Lower bounds on precedence-constrained scheduling for parallel processors

Ivan D. Baev and Waleed M. Meleis

Department of Electrical and Computer Engineering, Northeastern University, Boston MA, USA 02115

Alexandre Eichenberger

Department of Electrical and Computer Engineering, NC State University, Raleigh NC, USA 27695

Abstract

We consider two general precedence-constrained scheduling problems that have wide applicability in the areas of parallel processing, high performance compiling, and digital system synthesis. These problems are intractable so it is important to be able to compute tight bounds on their solutions. A tight lower bound on makespan scheduling can be obtained by replacing precedence constraints with release and due dates, giving a problem that can be efficiently solved. We demonstrate that recursively applying this approach yields a bound that is provably tighter than other known bounds, and experimentally shown to achieve the optimal value at least 90.3% of the time over a synthetic benchmark.

We compute the best known lower bound on weighted completion time scheduling by applying the recent discovery of a new algorithm for solving a related scheduling problem. Experiments show that this bound significantly outperforms the linear programming-based bound. We have therefore demonstrated that combinatorial algorithms can be a valuable alternative to linear programming for computing tight bounds on large scheduling problems.

Key words: Scheduling, Parallel processing, Lower bounds

1 Introduction

Precedence-constrained scheduling problems on parallel processors are often intractable and combinatorial lower bounds can be used to guide solvers and evaluate heuristics. These approaches are particularly useful for engineering applications such as parallel processing, digital system synthesis, and

Preprint submitted to Elsevier Preprint 1 October 2001

high performance compiling that need to quickly estimate required computational resources [7,11,6]. We consider two general precedence-constrained job scheduling problems on parallel processors: minimizing the maximum completion time (makespan), and minimizing the total weighted completion time (WCT).

The *makespan* scheduling problem is to schedule n unit latency jobs under p precedence constraints on m identical parallel processors so as to minimize the maximum job completion time. The completion time of job i in a schedule is denoted C_i. A directed acyclic graph (DAG) describes the precedence constraints among the jobs such that an edge from job i to job j in the graph implies that $C_i < C_j$. In a feasible schedule, no more than one job executes on any processor at any time, each job is scheduled nonpreemptively, and the precedence constraints are satisfied. The goal of the makespan problem is to find a feasible schedule of the n jobs that minimizes $C_{max} = \max_{i=1...n}\{C_i\}$. The problem is NP-hard for arbitrary m [14], but efficient solutions exist for tree-structured precedence constraints [9] or when $m = 2$ [5].

A natural extension of the makespan problem is the *weighted completion time* scheduling problem, where in addition each job i has a positive weight w_i that expresses the importance of that job. The objective here is to minimize the total weighted job completion time, i.e. under the same constraints described above for the makespan problem, the WCT problem finds a feasible schedule that minimizes $\sum_{i=1}^{n} w_i C_i$. The problem is NP-hard even for $m = 2$ and the empty precedence graph [4].

This paper analytically establishes the tightness of several recently proposed bounds on the makespan problem relative to well-known bounds. A tight lower bound can be computed by replacing precedence constraints with release and due dates, giving a problem that can be efficiently solved. We demonstrate that recursively applying this approach yields a bound that is provably better than other known bounds, and experimentally shown to achieve the optimal value at least 90.3% of the time over a synthetic benchmark.

Unlike the makespan problem, most relaxations of the weighted completion time problem are NP-hard, so tight combinatorial bounds for this problem have not been available. We compute the best known lower bound by applying the recent discovery of a new algorithm for solving a related scheduling problem. Experiments show that this bound significantly outperforms the linear programming-based bound. We have therefore demonstrated that combinatorial algorithms can be a valuable alternative to linear programming for computing tight bounds on large scheduling problems.

The organization of our paper is as follows. We describe four makespan bounds in the next section, and prove their relative tightness in Section 3. Section 4 presents two new combinatorial WCT bounds; the six bounds are experimentally evaluated in Section 5.

2 Bounds on makespan

In the following discussion we let the release date $r_i = 0$ for any job i with no predecessors, and for all other jobs $r_i = \max\{r_j\} + 1$ over all immediate predecessors j of i. For any job i, $r_i + 1$ is a lower bound on the completion time C_i in any schedule. The length of the critical path $CP = \max_{i=1...n}\{r_i\} + 1$ represents the simplest lower bound on C_{max} (*CP* bound). We let the due date $d_i = CP$ for any job i with no successors, and for all other jobs $d_i = \min\{d_j\} - 1$ over all immediate successors j of i.

Hu bound (Hu): A simple bound was described by Hu [9]. For any integer k, $0 \leq k \leq CP$, we let $n(k)$ be the number of jobs with $d_i \leq k$. A lower bound on the completion time of these jobs is $\lceil n(k)/m \rceil$. An additional time of at least $CP - k$ is needed to schedule the remaining jobs, so a lower bound on the completion time of all jobs is therefore $\max_{k=0...CP}\{\lceil n(k)/m \rceil + CP - k\}$. Note that this expression, for $k = 0$, includes the *CP* bound. The *Hu* bound can be computed in $O(n + p)$ time.

The next two bounds replace the precedence constraints with appropriate release dates and due dates so that the relaxed scheduling problem can be efficiently solved.

Rim & Jain bound (RJ): Rim and Jain observe that tight lower bounds on the solution to the makespan scheduling problem can be computed by minimizing the maximum lateness of any job in the relaxed problem [12]. Since $C_{max} - CP \geq \max_{i=1...n}\{C_i - d_i\}$, minimizing the maximum lateness of any job in this relaxed problem and adding CP gives a lower bound on C_{max} in the original problem.

The relaxed problem can be solved by list scheduling, with the priority of job i set to d_i, such that $C_i \geq r_i + 1$ [13]. That is, jobs are considered in order of nondecreasing due dates, and each job i is scheduled at the earliest time greater than or equal to its release date such that the resource constraints are satisfied. The lower bound is then equal to the maximum of $CP + (C_i - d_i)$ over all jobs i. Note that the bound is not simply C_{max} for the relaxed problem. The *RJ* bound can be computed in $O(n^2)$ time.

Langevin & Cerny bound (LC): A better bound on the completion time of a job can be computed by recursively using the *RC* bound to find tighter lower bounds on the earliest completion time of all preceding jobs [10]. That is, the release dates that were originally computed by finding longest paths in the precedence graph can themselves be computed using the *RJ* algorithm.

The lower bound on the start time of job i, r_i, used by the *RJ* bound is tightened by applying the same technique to a subset of the original problem. We let r'_i be this tighter bound for each job i. We let $r'_i = 0$ for any job i with no predecessors, as before. For all other jobs we let r'_i equal one less than the *RJ* lower bound computed for the subproblem consisting of i and all its predecessors and the tightened release dates r'_i (we subtract one to account for the unit latency). The value of r'_i is computed for job i only after r' has been computed for all predecessors of i. Since the LC relaxation is more constrained than the RJ relaxation, LC is a tighter bound. The LC bound can be computed in $O(n^3)$ time.

Brucker, Garey & Johnson bound (Br): Another relaxation of the makespan problem is obtained by deleting edges of the precedence graph so that the remaining edges form an intree. We again minimize the maximum lateness of any job.

The relaxed tree scheduling problem can be efficiently solved by list scheduling [3]. The priority of job i is set to d_i calculated from the original DAG precedence graph, but the schedule satisfies only the tree precedence constraints. While there are many ways to extract a tree from a DAG, in this paper we only consider a tree formed by connecting each job to a single successor with the smallest due date. We refer to a tree constructed in this way as a Brucker tree. Note that the due date for each job is unchanged in this tree. The lower bound is then equal to the maximum lateness of any job, plus CP. The Br bound can be computed in $O(n + p)$ time.

3 Relationships between bounds

We first show that the Hu and Br bounds are equal, and then show that the Hu bound is less than or equal to the RJ bound for any instance.

Lemma 1 . *For any instance, $Hu = Br$.*

PROOF. We first show that $Br \geq Hu$. We apply the Br algorithm and find the corresponding schedule. Let k be some integer in $[1 \ldots CP]$ and let j be the job with the largest completion time among all jobs such that $d_i = k$. Clearly $C_j - d_j \geq C_i - d_i$ for all jobs i such that $d_i = k$, so it suffices to show that $(C_j - d_j) + CP \geq \lceil n(k)/m \rceil + CP - k$. This is true because $C_j \geq \lceil n(k)/m \rceil$ and $d_j = k$. The same argument applies for all values of k, so $Br \geq Hu$.

We now show that $Br \leq Hu$. The Hu bound is the same for the original graph and a Brucker tree because the due dates d_i are unchanged and therefore the values of $n(k)$ are unchanged. The same is true for the Br bound. But for a Brucker tree, which is an intree, the Hu lower bound is in fact the optimal value for the problem [9]. Therefore $Br \leq Hu$.

Lemma 2 *For any instance, $Hu \leq RJ$.*

PROOF. It suffices to show that $\lceil n(k)/m \rceil + CP - k \leq RJ$ for all integers $k \in [0 \ldots CP]$. We let RJ' equal the bound computed by the Rim and Jain algorithm when applied to the following relaxed scheduling problem: delete all jobs i with $d_i > k$, set $d_i = k$ for the remaining jobs, set $r_i = 0$ and leave CP unchanged. Notice that this new scheduling problem is a relaxation of the scheduling problem solved by the original Rim and Jain algorithm, so

$RJ' \leq RJ$. When applied to this new problem, the Rim and Jain algorithm schedules all the jobs as early as possible with a maximum completion time equal to $\lceil n(k)/m \rceil$. Then the bound RJ' is equal to the maximum lateness plus the critical path, so $\lceil n(k)/m \rceil - k + CP = RJ' \leq RJ$ for all k.

Therefore we have established that for any instance, $CP \leq Hu = Br \leq RJ \leq LC$.

4 Bounds on weighted completion time

There are several integer linear formulations of the WCT scheduling problem [8,1]. The corresponding LP-relaxations give tight lower bounds. However, their computation involves solving a linear program, and therefore requires substantial time which is not appropriate for many applications. In this section we present two fast combinatorial algorithms for computing a lower bound on the weighted completion time. The quality of our first bound is close to that of the LP-based bound, while the second one outperforms the LP-based bound.

The first group of algorithms uses the makespan lower bounds described in Section 2. By applying any of these algorithms to a job and its predecessors, a lower bound on the completion time of that job can be computed. The weighted sum of these values over all jobs then gives a lower bound on the weighted completion time of all jobs. We restrict our study to the LC makespan bound since we have shown it is analytically better than the other bounds.

Langevin & Cerny weighted completion time bound (WCT-LC): First we use the LC algorithm to compute a lower bound on makespan for each job. Multiplying the bound by the job weight and summing over all jobs then gives a lower bound on the weighted completion time of all jobs. The *WCT-LC* bound can be computed in $O(n^3)$ time.

Our second algorithm uses a relaxation of the WCT problem which does not rely on an LP formulation. In general, one deletes a constraint from the original problem and uses a polynomial algorithm to find an optimal solution of the relaxed problem; the cost of this optimal solution represents a lower bound on the initial problem. Unfortunately, most of the WCT problem relaxations along the natural dimensions are also NP-hard. One exception is the WCT problem with release dates but without precedence constraints for which Baptiste recently discovered a polynomial-time algorithm [2].

We outline the two key observations behind Baptiste's algorithm. First, the times at which jobs start and end in the optimal schedule belong to the set of release dates and their multiples (up to a factor of n). Next, a resource profile ξ is defined as a vector $(\xi_1, \xi_2, \ldots, \xi_m)$ such that $\xi_1 \leq \xi_2 \leq \cdots \leq \xi_m$ and $\xi_m - \xi_1 \leq 1$. Each entry ξ_i in the resource profile divides the available execution times for processor i. Given two resource profiles ξ and ξ', $\xi \ll \xi'$ denotes that for any index i in $\{1, \ldots, m\}$, $\xi_i \leq \xi'_i$. Baptiste proves that an optimal partial schedule for the first k jobs between any two resource profiles

$\xi \ll \xi'$ can be found recursively from optimal partial schedules for the first $k-1$ jobs between all resource profiles θ and θ' such that $\xi \ll \theta \ll \theta' \ll \xi'$. While this gives a dynamic programming algorithm whose time and space complexities are respectively $O(n^{3m+4})$ and $O(n^{2m+2})$, we show below that the actual runtime and memory usage are much lower.

Dynamic programming bound (DP): First we apply the *LC* algorithm to find tight release dates for the jobs. Then we use a version of Baptiste's algorithm where the times at which jobs start and end in the optimal schedule belong to $\{0, 1, \ldots, n\}$.

We compare our two lower bounds on weighted completion time to an LP-based lower bound. The following time-indexed LP formulation is used.

$$\text{Minimize} \sum_{i=1}^{n} \sum_{t=1}^{T} w_i \cdot x_{i,t} \cdot t,$$

subject to

$$\begin{aligned}
\sum_{t=1}^{T} x_{i,t} &= 1, && (i = 1 \ldots n) \\
\sum_{i=1}^{n} x_{i,t} &\leq m, && (t = 1 \ldots T) \\
\sum_{t=1}^{T} (x_{i,t} \cdot t) &\leq \sum_{t=1}^{T} (x_{j,t} \cdot t) - 1, && ((i,j) \in R) \\
x_{i,t} &\in \{0, 1\}, && (i = 1 \ldots n,\ t = 1 \ldots T).
\end{aligned}$$

The binary variables $x_{i,t}$ indicate whether job i completes at time t. The first two constraints ensure that every job completes exactly once and that no more than m jobs are executing at any time. There is a pair (i, j) in R for each precedence edge from job i to job j, and for each such pair the third constraint ensures that the completion time of i is before the completion time of j. The formulation minimizes the weighted sum of job completion times. We compute a lower bound on the optimal value using a relaxation of this formulation where the variables $x_{i,t}$ can take on any values between 0 and 1.

5 Experiments

In this section we present a series of experiments that evaluate the quality of the six makespan and WCT bounds.

We evaluate the lower bounds for the makespan problem by constructing a set of 100 hard synthetic instances. The instances have an average of 1500 jobs each, and the number of processors m varies from 3 to 8. We compare all lower bounds to the upper bound computed by the critical path list scheduling algorithm [9]. The upper bound is computed by applying list scheduling to each

Table 1
Quality of lower bounds vs. upper bound: average absolute difference / % instances equal

m	*Density*	*Hu, Br*	*RJ*	*LC*
3	6.5%	13.60 / 0%	9.18 / 1%	0.38 / 70%
4	5.0%	8.29 / 11%	3.31 / 27%	0.20 / 81%
5	4.0%	3.95 / 28%	0.58 / 77%	0.03 / 97%
6	3.0%	2.50 / 44%	0.26 / 88%	0.02 / 98%
7	2.5%	2.00 / 59%	0.14 / 92%	0.03 / 97%
8	2.0%	1.05 / 74%	0.02 / 98%	0.01 / 99%
Average:		5.23 / 36%	2.25 / 64%	0.11 / 90%

instance with the priority of job i set equal to r_i. The critical path upper bound is then equal to the maximum completion time of any job in this schedule. Note that this critical path upper bound is different from the CP lower bound described in Section 2.

In our first experiment, we use graph edge densities to construct hard instances that expose the worst-case behavior of the bound algorithms. The edge density of a graph is the probability that any edge is present in the instance. For each fixed number of processors m, we select the edge density that maximizes sum of the absolute differences between the lower bounds and the critical path upper bound. We considered a range of edge densities between 1% and 10% in increments of 0.5%. The results in Table 1 give the average absolute difference between each lower bound and the upper bound, the percentage of the instances for which the lower bound equals the upper bound, and the edge density that gave the results indicated. The average runtime of the algorithms was 0.28 sec, 0.46 sec, and 313.81 sec, for the *Hu* bound, *RJ* bound, and *LC* bound, respectively. These experiments were run on a 333 MHz Sun Ultra 10 workstation with 256 MB of memory.

The results indicate that relaxing precedence constraints into release dates gives tight lower bounds when combined with an algorithm that minimizes maximum lateness. Even better bounds can be computed by recursively tightening release dates: the LC lower bound is on average only 0.11 from the upper bound.

We evaluate the WCT-LC bound by constructing two sets of 20 synthetic instances. The instances in the first set have 250 jobs each (small instances), and the instances in the second set have 350 jobs each (large instances). In each case, the weight of each job is uniformly distributed in $[1, 100]$, and m varies from 2 to 10. The results in Tables 2 and 3 demonstrate the high quality of the combinatorial WCT-LC bound. The WCT-LC bound, which can be efficiently

Table 2
WCT-LC vs. *LP* lower bounds, small instances

m	*WCT-LC*	*LP*	*difference*
2	640,137	751,458	14.81%
3	435,905	506,052	13.86%
4	336,969	383,379	12.11%
5	284,379	310,292	8.35%
6	265,154	271,938	2.49%
7	263,161	264,136	0.37%
8	263,156	263,483	0.12%
9	263,156	263,274	0.04%
10	263,155	263,195	0.02%
Average:			5.80%

Table 3
WCT-LC vs. *LP* lower bounds, large instances

m	*WCT-LC*	*LP*	*difference*
2	1,344,833	1,501,396	10.43%
3	908,919	1,009,132	9.93%
4	694,946	763,031	8.92%
5	573,577	615,439	6.80%
6	510,862	525,063	2.70%
7	492,231	493,627	0.28%
8	490,479	490,887	0.08%
9	490,479	490,579	0.02%
10	490,479	490,498	0.00%
Average:			4.38%

computed, is on average only 5.80% from the *LP* bound on the small instances, and 4.38% from the *LP* bound on the large instances. The average runtime of the *WCT-LC* bound was 2.37 sec for the small instances and 6.00 sec for the large instances on the same platform described above.

Table 4
DP vs. *LP* lower bounds

Density	*LP*	*DP*	*difference*
10%	10,261	11,187	9.0%
30%	11,010	11,285	2.5%
50%	13,006	13,345	2.6%

We next evaluate the *DP* bound against the *LP* bound on two processors. We use a set of 100 instances each with 90 jobs, and the weight of each job is uniformly distributed in $[1, 10]$. Here, we also vary the edge density. The average runtime for these instances is 12.90 seconds. These experiments were run on a 450 MHz Intel Pentium II Xeon MP system with 2GB of memory.

The results in Table 4 show that the *DP* bound clearly outperforms the *LP* bound across a variety of edge densities. While the *DP* algorithm has an unappealing asymptotic time and space complexity, efficient implementations are possible for small values of m.

6 Conclusion

While our work here considers scheduling problems with unit latency jobs, the same algorithms can be applied to non-unit latency problems by first preprocessing the jobs. Each job is replaced by a chain of unit latency jobs with the same total latency. The solution to the modified problem yields a lower bound for the non-unit latency problem.

This paper analytically and experimentally evaluated a group of algorithms for computing lower bounds on the makespan and WCT scheduling problems. We demonstrated that combinatorial algorithms can be a valuable alternative to linear programming for computing tight bounds on large scheduling problems.

References

[1] I. Baev, W. Meleis, and A. Eichenberger. Algorithms for total weighted completion time scheduling. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 852–853, Baltimore, Maryland, 1999.

[2] P. Baptiste. Scheduling equal-length jobs on identical parallel machines. *Research Report UTC*, 1998.

[3] P. Brucker, M. Garey, and D. Johnson. Scheduling equal-length tasks under treelike precedence constraints to minimize maximum lateness. *Mathematics of Operations Research*, 2:275–284, 1977.

[4] J. Bruno, E. G. Coffman Jr, and R. Sethi. Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, 17:382–387, 1974.

[5] E. Coffman and R. Graham. Optimal scheduling for two-processor systems. *Acta Inform.*, 1:200–213, 1972.

[6] A. Eichenberger and W. M. Meleis. Balance scheduling: Weighting branch tradeoffs in superblocks. In *32nd Annual International Symposium on Microarchitecture (IEEE/ACM)*, pages 272–283, Haifa, Israel, 1999.

[7] E. Fernandez and B. Bussell. Bounds on the number of processors and time for multiprocessor optimal schedules. *IEEE Trans. on Computers*, pages 745–751, 1973.

[8] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22:513–549, 1997.

[9] T. Hu. Parallel sequencing and assembly line problems. *Operations Research*, 9:841–848, 1961.

[10] M. Langevin and E. Cerny. A recursive technique for computing lower-bound performance of schedules. *ACM Trans. on Design Automation*, 1:443–455, 1996.

[11] R. Rabaey and M. Potkonjak. Estimating implementation bounds for real time DSP application specific circuits. *IEEE Trans. on CAD*, 13:669–683, 1994.

[12] M. Rim and R. Jain. Lower-bound performance estimation for the high-level synthesis scheduling problem. *IEEE Trans. on CAD*, 13:452–459, 1994.

[13] B. Simons. Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines. *SIAM Journal on Computing*, 12:294–299, 1983.

[14] J. D. Ullman. NP-complete scheduling problems. *Journal of Computing Systems and Sciences*, 10:384–393, 1975.