Parallel Hybrid Best-First Search

A. Beldjilali, P. Montalbano, D. Allouche, G. Katsirelos , S. de Givry

INRAE - MIAT

August 2nd, 2022

Plan

Cost Function Network

CFN

(X, D, F, k) is a CFN :

- $X = \{x_1, \ldots, x_n\}$ set of n **variables**
- $D = \{D_1, \ldots, D_n\}$ set of n **finite domains** (maximum size d)
- $F = \{f_0, \ldots, f_e\}$ set of e **cost functions**
  - f_S a cost function, with scope $S \subseteq X$
  - $f_S : D^S \mapsto \{0, \ldots, k\}$
  - $k > 0$ is an integer value associated with **forbidden assignments**

Optimization Task

$$Minimize_{t \in assignment(X)} \quad \sum_{f_S \in F} f_S(t[S])$$

NP-hard problem

Solving Cost Function Networks

Exact Methods

- Depth-First Branch-and-Bound with Equivalence Preserving Transformations (incremental lower bounds such as EDAC [4])
- AND/OR Search (exploit problem structure [8, 5, 6])
- Depth-First Search (reformulated in Constraint Programming)
- Best-First Search with Probes (reform. in Integer Programming or [1])

Approximate Methods

- Large Neighborhood Search [11]
- Other metaheuristics (INCOP [9], PILS [3],. . .)

Parallel Exact Solving Methods

Parallel Branch-and-Bound

- Parallel Depth-First Branch-and-Bound [7]
- Parallel AND/OR Search [2, 10]
- Parallel Depth-First Search (gecode)
- Parallel Best-First Search with Probes (cplex)

Other approaches

- Embarrassingly Parallel Search (EPS) [12]
- Portfolios (choco)

Load balancing

- Problem decomposition
- Bounded DFS (probe)
- Work stealing

Plan

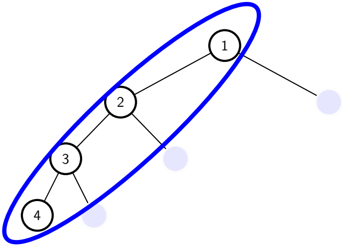
1. Cost Function Network
2. HBFS
3. Parallel HBFS
4. Experimental Results
5. Conclusion

Hybrid Best-First Search

BFS with adaptive DFS probes

- Initial BFS with 1 root node

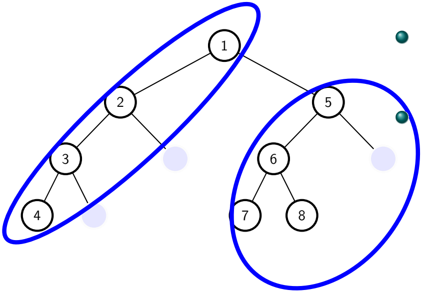
Hybrid Best-First Search



BFS with adaptive DFS probes

- Initial BFS with 1 root node
- First probe limited to 1 backtrack
  → add **3 open nodes** to BFS

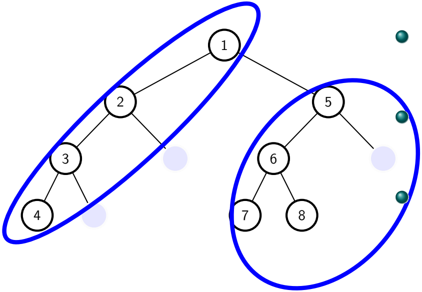
Hybrid Best-First Search



BFS with adaptive DFS probes

- Initial BFS with 1 root node
- First probe limited to 1 backtrack
  → add **3 open nodes** to BFS
- Second probe limited to 2 backtracks
  → add **1 open node** to BFS

Hybrid Best-First Search



BFS with adaptive DFS probes

- Initial BFS with 1 root node
- First probe limited to 1 backtrack
  $\rightarrow$ add **3 open nodes** to BFS
- Second probe limited to 2 backtracks
  $\rightarrow$ add **1 open node** to BFS
- Maximum number of backtracks limited to **16,384**

Hybrid Best-First Search

Function HBFS(*clb*,*cub*) : integer

$open := \{\nu(\delta = \varnothing, lb = clb)\}$; */* Initializes the open list with a root node */*

while ($open \neq \varnothing$ **and** $clb < cub$) **do**

ν :=pop(*open*) ; */* Chooses a node with minimum lower bound and maximum depth */*

Restores state $\nu.\delta$, leading to assignment A_ν, maintaining EDAC ;

$NodesRecompute := NodesRecompute + \nu.depth$;

cub :=DFS(A_ν,*cub*,*Z*) ; /* **Probe: Bounded Depth-First Search** */

$clb := \max(clb, lb(open))$;

if ($NodesRecompute/Nodes > \beta$ **and** $Z \leq N$) **then** $Z := 2 \times Z$;

else if ($NodesRecompute/Nodes < \alpha$ **and** $Z \geq 2$) **then** $Z := Z/2$;

return *cub*;

Relation between the number of open nodes and DFS backtrack limit

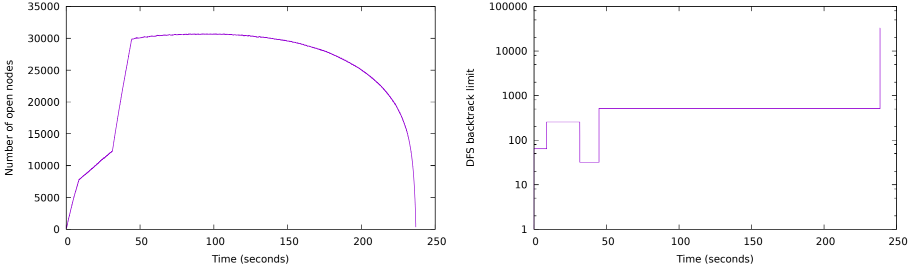
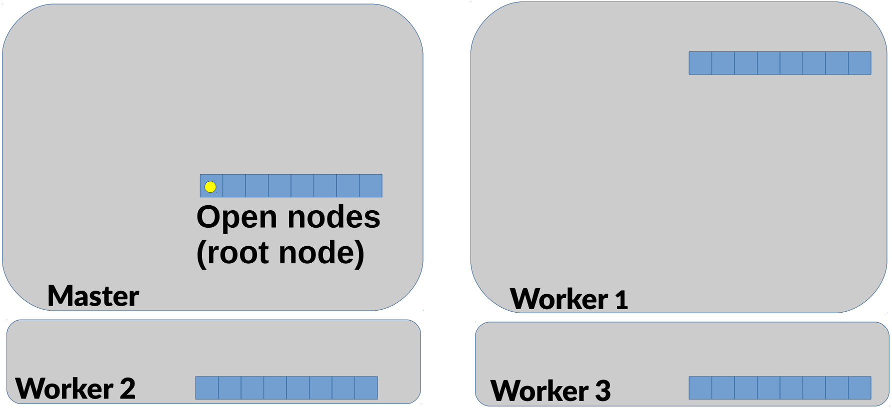


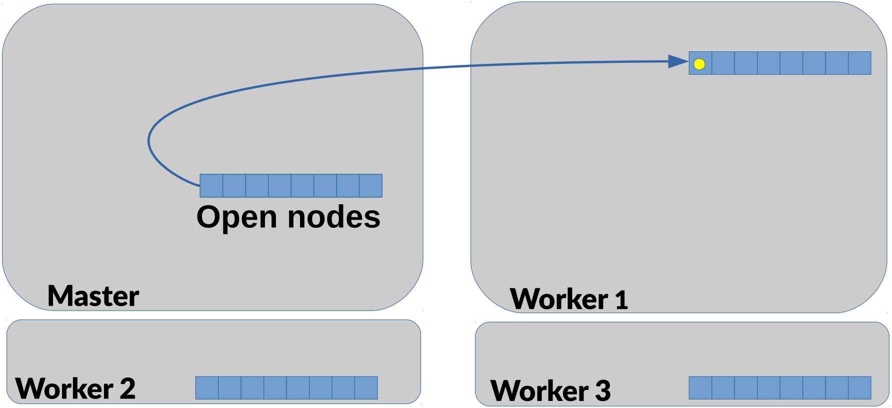
Figure – Quadratic assignment problem (nug12 with 12 variables and domain size of 12, solved in 4,615,297 backtracks and 10,269,978 nodes, and 236.694 seconds on a single core).

Plan

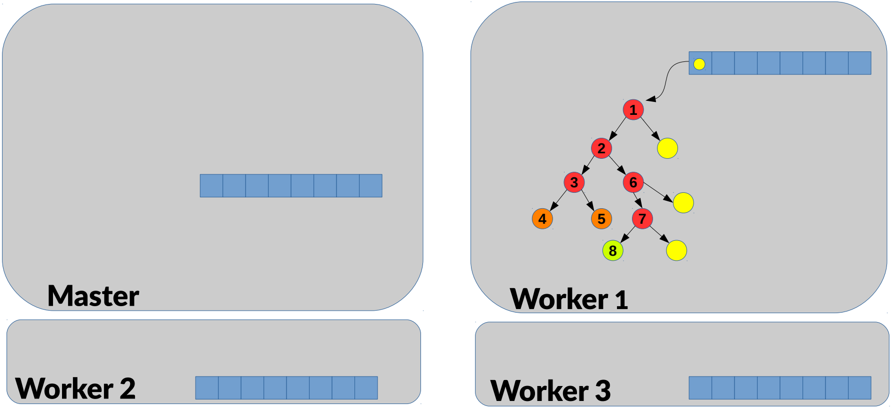
Parallel HBFS



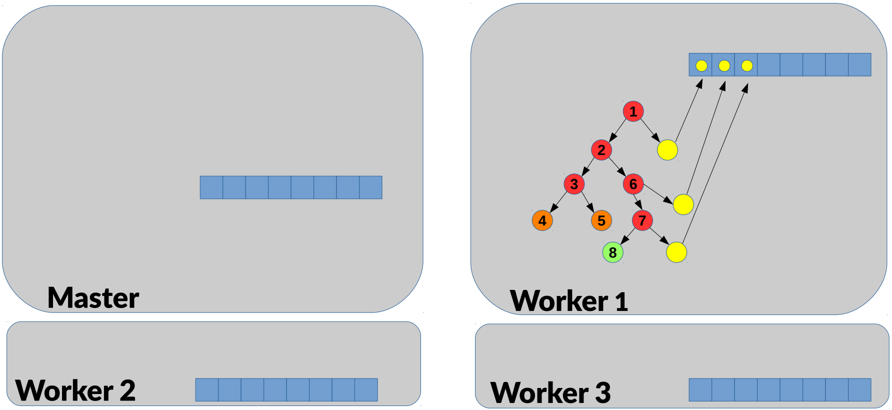
Parallel HBFS



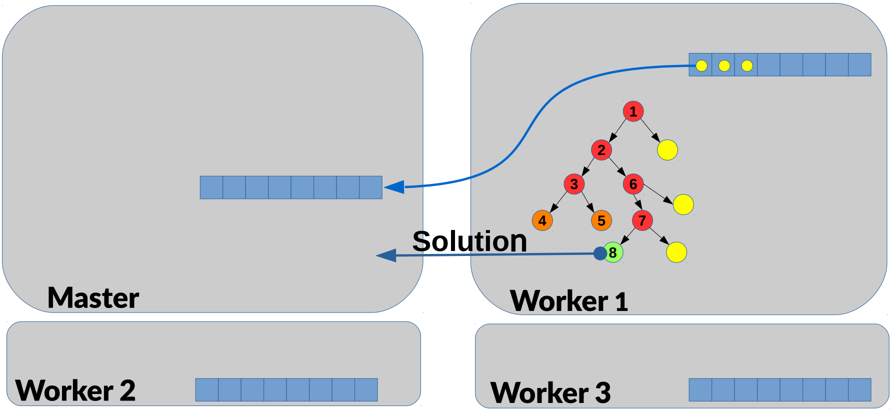
Parallel HBFS



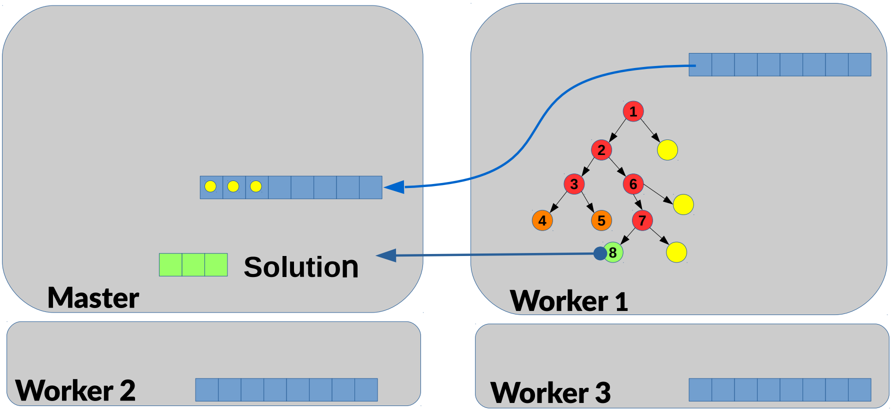
Parallel HBFS



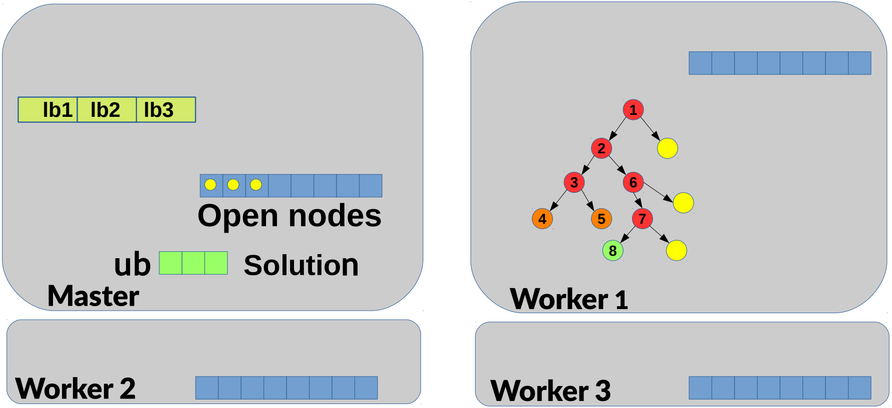
Parallel HBFS



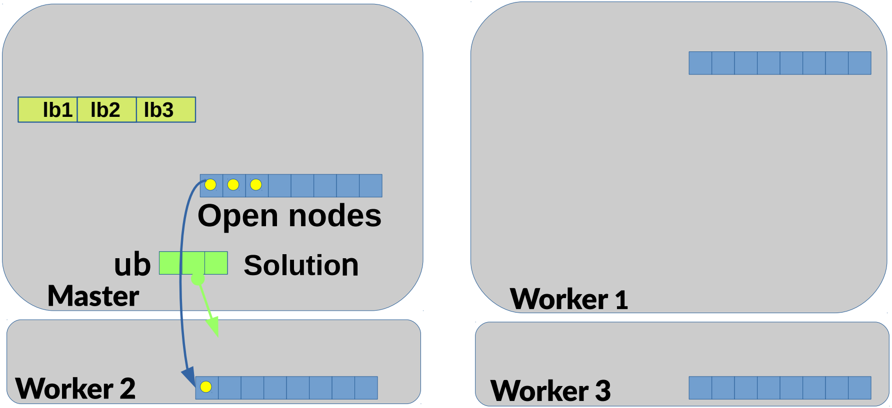
Parallel HBFS



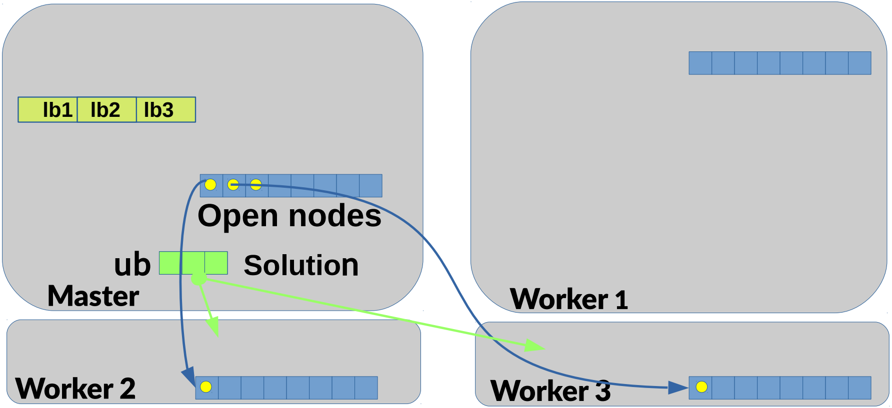
Parallel HBFS



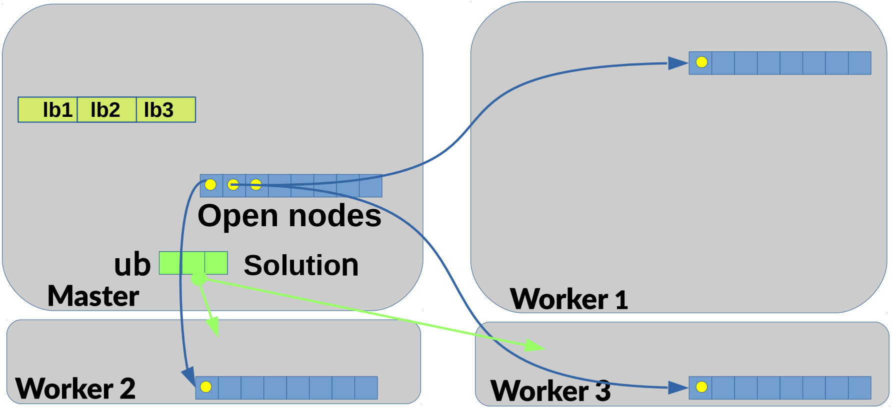
Parallel HBFS



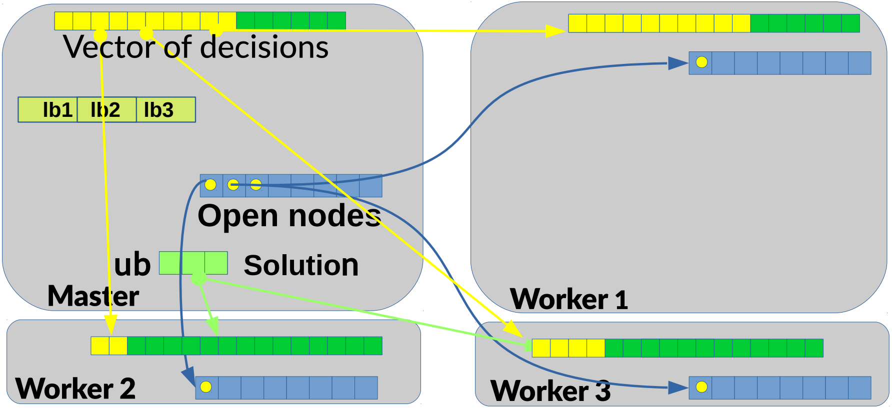
Parallel HBFS



Parallel HBFS



Parallel HBFS



Master (BFS) / Worker

```
Function HBFS-Master(clb, cub, S) : integer
    open := {ν(δ = ∅, lb = clb)} ;          /* Initializes the open list */
    I := S ;                                 /* Queue of idle workers */
    A := ∅ ;      /* Maps active workers to open nodes currently being
     processed */
    while ((open ≠ ∅ or A ≠ ∅) and clb < cub) do
10      while (open ≠ ∅ and I ≠ ∅) do
            ν :=pop(open) ;  /* Chooses a node with minimum lower bound and
             maximum depth */
            i :=popFront(I) ;             /* Unqueue the first idle worker */
            A := A ∪ {(i, ν)} ;
            Send ν and best solution cub to Worker i ;
11      Receive a list of open nodes 𝒱 and solution cub′ by worker j ;  /* Wait
         for message from any active worker */
        push(open, 𝒱) ;  /* Adds worker open nodes to the Master open list
         */
        cub := min(cub, cub′) ;       /* Checks if a better solution found */
12      pushBack(I, j) ;    /* Pushes Worker j at the end of idle queue I */
        A := A \ {(j, A[j])} ;       /* Removes Worker j from active workers */
13      clb := max(clb, min(lb(open), min{lb(ν) for (i, ν) ∈ A})) ;    /* Global
         LB */
    return cub;
```

Master / **Worker (DFS)**

Procedure HBFS-Worker(*cub*,*rank*)

while (**true**) **do**

  $open_i := \varnothing$; */* local open list of Worker i */*

  Receive an open node ν and solution cub' by Master ; */* Wait for message */*

  $cub := \min(cub, cub')$; */* Updates cub and best solution if any */*

  Restores state $\nu.\delta$, leading to assignment A_ν, maintaining soft local consistency ;

  $NodesRecompute := NodesRecompute + \nu.depth$;

  $cub :=$DFS(A_ν,cub,Z_i) ; /* **Probe: Bounded Depth-First Search** */

  if ($NodesRecompute > 0$) **then**

    if ($NodesRecompute/Nodes > \beta$ **and** $Z_i \leq N$) **then** $Z_i := 2 \times Z_i$;

    else if ($NodesRecompute/Nodes < \alpha$ **and** $Z_i \geq 2$) **then** $Z_i := Z_i/2$;

  Send $open_i$ and best solution cub to the Master ;

Automatic tuning of DFS backtrack limit

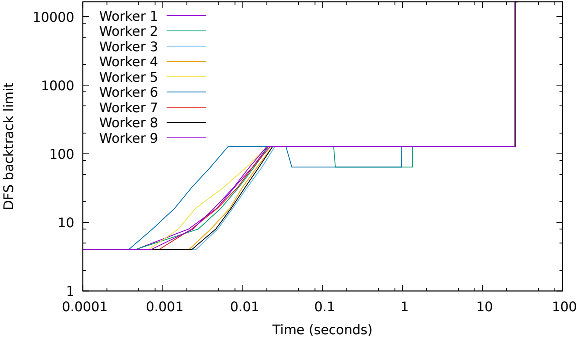
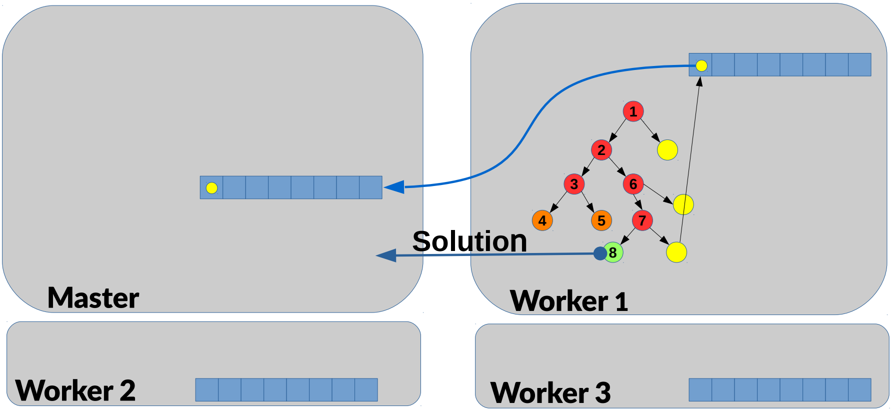


Figure – Evolution of DFS backtrack limit as time passes on a quadratic assignment problem (nug12 with 12 variables and domain size of 12, solved in 4,474,971 backtracks and 10,764,877 nodes, and 25.948 seconds on a 10-core server).

Improve ramp-up phase (burst mode)



Plan

Experimental setup

- Benchmarks : 134 instances
  - Warehouses (15), MaxClique (62)
  - Linkage (22), Computational Protein Design (35)
- Parallel architectures :
  - server ($<$24 cores, 256GB)
  - cluster ($<$13,464 cores, 192GB/36-core, Infiniband EDR 100Gb/s)
- Solvers :
  - CFN : toulbar2 v1.2.0 (parallel HBFS using MPI)
  - ILP : cplex v20.1 (multi-threading)
- Time limit : 1 hour (except sequential version on cluster with 10h)

Burst-mode effect

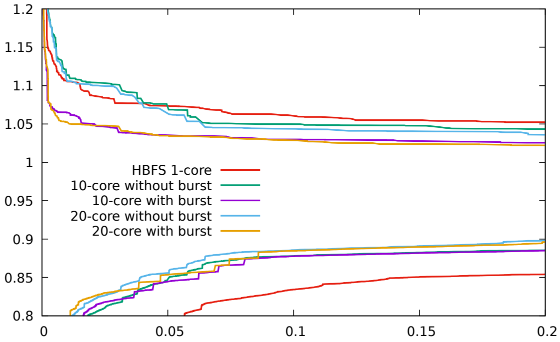


Figure – Comparison on a medium-scale computer between sequential versus parallel HBFS with or without burst mode. The x-axis represents normalized time (with 0.2 corresponding to 720 seconds). The y-axis corresponds to normalized lower and upper bounds on 134 instances (with 1 corresponding to the optimum or best known cost).

Load-balancing analysis of worker idle times

10-core (on 31 instances)	20-core (29 inst.)	180-core (8 inst.)
1.3% +- 2.22	2.7% +- 4.81	8.8% +- 3.75

Table – Average waiting/idle time percentage by a worker of total solving real-time (minus sequential preprocessing time) for different number of cores on instances solved with more than $1,000$ backtracks and 1 second (resp. 100 sec. for 180-core) overall time.

Parallel HBFS versus parallel integer programming

Method	CPD (35)		Warehouses (15)		Linkage (22)		MaxClique (62)	
		Speed-up		*Speed-up*		*Speed-up*		*Speed-up*
HBFS-1	30 (43.44s)		15 (128.96s)		20 (23.24s)		37 (364.25s)	
HBFS-10	30 (8s)	*5.43*	15 (80.174s)	*1.61*	21 (3.5s)	*6.64*	38 (40.24s)	*9.05*
HBFS-20	30 (4.43s)	*9.81*	15 (85.39s)	*1.51*	21 (2s)	*11.62*	40 (19.9s)	*18.3*
cplex-1	24 (331.2s)		15 (123.83s)		22 (8.04s)		42 (282.16s)	
cplex-10	24 (226.51s)	*1.46*	**15 (68.82s)**	*1.8*	22 (2.56s)	*3.14*	45 (55.48s)	*5.08*
cplex-20	24 (198.49s)	*1.67*	15 (72.06s)	*1.72*	**22 (2.29s)**	*3.51*	**46 (71.47s)**	*3.95*
HBFS-1 (cluster)	30 (66.46s)		15 (392.30s)		21 (427.21s)		37 (504s)	
HBFS-180 (cluster)	**30 (3.7s)**	*17.96*	15 (126s)	*3.11*	22 (4.15s)	*102.94*	45 (6.44s)	*78.26*

Table – Solved instances within 1 h (except for sequential HBFS-1 with a larger timeout of 10 hours) and their average time in seconds in parentheses.

Anytime curves on Computational Protein Design

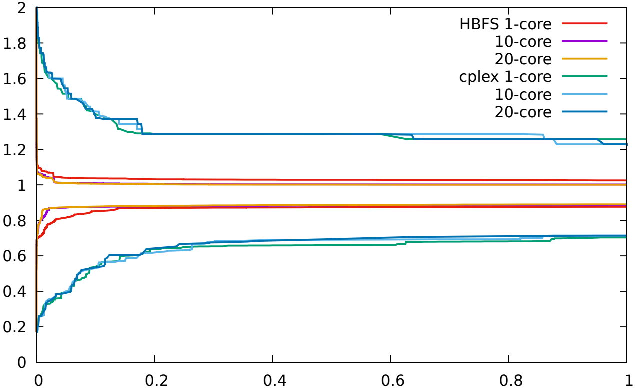


Figure – The x-axis represents normalized time (with 1 corresponding to 3, 600 seconds). The y-axis corresponds to normalized lower and upper bounds on 35 CPD instances.

Anytime curves on Linkage Analysis

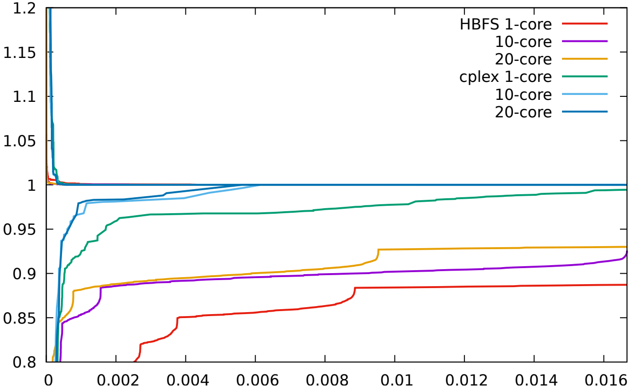


Figure – The x-axis represents normalized time (with 1 corresponding to 3, 600 seconds). The y-axis corresponds to normalized lower and upper bounds on 22 Linkage instances.

Anytime curves on Uncapacitated Warehouse Location

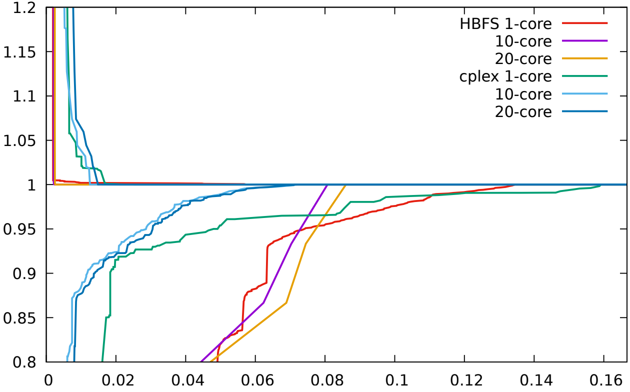


Figure – The x-axis represents normalized time (with 1 corresponding to 3, 600 seconds). The y-axis corresponds to normalized lower and upper bounds on 15 Warehouses instances.

Anytime curves on Maximum Clique Problem

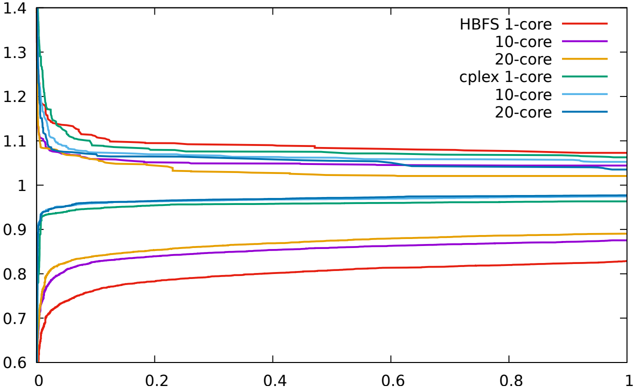


Figure – The x-axis represents normalized time (with 1 corresponding to 3, 600 seconds). The y-axis corresponds to normalized lower and upper bounds on 55 MaxClique instances.

Comparison of parallel HBFS with EPS

instance	n	d	av. time	max. t.	#fail(depth)	EPS-180	HBFS-180
linkage/pedigree19	259	5	20.57	-	1 (4)	-	**69.1**
linkage/pedigree40	274	6	101.99	-	49 (21)	-	**1680**
linkage/pedigree51	295	5	0.61	497.38	0	499	**5.7**
cpd/1BRS	38	178	2.94	38.90	0	44	**37.5**
cpd/1CDL	38	170	6.66	79.04	0	79	**18.3**
cpd/1GVP	52	170	14.59	170.66	0	171	**17.0**
maxcl./brock400_1	400	2	63.95	-	12 (149)	-	**1812**
maxcl./brock400_2	400	2	65.27	-	18 (149)	-	**880**
maxcl./san400_0.5_1	400	2	5.07	414.96	0	3652	**1220**

Table – EPS and HBFS-180 results on hard instances (with n variables and maximum domain size d). A '-' indicates that some (see #failed) subproblems could not be solved in less than $3,600$sec.

Comparison of parallel HBFS with 1,800 cores

instance	n	d	HBFS-180	HBFS-1800
linkage/pedigree19	259	5	**69.1**	201
linkage/pedigree40	274	6	**1680**	2753
linkage/pedigree51	295	5	**5.7**	8.4
cpd/1BRS	38	178	37.5	**15.2**
cpd/1CDL	38	170	18.3	**14.9**
cpd/1GVP	52	170	**17.0**	24.1
maxclique/brock400_1	400	2	1812	**947**
maxclique/brock400_2	400	2	880	**686**
maxclique/san400_0.5_1	400	2	1220	**630**

Table – HBFS-180 and HFBS-1800 results on hard instances with n variables and maximum domain size d. See *Supplementary Material (Results)*.

Conlusion & perspectives

Conclusion

- Speed-up depends on the instance, significant gains have been observed
- Scalable to a larger number of cores due to the minimal size of the information shared (tested on 1,800 cores, see *Supplementary Materials*)

Future work

- Combines parallel HBFS and parallel variable neighborhood search [11]
- Parallelizing HBFS with Tree Decomposition (BTD-HBFS [1]) sharing learnt (no)goods

References

D Allouche, S de Givry, G Katsirelos, T Schiex, and M Zytnicki.
Anytime Hybrid Best-First Search with Tree Decomposition for Weighted CSP.
In *Proc. of CP-15*, pages 12–28, Cork, Ireland, 2015.

D. Allouche, S. de Givry, and T. Schiex.
Towards parallel non serial dynamic programming for solving hard weighted csp.
In *Proc. of CP-10*, St Andrews, Scotland, 2010.

François Beuvin, Simon de Givry, Thomas Schiex, Sébastien Verel, and David Simoncini.
Iterated local search with partition crossover for computational protein design.
Proteins : Structure, Function, and Bioinformatics, 2021.

S. de Givry, M. Zytnicki, F. Heras, and J. Larrosa.
Existential arc consistency : Getting closer to full arc consistency in weighted CSPs.
In *Proc. of IJCAI-05*, pages 84–89, Edinburgh, Scotland, 2005.

Simon de Givry, Thomas Schiex, and Gérard Verfaillie.
Exploiting tree decomposition and soft local consistency in weighted csp.
In *Proc. of AAAI-06*, Boston, MA, 2006.
http ://www.inra.fr/mia/T/degivry/VerfaillieAAAI06pres.pdf (slides).

Philippe Jégou, Hélène Kanso, and Cyril Terrioux.
Adaptive and opportunistic exploitation of tree-decompositions for weighted csps.
11 2017.
doi:10.1109/ICTAI.2017.00064.

Bernard Mans, Thierry Mautor, and Catherine Roucairol.
A parallel depth first search branch and bound algorithm for the quadratic assignment problem.
European Journal of Operational Research, 81(3) :617–628, 03 1995.
URL : https://ideas.repec.org/a/eee/ejores/v81y1995i3p617-628.html.

R. Marinescu and R. Dechter.
And/or branch-and-bound for graphical models.
In *Proc. of IJCAI-05*, pages 224–229, Edinburgh, Scotland, 2005.

B. Neveu and G. Trombettoni.
INCOP : An Open Library for INcomplete Combinatorial OPtimization.
In *Proc. of CP-03*, pages 909–913, Cork, Ireland, 2003.

L Otten and R Dechter.
And/or branch-and-bound on a computational grid.
JAIR, 59 :351–435, 2017.

Abdelkader Ouali, David Allouche, Simon de Givry, Samir Loudni, Yahia Lebbah, Francisco Eckhardt, and Lakhdar Loukil.
Iterative Decomposition Guided Variable Neighborhood Search for Graphical Model Energy Minimization.
In *Proc. of UAI-17*, pages 550–559, Sydney, Australia, 2017.

Jean-Charles Régin and Arnaud Malapert.
Parallel Constraint Programming.
Springer, 2018.