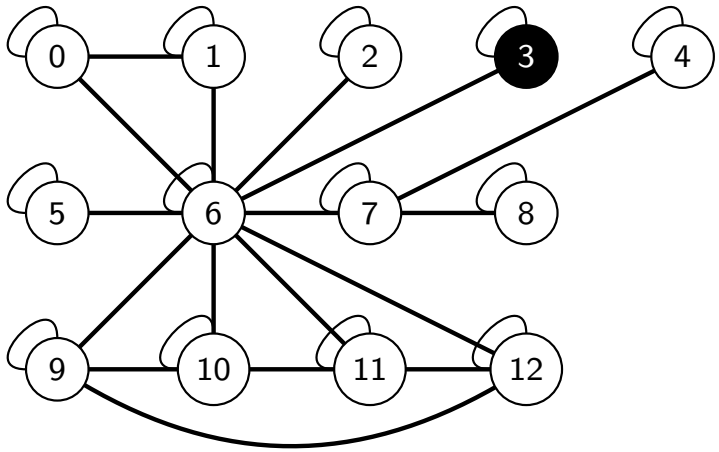
- Let $T = 5$, $y_0 = 3$, $poc_1(4) = 1.0$, $pod(y_t) = 0.9$ $(\forall t \in \mathcal{T})$, and assume a uniform Markovian motion model between accessible vertices.
- P^* is the optimal search plan:

$$P^* = [y_0, y_1, \ldots, y_5] = [3, 6, 7, 7, 7, 7]\,.$$

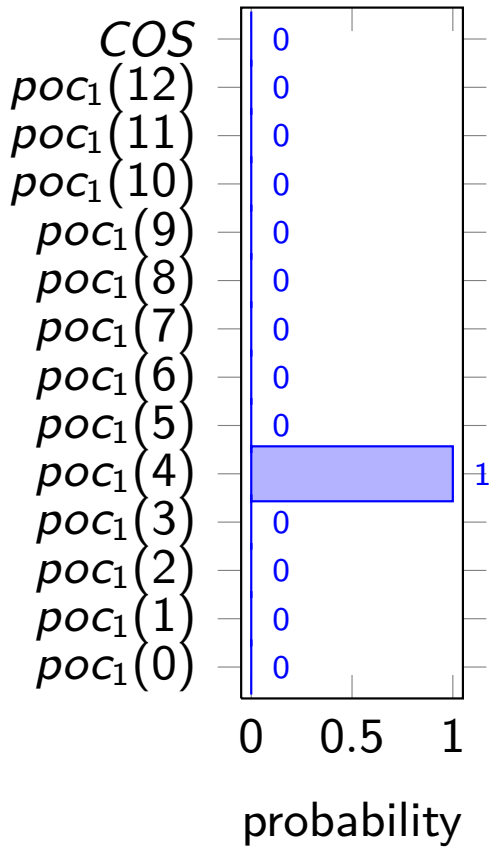
Probability distribution at $t = 0$



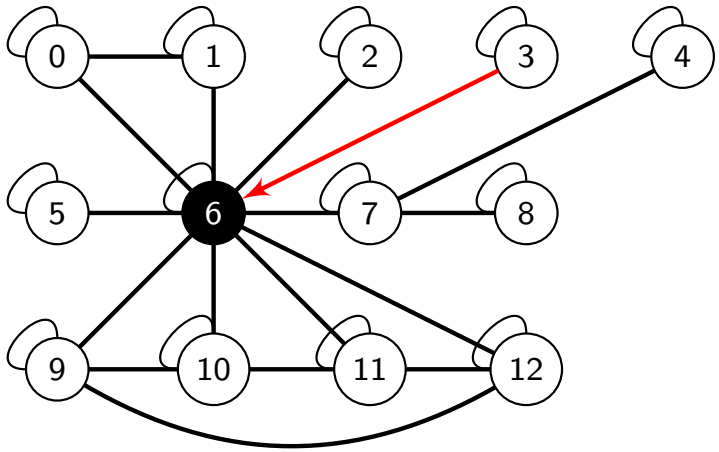
$$P^* = [y_0, y_1, \ldots, y_5] = [\mathbf{3}, 6, 7, 7, 7, 7]\,.$$



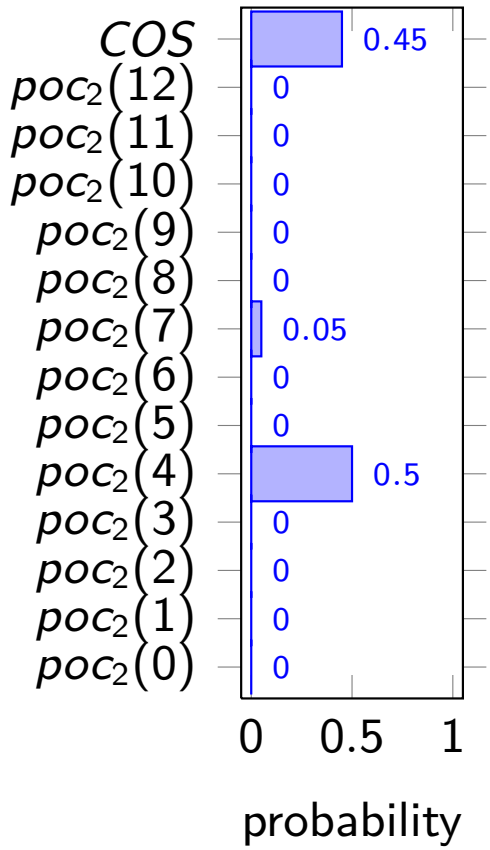
Probability distribution at $t = 1$



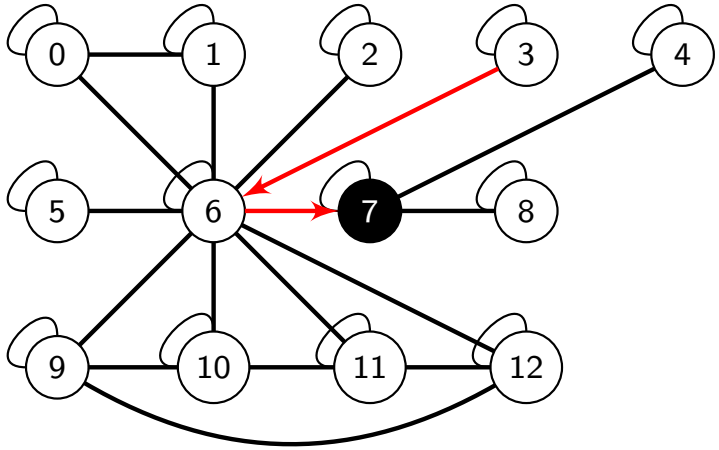
$$P^* = [y_0, y_1, \ldots, y_5] = [3, \mathbf{6}, 7, 7, 7, 7]\,.$$



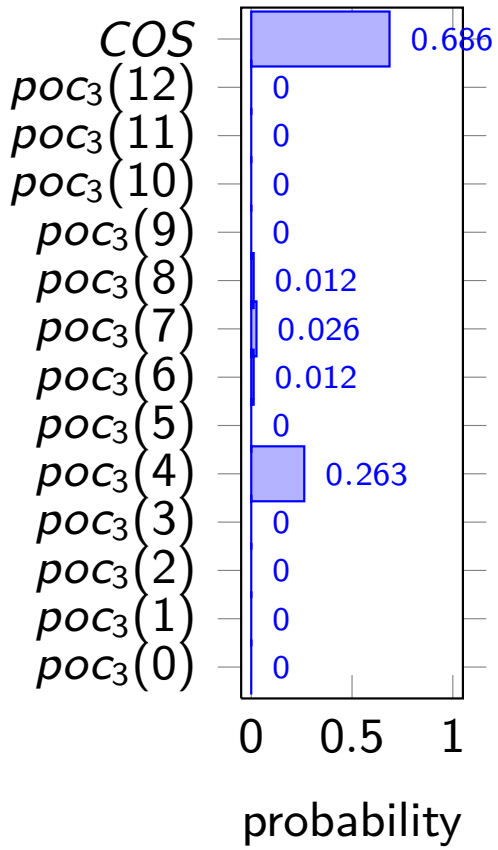
Probability distribution at $t = 2$



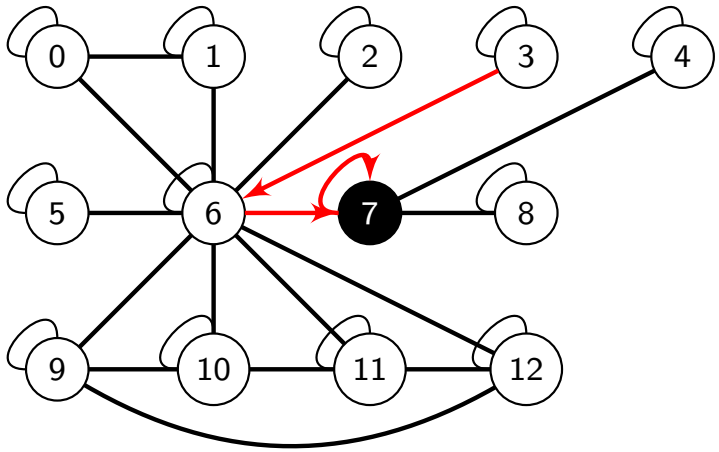
$$P^* = [y_0, y_1, \ldots, y_5] = [3, 6, \mathbf{7}, 7, 7, 7]\,.$$



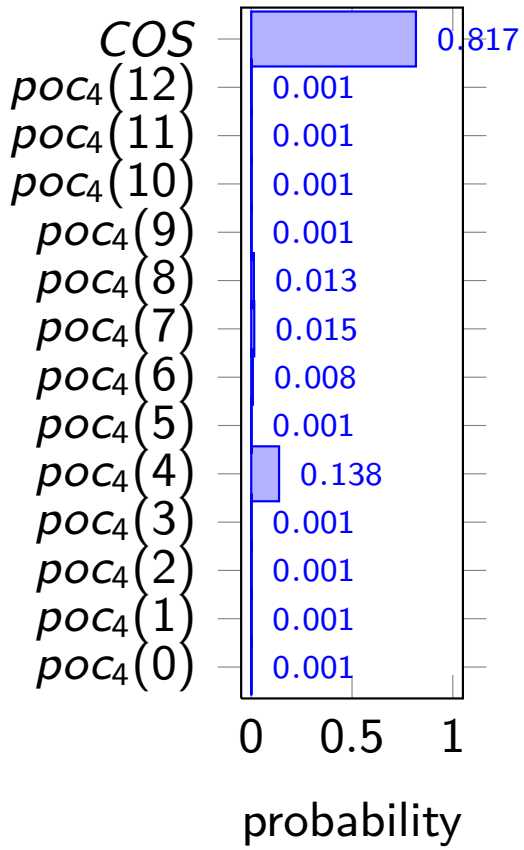
Probability distribution at $t = 3$



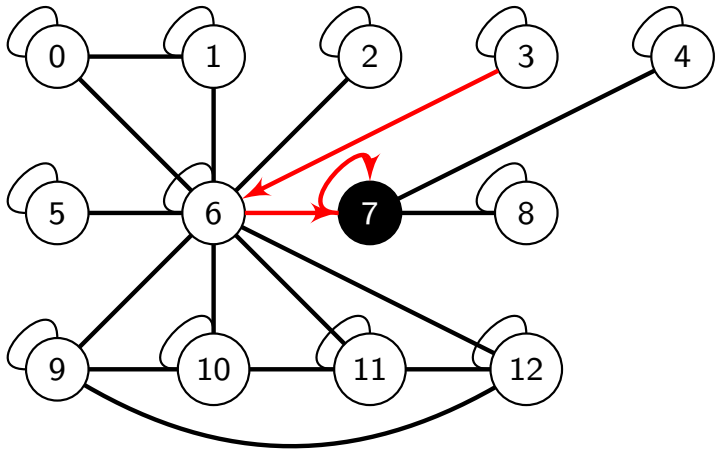
$$P^* = [y_0, y_1, \ldots, y_5] = [3, 6, 7, \mathbf{7}, 7, 7]\,.$$



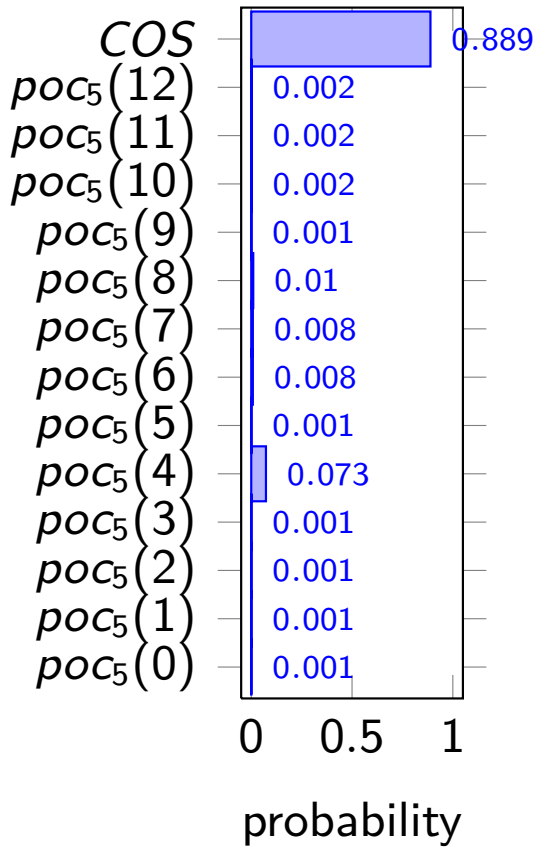
Probability distribution at $t = 4$



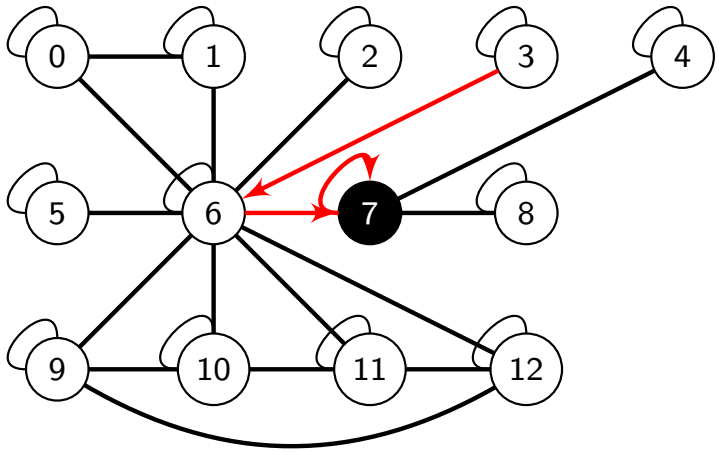
$$P^* = [y_0, y_1, \ldots, y_5] = [3, 6, 7, 7, \mathbf{7}, 7]\,.$$



Probability distribution at $t = 5$



$$P^* = [y_0, y_1, \ldots, y_5] = [3, 6, 7, 7, 7, \mathbf{7}] .$$



A CP Model for the OSP

- The *variables* and the *constraints* are given by the problem definition.
- Two *equivalent* objective functions with a *different* performance:
  - First choice: The double sum definition

$$\max COS,$$
$$COS = \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{V}(G_A)} POS_t(r).$$

  - Second choice: The sum and max definition

$$\max COS,$$
$$COS = \sum_{t \in \mathcal{T}} \max_{r \in \mathcal{V}(G_A)} POS_t(r).$$

A CP Model for the OSP

Two equivalent objective functions

- The searcher searches one vertex per time step.
- Thus, there is only one vertex r such that $POS_t(r) \neq 0$.
- Consequently,

$$\max \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{V}(G_A)} POS_t(r) \equiv \max \sum_{t \in \mathcal{T}} \max_{r \in \mathcal{V}(G_A)} POS_t(r).$$

A CP Model for the OSP

A different performance

- First choice: Poor filtering = poor bound:

$$\sup(\mathrm{dom}(COS)) = \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{V}(G_A)} \sup(\mathrm{dom}(POS_t(r))).$$

- Second choice: Better filtering = better bound:

$$\sup(\mathrm{dom}(COS)) = \sum_{t \in \mathcal{T}} \max_{r \in \mathcal{V}(G_A)} \sup(\mathrm{dom}(POS_t(r))).$$

A CP Model for the OSP

A different performance

- Suppose $T = 2$, and assume a 2 vertices graph. Given the following non null success probabilities variables:

$$\begin{aligned}
\text{dom}(POS_1(1)) &= [0.0, 0.2], \\
\text{dom}(POS_1(2)) &= [0.3, 0.4], \\
\text{dom}(POS_2(1)) &= [0.0, 0.1], \\
\text{dom}(POS_2(2)) &= [0.1, 0.2],
\end{aligned}$$

- What is the value of $\sup(\text{dom}(COS))$ with the $\sum$ function?

A CP Model for the OSP

A different performance

- Suppose $T = 2$, and assume a 2 vertices graph. Given the following non null success probabilities variables:

$$\begin{aligned}
\text{dom}(POS_1(1)) &= [0.0, 0.2], \\
\text{dom}(POS_1(2)) &= [0.3, 0.4], \\
\text{dom}(POS_2(1)) &= [0.0, 0.1], \\
\text{dom}(POS_2(2)) &= [0.1, 0.2],
\end{aligned}$$

- What is the value of $\sup(\text{dom}(COS))$ with the $\sum$ function?
- The upper bound of the $\sum$ objective function is

$$\sup(\text{dom}(COS)) = \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{V}(G_A)} \sup(\text{dom}(POS_t(r))),$$

$$\sup(\text{dom}(COS)) = 0.2 + 0.4 + 0.1 + 0.2 = 0.9.$$

A CP Model for the OSP

A different performance

- Suppose $T = 2$, and assume a 2 vertices graph. Given the following non null success probabilities variables:

$$\begin{aligned}
\text{dom}(POS_1(1)) &= [0.0, 0.2], \\
\text{dom}(POS_1(2)) &= [0.3, 0.4], \\
\text{dom}(POS_2(1)) &= [0.0, 0.1], \\
\text{dom}(POS_2(2)) &= [0.1, 0.2],
\end{aligned}$$

- What is the value of $\sup(\text{dom}(COS))$ with the max function?

A CP Model for the OSP

A different performance

- Suppose $T = 2$, and assume a 2 vertices graph. Given the following non null success probabilities variables:

$$\begin{aligned}
\text{dom}(POS_1(1)) &= [0.0, 0.2], \\
\text{dom}(POS_1(2)) &= [0.3, 0.4], \\
\text{dom}(POS_2(1)) &= [0.0, 0.1], \\
\text{dom}(POS_2(2)) &= [0.1, 0.2],
\end{aligned}$$

- What is the value of $\sup(\text{dom}(COS))$ with the max function?
- The upper bound of the max objective function is

$$\sup(\text{dom}(COS)) = \sum_{t \in \mathcal{T}} \max_{r \in \mathcal{V}(G_A)} \sup(\text{dom}(POS_t(r))),$$

$$\sup(\text{dom}(COS)) = 0.4 + 0.2 = 0.6.$$

The Total Detection Heuristic

- Ignore negative information when searching.
- What is the most promising vertex?
  - The one with the highest *total probability* of detecting the object in the remaining time.

The Total Detection Heuristic

Variables and Values Ordering

- Decision variables order: $Y_0, Y_1, \ldots, Y_T$.
- Values order:

$$\underset{y' \in \text{dom}\,(Y_t)}{\text{argmax}} \sum_{o \in \mathcal{V}(G_A)} w_t(y', o) POC_t(o), \qquad \forall t \in \mathcal{T}.$$

- $w_t(y', o)$ is the conditional probability that the searcher detects the object before the end of the search given that, at time t, the searcher is in y' and the object in o.
- $w_t(y', o)$ is computed using dynamic programming and the following data:
  - the Markovian motion model matrix **M**;
  - the probability of detection *pod*.

- Let $w_t(y, o)$ be the conditional probability that the searcher detects the object before the end of the search given that, at time t, the searcher is in y and the object in o:

$$w_t(y,o) \stackrel{\text{def}}{=} \begin{cases} pod(o), & \text{if } o = y \text{ and } t = T, \\ 0, & \text{if } o \neq y \text{ and } t = T, \\ p_t(y,o), & \text{if } o \neq y \text{ and } t < T, \\ pod(o) + (1 - pod(o))p_t(y,o), & \text{if } o = y \text{ and } t < T. \end{cases}$$

where

$$p_t(y,o) = \sum_{o' \in \mathcal{N}(o)} \mathbf{M}(o,o') \max_{y' \in \mathcal{N}(y)} w_{t+1}(y',o').$$

The Total Detection Heuristic

Summary

- Decision variables order: $Y_0, Y_1, \ldots, Y_T$
- Values order:

$$\underset{y' \in \mathrm{dom}\,(Y_t)}{\mathrm{argmax}} \sum_{o \in \mathcal{V}(G_A)} w_t(y', o) POC_t(o), \qquad \forall t \in \mathcal{T}.$$

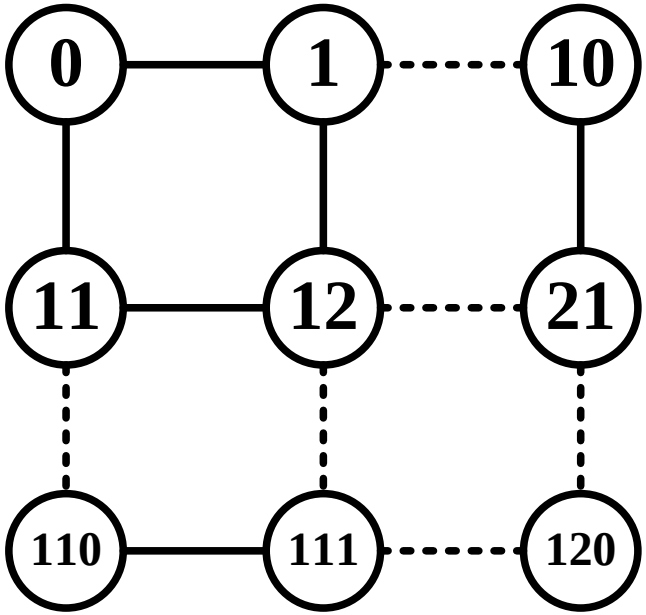
Experimentation

- Three different probabilities of detection: $pod(r) \in \{0.3, 0.6, 0.9\}$ $(\forall r \in \mathcal{V}(G_A))$.
- Three different motion models:

$$\mathbf{M}(s,r) = \begin{cases} \frac{1-\rho}{\deg(s)-1}, & \text{if } (s,r) \in \mathcal{E}(G_A), \\ \rho, & \text{if } s = r, \end{cases}$$

  where $\deg(s)$ is the degree of s and $\rho \in \{0.3, 0.6, 0.9\}$ is the probability that the object stays in its current location.
- Six different allowed time values: $T \in \{9, 11, 13, 15, 17, 19\}$.
- Three different graph structures...

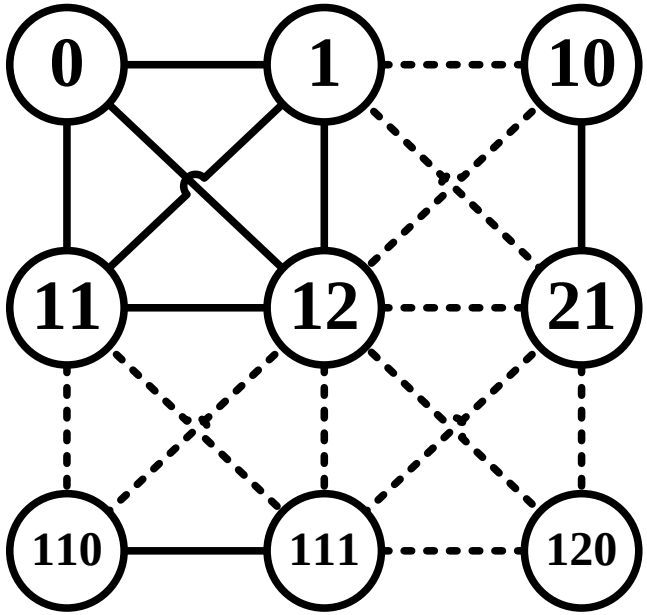
- The 11×11 grid G^+



$$poc_1(60) = 1$$

$$y_0 = 0$$

- The 11×11 grid G^*



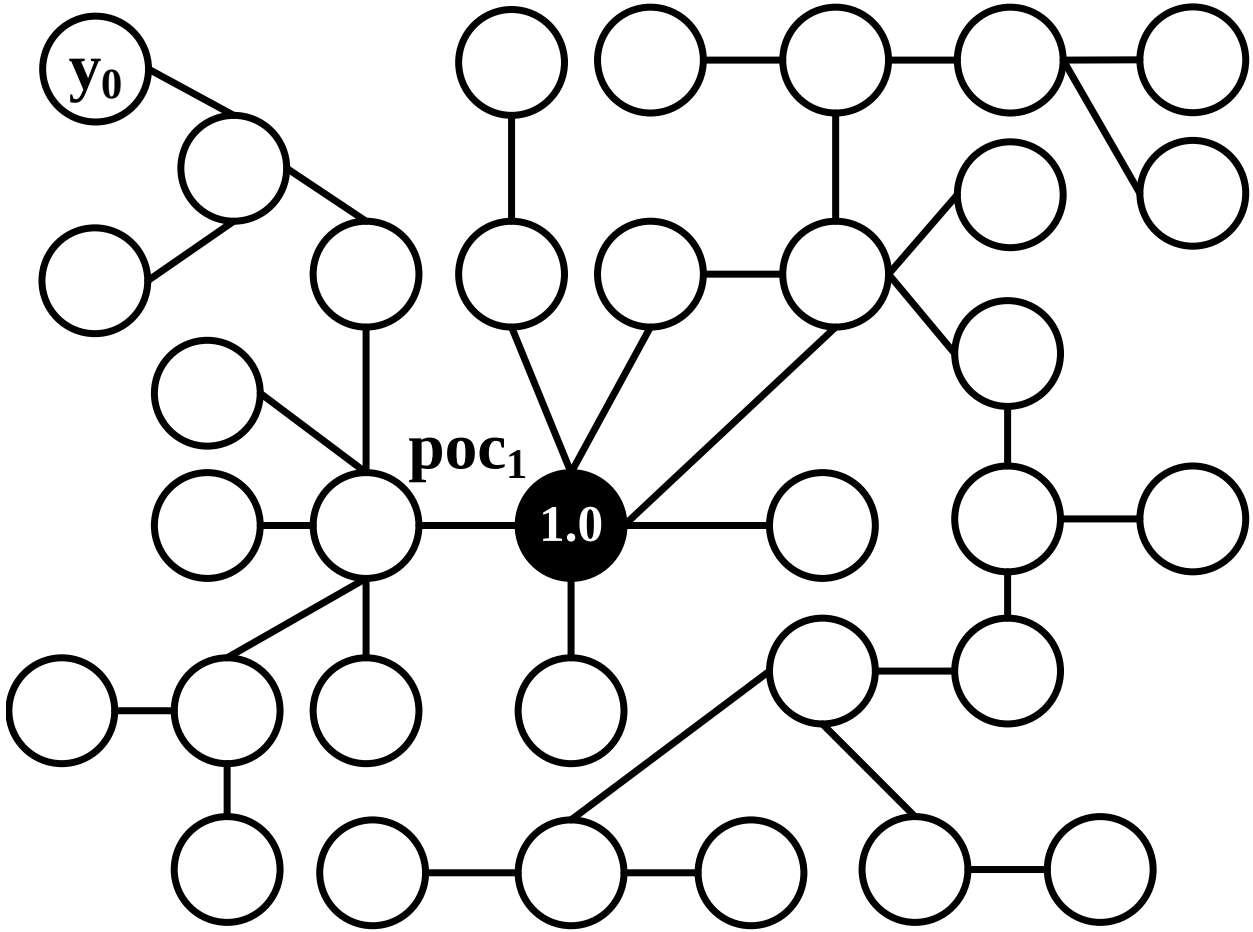
$$poc_1(60) = 1$$

$$y_0 = 0$$

Experimentation

Graph Structures

- The graph G^L (the Université Laval tunnels map)



Experimentation

- Java implementation:
  - Choco solver (Laburthe and Jussien [2012])
  - Java Universal Network/Graph (JUNG) 2.0.1 framework (O'Madadhain et al. [2010])
- 20 minutes time limit
- A maximum of 5,000,000 backtracks

- The CpMax model uses the max objective function.
- The CpSum model uses the $\sum$ objective function.

Table: CpMax vs CpSum on a 11×11 G^+ grid with $T = 17$.

		CpMax		CpSum	
$pod(r)$	ρ	Time to last incumbent (s)	*COS* value	Time to last incumbent (s)	*COS* value
0.3	0.6	**1199**	**0.128**	991	0.127
	0.9	**1026**	**0.338**	1166	0.338
0.6	0.6	**1169**	**0.220**	1016	0.217
	0.9	**1166**	**0.512**	942	0.501
0.9	0.6	**692**	**0.315**	728	0.315
	0.9	**1170**	**0.628**	880	0.625

Results and Discussion

Comparing the CpMax Model and Total Detection

- The CpMax model uses the max objective function.
- The TDValSel+CpMax model uses the Total Detection value selection heuristic.

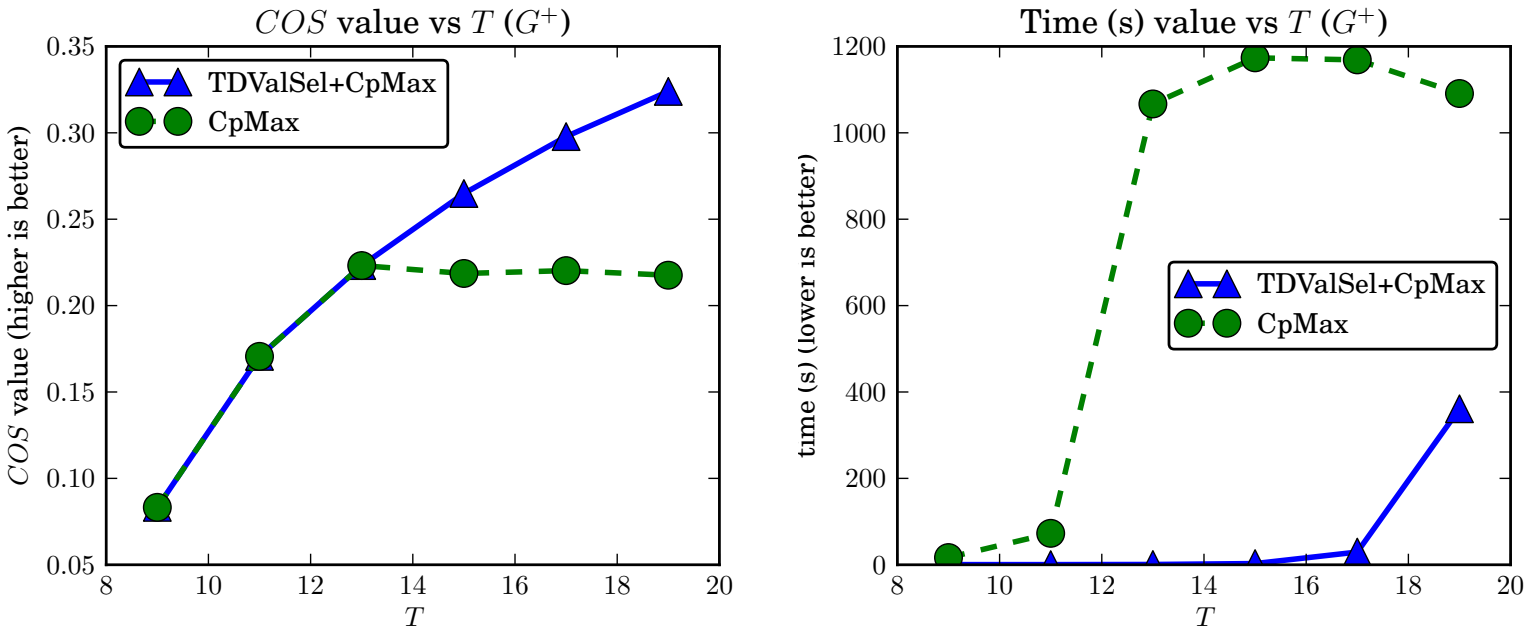
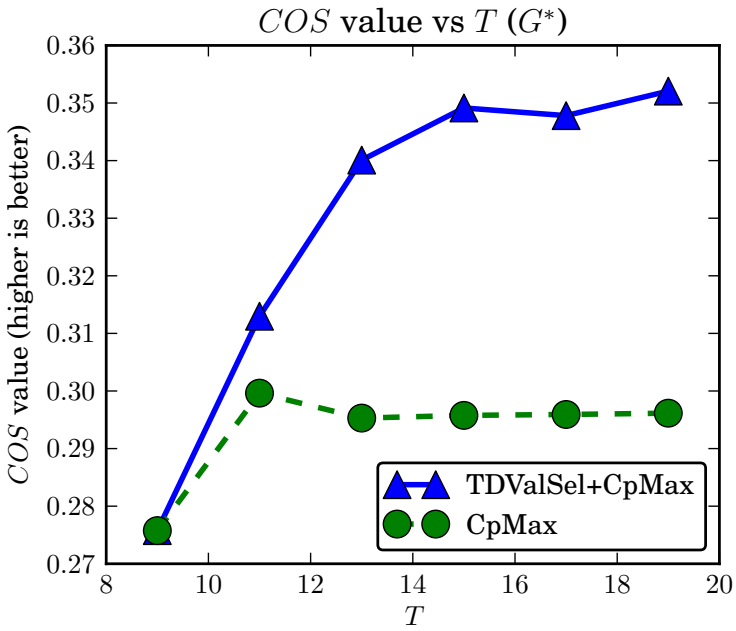


Figure: CpMax vs Total Detection on a 11×11 G^+ instance where $pod(y_t) = 0.6$ $(\forall t \in \mathcal{T})$, and $\rho = 0.6$.

Results and Discussion

Comparing the CpMax Model and Total Detection

- The CpMax model uses the max objective function.
- The TDValSel+CpMax model uses the Total Detection value selection heuristic.



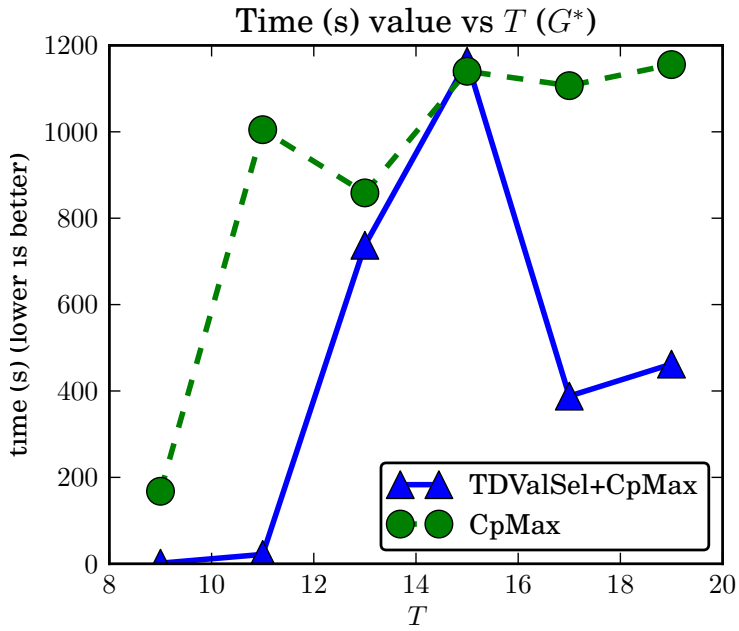


Figure: CpMax vs Total Detection on a 11×11 G^* instance where $pod(y_t) = 0.6$ $(\forall t \in \mathcal{T})$, and $\rho = 0.6$.

- The CpMax model uses the max objective function.
- The TDValSel+CpMax model uses the Total Detection value selection heuristic.

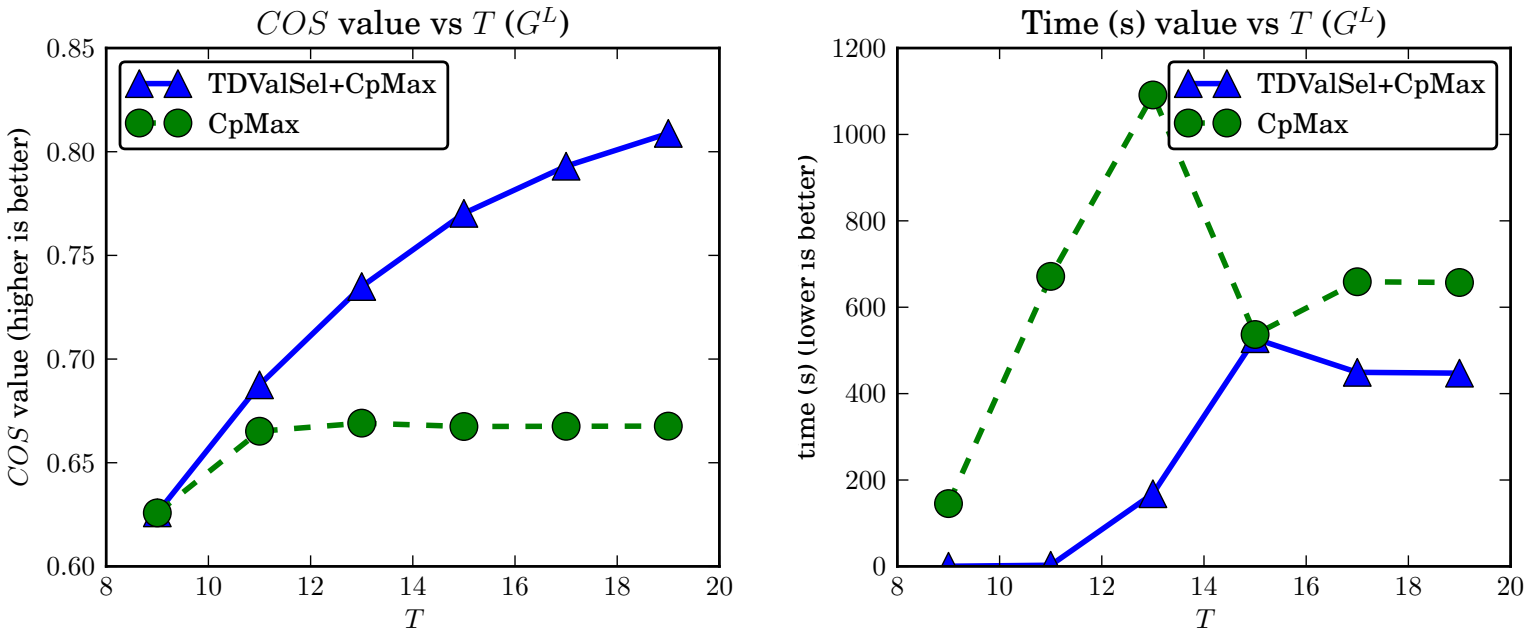


Figure: CpMax vs Total Detection on a G^L instance where $pod(y_t) = 0.6$ $(\forall t \in \mathcal{T})$, and $\rho = 0.6$.

- Indivisible effort:
  - Branch and bound (BB) algorithm [Stewart, 1979]
    - Path constraints are relaxed, use [Brown, 1980] algorithm (sub-optimal) [Stewart, 1979]
    - Indivisibility constraint relaxation, path constraints are maintained [Eagle and Yee, 1990]
    - Reduction to a longest path problem [Martins, 1993] [Lau et al., 2008]
    - Lagrangian relaxation [Sato, 2008]
  - Dynamic programming [Eagle, 1984]

- Infinitely divisible
  - Network flow [Stewart, 1979]
- Arbitrarily divisible
  - Sequential effort allocation (sub-optimal) (small amount of effort) [Stewart, 1979]
  - Network flow (sub-optimal) (large amount of effort) [Stewart, 1979]

Conclusion

- Contributions and novelties:
  - A new CP model to solve the OSP problem
  - A tighter bound using the max objective function encoding
  - The Total Detection heuristic
- Future work:
  - Use the concept of the Total Detection heuristic to develop a better bounding technique for the objective function.

This presentation is based on Morin et al. [2012]:

M. Morin, A.P. Papillon, F. Laviolette, I. Abi-Zeid, and C.G. Quimper, "*Constraint Programming for Path Planning with Uncertainty: Solving the Optimal Search Path problem*," in Proceedings of the 18th Conference on Principles and Practice of Constraint Programming, Québec, Qc, Canada, 2012, pp. 988-1003.

Thank you!

Photography by Yann Arthus-Bertrand

UNIVERSITÉ LAVAL

Stay tuned! :)
http://www.MichaelMorin.info

References I

S. Brown. Optimal search for a moving target in discrete time and space. *Operations Research*, 28(6):1275–1289, 1980.

J. Eagle. The optimal search for a moving target when the search path is constrained. *Operations Research*, 32(5):1107–1115, 1984.

J. Eagle and J. Yee. An optimal branch-and-bound procedure for the constrained path, moving target search problem. *Naval Research Logistics*, 38(1):110–114, 1990.

F. Laburthe and N. Jussien. *Choco Solver Documentation*, 2012. `http://www.emn.fr/z-info/choco-solver/`.

H. Lau, S. Huang, and G. Dissanayake. Discounted mean bound for the optimal searcher path problem with non-uniform travel times. *European journal of operational research*, 190(2):383–397, 2008.

G. Martins. A new branch-and-bound procedure for computing optimal search paths. Technical report, Naval Postgraduate School, 1993.

References II

M. Morin, A. P. Papillon, F. Laviolette, I. Abi-Zeid, and C. G. Quimper. Constraint programming for path planning with uncertainty. In M. Milano, editor, *Principles and Practice of Constraint Programming*, pages 988–1003. Springer Berlin Heidelberg, 2012.

J. O'Madadhain, D. Fisher, T. Nelson, S. White, and Y. Boey. Jung: Java universal network/graph framework. `http://jung.sourceforge.net`, 2010.

H. Sato. *Path optimization for single and multiple searchers: models and algorithms*. PhD thesis, Naval Postgraduate School, Monterey, California, 2008.

T. Stewart. Search for a moving target when the searcher motion is restricted. *Computers and Operations Research*, 6:129–140, 1979.

L. Stone. *Theory of Optimal Search*. Academic Press, New York, 2004.

K. Trummel and J. Weisinger. The complexity of the optimal searcher path problem. *Operations Research*, 34(2):324–327, 1986.