Precedence Constrained Scheduling to Minimize Weighted Completion Time on a Single Machine

Chandra Chekuri[*] Rajeev Motwani [†]

Dept. of Computer Science
Stanford University

Abstract

We consider the problem of scheduling a set of jobs on a single machine with the objective of mimizing weighted (average) completion time. The problem is NP-hard when there are precedence constraints between jobs, [12] and we provide a simple and efficient combinatorial 2-approximation algorithm. In contrast to our work, earlier approximation algorithms [9] achieving the same ratio are based on solving a linear programming relaxation of the problem.

1 Introduction

We consider the problem of scheduling a set of jobs in the presence of precedence constraints with the objective of minimizing the weighted completion time. Specifically, given a set of jobs $\{J_1, J_2, \ldots, J_n\}$ with execution times p_i and weights w_i to be scheduled on a single machine, find a *non-preemptive* schedule S (or equivalently an ordering) to minimize $\sum_i w_i C_i$, where C_i is the completion time of J_i in the schedule S. Precedence constraints between jobs are presented as a directed acyclic graph whose vertices correspond to the jobs, and whose edges represent the precedence constraints.

The problem is NP-hard if we permit arbitrary precedence constraints on the jobs [8, 12]. It is polynomially solvable when the precedence graph is a forest [10] or a generalized series-parallel graph [12, 1]. The best known approximation algorithm for the general DAG case until recently had a ratio of $O(\log n \log L)$ where $L = \sum_i p_i$ is the sum of the execution times of the jobs [14]. Recently, Hall et al [9] gave constant factor approximations using linear programming relaxations. It is interesting to note that several different formulations give the same bound of 2 [9]. Our analysis might give some insight as to why this is the case.

A more general version of the problem is to schedule the jobs on a set of m identical machines. For $m \geq 2$, the problem is NP-hard even without precedence constraints, unless the weights are all identical in which case it is polynomially solvable; on the other hand, the problem is strongly NP-hard even when all weights are identical and the precedence

[*]E-mail: `chekuri@cs.stanford.edu`. Supported by NSF Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

[†]E-mail: `rajeev@cs.stanford.edu`. Supported by an Alfred P. Sloan Research Fellowship, an IBM Faculty Partnership Award, an ARO MURI Grant DAAH04-96-1-0007, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

graph is a collection of chains [5]. An approximation ratio of 5.33 is achievable even if there are release times on the jobs [2]. Chekuri et al., [3] gave an algorithm which given an α approximate schedule to the single machine problem converts it to a $2\alpha + 2$ approximate schedule for the multimachine problem. This algorithm, based on a novel modification to list scheduling is combinatorial and extremely simple.

In this paper we give a simple combinatorial 2 approximation algorithm for the single machine problem which matches the best ratio achieved in [9]. The advantage of our algorithm is twofold. First, the algorithms in [9] are based on solving linear programming relaxations and our simple combinatorial algorithm is more efficient both in theory and practice. Secondly, combining this with the conversion algorithm in [3], we get a simple combinatorial algorithm for the multiple machine case as well.

2 Preliminaries and Notation

Let $G = (V, E)$ denote the precedence graph where V is the set of jobs. We will use jobs and vertices interchangeably. We say that i immediately precedes j, denoted $i \prec j$, if and only if there is an edge from i to j in the graph. A vertex i precedes a vertex j, denoted $i \overset{*}{\prec} j$, if and only if there is a path from i to j. For any vertex $i \in V$, let G_i denote the subgraph of G induced by the set of vertices preceding i.

Definition 1 *The* rank *of a job J_i denoted by r_i is defined as $r_i = p_i/w_i$. Similarly, the* rank *of a set of jobs A denoted by $r(A)$ is defined as $p(A)/w(A)$ where $p(A) = \sum_{J_i \in A} p_i$ and $w(A) = \sum_{J_i \in A} w_i$.*

Definition 2 *A subdag G' of G is said to be* precedence closed *if for every job $J_i \in G'$, G_i is a subgraph of G'.*

Definition 3 *We define G^* to be a precedence-closed subgraph of G of minimum rank, i.e., among all precedence-closed subgraphs of G, G^* is of minimum rank.*

Note that G^* could be the entire graph G.

3 A Characterization of the Optimal Schedule

Smith's rule for a set of independent jobs states that there is an optimal schedule that schedules jobs in non-decreasing order of their ranks. We show that we can generalize this rule for the case of precedence constraints in the following way. The following theorem has been discovered by Sydney in 1975 [16] but the authors rediscovered it and we present our own proof for the sake of completeness.

Theorem 1 *There exists an optimal sequential schedule where the optimal schedule for G^* occurs as a segment which starts at time zero.*

Proof: The theorem is trivially true for the case when $G*$ is the same as G. We consider the case when G^* is a proper subdag of G. Let S be an optimal schedule for G in which G^* is decomposed into a minimum number of maximal segments. Suppose that G^* is decomposed into two or more segments in S. For $k > 1$, let $A_1, A_2, \ldots, A_k$ be the segments of G^* in S, in increasing order of starting times. Let the segment between A_{i-1} and A_i be denoted by B_i. Note that B_1 could be empty in which case we assume that $p(B_1) = w(B_1) = 0$. Let $r(G^*) = \alpha$ and B^j denote the union of the blocks $B_1, B_2, \ldots, B_j$. From the definition of G^* it follows that $r(B^j) \geq \alpha$ since otherwise the union of B^j and G^* would have rank less than α. Let A^j similarly denote the union of the blocks $A_1, A_2, \ldots, A_j$. It follows that $r(A^k - A^j) \leq \alpha$ for otherwise $r(A^j) < \alpha$.

Let S' be the schedule formed from S by moving all the A_i's ahead of B_i's while preserving their order within themselves. The schedule S' is legal since G^* is precedence closed. Let Δ denote the difference in the total weighted completion times of S and S'. We will show that $\Delta \geq 0$ which will complete the proof. While comparing the two schedules, we can ignore the contribution of the jobs that come after A_k since their status remains the same in S'. Let $\Delta(A_j)$ and $\Delta(B_j)$ denote the difference in weighted completion time of B_j and A_j respectively in S and S'. It follows that $\Delta = \sum_{j \leq k} \Delta(A_j) + \Delta(B_j)$. It is easy to see that

$$\Delta(A_j) = w(A_j)p(B^j)$$

and

$$\Delta(B_j) = -w(B_j)p(A^k - A^{j-1}).$$

¿From our earlier observations on $r(B^j)$ and $r(A^k - A^j)$ we have $p(B^j) \leq \alpha w(B^j)$ and $p(A^k - A^j) \geq \alpha w(A^k - A^j)$. Therfore

$$\begin{aligned}
\Delta &= \sum_{j \leq k} \Delta(A_j) + \Delta(B_j) \\
&= \sum_{j \leq k} w(A_j)p(B^j) - \sum_{j \leq k} w(B_j)p(A^k - A^{j-1}) \\
&\geq \alpha \sum_{j \leq k} w(A_j)w(B^j) - \alpha \sum_{j \leq k} w(B_j)w(A^k - A^{j-1}) \\
&= \alpha \sum_{j \leq k} w(A_j) \sum_{i \leq j} w(B_j) - \alpha \sum_{j \leq k} w(B_j) \sum_{i \geq j} w(A_i) \\
&= 0.
\end{aligned}$$

The third inequality above follows from our observations about $r(B^j)$ and $r(A - A^j)$ and the last equality follows from a simple change in the order of summation. ■

Remark 1 *Note that when G^* is the same as G this theorem does not help in reducing the problem.*

4 A 2 approximation

Theorem 1 suggests the following natural algorithm. Given G, compute G^* and schedule G^* and $G - G^*$ recursively. Of course it is not clear that G^* can be computed in polynomial time but we will show that we can reduce this problem to solving a minimum cut problem. The second and more important problem that needs to be solved before we have an algorithm is take care of the case when G^* is same as G. Since the problem is NP-hard, it is clear that we have to settle for an approximation in this case for otherwise we would have a polynomial time algorithm to compute the optimal schedule.

The following lemma helps us in taking care of the case when $G^* = G$.

Lemma 1 *If G^* is the same as G,* OPT $\geq w(G) \cdot p(G)/2$.

Proof: Let $\alpha = r(G)$. Suppose S is an optimal schedule for G. Without loss of generality assume that the ordering of the jobs in S is $J_1, J_2, \ldots, J_n$. Observe that for any j, $1 \le j \le n$, that $C_j = \sum_{i \le j} p_i \ge \alpha \sum_{i \le j} w_i$. This is because the set of jobs $J_1, J_2, \ldots, J_j$ form a precedence closed subdag and from our assumption on G^* it follows $\sum_{i \le j} p_j / \sum_{i \le j} w_i \ge \alpha$. Using this we can bound the value of the optimal as follows.

$$\begin{aligned}
\text{OPT} &= \sum_j w_j C_j \\
&\ge \sum_j w_j \sum_{i \le j} \alpha w_i \\
&= \alpha \left(\sum_j w_j^2 + \sum_{i<j} w_i w_j \right) \\
&= \alpha \left((\sum_j w_j)^2 - \sum_{i<j} w_i w_j \right) \\
&\ge \alpha \left(w(G)^2 - w(G)^2/2 \right) \\
&= \alpha w(G)^2/2 \\
&= w(G)p(G)/2
\end{aligned}$$

■

Lemma 2 *Any feasible schedule with no idle time has a total completion time of at most $w(G) \cdot p(G)$.*

Proof: Obvious. ■

Theorem 2 *If G^* for a graph can be computed in time $O(T(n))$, then there is a 2 approximation algorithm for computing the minimum weighted completion time schedule which runs in time $O(nT(n))$.*

Proof: Given G, we compute G^* in time $O(T(n))$. If G^* is the same as G we schedule G arbitrarily and Lemmas 1 and 2 guarantee that we have a 2 approximation. If G^* is a proper subdag we recurse on $G*$ and $G - G^*$. From Theorem 1 we have $\text{OPT}(G) = \text{OPT}(G^*) + p(G^*) \cdot w(G - G^*) + \text{OPT}(G - G^*)$. Inductively if we have 2 approximations for G^* and $G - G^*$ it is clear that we have a 2 approximation of the overall schedule. To show the running time bound, we observe that the $G^{**} = G^*$. Therefore we make at most n calls to the routine to compute G^* and the bound follows. ■

All that remains is to show how to compute G^* in polynomial time.

4.1 Computing G^*

To compute G^* we consider the more general problem of finding a subdag of rank at most $\lambda > 0$ if one exists. We show how we can reduce the later problem to the problem of computing an $s - t$ mincut in an associated graph. The following defines the associated graph.

Definition 4 *Given a dag $G = (V, E)$, and a real number $\lambda > 0$, we define a capacitated directed graph $G_\lambda = (V \cup \{s, t\}, E', c)$ where the edge set E' is defined by $E' = \{(s, i), (i, t) \mid 1 \leq i \leq n\} \cup \{(i, j) \mid j \overset{*}{\prec} i\}$ and the capacities are defined by*

$$c(e) = \begin{cases} p_i & \textit{if } e = (i, t) \\ \lambda w_i & \textit{if } e = (s, i) \\ \infty & \textit{otherwise} \end{cases}$$

Lemma 3 *Given a dag G, there is a subdag of rank at most λ if and only if the $s-t$ mincut in G_λ is at most $\lambda w(G)$. If (A, B) is a cut whose value is bounded by $\lambda w(G)$, $r(A - \{s\}) \leq \lambda$ and $A - \{s\}$ is precedence closed in G.*

Proof: Let (A, B) be an $s - t$ cut in G_λ whose value is bounded by $\lambda w(G)$. We first claim that $A - \{s\}$ is precedence closed in G. Suppose not. Then there is a pair of vertices (i, j) such that $i \overset{*}{\prec} j$ and $j \in A$ and $i \notin A$. But then $c(j, i) = \infty$ which is a contradiction since $c(A, B) \leq \lambda w(G)$. From this fact and the definition of G_λ, it follows that

$$\begin{aligned} c(A, B) &= \sum_{i \in A} p_i + \sum_{i \notin A} \lambda w_i \\ &= \sum_{i \in A} (p_i - \lambda w_i) + \sum_{i \in V} \lambda w_i \\ &= \sum_{i \in A} (p_i - \lambda w_i) + \lambda w(G) \end{aligned}$$

Since $c(A, B) \leq \lambda w(G)$ it follows that $\sum_{i \in A}(p_i - \lambda w_i) \leq 0$ which implies that $r(A - \{s\}) \leq \lambda$. It is quite easy to see using similar arguments as above that a precedence closed subdag A in G whose rank is less than λ induces a cut of value at most $\lambda w(G)$ in G_λ. ∎

Lemma 4 *G^* can be computed in time $O(mn \log(n^2/m))$.*

Proof: Computing the minimum cut in G_λ for each $\lambda > 0$ can be viewed as a parametric maxflow computation. There are at most n values of λ for which the minimum cut changes in the graph and a result of Gallo et al. [6] shows that it is possible to obtain all the such values of λ in the time it takes to do one maximum flow computation using the push-relabel algorithm of Goldberg and Tarjan. The running time bound follows. ∎

The algorithm to compute G^* using a maximum flow computation is present in Lawler's book [13] where a binary search is performed to find the optimum λ. The improved strongly polynomial running time is due to the parametric flow techniques of Gallo et al. based on the push-relable algorithm of Goldberg and Tarjan.

5 Conclusions

In a recent paper, Chudak and Hochbaum [4] show a half integral linear programming relaxation for the same problem. They achieve a similar approximation ratio of 2. The half-integral program can be solved using a minimum cut computation, but the running time obtained is worse than that of our algorithm, by a factor of n. It is possible to show examples where the optimal solution to their linear programming relaxation is a factor of 2 away from the optimal integral solution. It would be interesting to show that the scheduling problem we consider is hard to approximate within some absolute constant factor (Max-SNP hard). We conjecture that this is the case.

References

[1] D. Adolphson. Single machine job sequencing with precedence constraints. *SIAM Journal on Computing*, 6:40–54 (1977).

[2] S. Chakrabarti, C.A. Phillips, A.S. Schulz, D.B. Shmoys, C. Stein, J. Wein. Improved scheduling algorithms for minsum criteria. *ICALP*, Springer Verlag (1996).

[3] C. Chekuri, R. Motwani, B. Natarajan and C. Stein. Approximation Techniques for Average Completion Time Scheduling. *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1997.

[4] F. Chudak and D. Hochbaum. A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. *(submitted)*, August 1997.

[5] J. Du, J.Y.T. Leung, and G.H. Young. Scheduling chain structured tasks to minimize makespan and mean flow time. *Information and Computation*, 92:219–236 (1991).

[6] G. Gallo, M.D. Grigoriadis, and R. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J. on Comput.*, 18:30–55, 1989.

[7] M.R. Garey. Optimal task sequencing with precedence constraints. *Discrete Mathematics*, 4:37–56 (1973).

[8] M.R. Garey and D.S. Johnson. **Computers and Intractability: A Guide to the Theory of NP-completeness,** Freeman, San Francisco (1979).

[9] L.A. Hall, A.S. Schulz, D.B. Shmoys, J. Wein. Scheduling to minimize average completion time: Offline and online algorithms. *Journal submission.*

[10] W.A. Horn. Single-machine job sequencing with treelike precedence ordering and linear delay penalties. *SIAM Journal of Applied Mathematics*, 23:189–202 (1972).

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

[12] E.L. Lawler. Sequencing jobs to minimize total weighted completion time. *Annals of Discrete Mathematics*, 2:75–90 (1978).

[13] E.L. Lawler. **Combinatorial Optimization**. Holt, Rinehart, and Winston (1976).

[14] R. Ravi, A. Agrawal, P. Klein. Ordering problems approximated: single-processor scheduling and interval graph completion. *ICALP*, Springer Verlag, (1991).

[15] W. Smith. Various optimizers for single-stage production. *Naval Res. Logist. Quart.*, 3:59–66 (1956).

[16] J. Sydney. Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs. *Operations Research*, 23(2):283–298 (1975).