A Conflict-Free Learning Approach for MILP and WCSP Optimisation

1st International Workshop on Solving Linear Optimization Problems for Pseudo-Booleans and Yonder (SLOPPY '24)

Pierre Montalbano[1,2], Simon de Givry[1], and George Katsirelos[3]

[1] Université Fédérale de Toulouse, ANITI, INRAE, UR 875, 31326 Toulouse, France
simon.de-givry@inrae.fr
[2] Université de Tours, LIFAT, ROOT, 37200 Tours, France
pierre.montalbano@univ-tours.fr
[3] Université Fédérale de Toulouse, ANITI, INRAE, MIA Paris, AgroParisTech, 75231 Paris, France
gkatsi@gmail.com

Keywords: graphical model · conflict free learning · combinatorial optimisation · Mixed integer Linear Programming

1 Abstract

Weighted Constraint Satisfaction Problems (WCSP) are an extension of the classic Constraint Satisfaction Problem (CSP), where costs (or weights) are associated with constraint violations. The objective is to find an assignment that minimizes the total constraint violation. Solving WCSPs typically relies on backtracking search and constraint propagation to compute lower bounds. It has the ability to handle global constraints, but it often requires a dedicated algorithm to propagate them. Guided by the success of conflict-based learning methods in multiple domains (such as SAT, Pseudo-Boolean Optimization, or ILP), we designed a new conflict-free learning mechanism. This mechanism aims to memorize, through a linear constraint, the lower bounds of encountered sub-problems. If the same sub-problem (or a similar one) reappears in the search, propagating the previously learned constraint will help to obtain a stronger lower bound. This learning mechanism is conflict-free, meaning that it doesn't require a conflict to be triggered and can be used to learn a constraint at every node of a search tree. Our approach integrates techniques from CP, ILP, and PBO, and we show how it can be embedded inside a classic MILP solver before extending it to WCSP solvers. We implemented this learning mechanism in a simple solver performing branch-and-bound and using CPLEX to solve the LP at every node. We show that it can significantly decrease the number of visited nodes for the knapsack problem or kb-tree (instances with bounded treewidth).

A Conflict-Free Learning Approach for MILP and WCSP Optimization

SLOPPY Workshop

Pierre Montalbano[13], Simon de Givry[1], George Katsirelos[2]

[1]Université Fédérale de Toulouse, ANITI,INRAE, MIAT, UR875,Toulouse, France

[2]Université Fédérale de Toulouse, ANITI,INRAE, MIA Paris, AgroParisTech, France

[3]Laboratoire LIFAT, 64 avenue Jean Portalis, 37200 Tours, France

05/11/2024

Conflict-free learning in MILP

Conflict-free learning

Objective

- Learn the lower bounds of subproblems visited during search

Example

$$\min 3x_1 + 4x_2 + 6x_3 + 8x_4 + x_5 + 6x_6$$

$$s.t$$

$$2x_1 + 2x_2 + 3x_3 + 4x_4 + x_5 + 6x_6 \geq 10$$

$$x_1 + x_2 = 1$$

$$x_i \leq 1 \qquad \forall i \in [1, 6]$$

$$x_i \in \{0, 1\} \qquad \forall i \in [1, 6]$$

MILP standard form

$$\min 3x_1 + 4x_2 + 6x_3 + 8x_4 + x_5 + 6x_6$$

$$s.t$$

$$2x_1 + 2x_2 + 3x_3 + 4x_4 + x_5 + 6x_6 - s = 10$$

$$x_1 + x_2 = 1$$

$$x_i + \overline{x_i} = 1 \qquad \forall i \in [1, 6]$$

$$x_i \in \{0, 1\} \qquad \forall i \in [1, 6]$$

$$\overline{x_i} \geq 0 \qquad \forall i \in [1, 6]$$

$$s \geq 0$$

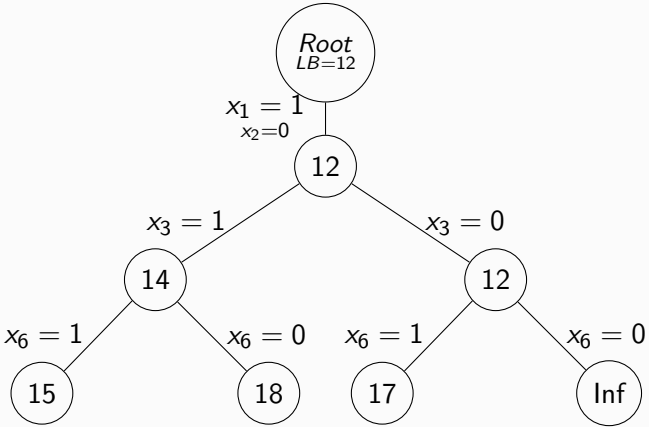
Domain restriction

To remove a variable we increase its coefficient in the objective function by a large value.
Removing x_1:

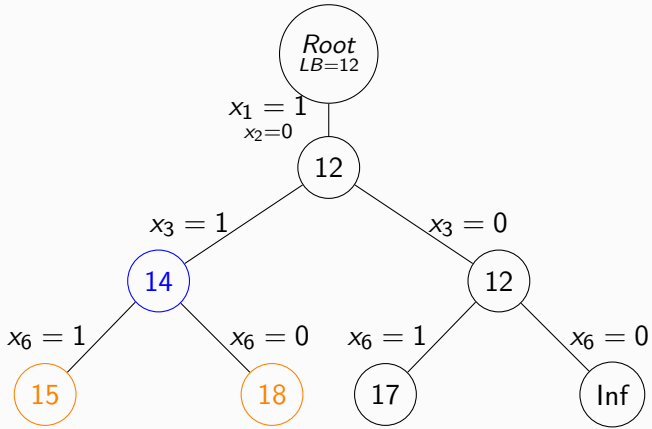
$$\min 100x_1 + 4x_2 + 6x_3 + 8x_4 + x_5 + 6x_6$$

$\rightarrow$ Sets of primal constraints/dual variables are the same at every node of the search tree.

Example

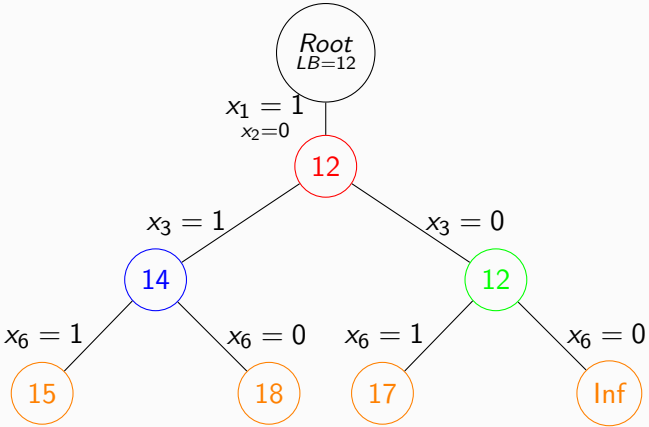


Example



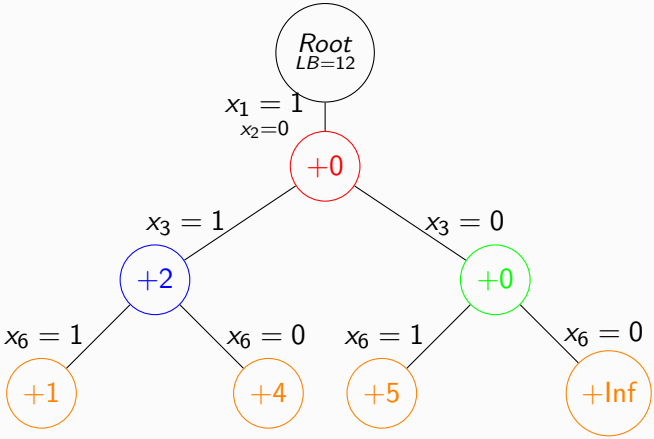
Learn a constraint capturing that the LB of the blue node is 15.

Example



Learn a constraint capturing that the LB of the blue node is 15.
Learn a constraint capturing that the LB of the green node is 17.
Learn a constraint capturing that the LB of the red node is 15.

Example



Learn constraint capturing that we can increase the LB by 3 at blue node.
Learn constraint capturing that we can increase the LB by 5 at green node.
Learn constraint capturing that we can increase the LB by 3 at red node.

Conflict-free learning

Defined for MILP

Farkas constraint [Farkas, 1902]

- Linear combination of primal constraints guided by an LP (dual) solution
- Explain a lower bound

Conflict-free learning [Witzig, 2022]

- Modify the Farkas constraint according to information obtained deeper in the tree $\rightarrow$ tighten the Farkas constraint
- Learned constraints are useful only to prune more values

Memo Constraint

Definition of Memo Constraint

- A constraint $w^T z = b$ is a memo constraint if $w \leq obj$
- A memo constraint $w^T z = b$ explains an LB of b

There exists a family of Farkas constraints that are memo constraints.

Example

$$
\begin{array}{c|c}
\min 3x_1 + 4x_2 + 6x_3 + 8x_4 + x_5 + 6x_6 & \\
2\ x_1 + 2x_2 + 3x_3 + 4x_4 + x_5 + 6x_6 - s = 10 & 2 \\
x_1 + \overline{x_1} = 1 & -1 \\
x_2 + \overline{x_2} = 1 & \\
x_3 + \overline{x_3} = 1 & \\
x_4 + \overline{x_4} = 1 & \\
x_5 + \overline{x_5} = 1 & -1 \\
x_6 + \overline{x_6} = 1 & -6
\end{array}
$$

Example

$$
\begin{array}{c|c}
\min 3x_1 + 4x_2 + 6x_3 + 8x_4 + x_5 + 6x_6 & \\
2\ x_1 + 2x_2 + 3x_3 + 4x_4 + x_5 + 6x_6 - s = 10 & 2 \\
x_1 + \overline{x_1} = 1 & -1 \\
x_2 + \overline{x_2} = 1 & \\
x_3 + \overline{x_3} = 1 & \\
x_4 + \overline{x_4} = 1 & \\
x_5 + \overline{x_5} = 1 & -1 \\
x_6 + \overline{x_6} = 1 & -6
\end{array}
$$

Farkas Constraint: $3x_1 + 3x_2 + 6x_3 + 8x_4 + x_5 - \overline{x_5} + 6x_6 - 6\overline{x_6} - 2s = 12$

Capturing the increase of lower bound ?

From the LP optimal solution rewrite the objective to obtain an **equivalent** problem
$\rightarrow$ We use the reduced costs

- The Farkas constraint saves a subset of costs justifying the increase of LB
- Then we remove that information from the objective function

Example

Problem P :

$$\min 3x_1 + 4x_2 + 6x_3 + 8x_4 + x_5 + 6x_6$$

$$s.t$$

$$2x_1 + 2x_2 + 3x_3 + 4x_4 + x_5 + 6x_6 - s = 10$$
$$x_1 + x_2 = 1$$
$$x_i + \overline{x_i} = 1$$
$$x_i \in \{0, 1\}$$
$$\overline{x_i} \geq 0$$
$$s \geq 0$$

Problem P' :

$$\min 0x_1 + x_2 + \overline{x_5} + 6\overline{x_6} + 2s + 12$$

$$s.t$$

$$2x_1 + 2x_2 + 3x_3 + 4x_4 + x_5 + 6x_6 - s = 10$$
$$x_1 + x_2 = 1$$
$$x_i + \overline{x_i} = 1 \qquad \forall i \in [1, 6]$$
$$x_i \in \{0, 1\} \qquad \forall i \in [1, 6]$$
$$\overline{x_i} \geq 0 \qquad \forall i \in [1, 6]$$
$$s \geq 0$$

Farkas constraint: $3x_1 + 3x_2 + 6x_3 + 8x_4 + x_5 - \overline{x_5} + 6x_6 - 6\overline{x_6} - 2s = 12$

Memo Resolution

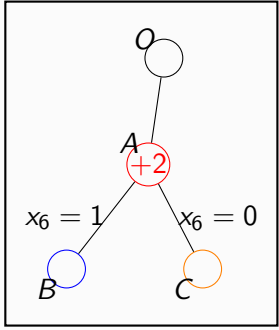
Fusion Resolution[Gocht, Nordstrom, and Buss,2021]

$$\frac{\overline{x_1} + 2x_3 + 4x_4 - x_5 \geq b \qquad x_1 + 2x_3 + 4x_4 - x_5 \geq b'}{2x_3 + 4x_4 - x_5 \geq \min(b, b')}$$

Memo Resolution (simplified)

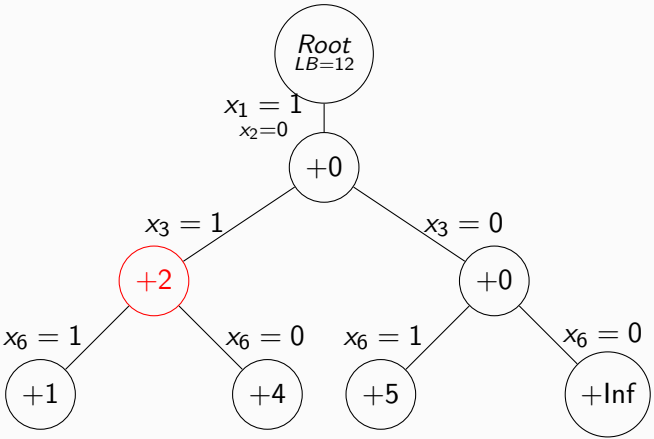
$$\frac{w_j\overline{x_j} + \sum_{i \neq j} w_i x_i = b, \quad w'_j x_j + \sum_{i \neq j} w'_i x_i = b'}{\sum_{i \neq j} \max(w_i, w'_i) x_i \geq \min(b, b')}$$

General Idea



1. Compute a Farkas constraint c_A at node A and transform the objective function. Then compute memo constraints for B and C.
2. Apply memo resolution on c_B and c_C
3. Sum the resulting constraint with the memo constraint c_A
4. Learn the resulting **memo** constraint and return it to node O

Example



Example

1) Compute memo constraint c_A.

$$\min 100x_2 + 100\overline{x_3} + \overline{x_5} + 6\overline{x_6} + 2s + 12$$

$$s.t$$

$$2x_1 + 2x_2 + 3x_3 + 4x_4 + x_5 + 6x_6 - s = 10$$

$$x_1 + x_2 = 1$$

$$x_i + \overline{x_i} = 1 \qquad \forall i \in [1, 6]$$

$$x_i \geq 0 \qquad \forall i \in [1, 6]$$

$$\overline{x_i} \geq 0 \qquad \forall i \in [1, 6]$$

$$s \geq 0$$

$$c_A : \quad 3\overline{x_3} + \overline{x_5} + 6\overline{x_6} - 4x_4 + s = 2$$

Example

1) Compute memo constraint c_A, and transform the objective function:

$$\min 100x_2 + 100\overline{x_3} + 4x_4 + s + 14$$

$$s.t$$

$$2x_1 + 2x_2 + 3x_3 + 4x_4 + x_5 + 6x_6 - s = 10$$

$$x_1 + x_2 = 1$$

$$x_i + \overline{x_i} = 1 \qquad \forall i \in [1, 6]$$

$$x_i \geq 0 \qquad \forall i \in [1, 6]$$

$$\overline{x_i} \geq 0 \qquad \forall i \in [1, 6]$$

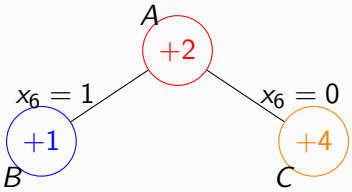
$$s \geq 0$$

$$c_A : \quad 3\overline{x_3} + \overline{x_5} + 6\overline{x_6} - 4x_4 + s = 2$$

General Idea

1) Compute memo constraints c_B, c_C at nodes B and C

$$c_A : \quad 3\overline{x_3} + \overline{x_5} + 6\overline{x_6} - 4x_4 + s = 2$$
$$c_B : \quad 6\overline{x_6} + 3\overline{x_3} - 4x_4 - x_5 + s = 1$$
$$c_C : \quad 6x_6 + x_2 + 4x_4 - 2\overline{x_1} - 3\overline{x_3} - \overline{x_5} - s = 4$$



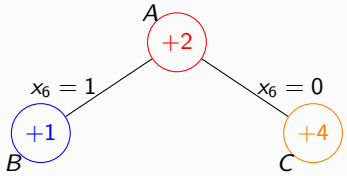
General Idea

2) Apply memo resolution on c_B and c_C.

$$\frac{c_B : 6\overline{x_6} + 3\overline{x_3} - 4x_4 - x_5 + s = 1 \quad c_C : 6x_6 + x_2 + 4x_4 - 2\overline{x_1} - 3\overline{x_3} - \overline{x_5} - s = 4}{c_{BC} : 3\overline{x_3} + x_2 + 4x_4 + s \geq 1}$$

$$c_{BC} : \quad 3\overline{x_3} + x_2 + 4x_4 + s - s_1 = 1$$

c_{BC} is a memo constraint at node A' (**after objective reformulation**).

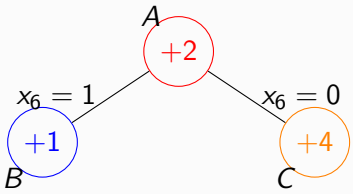


General Idea

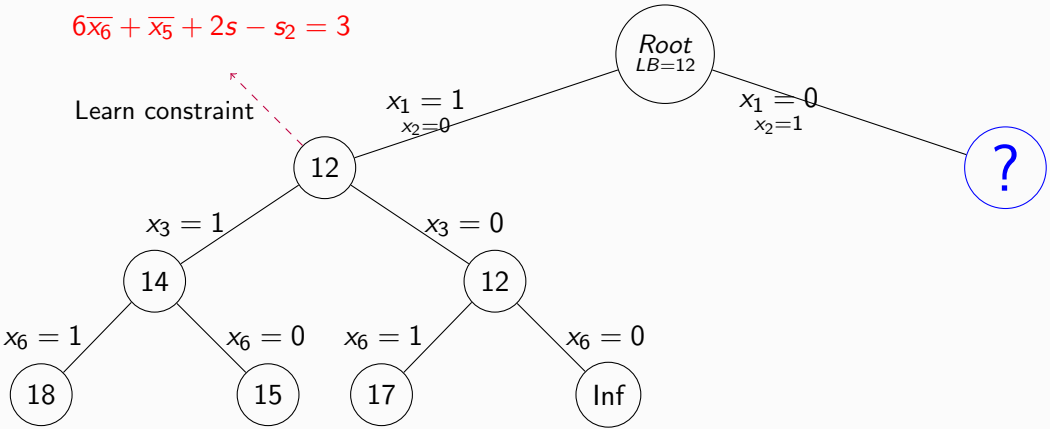
3) Sum the resulting **memo** constraint with the constraint c_A

$$
\begin{aligned}
c_{BC}: \quad & 3\overline{x_3} + x_2 + 4x_4 + s - s_1 = 1 \\
& + \\
c_A: \quad & 3\overline{x_3} + \overline{x_5} + 6\overline{x_6} - 4x_4 + s = 2 \\
\hline
c_{ABC}: \quad & 6\overline{x_3} + x_2 + \overline{x_5} + 6\overline{x_6} + 2s - s_1 = 3
\end{aligned}
$$

c_{ABC} is a memo constraint at node A.



Example


$6\overline{x_6} + \overline{x_5} + 2s - s_2 = 3$
Learn constraint
Root
LB=12
$x_1 = 1$
x_2=0
$x_1 = 0$
x_2=1
?
12
$x_3 = 1$
$x_3 = 0$
14
12
$x_6 = 1$
$x_6 = 0$
$x_6 = 1$
$x_6 = 0$
18
15
17
Inf

Example

$$\min 100x_1 + x_2 + \overline{x_5} + 6\overline{x_6} + 2s + 12$$

$$s.t$$

$$2x_1 + 2x_2 + 3x_3 + 4x_4 + x_5 + 6x_6 - s = 10$$

$$x_1 + x_2 = 1$$

$$6\overline{x_6} + \overline{x_5} + 2s - s_2 = 3$$

$$x_i + \overline{x_i} = 1 \qquad \forall i \in [1, 6]$$

$$\overline{x_i}, x_i \geq 0 \qquad \forall i \in [1, 6]$$

$$s, s_1 \geq 0$$

Without the red constraint: OPT=13 $\rightarrow$ the search continue
With the red constraint: OPT=16 $\rightarrow$ end of search

Implementation

Python script based on CPLEX Python API

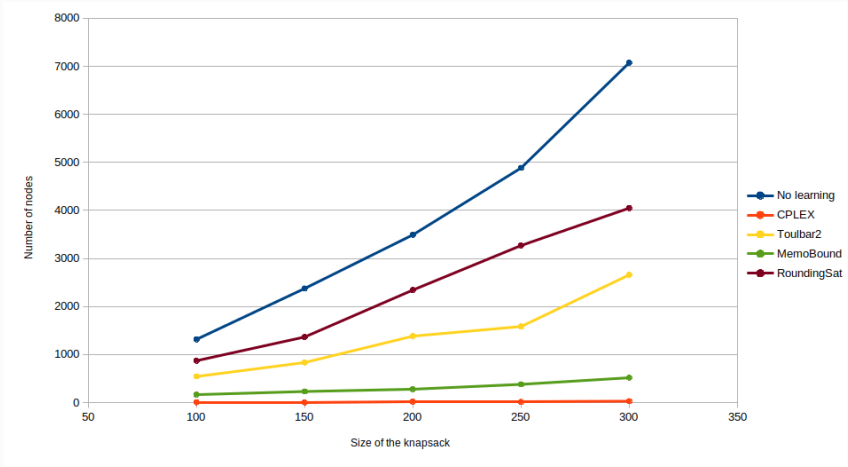
No learning	Depth-first search B&B Branch on the first fractional variable Value removal by bound propagation Solve the LP
MemoBound	+ Rewrite the objective function Value removal by node consistency Learn memo constraints

Results

Knapsack problem

- 100-300 variables
- Random weights and profits

Results


8000
7000
6000
5000
4000
3000
2000
1000
0
Number of nodes
50
100
150
200
250
300
350
Size of the knapsack
No learning
CPLEX
Toulbar2
MemoBound
RoundingSat

Experimental results

Knapsack Problem with Conflict Graph (KPCG)

- 120 or 250 Boolean variables.
- One linear constraint of size equal to the number of variables.
- Weights are uniformly distributed. Sometimes correlated to the profit (instance C).
- A conflict graph (binary constraints) of varying density (0.1-0.3).
- 10 instances in each class

KPCG

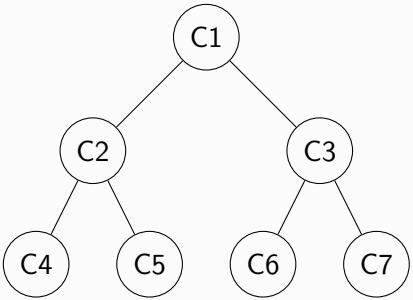
	No learning	MemoBound	TOULBAR2	RoundingSat	CPLEX
$R^1120-0.1$	1030	226	213	2660	0.3
$R^1120-0.2$	1054	270	260	2558	0
$R^1120-0.3$	1004	298	270	2546	1.7
$R^1250-0.1$	2123	418	522	8985	0
$R^3120-0.1$	2272	472	476	5330	0.8
$R^3120-0.2$	3064	906	1026	6061	39.2
$R^3120-0.3$	3115	1487	1886	6650	66.8
$R^3250-0.1$	9423	1223	1679	17532	10.3
$C^1120-0.1$	10989	2646	1580	6064	0
$C^1120-0.2$	8672	2292	1151	8779	5.8
$C^1120-0.3$	6437	2156	1537	8043	73

Table: Average number of nodes developed to solve different configurations of the KPCG problem.

Results

Kbtree problem

- 44 to 188 Boolean variables
- 190 to 838 binary constraints
- Very specific structure (bounded tree-width)



- Each cluster is a complete graph.
- 2 clusters are connected by 2 *separators*.
- kb-7-3 indicates clusters of size 7 and tree height 3.

Results

	Variables	Constraints	No learning	MemoBound	TOULBAR2-BTD	ROUNDINGSAT	CPLEX
kb-7-3	44	190	7.79	5.9	5.58	4649	0 (573)
kb-7-4	92	406	43	17	16.33	64549	0 (1235)
kb-7-5	188	838	1240	262	37	-	0 (2556)
kb-8-3	51	246	13.89	8	12	7568	0 (758)
kb-8-4	107	526	128	40	40	153374	0 (1613)
kb-9-3	58	309	26	14	27	19153	0(979)
kb-9-4	122	661	457	156	95	-	0(2071)

Conclusion

- The proof of concept validates our theory

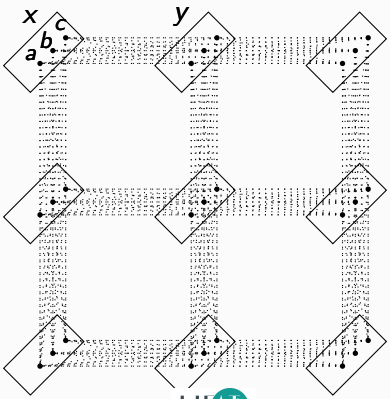
- Implementation in a fully functional solver?
- Heuristics?
  - Restart
  - Selecting the learned constraints
  - Constraint strengthening

- Theoretical results
  - Comparable to dynamic programming?

Conflict-free learning in WCSP

Graphical Models

- The nodes represent discrete domain variables
- (Hyper)-edges represent interactions between variables.



Different types of GM:

- Bayesian Networks (probabilities)
- Markov Random Fields (potentials)
- **Cost Function Networks** (costs)

Cost function

Definition: Cost function

- Scope A (a set of variables)
- Associate a cost to each tuple in the scope:
  - $f_A : \prod_{x \in A} D_x \rightarrow \mathbb{N} \bigcup \{\infty\}$

Unary cost function f_x

x	f_x
a	2
b	0

Binary cost function f_{xy}

x	y	f_{xy}
a	a	3
a	b	2
b	a	0
b	b	∞

Hard binary constraint f_{xz}

x	z	f_{xz}
a	a	∞
a	b	0
b	a	0
b	b	∞

Weighted Constraint Satisfaction Problem

Definition: Cost Function Network $P = (V, S, f)$

- Set V of discrete domain variables
- Set S of scopes
- For each scope $A \in S$, we define a cost function:
  - $f_A : \prod_{x \in A} D_x \rightarrow \mathbb{N} \bigcup \{\infty\}$

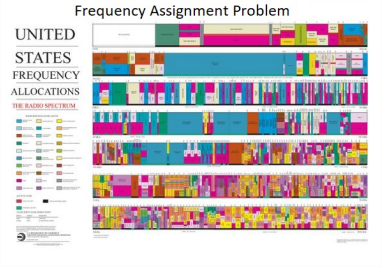
Objective:

Find a complete assignment v minimizing $\sum_{A \in S} f_A(v[A])$

$\rightarrow$ NP-Hard Problem

Cost functions in real life

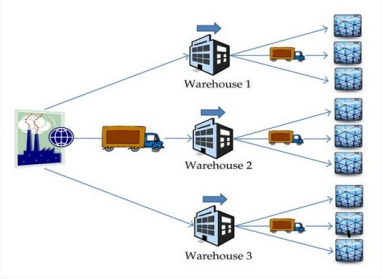
Frequency Assignment Problem



Grid operation-based outage maintenance planning

Protein Design

Warehouse location problem



Example

x	y	f_{xy}
a	a	3
a	b	2
b	a	0
b	b	∞

x	f_x
a	0
b	2

y	f_y
a	0
b	2

$$\min_{x,y} \quad f_x(x) + f_y(y) + f_{xy}(x, y)$$

Example

x	y	f_{xy}
a	a	3
a	b	2
b	a	0
b	b	∞

x	f_x
a	0
b	2

y	f_y
a	0
b	2

$$\min_{x,y} \quad f_x(x) + f_y(y) + f_{xy}(x, y)$$

$(x = b, y = a)$ is the optimal assignment with cost 2.

How to compute the lower bounds ?

- Soft arc consistency algorithms
- Equivalence preserving transformations

Equivalence Preserving Transformation

Equivalent WCSP

$P = (V, S, f)$ and $P' = (V, S, f')$ are equivalent if for any complete assignment v:

$$\sum_{A \in S} f_A(v[A]) = \sum_{A \in S} f'_A(v[A])$$

Equivalence Preserving Transformation (EPT)

Transform a WCSP P into an equivalent WCSP P' by moving costs from a cost function f_A to another cost function f_B.

Example

x	y	f_{xy}
a	a	3
a	b	2
b	a	0
b	b	∞

x	f_x
a	0
b	2

y	f_y
a	0
b	2

$$\min_{x,y} \quad f_x(x) + f_y(y) + f_{xy}(x, y)$$

Example

x	y	f_{xy}
a	a	3-2
a	b	2-2
b	a	0
b	b	∞

x	f_x
a	0+2
b	2

y	f_y
a	0
b	2

$$\min_{x,y} \quad f_x(x) + f_y(y) + f_{xy}(x, y)$$

Example

x	y	f_{xy}
a	a	1
a	b	0
b	a	0
b	b	∞

x	f_x
a	2
b	2

y	f_y
a	0
b	2

$$\min_{x,y} \quad f_x(x) + f_y(y) + f_{xy}(x, y)$$

Example

x	y	f_{xy}
a	a	1
a	b	0
b	a	0
b	b	∞

x	f_x
a	2-2
b	2-2

y	f_y
a	0
b	2

$$f_\emptyset = 2$$

$$\min_{x,y} \quad f_x(x) + f_y(y) + f_{xy}(x,y) + f_\emptyset$$

Example

x	y	f_{xy}
a	a	1
a	b	0
b	a	0
b	b	∞

x	f_x
a	0
b	0

y	f_y
a	0
b	2

$$f_\emptyset = 2$$

$$\min_{x,y} \quad f_x(x) + f_y(y) + f_{xy}(x,y) + f_\emptyset$$

How to Solve a WCSP?

Branch&Bound Algorithm

At each node of the search tree produce a sequence of EPTs maximizing $f_\emptyset$

The optimal sequence (using rational costs) can be obtained from the optimal solution of a linear problem: **The Local Polytope**.
However, solving this LP to optimality is often **too expensive**

Soft Local Consistency Algorithms

Reason on a 'local' level by considering only a subset of cost functions.

- Prune locally inconsistent values
- Define a sequence of EPTs increasing $f_\emptyset$

Examples:

Node Consistency, Soft Arc Consistency, Existential Directional Arc Consistency, **Virtual Arc Consistency (VAC)**,...

Global constraints

Examples: alldiff, among, clique, grammar, Pseudo-Boolean...

Global constraints

1. Hard Global Constraint
   - Representation is implicit
   - Design a dedicated propagator
2. Soft global constraint

Conflict-free learning in CFNs

- We know a LP: **The Local Polytope**
- Soft consistency algorithms
  - Find solutions of the The Local Polytope
  - Natively reformulate the problem

Thanks

Thanks for your attention! Questions ?

pierre.montalbano@univ-tours.fr


Farkas, J. (1902).
Theorie der einfachen ungleichungen.
Journal für die reine und angewandte Mathematik (Crelles Journal), 1902(124):1–27.


Witzig, J. (2022).
Infeasibility Analysis for MIP.
Technische Universitaet Berlin (Germany).