Zurich, Switzerland

Zurich, Switzerland

Zurich, Switzerland

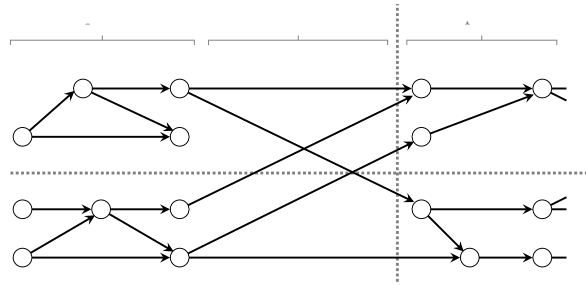
Zurich, Switzerland

ABSTRACT

CCS CONCEPTS

KEYWORDS

36th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2024),
https://doi.org/10.1145/3626183.3659972.



$$C_{work}(s) = \max_{p\in\{1,\dots,P\}} \sum_{\substack{\pi(v)=p\\ \tau(v)=s}} w(v)\,.$$

$$\sum_{\substack{(v,p_1,p_2,s_1)\in\Gamma\\ p_1=p\\ s_1=s}} c(v)\,,$$

$$\sum_{\substack{(v,p_1,p_2,s_1)\in\Gamma\\ p_2=p\\ s_1=s}} c(v)\,,$$

$$C_{comm}(s) = \max_{p\in\{1,\dots,P\}} \max\left(C_{send}(p,s),\, C_{rec}(p,s)\right)\,.$$

$$C(s) = C_{work}(s) + g\cdot C_{comm}(s) + \ell\,,$$

future. Once we cannot assign further nodes to at least half

- Source: A different greedy approach that in each step forms

algorithm uses a simple rule to cluster the original source

round-robin-based approach to assign the current source

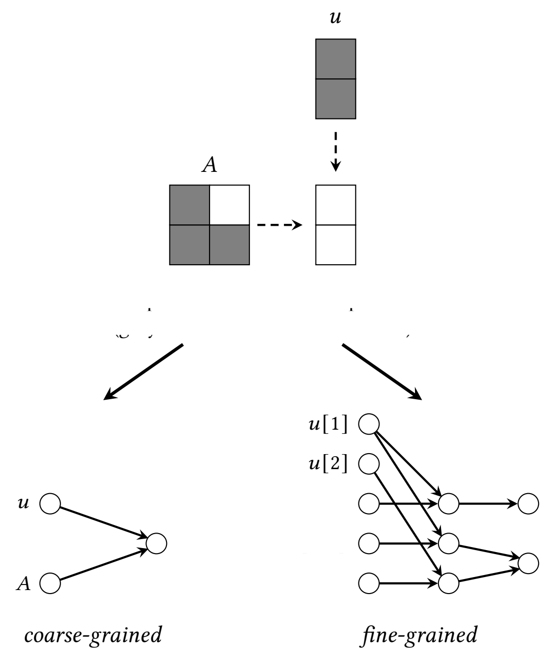
work costs between processors. Besides the current source

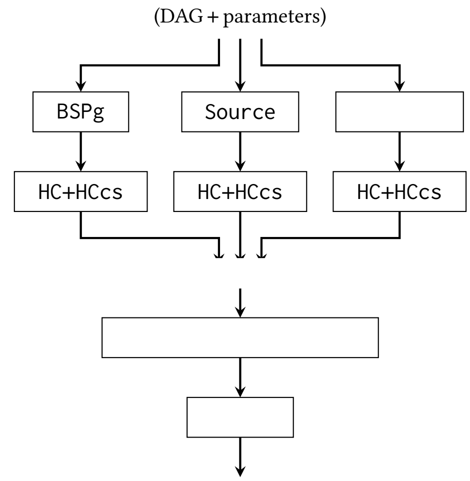
successors to the current superstep, if this requires no extra

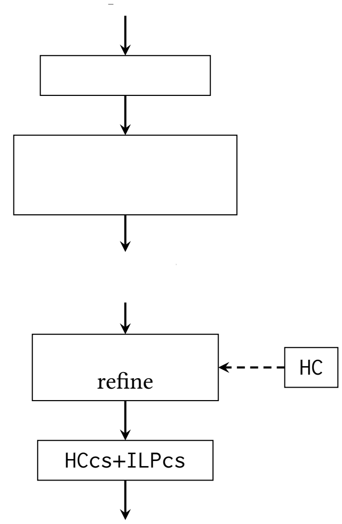
4.4 ILP-based approach

method (denoted HC) that begins from an initial solution (that is,

either a local minimum is found where none of the potential modification steps result in an improvement, or until a predefined time


u
A
u
A
u[1]
u[2]
coarse-grained
fine-grained

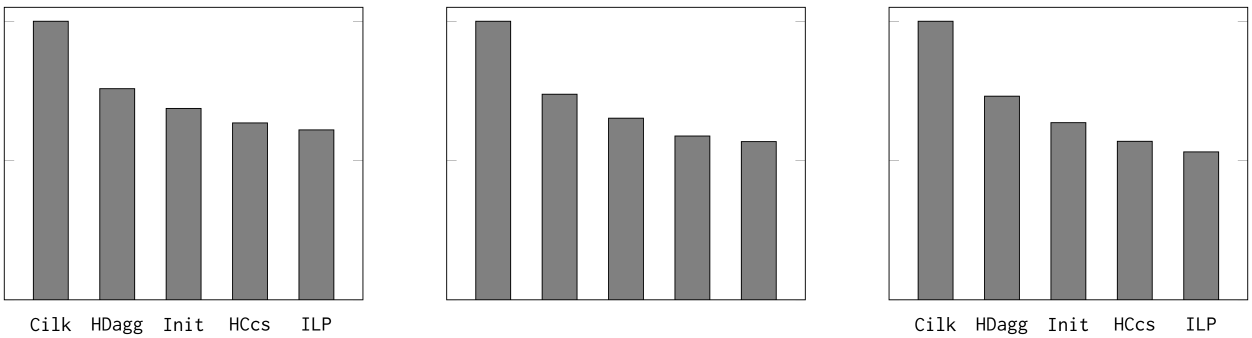

(DAG + parameters)
BSPg
Source
HC+HCcs
HC+HCcs
HC+HCcs



datasets, for $P \in \{4, 8, 16\}$, $g \in \{1, 3, 5\}$ and $\ell = 5$, without NUMA.

schedules separately (see Appendix C for details). Our experiments

all the parameters $P \in \{8, 16\}$ and $\Delta \in \{2, 3, 4\}$, our method provides

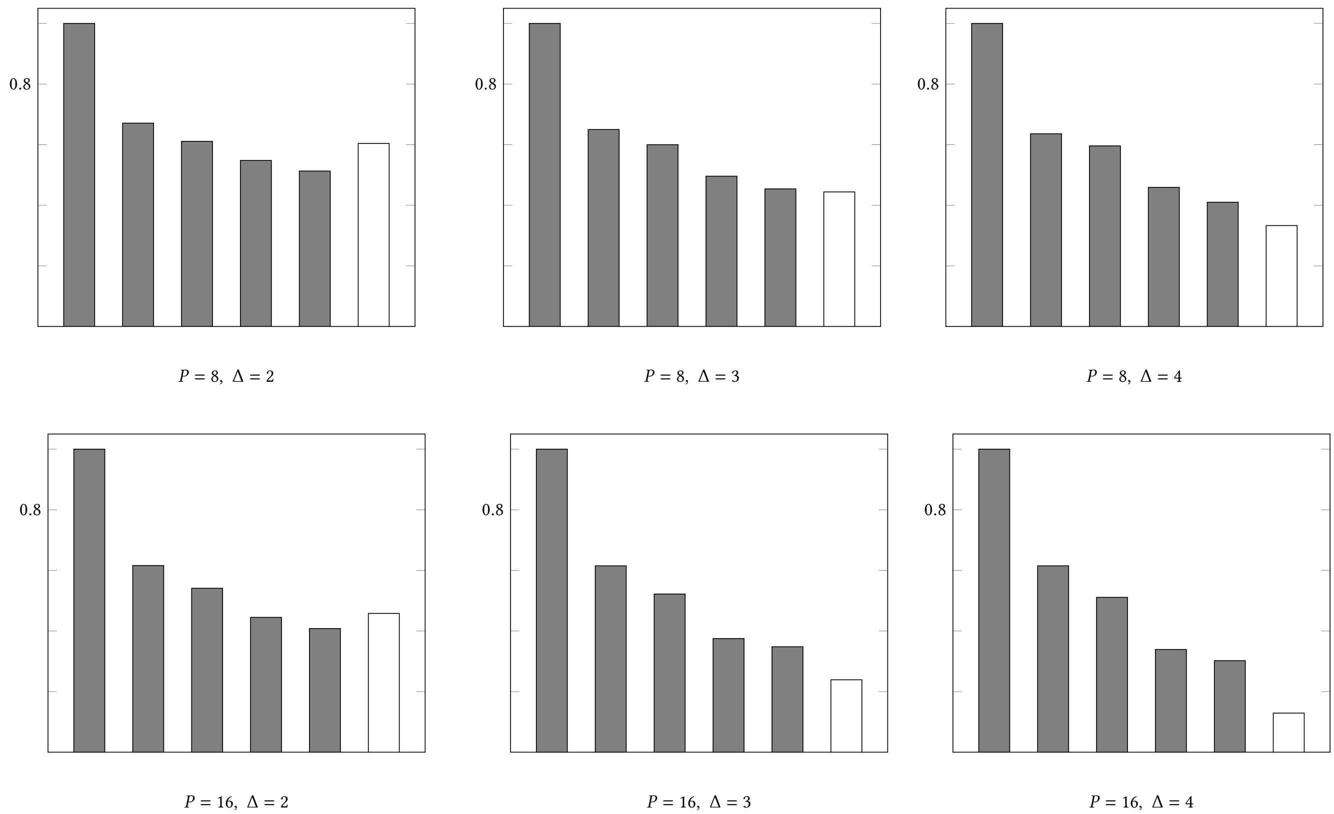


$g = 3$

$P = 8$	48% / 27%	55% / 35%	61% / 42%
$P = 16$	57% / 36%	67% / 51%	71% / 58%

be able to provide significantly better schedules for many relevant applications.

(e.g. if we have $\Delta = 4$, or if we only have $\Delta = 3$ but $P = 16$, and hence



and different NUMA increase factors $\Delta \in \{2, 3, 4\}$. The multilevel approach (`ML`) is shown separately at the end. This specific figure only covers the `small`, `medium` and `large` datasets, since `tiny` is too small to coarsify with `ML`; however, in general, our improvement factors in Section 7.2 and Appendix C also include `tiny`.

$P = 16$	54% / 26%	76% / 61%	87% / 79%

method is indeed a specialized tool, which is useful mostly when

work, the same approach could become an efficient tool for any

8 CONCLUSION

time-consuming: HC+HCcs and Multi typically take between 1-2

still be reduced significantly, in general, it is inevitable that this

- when the computation can be captured in a coarse-grained way by relatively small DAGs, but possibly still with large

suboptimal.

ACKNOWLEDGMENTS

We would like to thank Benjamin Lozes for his valuable help with

REFERENCES

Artifacts/tree/master/SPAA 2024 Efficient Multi-Processor Scheduling.

schedules for parallel processing systems. *Commun. ACM* 17, 12 (1974), 685–690.

9, 9 (1998), 872–892.

Using BSP. Oxford University Press, USA.

[6] David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten Von Eicken. 1993. LogP:

1–12.

[8] Alfredo Goldman, Gregory Mounié, and Denis Trystram. 1998. Near optimal

[10] Julien Herrmann, Jonathan Kho, Bora Uçar, Kamer Kaya, and Ümit V Çatalyürek.

371–380.

[11] Jonathan MD Hill, Bill McColl, Dan C Stefanescu, Mark W Goudreau, Kevin Lang, Satish B Rao, Torsten Suel, Thanasis Tsantilas, and Rob H Bisseling. 1998.

[12] JA Hoogeveen, Jan Karel Lenstra, and Bart Veltman. 1994. Three, four, five, six,

[13] Jing-Jang Hwang, Yuan-Chieh Chow, Frank D Anger, and Chung-Yee Lee. 1989.

times. *siam journal on computing* 18, 2 (1989), 244–257.

[14] Hidehiro Kanemitsu, Masaki Hanada, and Hidenori Nakazato. 2016. Clustering-

[15] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. 1997. Multi-

[16] Janardhan Kulkarni, Shi Li, Jakub Tarnawski, and Minwei Ye. 2020. Hierarchy-

effective technique for allocating task graphs to multiprocessors. *IEEE transactions on parallel and distributed systems* 7, 5 (1996), 506–521.

single machine with precedence constraints. *ORSA Journal on Computing* 2, 4 (1990), 346–352.

168–177.

[22] Quanquan C Liu, Manish Purohit, Zoya Svitkina, Erik Vee, and Joshua R Wang.

[24] William F McColl. 1995. Scalable computing. *Computer Science Today* (1995), 46–61.

[25] William F McColl and Alexandre Tiskin. 1999. Memory-efficient matrix multipli-

[27] M Yusuf Özkaya, Anne Benoit, Bora Uçar, Julien Herrmann, and Ümit V

processors using DAG partitioning. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 155–165.

[29] Pál András Papp, Georg Anegg, and Albert-Jan N Yzelman. 2023. Partitioning

[30] Christophe Picouleau. 1995. New complexity results on scheduling with small communication delays. *Discrete Applied Mathematics* 60, 1-3 (1995), 331–342.

[31] Merten Popp, Sebastian Schlag, Christian Schulz, and Daniel Seemaier. 2021.

for distributed-memory machines. *IEEE transactions on parallel and distributed systems* 13, 6 (2002), 648–658.

[33] Sebastian Schlag, Vitali Henne, Tobias Heuer, Henning Meyerhenke, Peter Sanders, and Christian Schulz. 2016. K-way hypergraph partitioning via n-level

answers about BSP. *Scientific Programming* 6, 3 (1997), 249–274.

[35] Ola Svensson. 2010. Conditional hardness of precedence constrained scheduling

[36] Haluk Topcuoglu, Salim Hariri, and Min-You Wu. 2002. Performance-effective

[37] Aleksandar Trifunović and William J Knottenbelt. 2008. Parallel multilevel

(2008), 563–581.

ACM 33, 8 (1990), 103–111.

ming. *International Journal of Parallel Programming* 42 (2014), 619–642.

specification, implementation, parallelisation, and evaluation. *arXiv preprint arXiv:1906.03196* (2020).

[46] Behrooz Zarebavani, Kazem Cheshmi, Bangtian Liu, Michelle Mills Strout, and Maryam Mehri Dehnavi. 2022. HDagg: hybrid aggregation of loop-carried

https://github.com/Algebraic-Programming/OneStopParallel.

Python scripts.

an external dependency [46].

processors become idle (it finishes computing a node), it takes the

phase can last at most until time t; hence we assign all nodes that

putations, i.e. solving sparse linear systems defined by a triangular

available, we directly apply this in our experiments: we convert end if

$free[\pi(v)] \leftarrow True$

$ready \leftarrow ready \cup \{u\}$

if $\forall\, (u_0, u) \in E$ we have either $\pi(u_0) = \pi(v)$ or $\tau(u_0) <$

end while

$superstep \leftarrow 0$

$sources \leftarrow$ all unassigned $v \in V$ with $\text{indeg}(v) = 0$

if v shares an out-neighbor with another $u \in sources$

end if

$\pi(v) \leftarrow p, \quad \tau(v) \leftarrow superstep$

$p \leftarrow (p+1)$ modulo P

$\pi(v) \leftarrow p, \quad \tau(v) \leftarrow superstep$

$p \leftarrow (p+1)$ modulo P

for all edges $(v, u) \in E$ with $v \in sources$ **do**

$\pi(u) \leftarrow \pi(v), \quad \tau(u) \leftarrow superstep$

end if

$superstep \leftarrow superstep + 1$

schedule.

Our hill climbing methods take an initial solution (BSP schedule)

node v. In particular, if we currently have $\pi(v) = p$ and $\tau(v) = s$, we consider all the solutions for where we have $\pi(v) \in \{1, ..., P\}$ (i.e. v on any processor) and $\tau(v) \in \{(s-1), s, (s+1)\}$ (i.e. v in the

$(s-1)$ (unless $p = \pi(v)$). This ensures that whenever we consider

predecessors/successors.

move option, also updating our data structures efficiently.

ules with $(v, \pi(v), p, s) \in \Gamma$ for different possible $s \in [\tau(v), s_0 - 1]$.

For both hill climbing methods, there are two possible variants:

A.4 ILP-based methods

u can potentially be even larger than $|V_0|$, hence if we add

solution.

munication steps (v, p_1, p_2, s) where v was computed before S_0, it is only required on p_2 after S_0, but our Γ still just hap-

As mentioned before, we use the CBC open-source solver [7] for solving the ILP problems described above. Since ILPfull tries

5 minutes, similarly to HC+HCcs. We select an even shorter time

step that satisfy this property. In particular, an edge $(u, v) \in E$ can

largest $c(u)$ value.

(possibly many) original nodes contracted into u, maybe only a few

based on summed weights $c(u)$, then this is only an upper bound

(u', v') such that there is a long directed path from u to v', and

However, this difference is only relevant from a theoretical modelling perspective, to emphasize the fact that even if a node v has

needs to be sent to p once; due to this, for partitioning problems, it

containing v and all its out-neighbors. For our work, this is simply

DAG, and hence it has no effect. In fact, all of our algorithms begin

B.1 Coarse-grained DAGs

[45], and extended this with a so-called HyperDAG backend. The

BLAS algorithm and gather meta-data during this run, which is then

solvers (e.g. Conjugate Gradient for positive definite systems, or

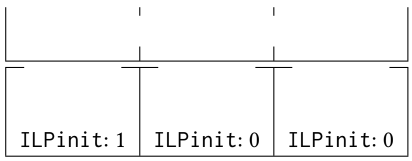
We note that while larger containers (matrices, vectors) are easy

represents combining indeg(v) distinct values, indeg$(v) - 1$ is a

$A^k \cdot u$, by executing k distinct spmv operations. This compu-

- CG: the well-known conjugate gradient method for finding a

iterations.


ILPinit: 1
ILPinit: 0
ILPinit: 0

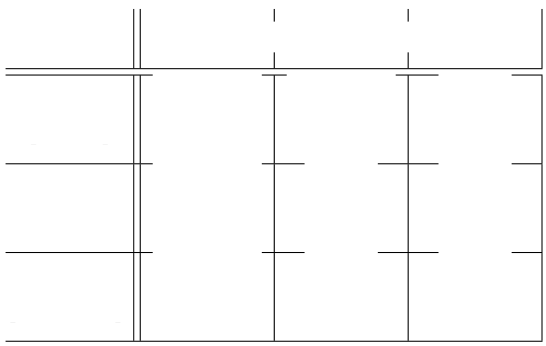


Table 6: Improvement achieved by our scheduler (without NUMA) for each combination of g, P and dataset, with respect to Cilk

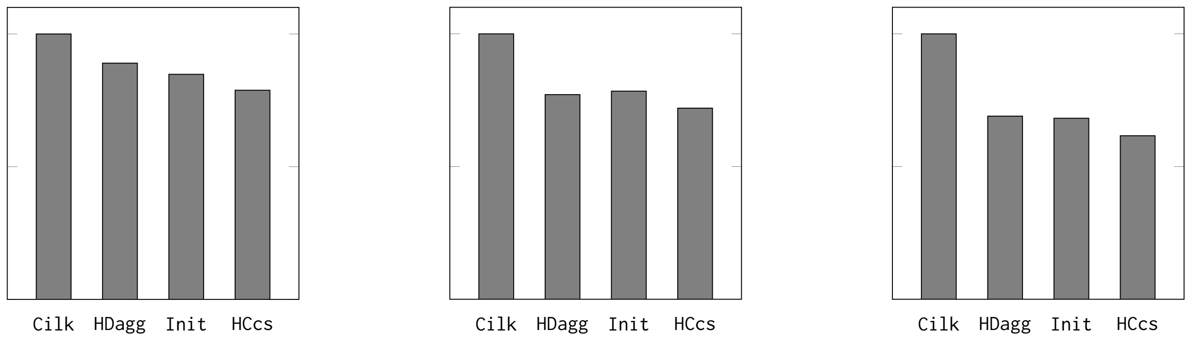
	$P = 4$	$P = 8$	$P = 16$	$P = 4$	$P = 8$	$P = 16$	$P = 4$	$P = 8$	$P = 16$
tiny	41%/34%	33%/28%	20%/16%	49%/43%	40%/36%	28%/26%	54%/49%	30%/36%	33%/32%
small	33%/23%	41%/25%	39%/20%	40%/28%	46%/31%	46%/30%	43%/30%	46%/32%	49%/35%
medium	31%/14%	43%/17%	53%/20%	38%/16%	47%/20%	56%/27%	42%/18%	47%/20%	58%/31%
large	27%/ 9%	41%/13%	53%/16%	34%/ 8%	46%/12%	56%/21%	38%/ 7%	46%/12%	58%/13%

	BL-EST	ETF	Cilk	HDagg	Init	HCcs	ILPpart	ILPcs
tiny	1.126	0.883	1	0.943	0.728	0.619	0.57	0.569
small	1.54	1.073	1	0.791	0.66	0.579	0.556	0.539
medium	1.896	1.254	1	0.658	0.592	0.542	0.529	0.506
large	2.142	1.517	1	0.609	0.591	0.547	0.542	0.521

$P = 8$	33%	31%	32%
$P = 16$	22%	27%	28%

$\ell = 2$	$\ell = 5$	$\ell = 10$	$\ell = 20$
38% / 16%	43% / 17%	50% / 19%	58% / 21%

margin.



C.4 Experiments with NUMA

tree, $P = 4$ gives a very shallow hierarchy with only two levels. Similarly, we only considered $a = 1$, since with $P = 16$ and $\Lambda = 4$.

reduction compared to `Cilk` (for $P = 4, 8, 16$, respectively). The local search methods then further improve this by 7%, 8% and 10%

these much larger DAGs, without applying our ILP-based methods,

`Init+HC+HCcs` provides an improvement of 11%, 7% and 11% (for

	$\Delta = 2$	$\Delta = 3$	$\Delta = 4$	$\Delta = 2$	$\Delta = 3$	$\Delta = 4$
tiny	43% / 39%	57% / 54%	66% / 64%	45% / 45%	68% / 68%	77% / 78%
small	48% / 31%	55% / 40%	60% / 47%	55% / 38%	66% / 52%	71% / 59%
medium	50% / 23%	55% / 30%	58% / 35%	61% / 34%	67% / 44%	69% / 49%
large	49% / 14%	54% / 18%	57% / 20%	61% / 28%	67% / 38%	69% / 42%

$P = 4$	15% / 9%	22% / 12%	26% / 13%
$P = 8$	24% / 7%	30% / 6 %	30% / 7%
$P = 16$	35% / 9%	39% / 11%	41% / 13%

$P = 8$	30% / 7%	34% / 7%	37% / 7%
$P = 16$	41% / 12%	45% / 16%	48% / 21%

Our multilevel algorithm answers this question, demonstrating

with NUMA costs ($P \in \{8, 16\}$, $\Delta \in \{2, 3, 4\}$), there were as many as

could not find a solution with lower than trivial cost; however, with

cases was only 8 out of 396. These numbers are understood over all datasets except tiny (where multilevel scheduling was not applied,

algorithm (separately for $P \in \{8, 16\}$, $\Delta \in \{2, 3, 4\}$) are shown in

approach clearly remains inferior when $\Delta = 2$, it is approximately equally strong when $\Delta = 2$ and $P = 8$, and is clearly superior when

and $g = 1$ (still understood over all datasets except tiny). While

itself (intuitively, to decide if coarsification is even necessary, or in

	$\Delta = 2$	$\Delta = 3$	$\Delta = 4$	$\Delta = 2$	$\Delta = 3$	$\Delta = 4$

	$\Delta = 2$	$\Delta = 3$	$\Delta = 4$	$\Delta = 2$	$\Delta = 3$	$\Delta = 4$
C_{15}	1.353	1.136	0.912	1.291	0.813	0.506
C_{30}	1.195	1.014	0.871	1.141	0.774	0.502
C_{opt}	1.179	0.979	0.812	1.122	0.711	0.429