

Combining exact WCSP techniques and VNS search for solving MPE

D. Allouche, A. Favier, S. de Givry, T. Schiex, M. Zytnicki
UBIA UR 875, INRA, F-31320 Castanet Tolosan, France
M. Fontaine, J-P Métivier, GREYC-CNRS, UMR 6072, U.
Caen, France
M. Sanchez (Spain), K. L. Leung (China)



FICOLOFO

August 2012

- 1 WCSP
- 2 MPE to WCSP
- 3 DFBBVE
- 4 VNS/LDS+CP
- 5 Results

WCSP

$(\mathcal{X}, \mathcal{D}, \mathcal{C}, T)$ is a WCSP:

- $\mathcal{X} = \{X_1, \dots, X_n\}$ set of variables,
- $\mathcal{D} = \{D_1, \dots, D_n\}$ set of their finite domains (max size d),
- $\mathcal{F} = \{f_1, \dots, f_e\}$ set of cost functions,
 - set of variables (scope) of f_i : $var(f_i)$,
 - cost function $f_i: \otimes_{j \in var(f_i)} D_j \rightarrow [0..T]$,
- $T \in [1, +\infty]$ is an integer value (maximum cost value)

Goal: to find a complete assignment minimizing the sum of the cost functions. (NP-hard)

Most Probable Explanation (MPE) in Bayesian Networks

Find a complete assignment A maximizing

$$\prod_{i \in [1, n]} P(A[i] | A[\text{Parent}(i)])$$

is translated into

WCSP

Find a complete assignment A minimizing

$$\sum_{i \in [1, n]} \lceil -C \times \log P(A[i] | A[\text{Parent}(i)]) \rceil$$

With C a positive constant (for real-to-integer precision)

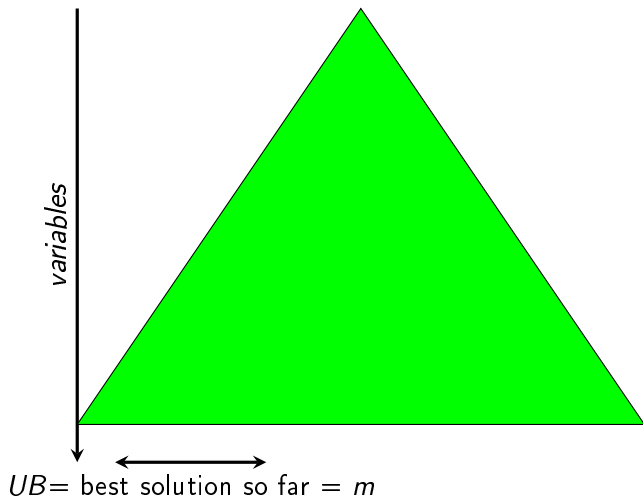
DFBBVE

- complete tree search
- use local consistency to compute lower bound
- preprocess the problem to remove low degree variables

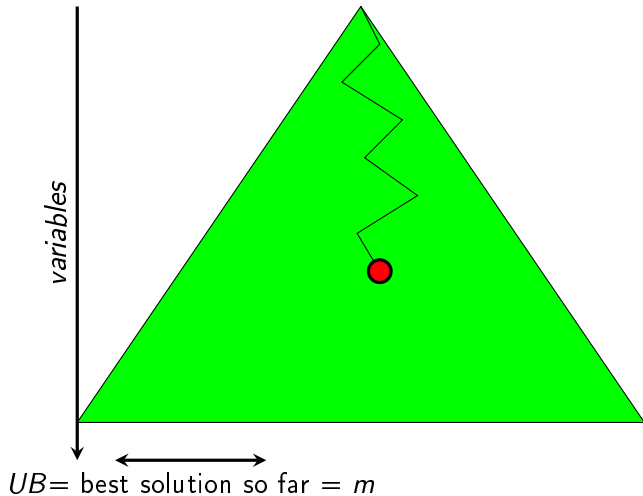
VNS/LDS+CP

- local search using large neighborhood of variable size (VNS)
- A partial tree search to explore the search space (LDS)
- A lower bound computation based on soft local consistency to prune the search space (CP)

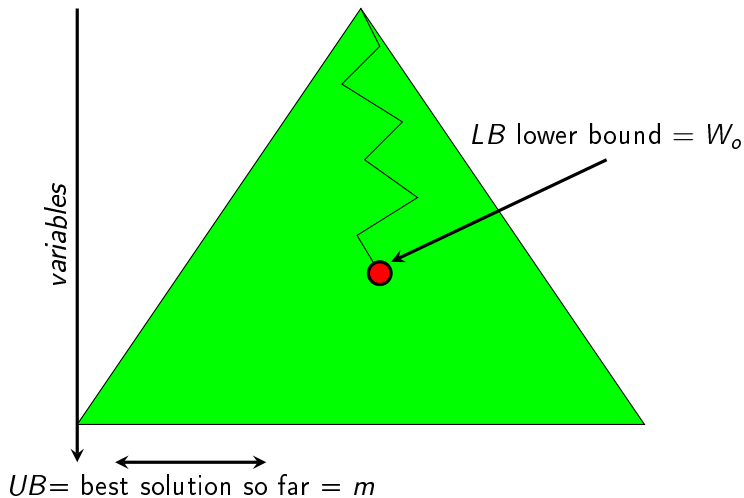
- Binary branching scheme instead of value enumeration
- Dynamic variable and value ordering heuristics (pseudo learning/CBJ: weighted degree/last conflict)



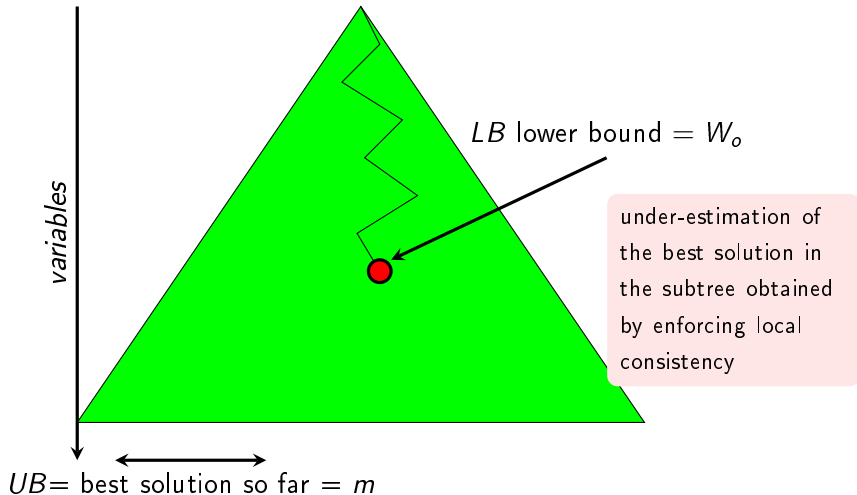
- Binary branching scheme instead of value enumeration
- Dynamic variable and value ordering heuristics (pseudo learning/CBJ: weighted degree/last conflict)



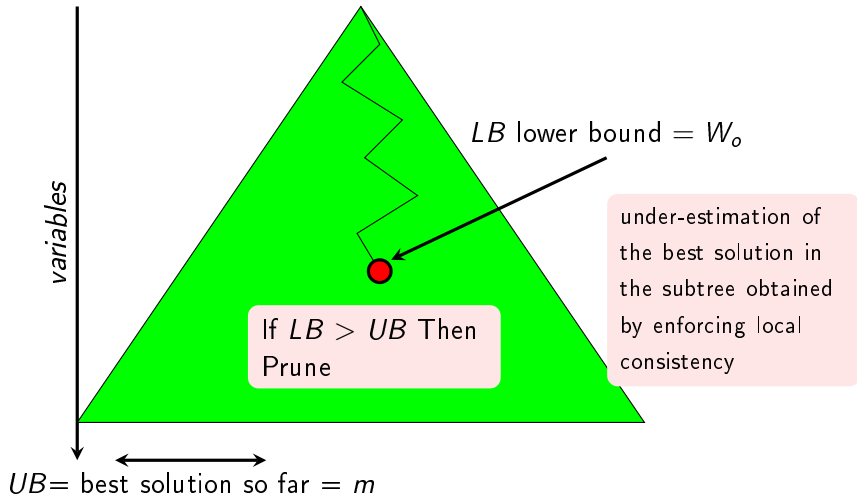
- Binary branching scheme instead of value enumeration
- Dynamic variable and value ordering heuristics (pseudo learning/CBJ: weighted degree/last conflict)



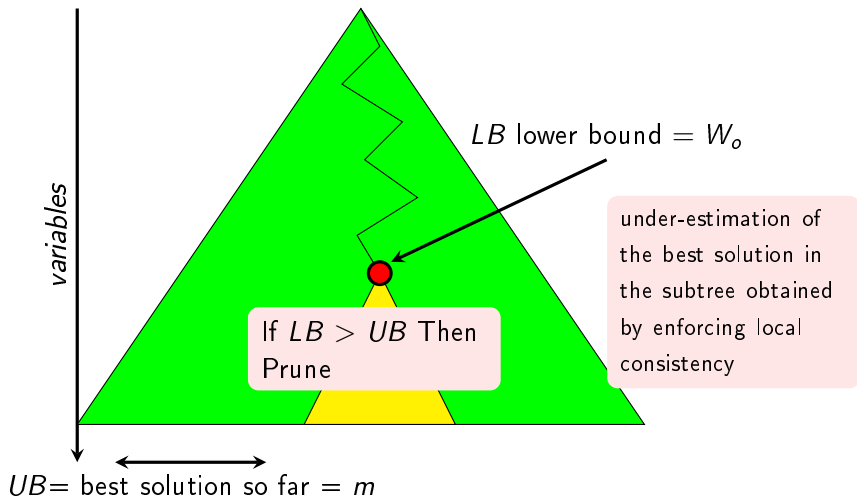
- Binary branching scheme instead of value enumeration
- Dynamic variable and value ordering heuristics (pseudo learning/CBJ: weighted degree/last conflict)



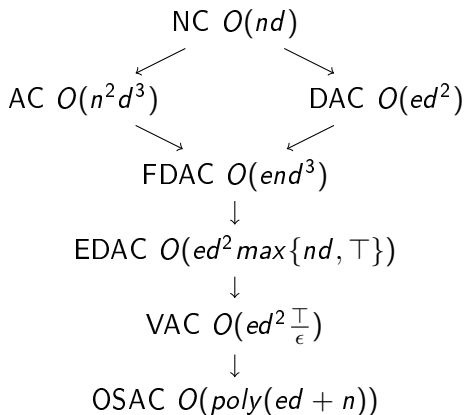
- Binary branching scheme instead of value enumeration
- Dynamic variable and value ordering heuristics (pseudo learning/CBJ: weighted degree/last conflict)



- Binary branching scheme instead of value enumeration
- Dynamic variable and value ordering heuristics (pseudo learning/CBJ: weighted degree/last conflict)

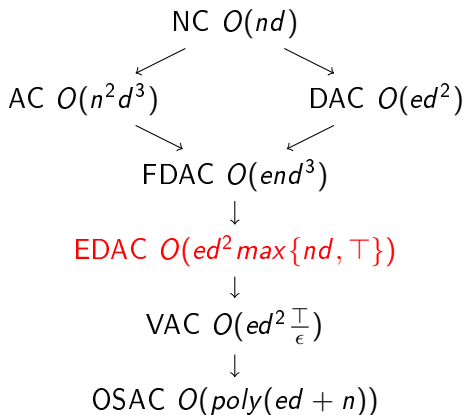


Hierarchy of soft arc consistencies (with time complexities for binary cost functions)



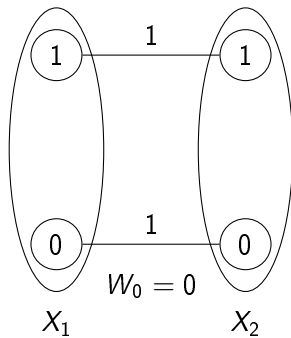
n : num. of variables, d : max domain size, e : num. of cost functions, T : maximum cost value

Hierarchy of soft arc consistencies (with time complexities for binary cost functions)



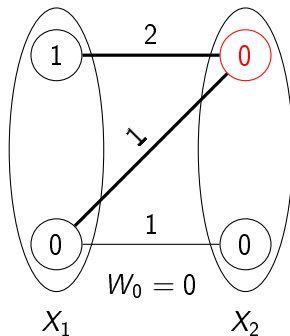
n : num. of variables, d : max domain size, e : num. of cost functions, T : maximum cost value

Enforcing soft arc consistency by Equivalence Preserving Transformations



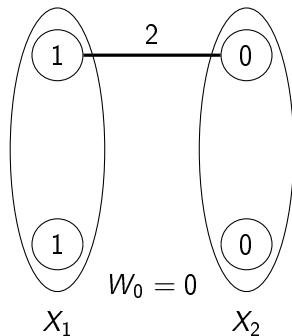
Enforcing soft arc consistency by Equivalence Preserving Transformations

Cost extension from a unary to a binary cost function



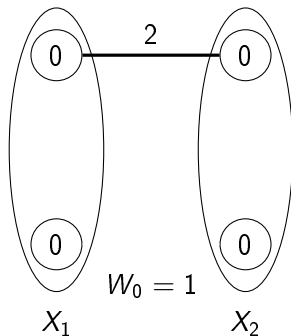
Enforcing soft arc consistency by Equivalence Preserving Transformations

Cost projection from a binary to a unary cost function



Enforcing soft arc consistency by Equivalence Preserving Transformations

Cost projection from a unary cost function to W_0



- Based on VNS [Mladenovic and Hansen, 1997].

Let N_k be the set of all combinations of k variables and S a current solution:

- Select in N_k a set $\mathcal{X}_{unassigned}$ of k variables to unfix in S .
- Rebuild S using a partial tree search (Limited Discrepancy Search) and constraint propagation (CP) on $\mathcal{X}_{unassigned}$.
- Neighborhood change:
 - if a better solution S' is found
 - **Intensification:** $S \leftarrow S'$ and $k \leftarrow k_{init}$
 - else
 - **Diversification:** $k \leftarrow k + 1$

¹(Loudni et Boizumault, EJOR 2008)

Algorithm 1: VNS/LDS+CP

```
function VNS/LDS+CP( $\mathcal{X}, \mathcal{C}, k_{init}, k_{max}, \delta_{max}$ ) ;
begin
1   |    $S \leftarrow \text{genInitSol}()$  ;
2   |    $k \leftarrow k_{init}$ ;
3   |   while ( $k < k_{max}$ )  $\wedge$  (not TimeOut) do
4   |       |    $\mathcal{X}_{unassigned} \leftarrow \text{Hneighborhood}(N_k, S)$  ;
5   |       |    $\mathcal{A}_k \leftarrow S \setminus \{(x_i = a) \mid x_i \in \mathcal{X}_{unassigned}\}$  ;
6   |       |    $S' \leftarrow \text{LDS+CP}(\mathcal{A}_k, \mathcal{X}_{unassigned}, \delta_{max}, f(S), S)$  ;
7   |       |   if  $f(S') < f(S)$  then
8   |       |       |    $S \leftarrow S'$  ;
9   |       |       |    $k \leftarrow k_{init}$  ; // intensification
10  |       |   end
11  |       |   else  $k \leftarrow k + 1$  ; // diversification
12  |   end
13  |   return  $S$  ;
14 end
```

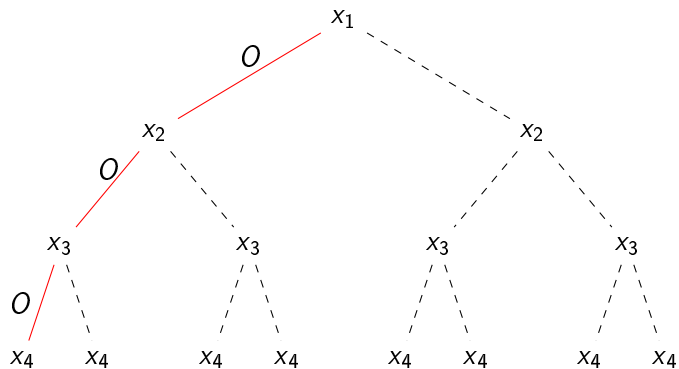
LDS + CP allows us :

- To invalidate early partial assignments which cannot lead to solutions of better quality.
- To obtain a faster and more balanced exploration of the search space.
- To guide the search towards the best neighbor solutions.
(powerful value heuristics deduced from propagation)

- Partial tree search
- Allows a certain number of heuristic discrepancies during search
- Better diversification of the exploration of the search space

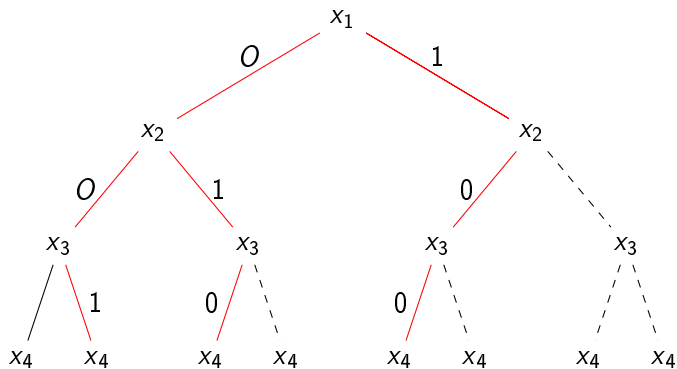
LDS example

$$\delta = 0$$



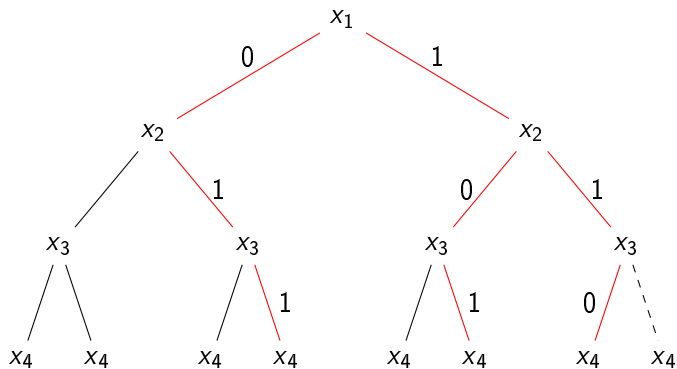
LDS example

$$\delta = 1$$



LDS example

$$\delta = 2$$



VNS

- $k_{min} = 3, k_{max} = n$
- initial solution: *LDS* with $\delta = 0$
- variable selection: conflict-var

LDS+CP

- LDS: $\delta = 3$
- CP: EDAC

DFBBVE

- preprocessing : removing variable of degree < 6
- MCS for DAC order
- *LDS* with $\delta = 4$
- restart when 10,000 nodes have been visited

Results

	VNS/LDS+CP	DFBBVE	best
20s	-6.0819	-6.0518	-6.1364
20min	-7.8000	-7.8041	-7.8519
1h	-8.3196	-8.3033	-8.3214

- Open source solver (last release 0.9.5 ; 30,000 lines of C++)
<http://mulcyber.toulouse.inra.fr/projects/toulbar2>
- Problem file formats: wcsp, uai08/uai10 (MAP for Markov Random Fields and MPE for Bayesian Networks), cnf, wcnf, qpbo, ...
- Large set of benchmarks and toulbar2 solver web service at <http://costfunction.org/>
- Links to ILOG Solver (C++ CP solver) and NumberJack (python interface to MIP/CP/SAT solvers)