Recherche complète à voisinages variables guidée par la décomposition arborescente pour la minimisation d'énergie dans les modèles graphiques

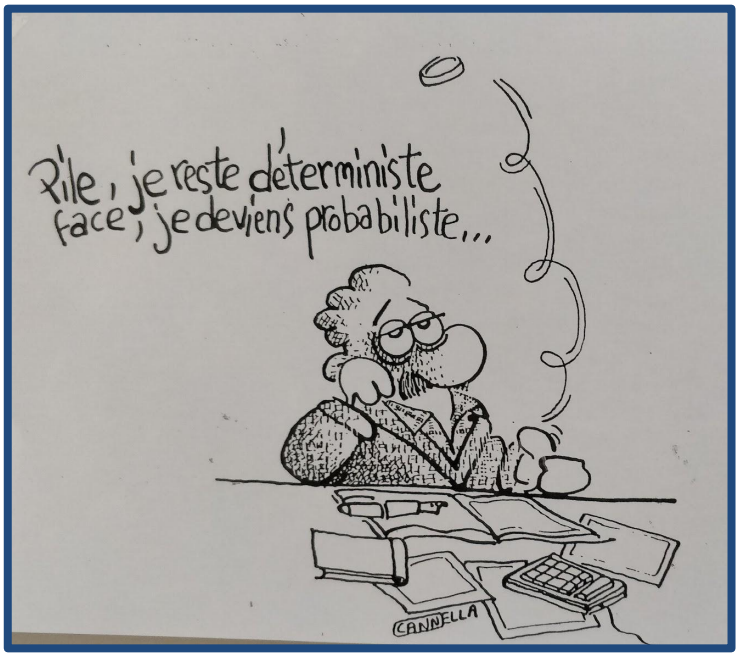
D. Allouche

A. Ouali, D. Allouche, S. de Givry , S. Loudni , F. Eckhardt, Y. Lebbah, L. loukil

UAI 2017 conf paper (sydney)

Outline

- **Graphical model :**
- **Unified Decomposition Guided Variable Neighborhood Search**
  - (UDGVNS : complete method)
- **EXPERIMENTAL RESULTS**
- **CONCLUSION**



Graphical Model

(Shapiro, Haralick, IEEE PAMI 81)

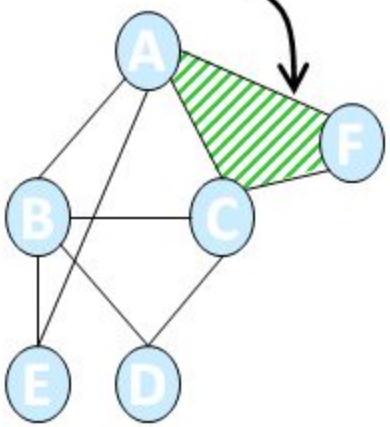
- *n* variables $X = \{X_1, \ldots, X_n\}$
  - finite domains $X_i \in D_i, |D_i| \leq d$
- *e* local/global functions $F = \{f_1, \ldots, f_e\}$
  - scope, function with costs

Energy:

- Costs $\in \{0, \ldots, +\infty\}$ finite or infinite integer

$F_i := P(F \mid A, C)$

$F_i := F = A + C$



Minimize $\sum_F f_i(X)$

NP-hard

- **Influence Diagram**
- **Markov Random Field**
- **Bayesian Network**

Markov Random Field

$$\mathbb{P}(x) = \frac{1}{Z} \prod_{f_S \in \mathcal{F}} f_S(x_S)$$

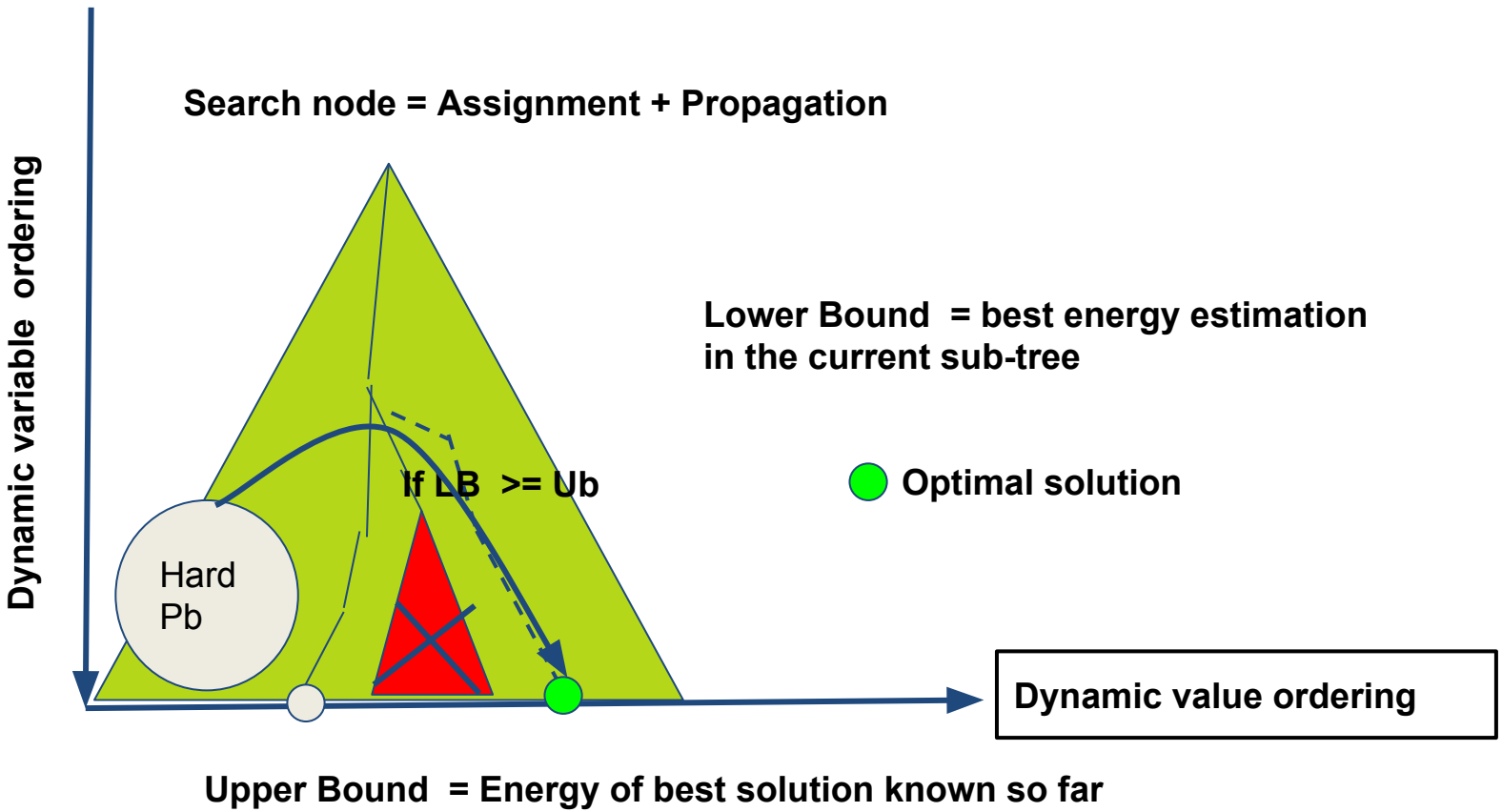
$$Z = \sum_{x \in \Delta} \prod_{f_S \in \mathcal{F}}$$

optimization → Markof Random Field → (**MAP = Maximum a posteriori**)

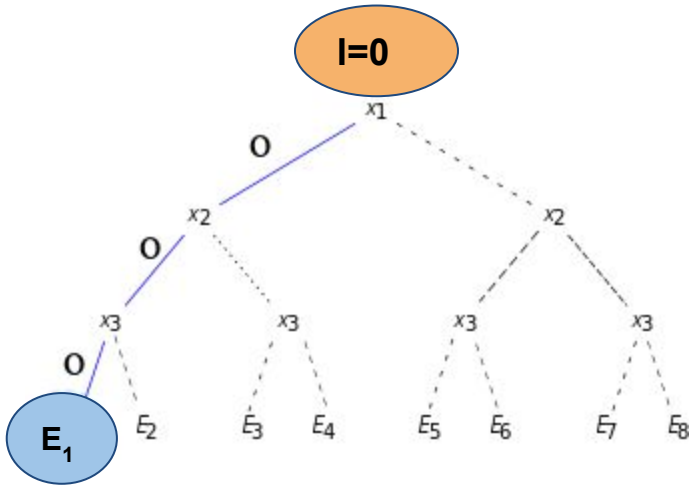
Cost Function Network equivalent after a **− log transform.**

for Bayesian Network → (MPE = Most Probable Explanation)

DFBB (*Depth-First Branch & Bound*)



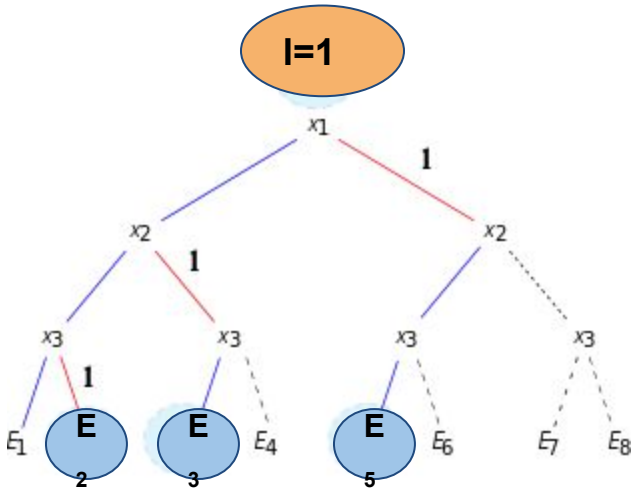
Limited Discrepancy Search *(Ginsberg 95)*



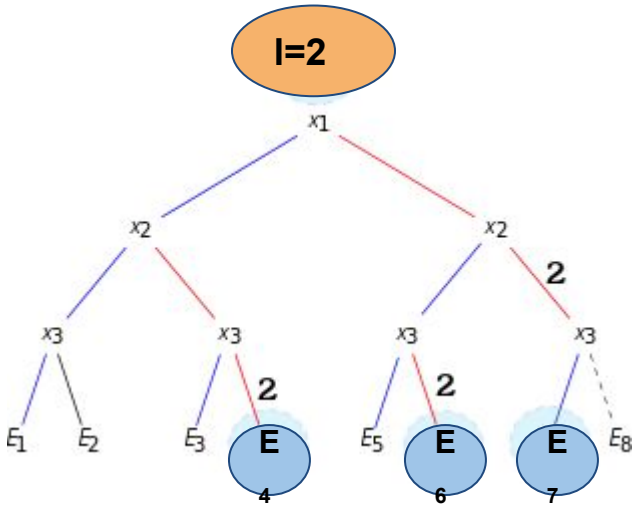
- **Small example with 3 variables and 2 values per domain**

Limited Discrepancy Search

- Small example with 3 variables and 2 values per domain



Limited Discrepancy Search *(Ginsberg 95)*

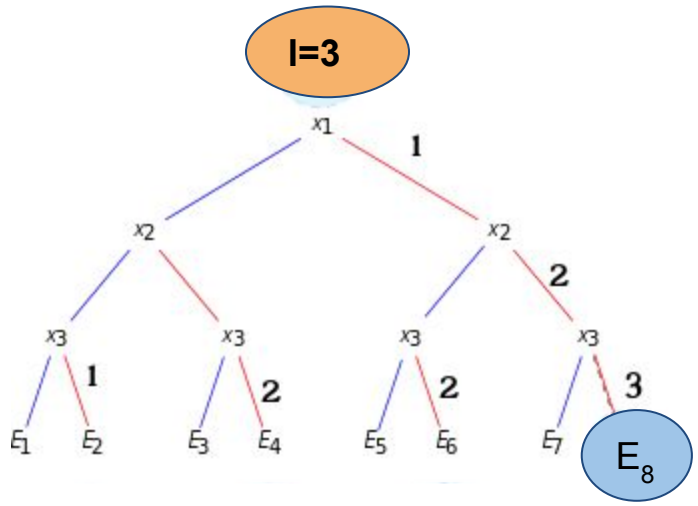


Limited Discrepancy Search *(Ginsberg 95)*

l=3 ⇒ optimality proof

$l_{max} = |\mathcal{X}| * (D_{max}-1)$: in this case $l_{max}=3*(2-1)=3$

Full exploration

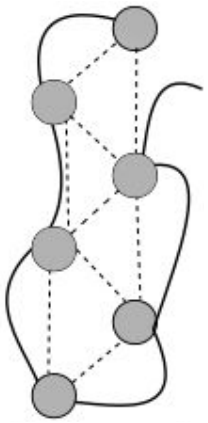


in the worst case : **LDS Complexity: $(k.l)^{l+1}$**

Variable Neighborhood Search *(Hansen 97)*

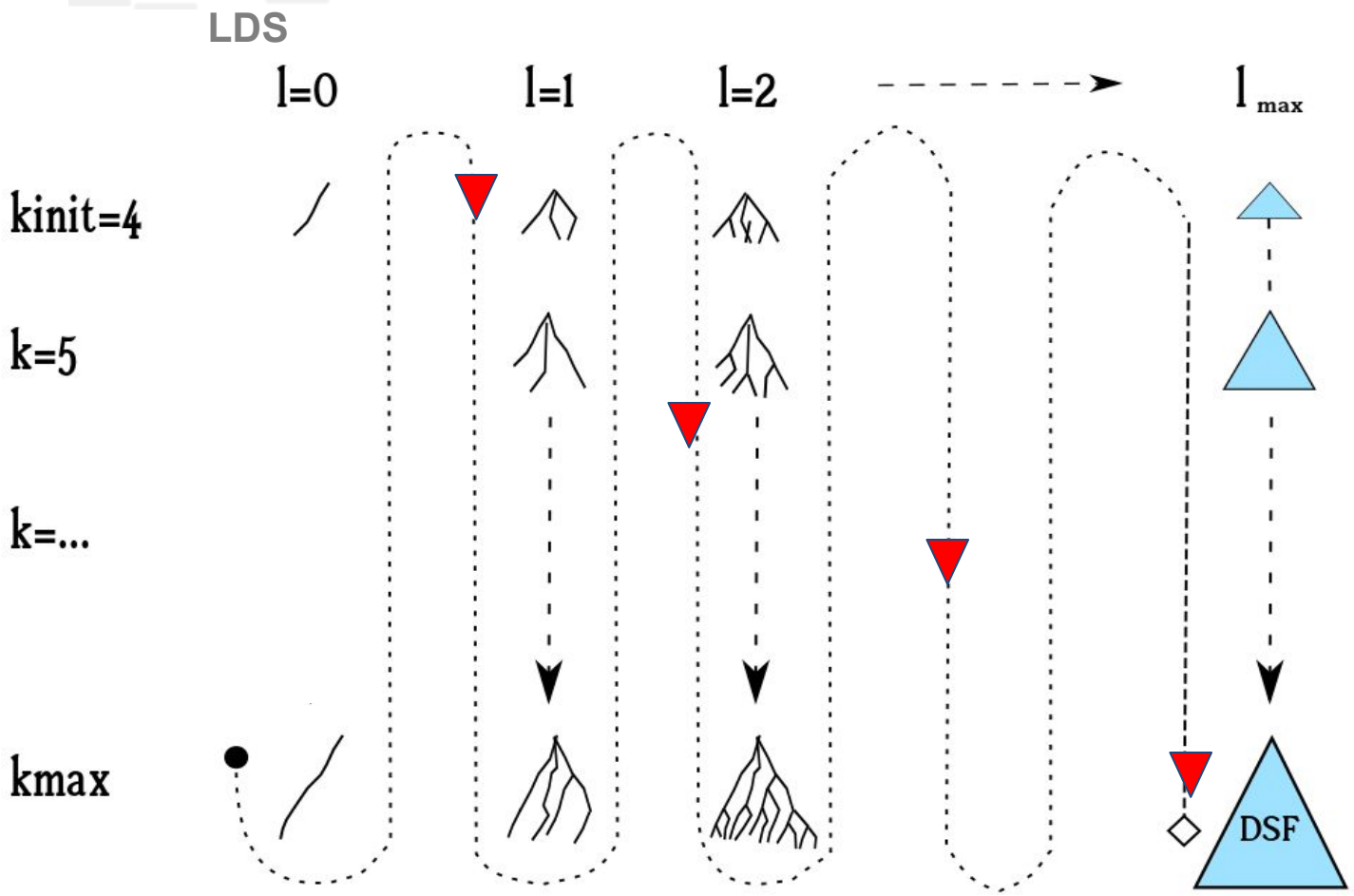
Neighborhood = Combinatorial sub space associted with Selected

New solution ***S'***
with energy ***E'***

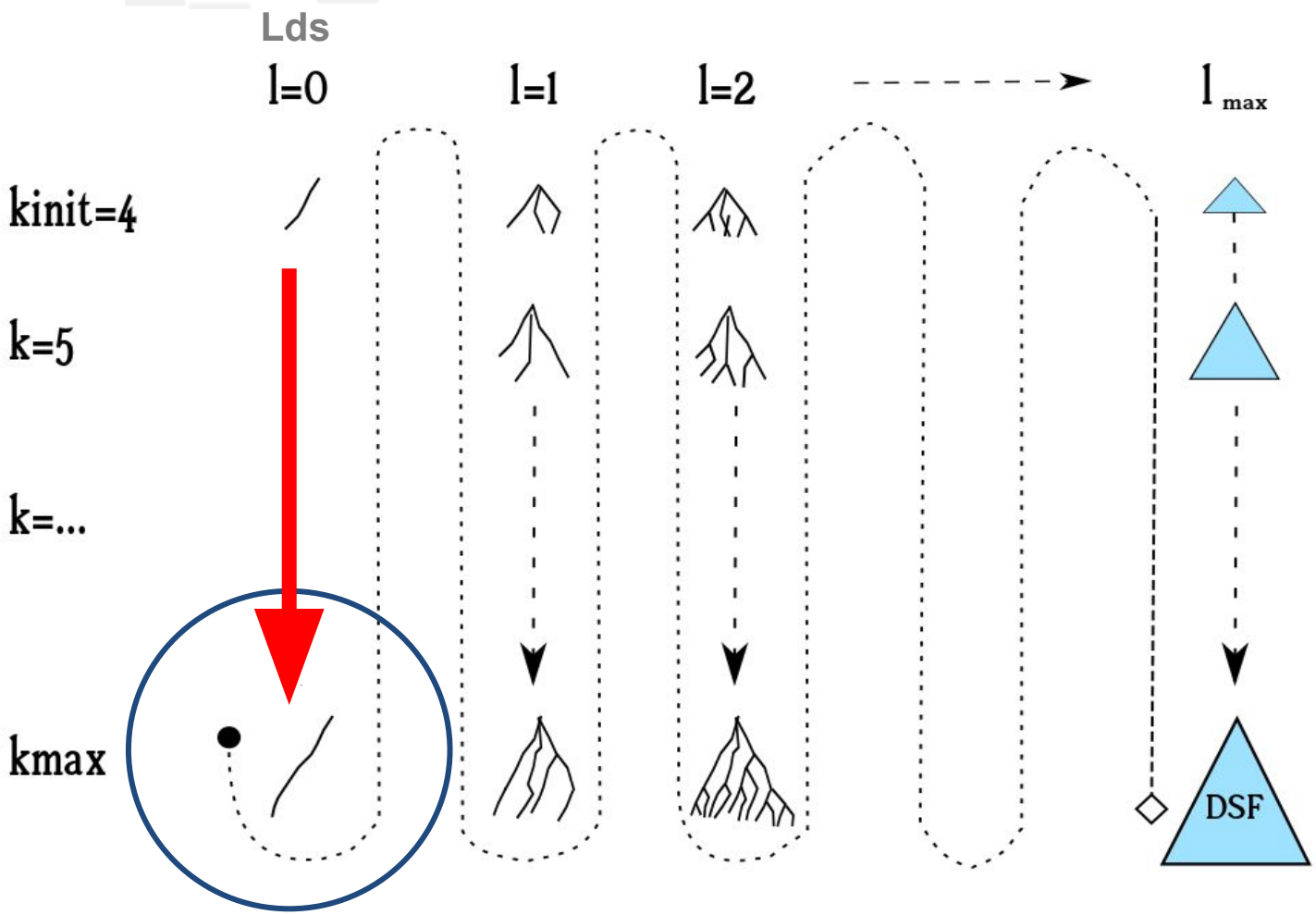


LDS SEARCH
with given discrepancy

UDGVNS : Exploration of both k and l dimensions

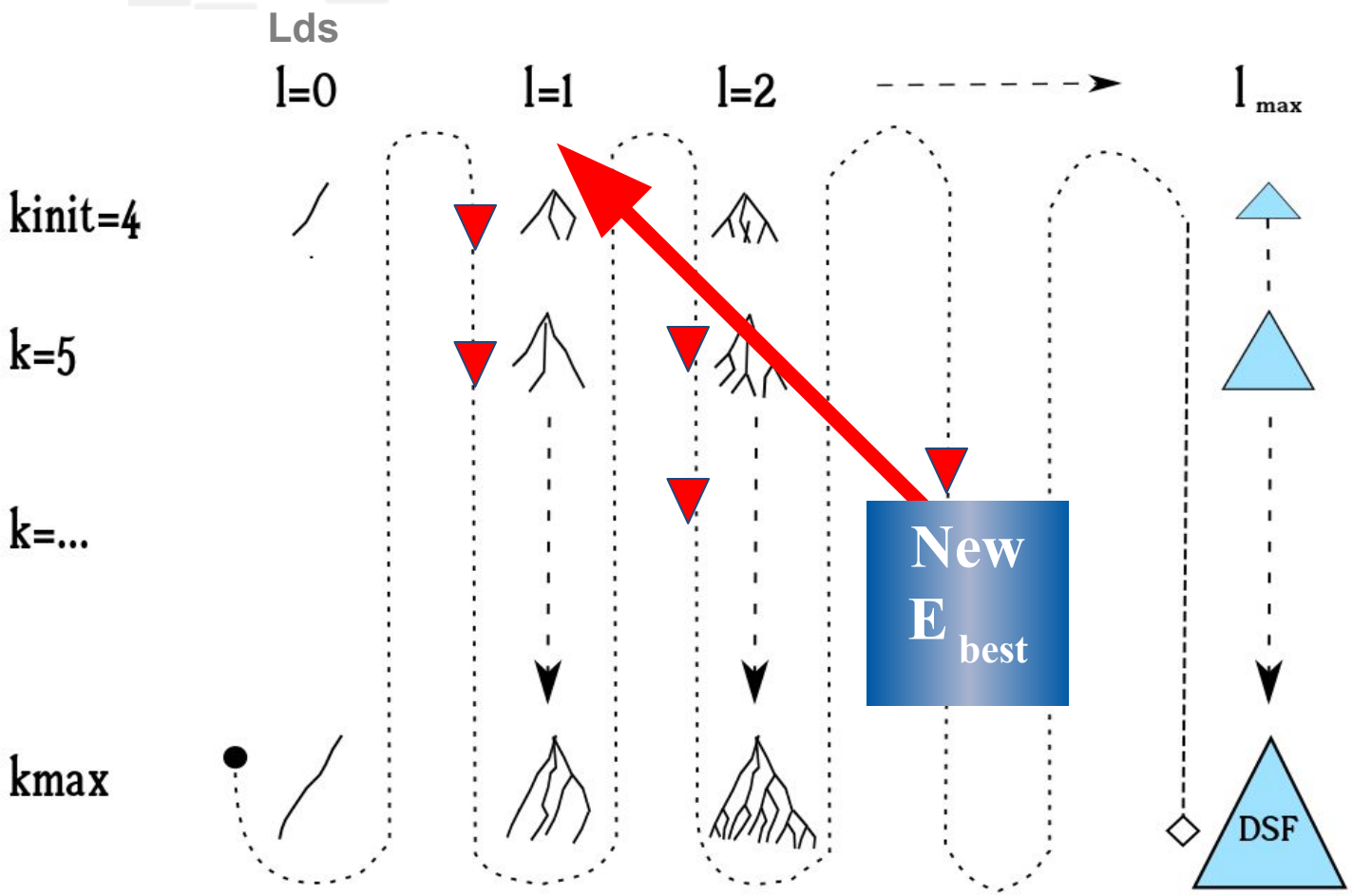


Step 0 : Initial solution

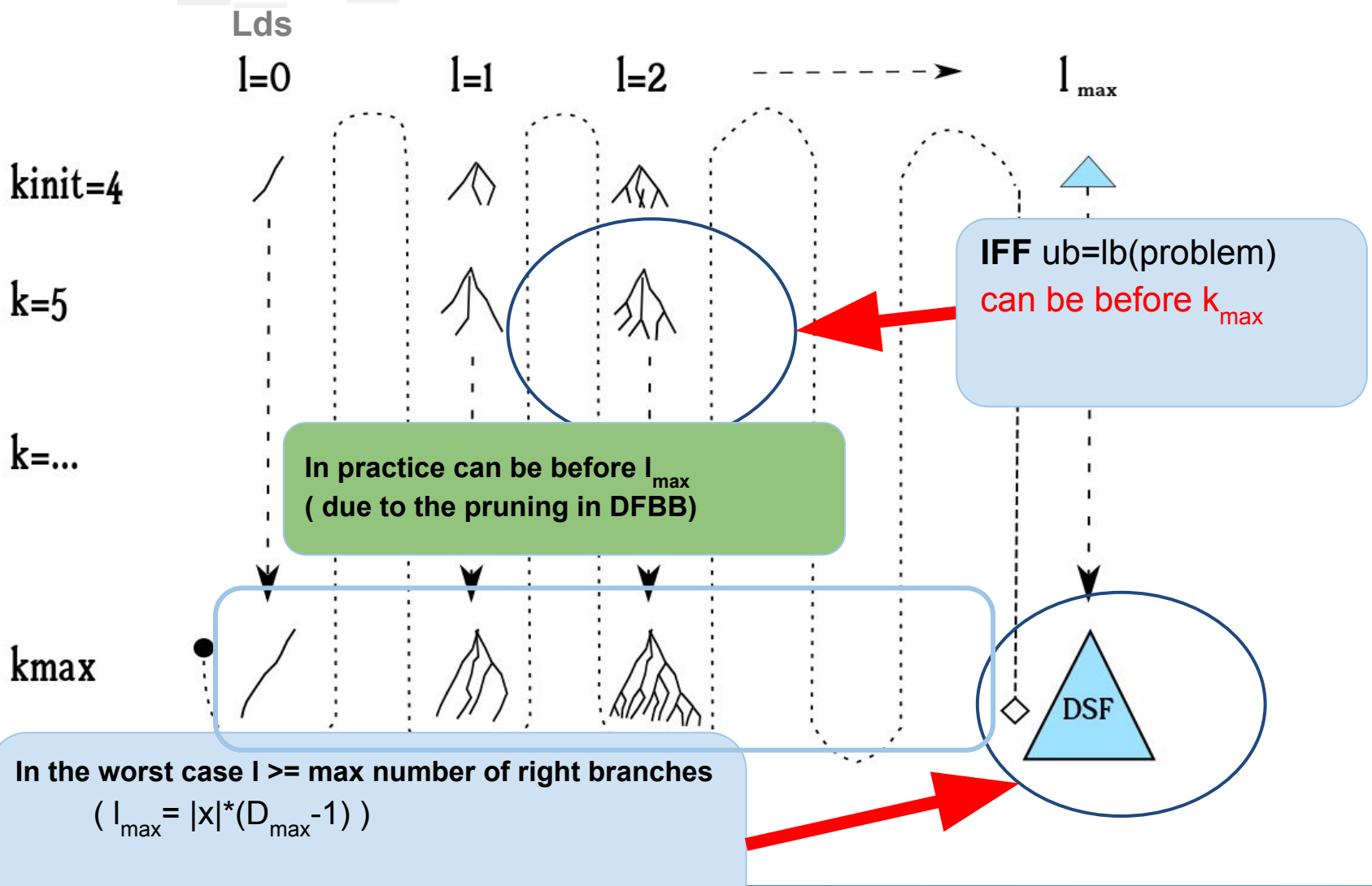


Greedy assignment

NEW SOLUTION WITH BETTER E → RESTART



Completeness restoration



In the worst case l >= max number of right branches

(l_{max}= |x|*(D_{max}-1))

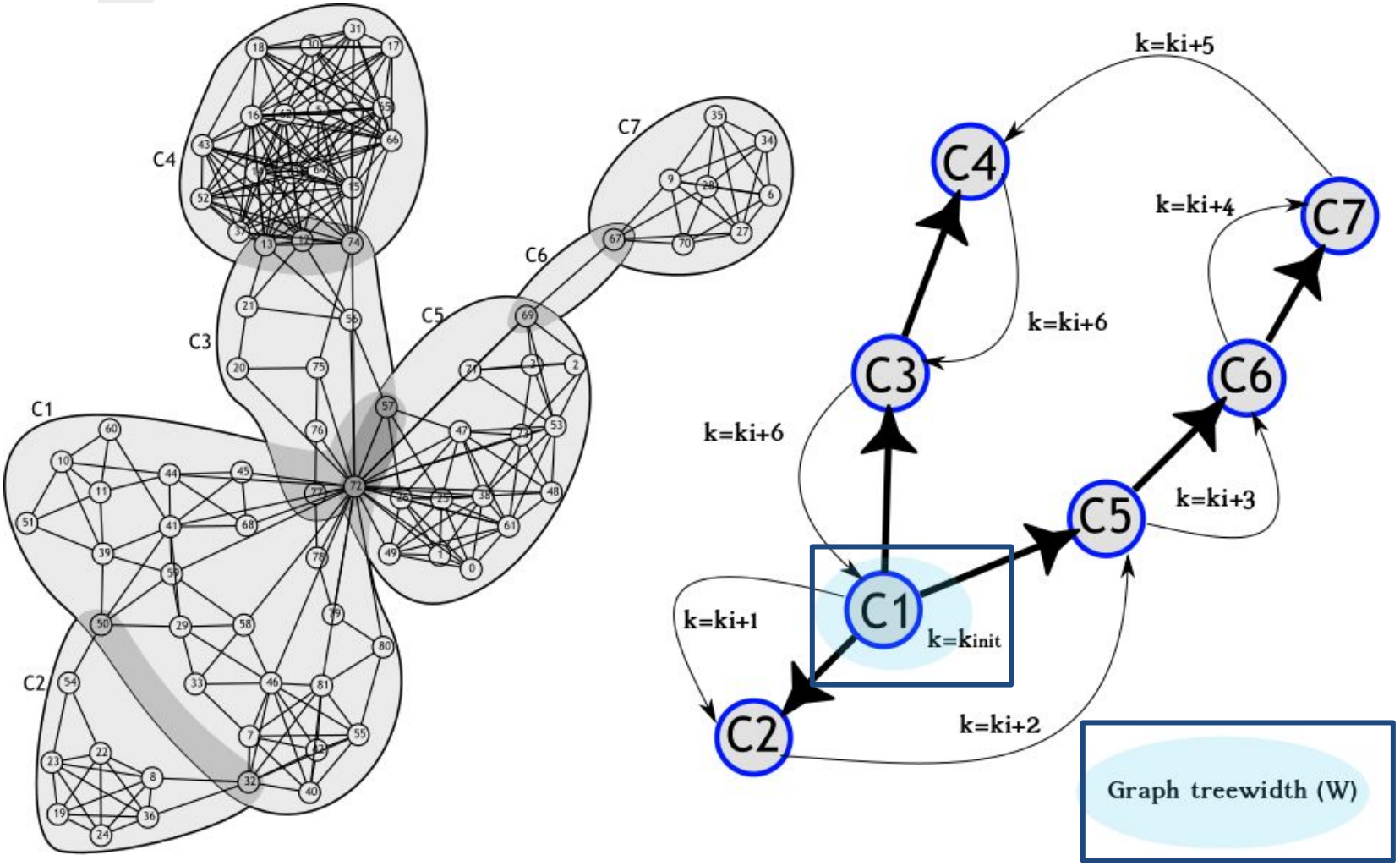
Iff **k** = k_{max} = problem size

Resolution Strategy :

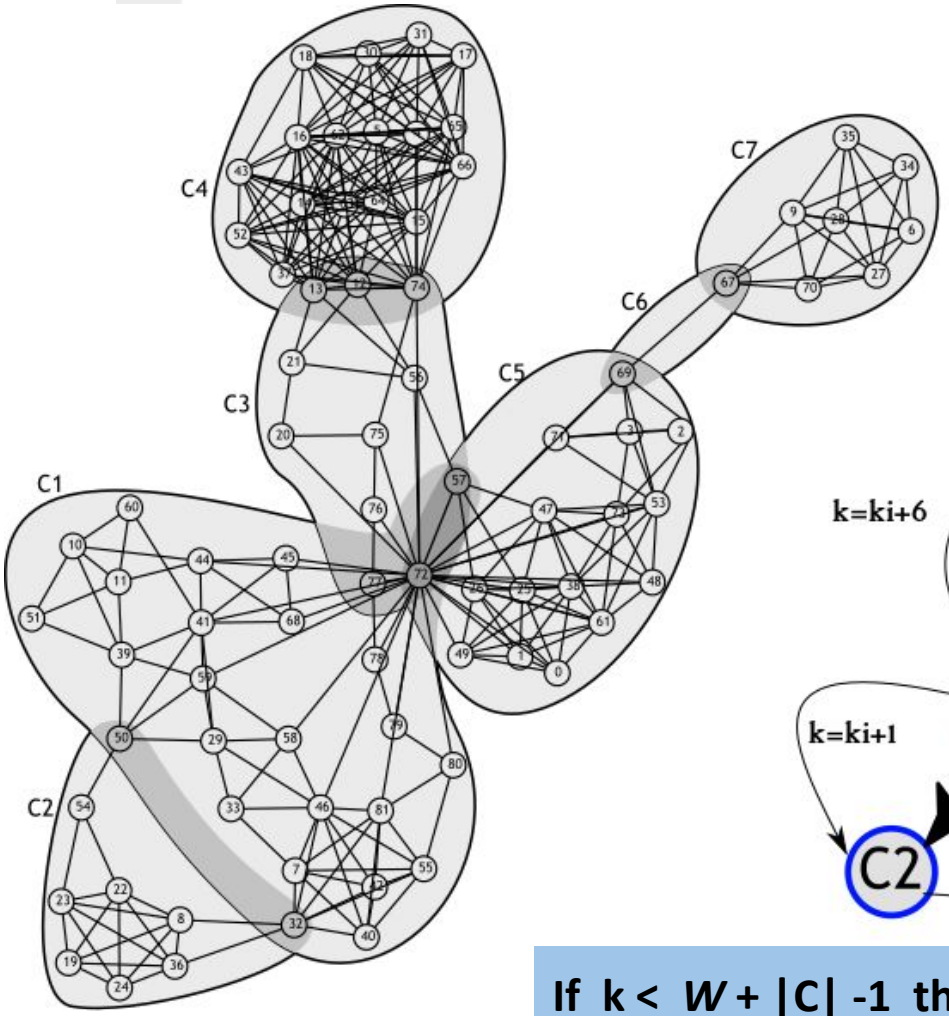
conditioned by (l, k) Operator:

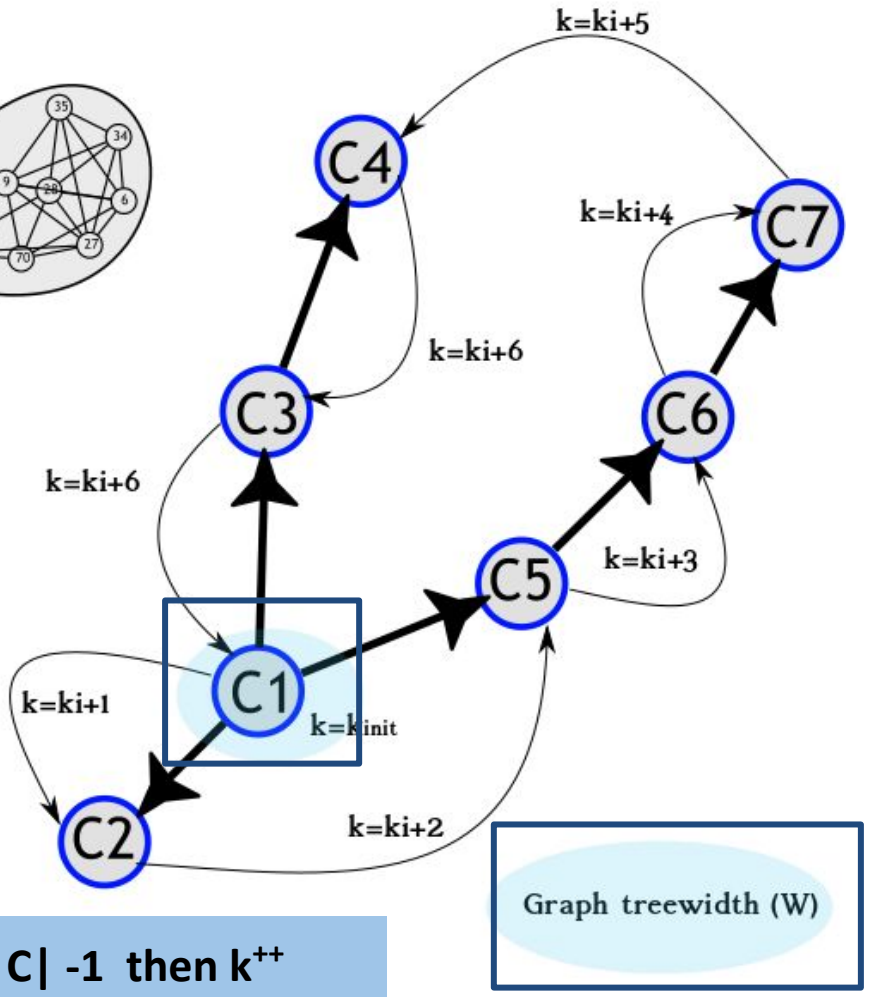
- **Linear** : k^{++}, l^{++}
- **Mult** * 2 : l*2, k*2
- **Luby** serie: (*Luby & al 1993)*
  - S=(1, 1,2, 1,1,2,4, 1,1,2,1,1,2,4,8, ….)
- **k JUMP** (Goal : solve completely the biggest cluster)

Cluster visit in a topological order :



k jump heuristic





If $k < W + |C| - 1$ then k^{++}
else
$k = |X| = k_{max}$

EXPERIMENTAL PROTOCOL

1 hour CPU time limit

Benchmark description:

Optimality proof: experiment on 1669 instances

- Probabilistic Inference Challenge (PIC 2011), UAI competitions
- CVPR instances (Computer Vision and Pattern Recognition)
- Cost Function Library

Anytime curves : Evolution of best energy over time

- subset of 114 difficult instances
  (unsolved after 1h search runtime with hbfs and BTD)

Solvers:

IBM ILOG CPLEX release 12.7.0.0 (MIP solver)

- parameter : Default
- precision threshold: EPAGAP , EPGAP, EPINT
  - (set to zero in order to enforce complete search)

DAOOPT: (Dechter & AL)

- GLS+ (Guided local search)
- AND/OR search based tree decomposition

INCOP + TOULBAR2 0.98 (previous release)

- IDWalk (local search in preprocessing)
- Dead End Elimination
- Hybrid Best First Search

Lib DAI rel 0.3.2 (UAI 2010 settings) :

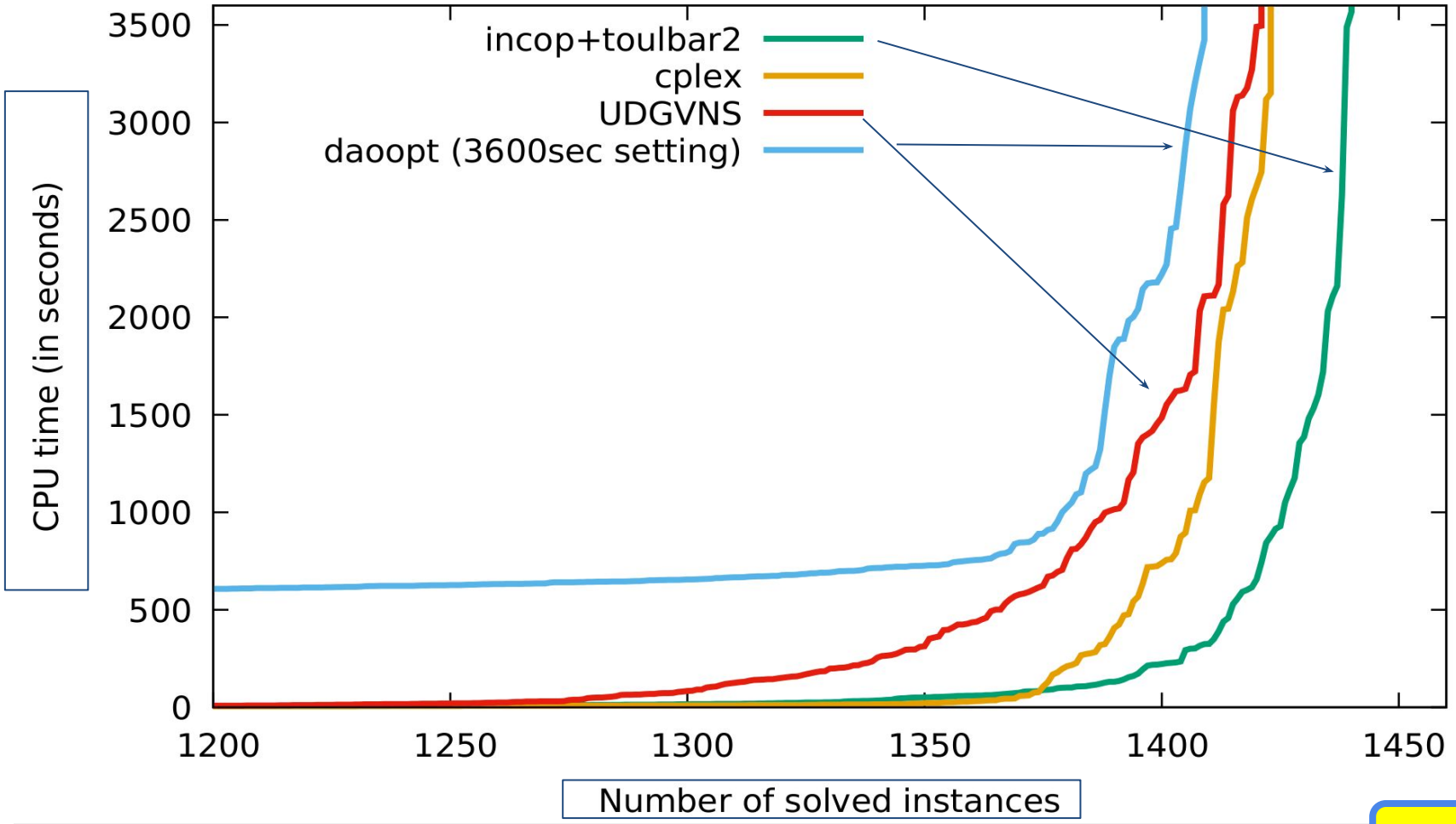
- Message passing algorithm with decimation

Toulbar2 1.0 : UDGVNS

- **Parameters:** $k_{min} = 4$; $k_{max} = |X|$; $lds_{max} = |X| * (|D_{max}|-1)$; D_{max} = Max Domain size
- **With best Operators:**
  - $k \rightarrow k{+}{+}$ / jump
  - LDS $\rightarrow$ mult*2

Cactus plot

Optimality proof efficiency

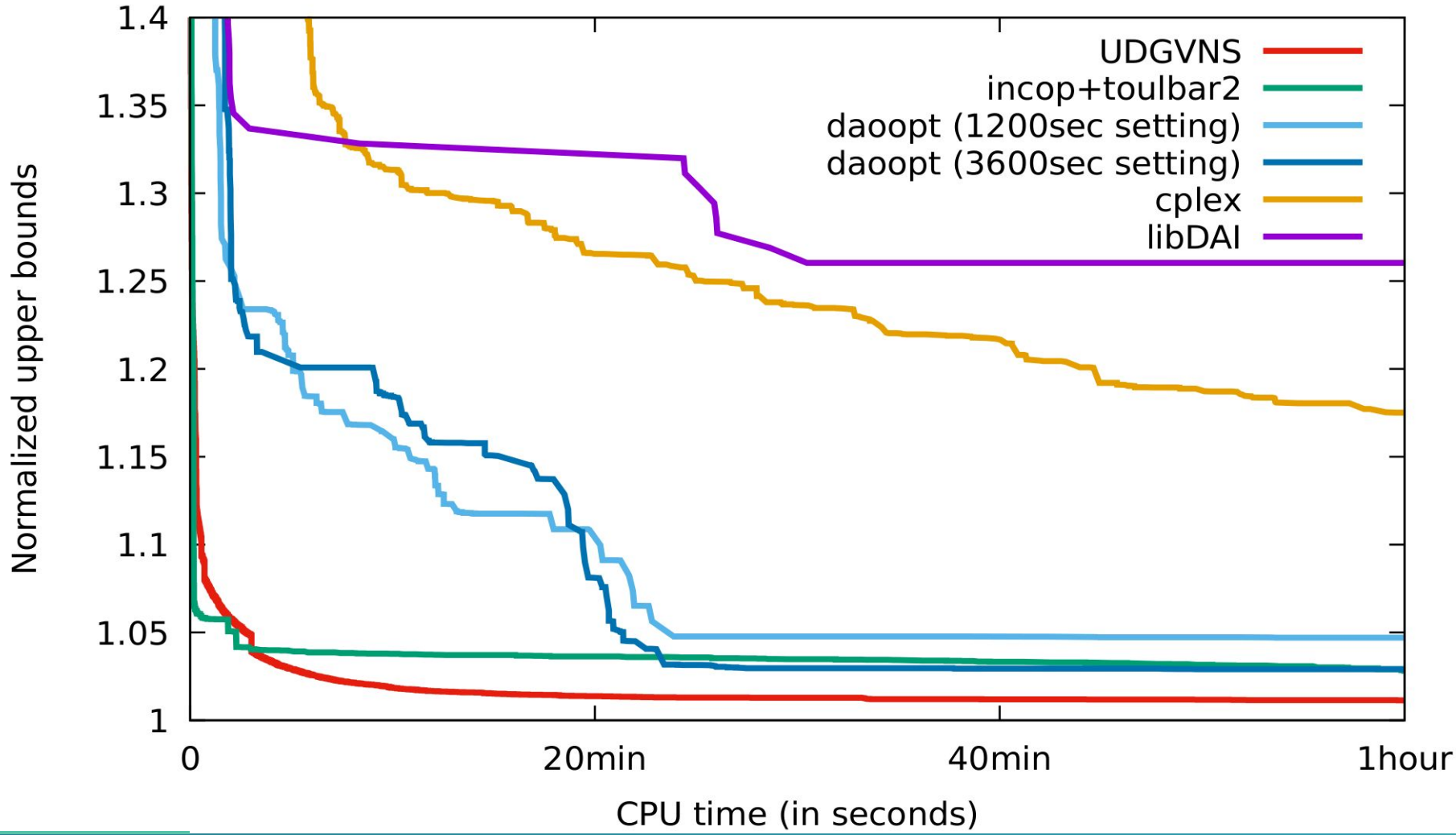


UDGVNS with k++ jump and lds mult*2

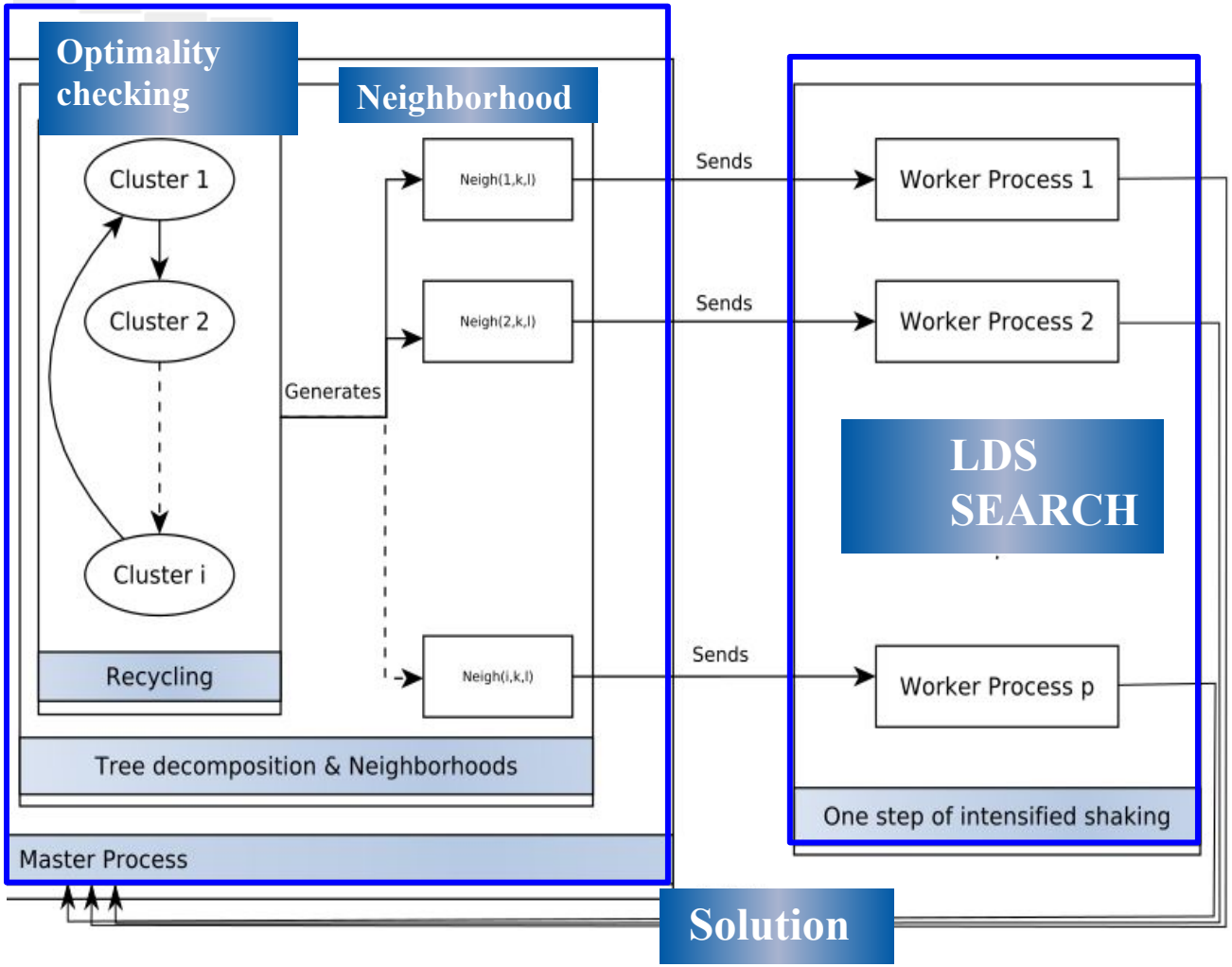
1669 instances

Anytime behavior

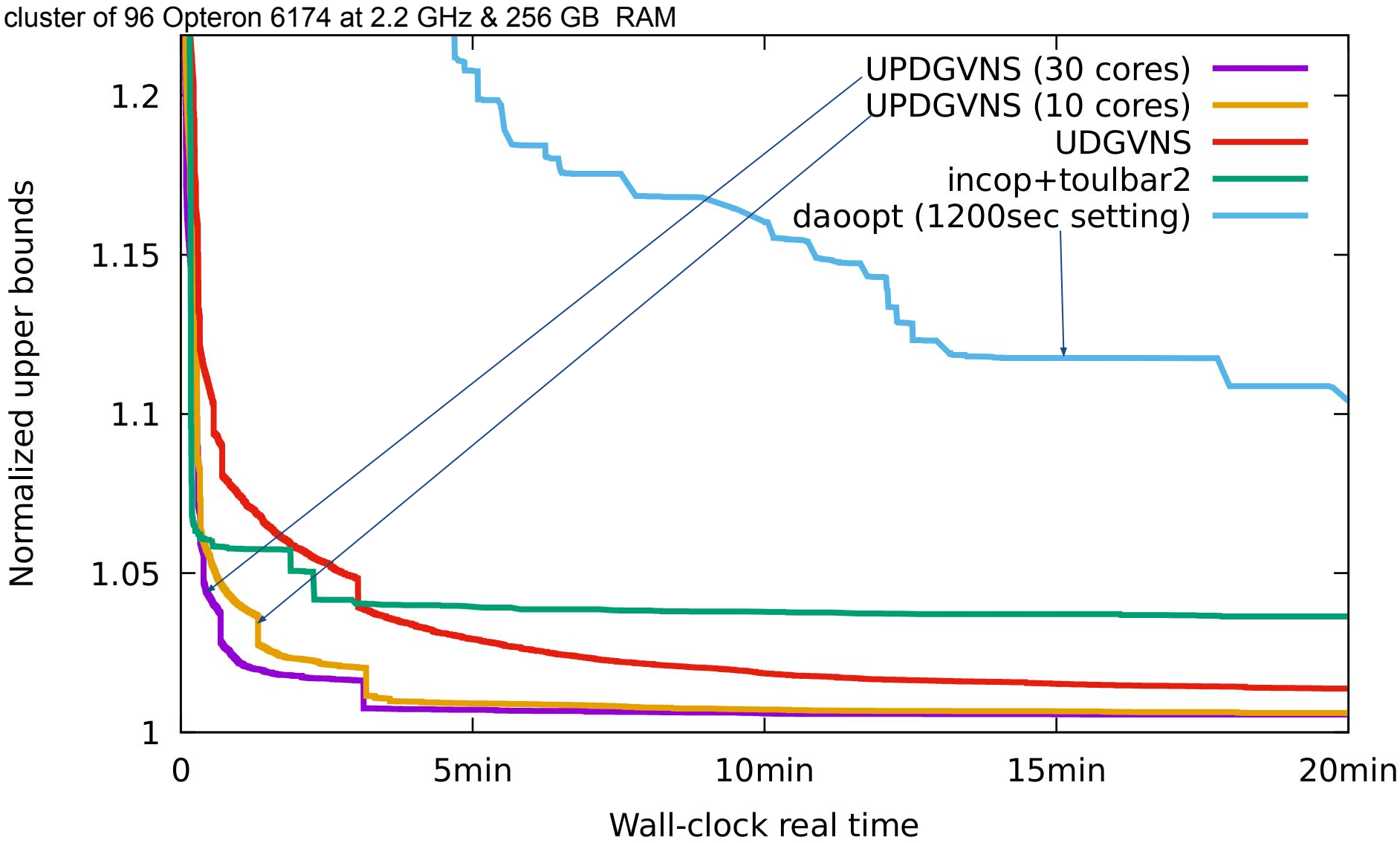
UDGVNS with k++ jump & l ← mult*2



Parallelization



Anytime Zoom with Parallele release



Conclusion

- UDGVNS **restores completeness**
  (with **various** LDS and neighborhood **evolution strategies**)

- UDGVNS the best **balance between anytime behaviour and optimality proof**
  (empirical results with **k jump and lds = mult*2**)

- **Parallel version improves its anytime behavior** (better search space covering due to multiple selection of neighborhood with same k value)

Acknowledgement

- **Ouali[2,3],**
- **S. de Givry[1], F. Eckhardt[1]**
- **S. Loudni[3],**
- **Y. Lebbah, L. Loukil[2]**

[1] MIAT UR 875, INRA, F-31320 Castanet Tolosan, France
[2] Lab. LITIO University of Oran 31000 Oran, Algeria
[3] CNRS, UMR 6072 GREYC, University of Caen Normandy, France

Computational support:

- HPC HAYTHAM of university of Oran 1
- Scientific and Technical Information center IBNBADIS
- Genotoul bioinformatic platform (Toulouse / occitanie)

Practical interest of graphical model

Image processing : (example from openGM2)

Segmentation

Form recognition

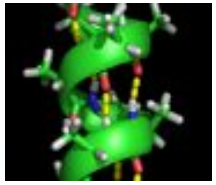
In-Painting

Chinese Characters recognition

Color Segmentation

.etc..

Protein modeling :



- **Computational Protein Design**
- **force fields tuning**

hamiltonian → objectif function <=> Energie → Probability = $e^{-\beta.E}$

merci

References

- [Larrosa et al., 2005] Existential arc consistency: getting closer to full arc consistency in weighted CSPs. IJCAI
- [Larrosa et al., 2016] Limited discrepancy AND/OR search and its application to optimization tasks in graphical models. IJCAI
- [Harvey and Ginsberg, 1995] W Harvey and M Ginsberg.Limited discrepancy search. IJCAI
- [Ouali et al., 2015] Replicated Parallel Strategies for Decomposition Guided VNS. ENDM.
- [Fontaine et al., 2013] Exploiting tree decomposition for guiding neighborhoods exploration for VNS. RAIRO OR
- [Davidovic and Crainic, 2012] MPI parallelization of variable neighborhood search. Electron. Notes in Discrete. Math.
- [Mladenović and Hansen, 1997] N Mladenović and P Hansen.Variable Neighborhood Search. Comput. Oper. Res.
- [Allouche et al., 2014] Computational Protein Design as an Optimization Problem. Artif. Intell.
- [Hurley et al., 2016] MultiLanguage Evaluation of Exact Solvers in Graphical Model Discrete Optimization. Constraints

INS multi stat

Variable Neighborhood Search

- 1997 : Variable Neighborhood Search (Mladevoc& Hansen 1997) (local search)
- 2003 : VNS /LDS + CP (Loudni& Boizunault, 2003) ⇒ (local search based on constraint programming)
- 2013 : DGVNS (fontaine & al 2013 RAIRO OR) ⇒ Decomposable Guided VNS tree decomposition (local search)

UDGVNS : Unified Decomposable Guided VNS (complet method)

Perspectives

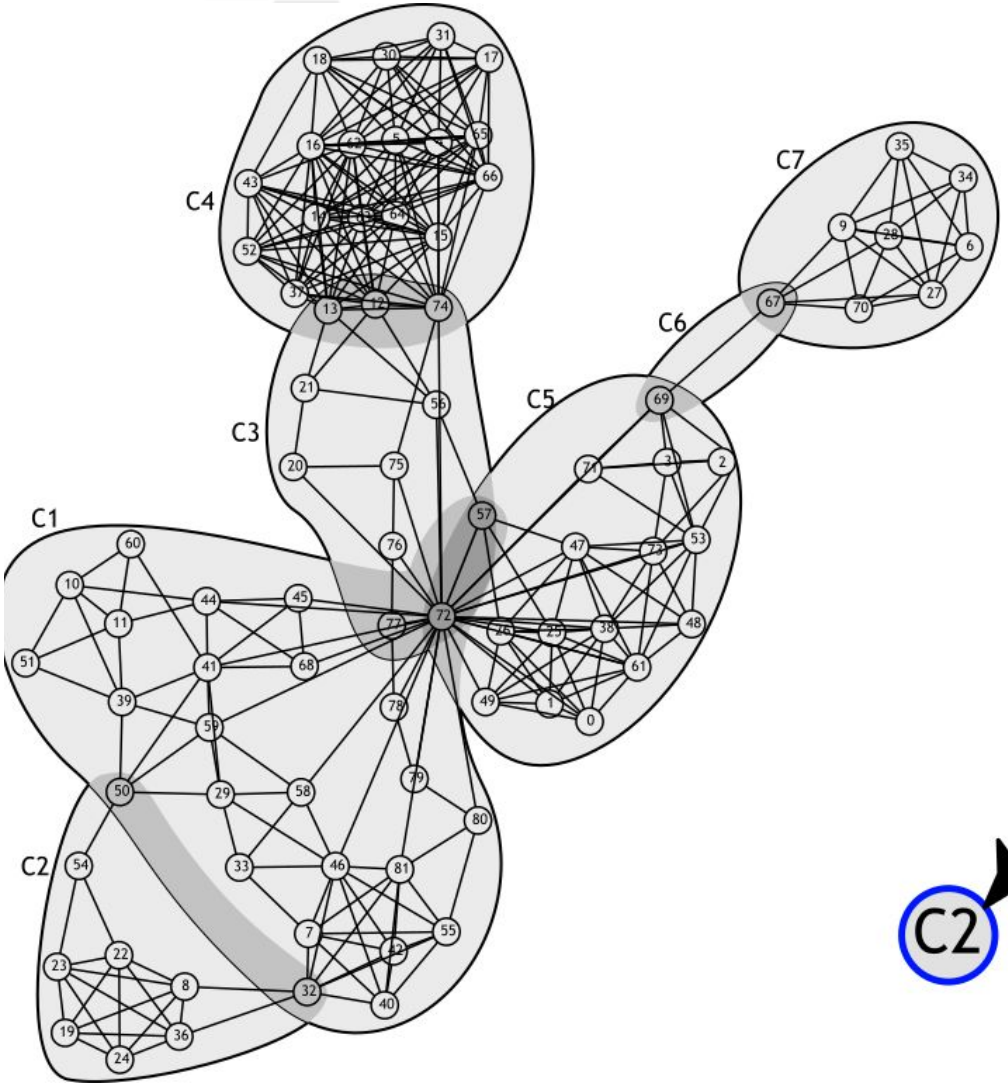
- Enhancing proof optimality in UDGVNS by exploiting story search in LDS of previous iterations.

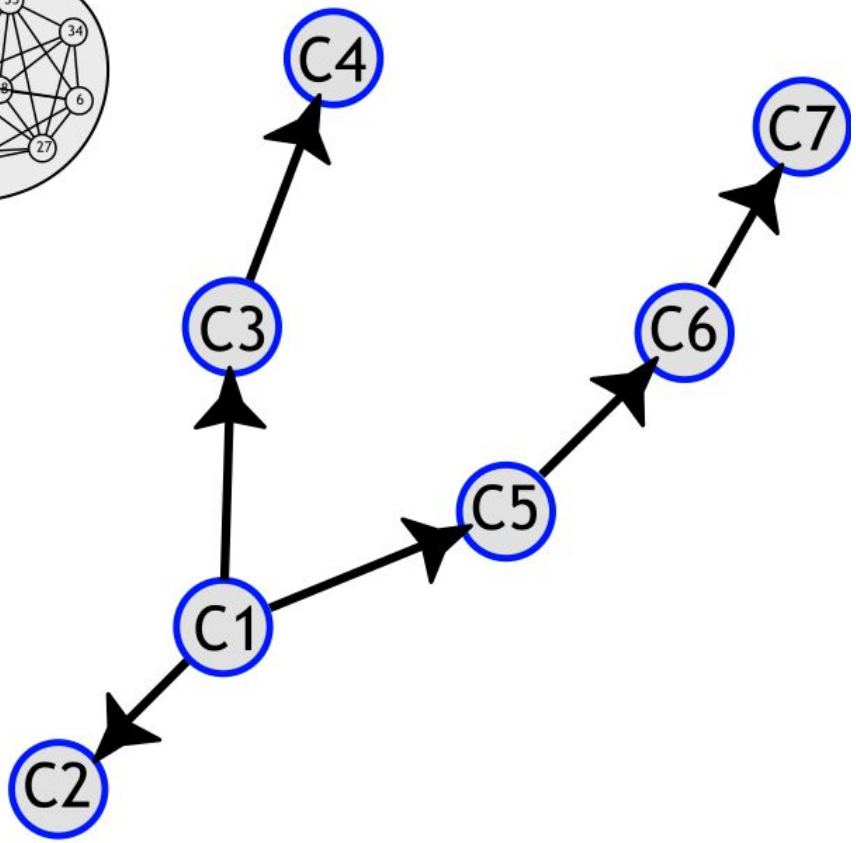
- Automatic tuning of the best parameter settings :
  - per instance / family of instances
    - based on dynamic metric during search
    - based on prior knowledge information
    - Tree decomposition impact

- Benchmarking on proteins design problem

(and others putatives applications ;)

Tree decomposition





Evaluation criteria

Anytime measure → convergence speed

Normalized energie:

$$E^*(t) = (E(t_0) - E(t_i)) / E_{best}$$

$$\bar{E}(t) = 1/N \cdot \Sigma_1^N E^*(t)$$

$E^*(t)$ = normalized energy
$E(t_0)$= initial upper bound energy
E_{best} =Energy of Best known solution

Cactus plot → optimality

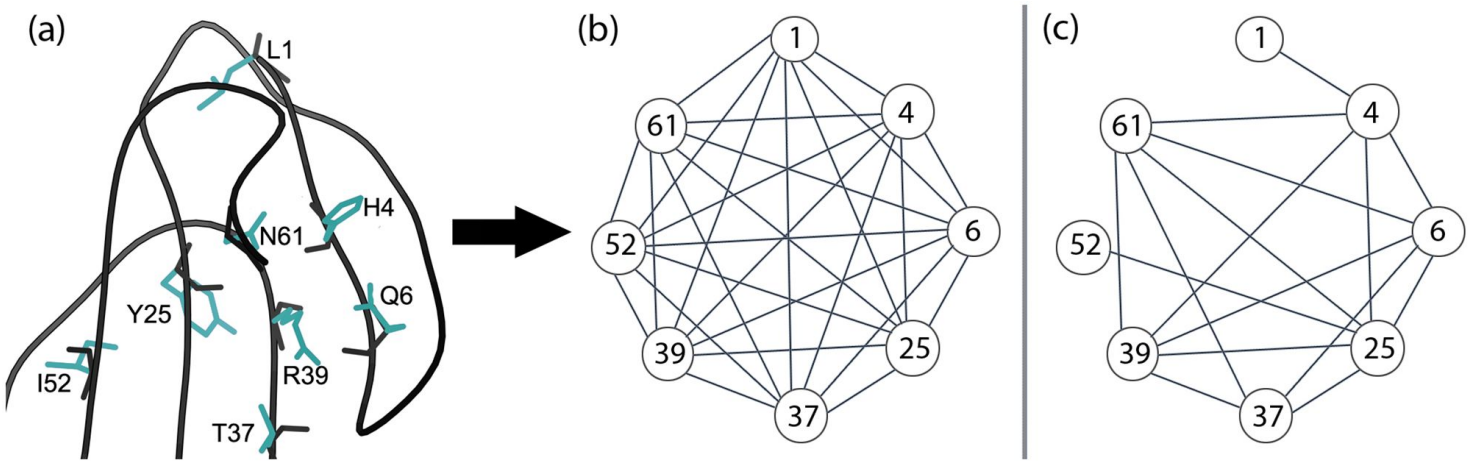
$$\Sigma_0^t \#solved = f(t)$$

Link with ranked probability skill score (RPSS) ??

Algorithme 1 : Unified DGVNS algorithm.

```
Function UDGVNS (ℓ_min, ℓ_max, +_ℓ, k_min, k_max, +_k, ub : In/Out, x : In/Out) : boolean
    let (C_T, T) be a tree decomposition of (𝒳, 𝒟, ℱ) ;
    opt ← true ;
1   LDS^r (∞, 𝒟, ub, x, opt) ;                          // initial solution
2   if (ub = lb(𝒟)) then opt ← true;
    c ← 1 ;                                             // current cluster index
    r ← 0 ;                                             // number of iterations
    ℓ ← ℓ_min ;                                         // initial discrepancy limit
    while (¬opt ∧ ℓ ≤ ℓ_max) do
        i ← 0 ;                                         // number of failed neighborhoods
3       k ← k_min ;                                     // initial neighborhood size
        while (¬opt ∧ k ≤ k_max) do
            A ← getNeighborhood(x, C_c, k) ;
            ub' ← ub, opt ← true ;
4           LDS^r (ℓ, A, ub', x', opt) ;                // neighborhood search
5           if (ub' = lb(𝒟)) then opt ← true;
6           else if (A ≠ 𝒟) then opt ← false;
7           if (ub' < ub) then
                x ← x',  ub ← ub' ;                     // new best solution
8               i ← 0,  k ← k_min ;
9               r ← 0,  ℓ ← ℓ_min ;
            else
                i ← i + 1 ;
                if (k < k_max) then
                    k ← min(k_max, k_min +_k i) ;
                else k ← ∞;
10          c ← 1 + c mod |C_T| ;                       // get next cluster
        r ← r + 1 ;
        if (ℓ < ℓ_max) then
            ℓ ← min(ℓ_max, ℓ_min +_ℓ r) ;
        else ℓ ← ∞;
11  return opt ;
```

Computational Protein Design



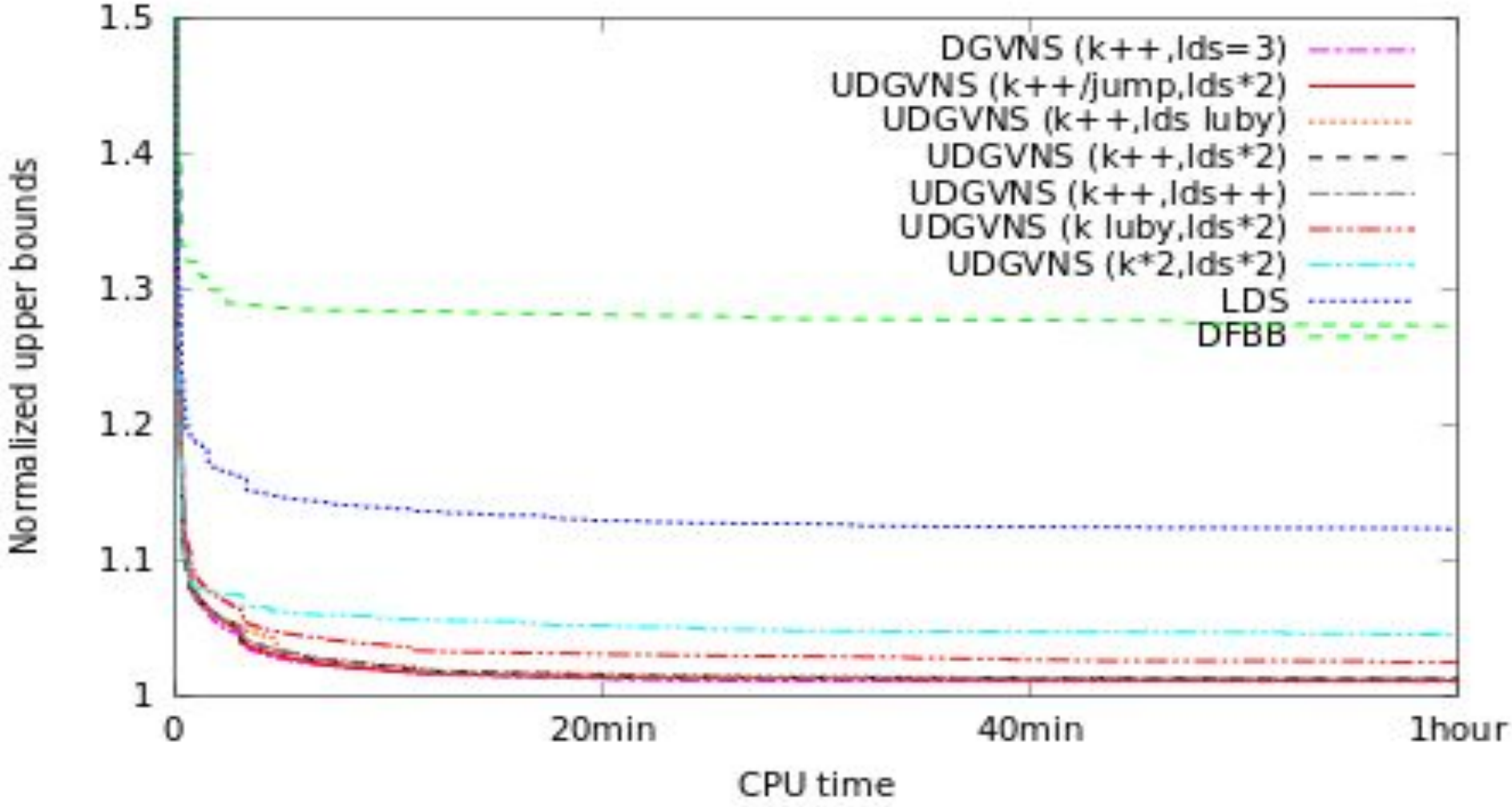
Instances contain from 130 up to 282 variables with maximum domain size from 383 to 438, and between 1706 and 6208 cost/energy functions. The tree width ranges from 21 to 68 (i.e. from 0.16 to 0.34 for a normalized tree width). Instances selected on the basis of 3D proteins critters (i.e. normalized Gyration radius)

CPD results

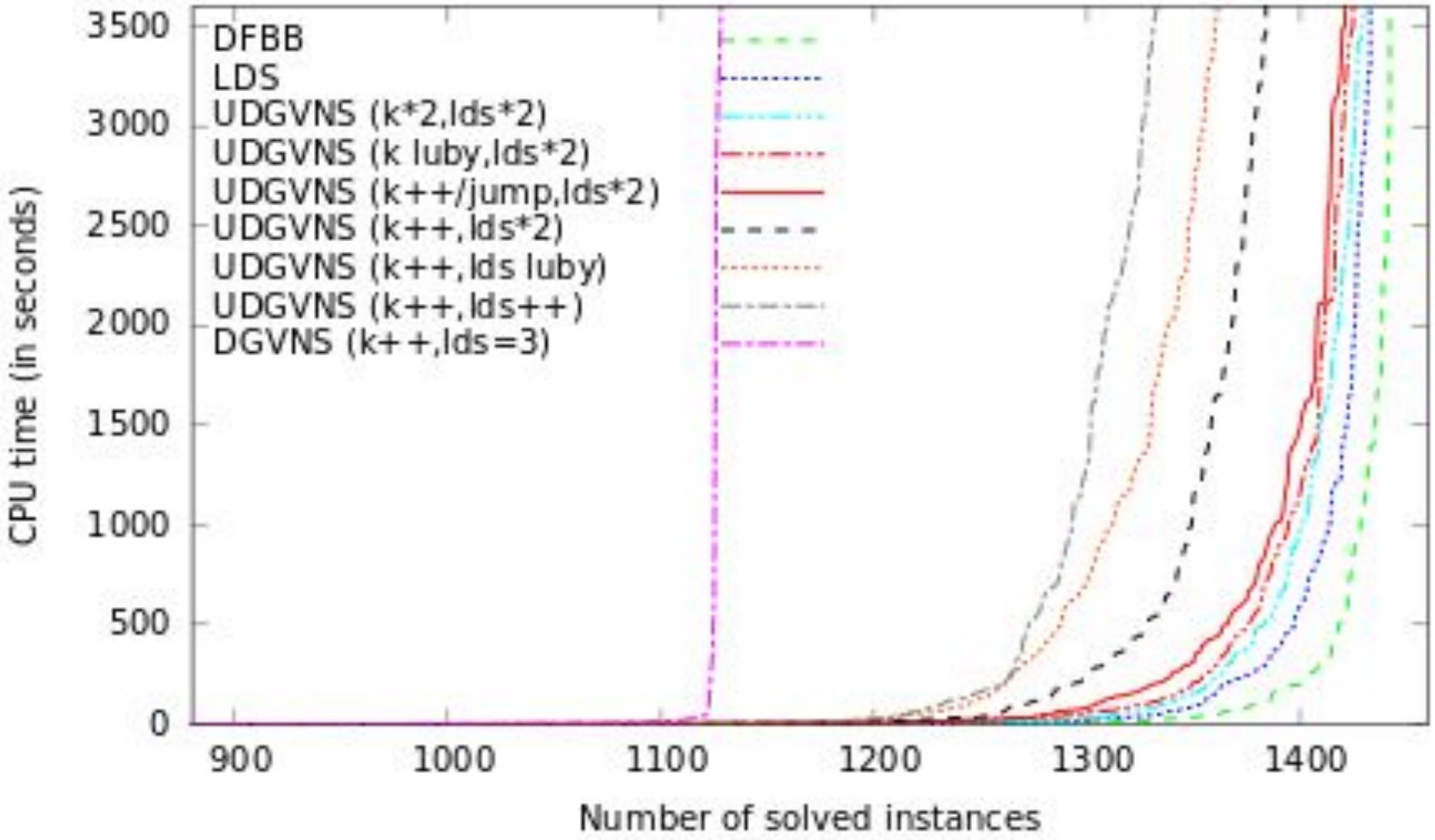
Instance	ncl	(1)		(2)		(3)		(4)	(5)		Speed-up		
		Succ.	Time (s)	Succ.	Time (s)	Succ.	Time (s)	ΔE	Time (s)	ΔE	(1/2)	(1/3)	(2/3)
5dbl	87	10/10	2761± 67	10/10	963± 53	10/10	149± 18	0.27	783	0	2.86	18.53	6.46
5jdd	168	0/10	TO	10/10	3248 ± 162	10/10	646± 72	4.08	20662	0.04	-	-	5.02
3r8q	157	0/10	TO	10/10	3478± 82	10/10	397± 47	4.19	12762	0	-	-	8.76
4bxp	108	10/10	1,213± 32	10/10	1352± 40	10/10	216± 24	0.26	2966	0	0.89	5.61	6.26
1f00	177	0/10	TO	10/10	2575± 29	10/10	542± 30	4.38	9749	0	-		4.75
2x8x	131	10/10	3312± 111	4/10	2801± 1,071	10/10	521± 58	4.16	69213	3.61	1.18	6.35	5.37
1xaw	66	0/10	TO	10/10	581± 38	10/10	260± 57	2.73	2804	0	-	-	2.23
5e10	74	10/10	1667± 86	10/10	1412± 21	10/10	132± 24	0.26	1171	0	1.18	12.62	10.7
1dvo	82	10/10	940± 22	10/10	940± 22	10/10	197± 25	2.90	34142	0.18	1	4.77	4.77
1ytq	67	10/10	2235± 211	10/10	1304± 40	10/10	280± 26	1.67	17063	0.31	1.71	7.98	4.65
2af5	140	0/10	TO	10/10	2659± 75	10/10	894± 247	4.37	86029	0.60	-	-	2.97
1ng2	86	10/10	1065± 69	10/10	547± 18	10/10	260± 26	1.14	38730	5.93	1.94	4.09	2.10
3sz7	79	0/10	TO	10/10	2952± 276	10/10	320± 50	3.11	82625	0.54	-	-	9.22
2gee	110	10/10	1,647± 8	10/10	1276± 19	10/10	286± 38	1.68	5021	0	1.29	5.75	4.46
5e0z	73	10/10	621± 15	10/10	995± 18	10/10	105± 8	0.16	999	0	0.62	5.91	9.47
1yz7	87	10/10	2148± 9	10/10	945± 24	10/10	457± 85	2.91	83816	3.20	2.27	4.70	2.06
3lf9	72	10/10	1636± 31	10/10	894± 41.17	10/10	216± 21	2.41	2667	0	1.82	7.57	4.13
3e3v	91	0/10	TO	10/10	2327± 621	10/10	263± 38	2.57	81574	0.15	-	-	8.84
1is1	107	10/10	3178± 51	10/10	2274± 421	10/10	329± 48	3.45	63832	0.42	1.39	9.65	6.91
5eqz	89	10/10	1849± 49	10/10	697± 6	10/10	225± 16	2.20	12768	0	2.65	8.21	3.097
4uos	118	10/10	2304± 988	10/10	2109± 62	10/10	469± 79	5.20	58589	17.86	1.09	4.91	4.5

TO: TimeOut **(1):** VNS/LDS+CP $(k{+}{+}, \ell = 3)$ **(2):** UDGVNS $(k{+}{+}, \ell = 3)$ **(3):** UPDGVNS $(npr, 1, k{+}{+}, \ell = 3)$ **(4):** FIXBB **(5):** TOULBAR2

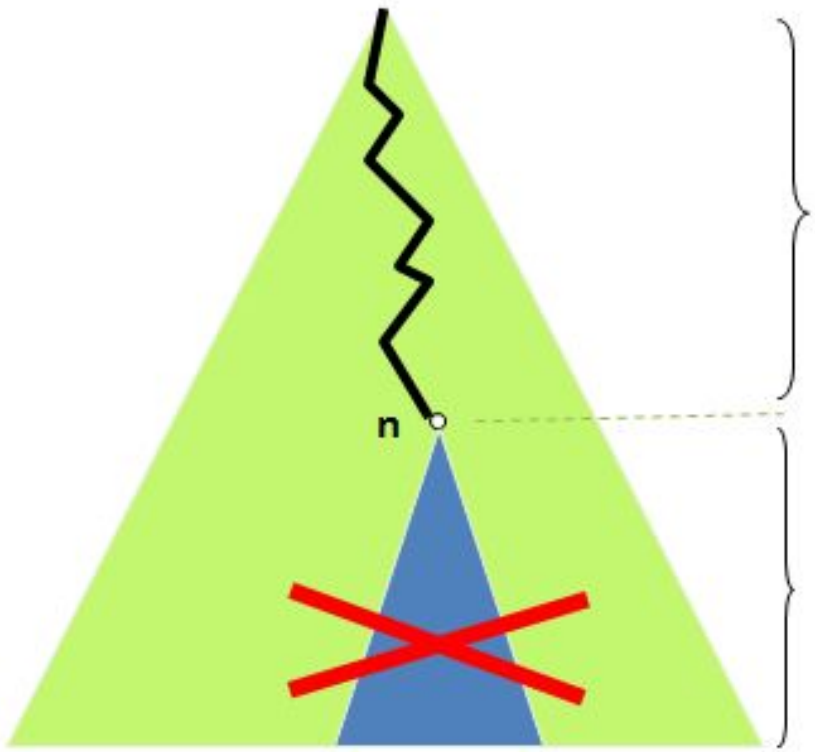
Anytime curve by operator



Cactus plot by operator



Branch-and-Bound Search



Depth first BB Tree
(Lawler & Wood66)

g(n) **cost of the search path to n**

h(n) **estimates the optimal cost below n**

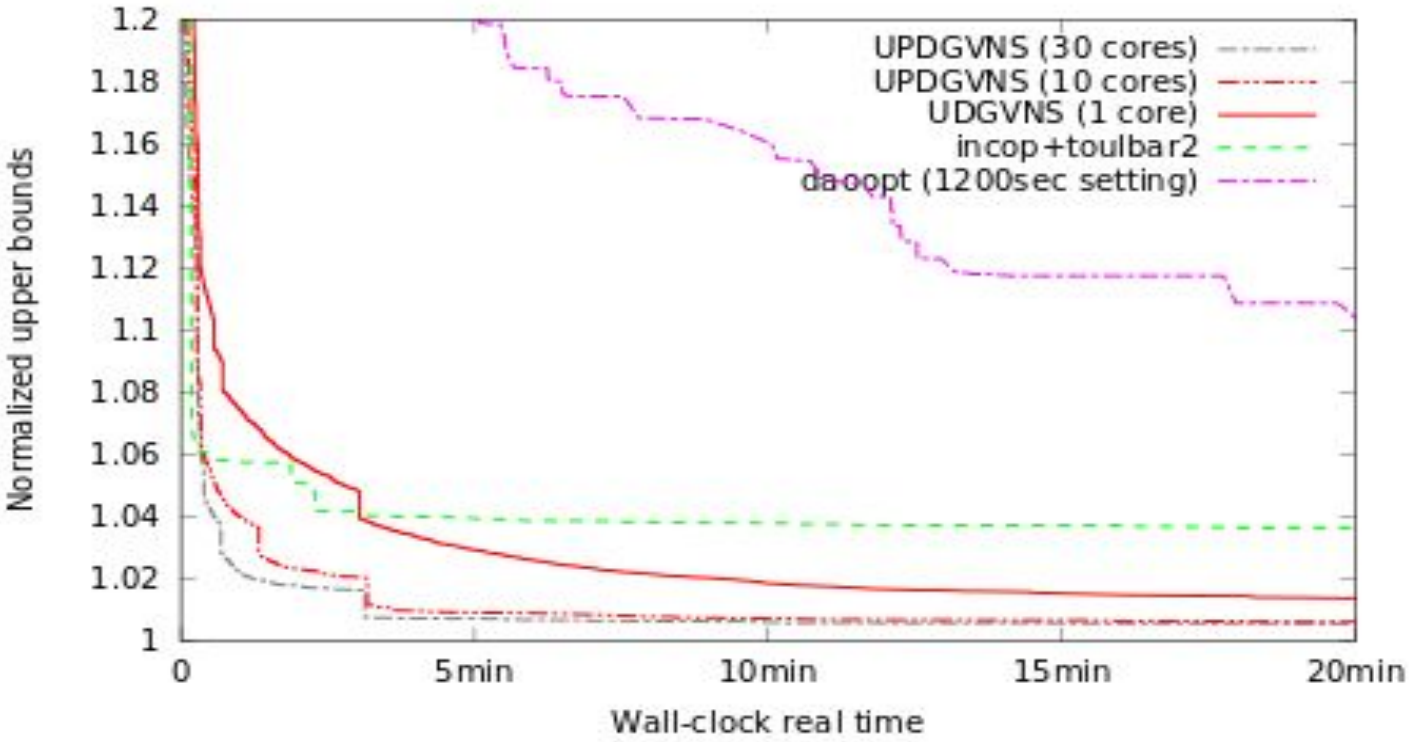
Upper Bound **UB**

Lower Bound **LB(n)**

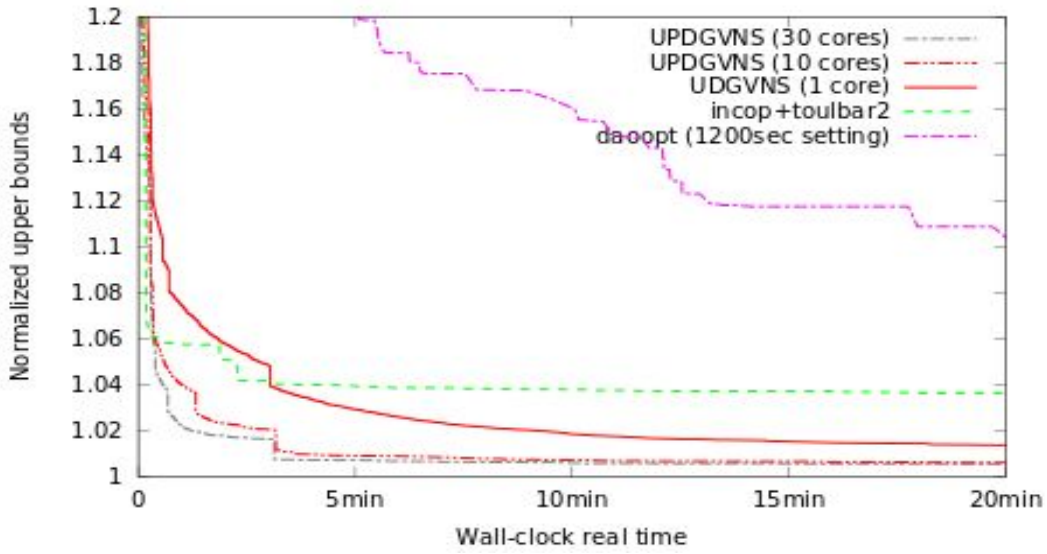
LB(n) = g(n) + h(n)

Prune if LB(n) ≥ UB

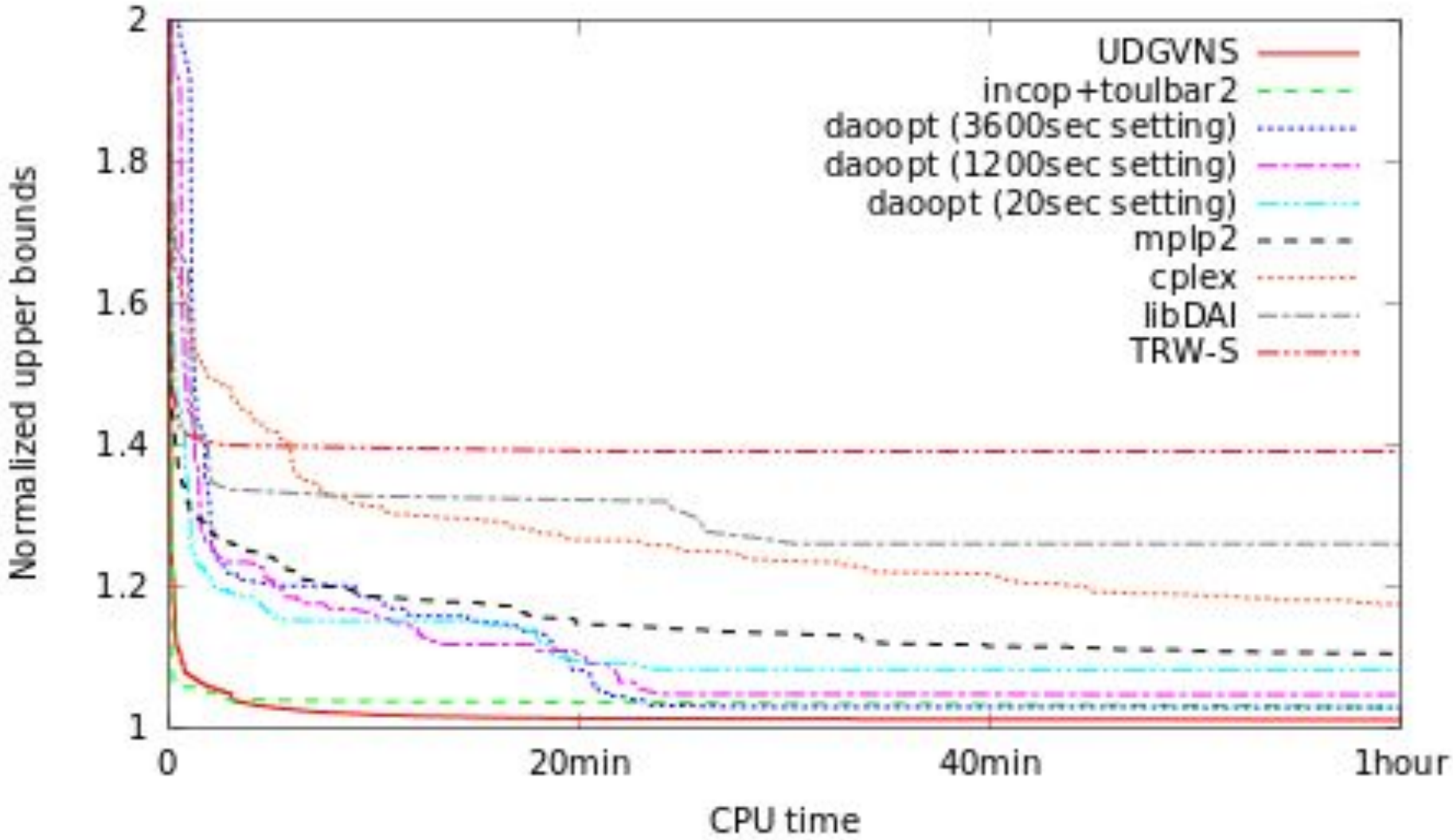
Parallelism enhances the best anytime Upper bound profil



Parallelism enhances the best anytime Upper bound profil

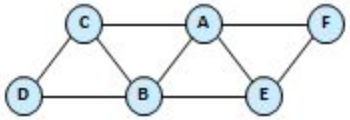


UDGVNS provides the best anytime Upper bound profil



Depth-First Branch and bound

The Search Space algorithm : branch and bound (dffbb, btd



A	B	f_1
0	0	2
0	1	0
1	0	1
1	1	4

A	C	f_2
0	0	3
0	1	0
1	0	0
1	1	1

A	E	f_3
0	0	0
0	1	3
1	0	2
1	1	0

A	F	f_4
0	0	2
0	1	0
1	0	0
1	1	2

B	C	f_5
0	0	0
0	1	1
1	0	2
1	1	4

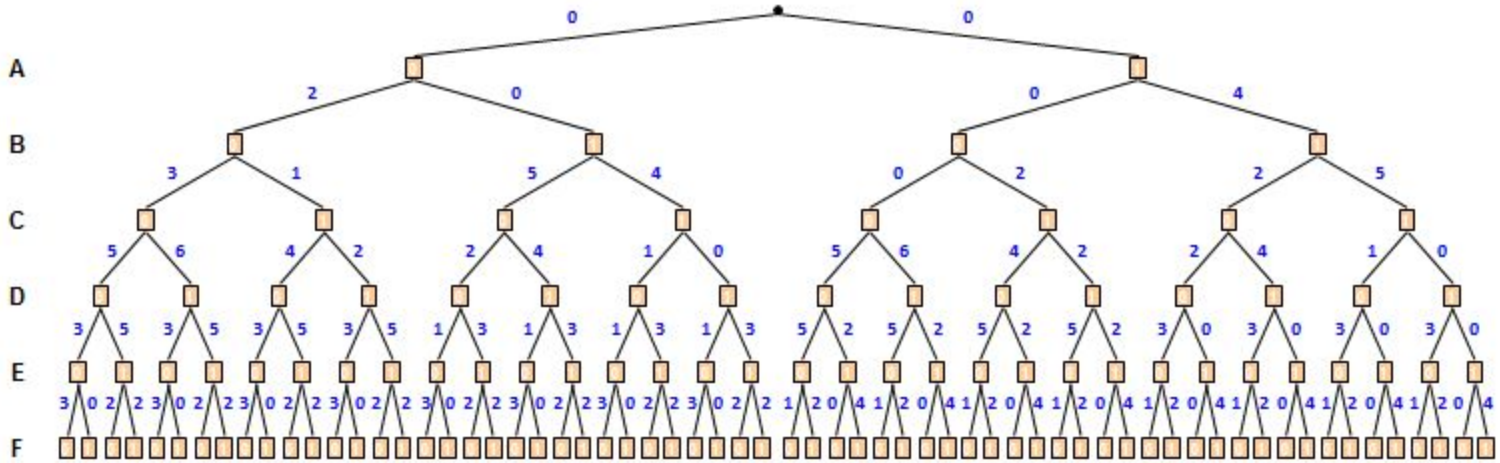
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

B	E	f_7
0	0	3
0	1	2
1	0	1
1	1	0

C	D	f_8
0	0	1
0	1	4
1	0	0
1	1	0

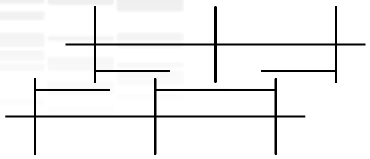
E	F	f_9
0	0	1
0	1	0
1	0	0
1	1	2

Objective function: $f(X) = \min_X \sum_{i=1}^{9} f_i(X)$



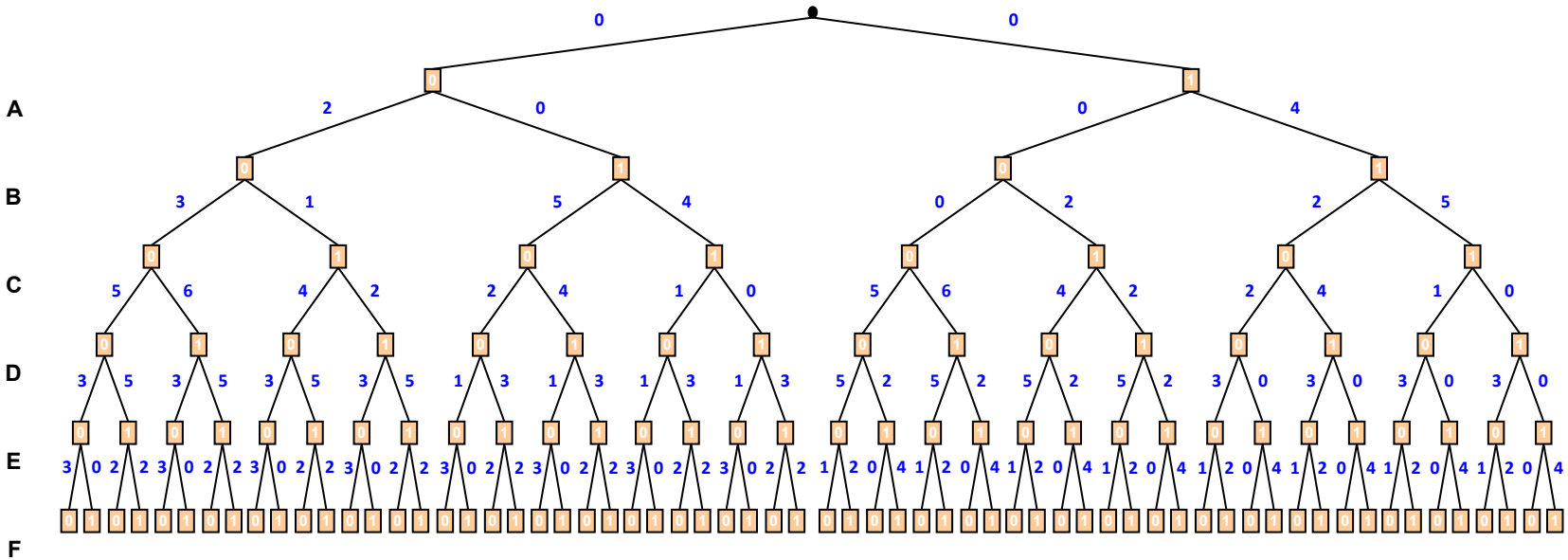
Arc-costs are calculated based on cost components

The Search Space algorithm : branch and bound (dffbb, btd



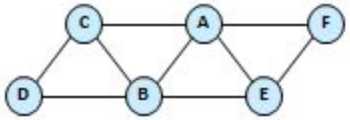
A	B	f_1	A	C	f_2	A	E	f_3	A	F	f_4	B	C	f_5	B	D	f_6	B	E	f_7	C	D	f_8	E	F	f_9
0	0	2	0	0	3	0	0	0	0	0	2	0	0	0	0	0	4	0	0	3	0	0	1	0	0	1
0	1	0	0	1	0	0	1	3	0	1	0	0	1	1	0	1	2	0	1	2	0	1	4	0	1	0
1	0	1	1	0	0	1	0	2	1	0	0	1	0	2	1	0	1	1	0	1	1	0	0	1	0	0
1	1	4	1	1	1	1	1	0	1	1	2	1	1	4	1	1	0	1	1	0	1	1	0	1	1	2

Objective function: $f(X) = \min_X \sum_{i=1}^{9} f_i(X)$



Arc-costs are calcul
ated based on cost components

The Search Space algorithm : branch and bound (dffbb, btd



A	B	f_1
0	0	2
0	1	0
1	0	1
1	1	4

A	C	f_2
0	0	3
0	1	0
1	0	0
1	1	1

A	E	f_3
0	0	0
0	1	3
1	0	2
1	1	0

A	F	f_4
0	0	2
0	1	0
1	0	0
1	1	2

B	C	f_5
0	0	0
0	1	1
1	0	2
1	1	4

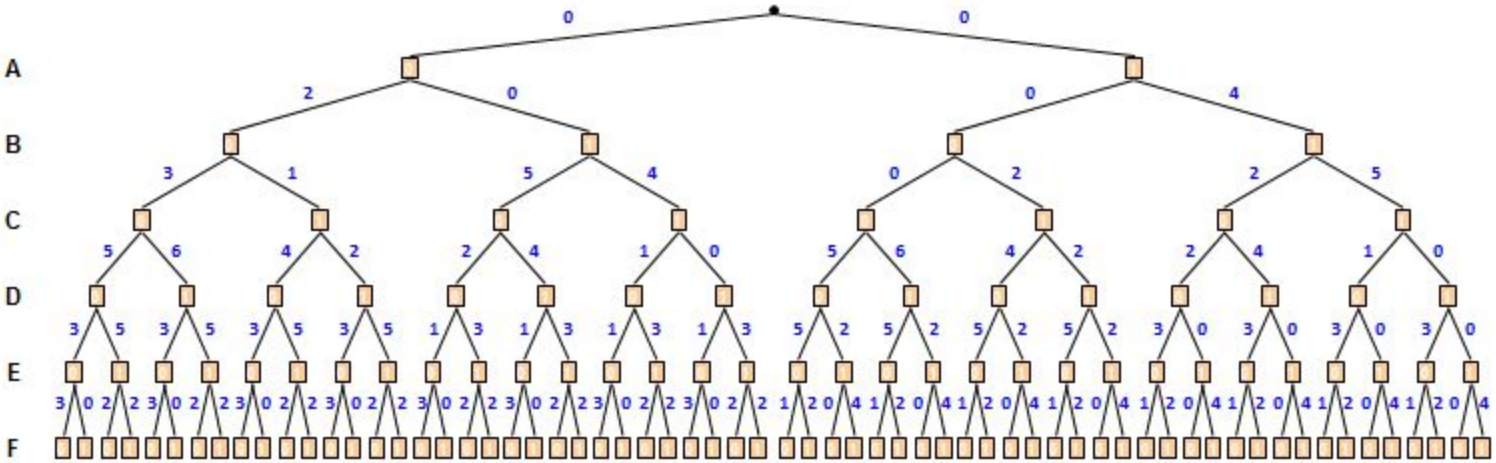
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

B	E	f_7
0	0	3
0	1	2
1	0	1
1	1	0

C	D	f_8
0	0	1
0	1	4
1	0	0
1	1	0

E	F	f_9
0	0	1
0	1	0
1	0	0
1	1	2

Objective function: $f(X) = \min_X \sum_{i=1}^{9} f_i(X)$



Arc-costs are calculated based on cost components

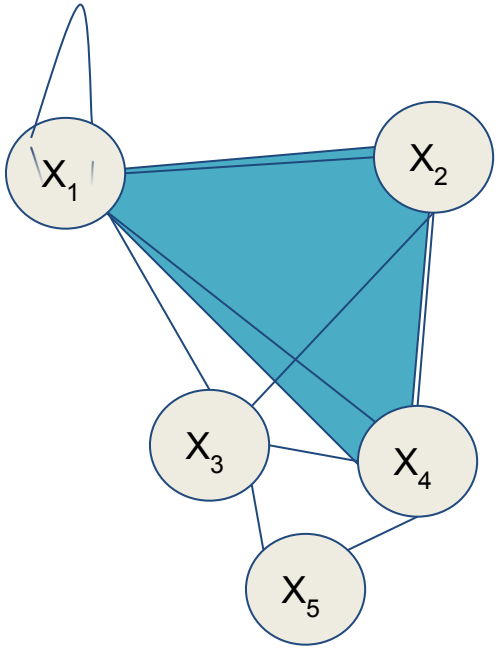
Graphical Model

(Shapiro, Haralick, IEEE PAMI 81)

- *n* variables $X = \{X_1, \ldots, X_n\}$
  - finite domains $X_i \in D_i, |D_i| \leq d$
- *e* local/global functions $F = \{f_1, \ldots, f_e\}$
  - scope, function with costs
- Costs $\in \{0, \ldots, +\infty\}$ finite or infinite integer

Minimize $\sum_F f_i(X)$

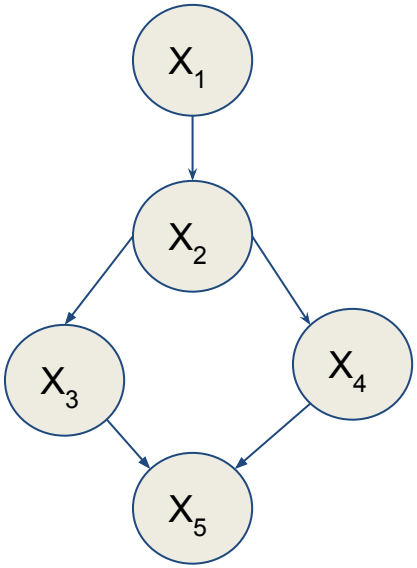
NP-hard



- Decision Diagramme
- Markov random field
- Belief propagation
- Bayesian Network

Bayesian network

- Direct acyclic Graph
- Normalized joint probability distribution
- Conditional probability



Random variables X with discrete domains joint normalized probability distribution p(X) defined as a product of positive real-valued functions:

- Optimal solution ⇒ Most Probable Explanation (MPE) (with evidence or not)