Combinatorial Algorithms for Minimizing the Weighted Sum of Completion Times on a Single Machine ⋆

James M. Davis[1], Rajiv Gandhi[2], and Vijay Kothari

[1] Department of Computer Science, Rutgers University-Camden, Camden, NJ 08102. `jamesdav@camden.rutgers.edu`.

[2] Department of Computer Science, Rutgers University-Camden, Camden, NJ 08102. `rajivg@camden.rutgers.edu`.

Abstract. In this paper we study the problem of minimizing the weighted sum of completion times of jobs with release dates on a single machine. We develop two algorithms that rely on "the simplest [linear program] relaxation" [8]. For the first algorithm we consider the online setting where we gain knowledge of a job on its release date and produce a schedule as the machine processes jobs. We develop an online dual fitting algorithm with an approximation guarantee of 3. This is the first online algorithm to use this LP as a lower bound. For the second algorithm we work in the off-line setting and develop a primal-dual algorithm with an approximation guarantee of 2.42. This algorithm provides the current best upper bound on the integrality gap of this simple LP formulation.

1 Introduction

Scheduling problems involve scheduling jobs on machines to optimize some objective function. Many scheduling problems have been studied in the literature, each defined by different objective functions and different constraints on admissible schedules. For an overview of these problems we refer the reader to the surveys of Queyranne and Schulz [8], Graham et al. [1] and Chekuri and Khanna [16].

In this paper we study the problem of minimizing the weighted sum of completion times on a single machine with release dates, denoted by $1|r_j|\sum_j w_jC_j$ [1]. In this problem we are given a set of n jobs $J = \{1, 2, \ldots, n\}$, each with a processing time $p_j > 0$, weight $w_j \geq 0$ and release date $r_j \geq 0$. Our aim is to schedule these jobs non-preemptively on a single machine to minimize $\sum_j w_jC_j$, where C_j denotes the completion time of job j in the schedule. This problem is NP-hard, even if the weight of every job is 1 [2]. When all release dates are 0 the problem is solved optimally by using Smith's rule.

A significant amount of work has been done on this problem. Polynomial time approximation schemes have been discovered [3, 16]. There are also several linear programming based approximations [14, 13, 17–19, 9, 12]. The linear programming based techniques derive primarily from three different LP formulations (discussed in detail in [8]): the completion time LP (LP1), the completion time LP with shifted parallel inequalities (LP2), and the preemptive time indexed LP (LP3). Most algorithms use LP2 and LP3 [14, 13, 17–19]. Goemans et al. [14] study these two LP formulations in detail, show their equivalence and present a LP-rounding algorithm with an approximation guarantee of 1.6853. LP1 is not so well studied. Schulz [12] studies LP1 and derives a constant factor approximation using it. Hall et al. [9] improves on this result and derives a 3 approximation algorithm using LP1, the current best guarantee for the problem using this LP. Like most other algorithms, the algorithms of Schulz and Hall et al. use the technique of LP rounding.

⋆ Research supported by Rutgers University Research Council Grant and by NSF award CCF-0830569

Our work focuses on understanding LP1; we provide two combinatorial algorithms who's approximation guarantees give an upper bound on LP1's integrality gap. The first algorithm is an online dual-fitting algorithm that achieves an approximation guarantee of 3. In the online setting we gain knowledge of a job on its release date and for each time t we must construct the schedule until time t without any knowledge of jobs that are released afterwards [9]. To the best of our knowledge this is the first online algorithm that uses LP1; hence it is the first online algorithm to give an upper bound on the integrality gap of LP1. The second algorithm is a primal-dual algorithm that yields an approximation guarantee of $1+\sqrt{2}$ (≈ 2.42), improving on the result of Hall et al. [9]. The second algorithm gives the current best upper bound on the integrality gap of LP1. Both algorithms are simple to implement and run in $O(n\log(n))$ time.

2 Linear Program Formulation

Several linear programming relaxations for the problem are well known [8]. The linear programming relaxation we use was first studied extensively by Schulz [12].

For a job j let C_j represent its completion time. For any set $S \subseteq J$ let $p(S) = \sum_{j\in S} p_j$ and $p^2(S) = \sum_{j\in S} p_j^2$. The completion time linear programming formulation is given by

$$\begin{aligned} \min \sum_{j\in J} w_j C_j & & \\ \text{subject to} \qquad C_j &\geq r_j + p_j, & \forall j \in J \\ \sum_{j\in S} p_j C_j &\geq \frac{p(S)^2 + p^2(S)}{2}, & \forall S \subseteq J \\ C_j &\geq 0, & \forall j \in J \end{aligned}$$

The justification for the second constraint is as follows. By the problem definition no two jobs can be scheduled at the same time. Consider any schedule for the jobs in $S \subseteq J$. Assume w.l.o.g. that the jobs are ordered by their completion time. If we set $r_j = 0$ for all jobs, then there is no time the machine is not processing a job. In this case we get that $C_j = \sum_{k=1}^{j} p_k$ and using algebra

$$\sum_{j=1}^{|S|} p_j C_j \geq \sum_{j=1}^{|S|} p_j \sum_{k=1}^{j} p_k = \sum_{j=1}^{|S|} \sum_{k=1}^{j} p_j p_k = \frac{P(S)^2 + p^2(S)}{2}.$$

Combining this with the fact that $\sum_{j=1}^{|S|} p_j C_j$ can only be greater when there are non-zero release times gives us the constraint.

The dual linear program is given by

$$\begin{aligned} \max \sum_{j\in J} \alpha_j (r_j + p_j) + \sum_{S\subseteq J} \beta_S \Big(\frac{p(S)^2 + p^2(S)}{2}\Big) & & \\ \text{subject to} \qquad \alpha_j + p_j \sum_{S \,\mid\, j\in S} \beta_S &\leq w_j, & \forall j \in J \\ \alpha_j &\geq 0, & \forall j \in J \\ \beta_S &\geq 0, & \forall S \subseteq J \end{aligned}$$

Notice that there is a dual variable α_j for every job j, a constraint for every job j, and a dual variable β_S for every subset of jobs S. The cost of any feasible dual solution is a lower bound on OPT.

3 Dual Fitting Algorithm

The first step in the algorithm below is to sort the jobs by non-increasing order of weight over processing time so that the set $J' = \{1, 2, \ldots, n\}$ of jobs satisfies $\frac{w_1}{p_1} \geq \frac{w_2}{p_2} \geq \ldots \frac{w_n}{p_n}$. The main idea behind the algorithm is that after a job is released it is forced to wait from time r_j to $r_j + p_j$, after which it is eligible to be scheduled. When a job has finished waiting we say that it is available. When the machine is free (no job is being processed), among the available jobs we select the job with the smallest index in J' to be scheduled.

Algorithm 1 Dual Fitting Algorithm

```
J' ← list of jobs sorted by non-increasing w_j/p_j
Q ← ∅
t ← 0
while J' ≠ ∅ do
  if t = r_j + p_j for some j ∈ J' then
    Q ← Q ∪ {j}
    J' ← J' − j
  end if
  if machine is not processing a job then
    if Q ≠ ∅ then
      j' ← job in Q that appears earliest in J'
      schedule j'
      Q ← Q − {j}'
    end if
  end if
end while
```

Although this algorithm runs in pseudopolynomial time it can easily be made to run in $O(n \log(n))$ time. We simply need to introduce a variable indicating when the machine will finish processing the current job, say s, and increment time steps by $\min\{\min_{j \in J'} r_j + p_j, s\}$.

Notice that this dual fitting algorithm is an online algorithm. At any time t we have only considered jobs with release times less than t to be scheduled. Also, at any time t the schedule before t cannot be altered. What remains to be analyzed is the performance guarantee achieved by this algorithm.

3.1 Analysis for Dual Fitting Algorithm

To analyze the performance guarantee we first construct a dual infeasible solution. We let S_j denote the first j jobs, $\{1, 2 \ldots j\}$, after the jobs have been sorted by non-increasing $\frac{w_j}{p_j}$ value. For convenience we let β_j denote β_{S_j}.

$$\begin{aligned} \alpha_j &= w_j, && \forall j \in J \\ \beta_j &= \frac{w_j}{p_j} - \sum_{k>j} \beta_k, && \text{for } j = n \text{ down to } 1 \\ \beta_S &= 0, && \forall S \mid \forall i \in J, S \neq S_i \end{aligned}$$

This infeasible solution is not a lower bound on OPT. We can, however, scale the values of the variables in the infeasible solution to create a feasible solution. This is formalized in Lemma 1.

Lemma 1. $\frac{2}{3}\sum_{j\in J}\alpha_j(r_j+p_j) + \frac{1}{3}\sum_{k\in J}\beta_k\Big(\frac{p(S_k)^2+p^2(S_k)}{2}\Big) \leq OPT$

Proof. We can verify that in the infeasible solution above, the nonzero α_j variables constitute a feasible dual solution. Similarly the set of nonzero β_S variables constitute a feasible solution. Since any convex combination of two feasible solutions yields a feasible solution, by scaling the α_j variables by a factor of $\frac{2}{3}$ and the β_S variables by a factor of $\frac{1}{3}$ we obtain a new feasible solution. Since any dual feasible solution is a lower bound on OPT the claim follows.

We will need two key lemmas to relate the cost of our solution to the dual feasible solution described in Lemma 1.

Lemma 2. *For any job j we have that $w_j = \alpha_j$ and $\frac{w_j}{p_j} = \sum_{i\geq j}\beta_i$.*

Proof. This follows directly from the construction of the dual infeasible solution.

Lemma 3. *The completion time, C_j, of any job j in our schedule satisfies $C_j \leq 2(r_j+p_j)+p(S_j)$.*

Proof. Note that job j is eligible for processing at time $r_j + p_j$. Let's first assume that no other job is being processed at $r_j + p_j$. After j has completed its idle time the most that j will have to wait before it begins processing is the amount of time it takes for j to be emptied from Q, which is at most $p(S_j)$. This implies that $C_j \leq r_j + p_j + p(S_j)$.

If at time $r_j + p_j$ another job k is processing, then job k must have already completed its idling period, so that $r_k + p_k \leq r_j + p_j$. Therefore, $p_k \leq r_j + p_j$ so that k will finish processing by $r_j + p_j + p_k \leq r_j + p_j + (r_j + p_j) = 2(r_j + p_j)$. Thus, $C_j \leq 2(r_j + p_j) + p(S_j)$ as desired.

We can now bound the cost of our solution.

Theorem 1. *The dual fitting algorithm is a 3-approximation.*

Proof. We will use use Lemma 2 and Lemma 3 to rewrite the cost of our solution in terms of α_j, β_j, p_j and r_j.

$$\begin{aligned} Cost = \sum_{j\in J} C_j w_j &= 2\sum_{j\in J} w_j(r_j+p_j) + \sum_{j\in J} p(S_j)\Big(\frac{w_j}{p_j}\Big)p_j && \text{(Lemma 3)} \\ &= 2\sum_{j\in J}\alpha_j(r_j+p_j) + \sum_{j\in J} p_j \sum_{k\geq j}\beta_k p(S_j) && \text{(Lemma 2)} \\ &= 2\sum_{j\in J}\alpha_j(r_j+p_j) + \sum_{k\in J}\beta_k\Big(\frac{p(S_k)^2+p^2(S_k)}{2}\Big) \end{aligned}$$

Finally, we can use Lemma 1 to bound the solution cost.

$$
\begin{aligned}
Cost &\le 2\sum_{j\in J}\alpha_j(r_j+p_j)+\sum_{k\in J}\beta_k\Big(\frac{p(S_k)^2+p^2(S_k)}{2}\Big)\\
&=3\Big(\frac{2}{3}\sum_{j\in J}\alpha_j(r_j+p_j)+\frac{1}{3}\sum_{k\in J}\beta_k\Big(\frac{p(S_k)^2+p^2(S_k)}{2}\Big)\Big)\\
&\le 3\cdot OPT \qquad \square
\end{aligned}
$$

4 Primal-Dual Algorithm

The primal-dual algorithm is inspired by Gandhi et al. [20]. In the algorithm below a feasible schedule is built iteratively. Consider a particular iteration. Let J' be the set of jobs that aren't scheduled at the beginning of this iteration and let j be the job with the largest release time. Let κ be some constant that will be optimized later. If $r_j > \kappa \cdot p(J')$ we raise the dual variable α_j until the dual constraint for j becomes tight. We then schedule j to be processed before every previously scheduled job.

If $r_j \le \kappa \cdot p(J')$ we raise the dual variable $\beta_{J'}$ until one of the constraints becomes tight for some job $j' \in J'$. Job j' is scheduled to be processed before every previously scheduled job.

Algorithm 2 Primal-Dual

```
J' ← J
while J' ≠ ∅ do
  j ← job with largest r_j value
  if r_j > κ · p(J') then
    α_j ← w_j − p_j Σ_{S|j∈S} β_S
    J' ← J' − j
  else if r_j ≤ κ · p(J') then
    j' ← arg min_j { w_j/p_j − Σ_{S|j∈S} β_S }
    β_{J'} ← w_{j'}/p_{j'} − Σ_{S|j∈S} β_S
    J' ← J' − j'
  end if
  schedule the jobs in the reverse order that they were removed from J'
end while
```

This algorithm can be implemented in $O(n\log(n))$ time by maintaining two sorted lists of jobs: one sorted by non-increasing r_j value and the other sorted by non-increasing $\frac{w_j}{p_j}$ value. We then observe that when $r_j > \kappa \cdot p(J)$ the job with highest r_j value is removed and when $r_j \le \kappa \cdot p(J)$ the job with lowest $\frac{w_j}{p_j}$ value is removed.

4.1 Analysis for Primal-Dual Algorithm

At any time during the algorithm the nonzero variables constitute a feasible dual solution. Assume w.l.o.g. that the jobs in $J = \{1, 2, \ldots, n\}$ are indexed by their order in the schedule. That is, if j

and k are jobs with $j < k$ then j is scheduled before k. Let S_j be the set of jobs $\{1, 2, \ldots, j\}$. We let β_j denote β_{S_j} for convenience.

Lemma 4. *The following are properties of our algorithm.*

(a) Every nonzero β_S variable can be written as β_j for some job j.
(b) For every set S_j that has a nonzero β_j variable, if $i \leq j$ then $r_i \leq \kappa \cdot p(S_j)$.
(c) For every job j that has a nonzero α_j variable, $r_j > \kappa \cdot p(S_j)$.
(d) For every job j that has a nonzero α_j variable, if $i \leq j$ then $r_i \leq r_j$.

Each of these observations can easily be verified. We will also need two lemmas that relate our algorithm to the constructed feasible dual solution.

Lemma 5. *For every job j, $w_j = \alpha_j + p_j \sum_{k \geq j} \beta_k$.*

Proof. To prove the lemma we simply take note of the fact that a job j isn't removed from J' until the constraint for j becomes tight. Since all jobs are removed from J' all constraints are tight.

Lemma 6. *For every job j, $C_j \leq \max_{i \leq j}\{r_i\} + p(S_j)$.*

Proof. Let $r = \max_{i \leq j}\{r_i\}$. After time r, all jobs in S_j are released. Hence, after time r job j will take at most $p(S_j)$ additional time to complete. The lemma follows.

Theorem 2. *The algorithm above gives a $(1 + \sqrt{2})$-approximation algorithm for $1|r_j|\sum_{j \in J} w_j C_j$.*

Proof. We use Lemma 5 to rewrite the cost of our solution in terms of the dual variables.

$$\begin{aligned} Cost = \sum_{j \in J} w_j C_j &= \sum_{j \in J} (\alpha_j + p_j \sum_{k \geq j} \beta_k) C_j \\ &= \sum_{j \in J} \alpha_j C_j + \sum_{j \in J} p_j \sum_{k \geq j} \beta_k C_j \end{aligned} \tag{1}$$

We will first bound $\sum_{j \in J} \alpha_j C_j$.

$$\begin{aligned} \sum_{j \in J} \alpha_j C_j &\leq \sum_{j \in J} \alpha_j (\max_{i \leq j}\{r_i\} + p(S_j)) && \text{(Using Lemma 6)} \\ &\leq \sum_{j \in J} \alpha_j (r_j + p(S_j)) && \text{(Using (d) of Lemma 4)} \\ &< (1 + \frac{1}{\kappa}) \sum_{j \in J} \alpha_j (r_j + p_j) && \text{(2) (Using (c) of Lemma 4)} \end{aligned}$$

Now we bound $\sum_{j\in J} p_j \sum_{k\geq j} \beta_k C_j$.

$$\begin{aligned}
\sum_{j\in J} p_j \sum_{k\geq j} \beta_k C_j &\leq \sum_{j\in J} p_j \sum_{k\geq j} \beta_k \big(\max_{i\leq j}\{r_i\} + p(S_j)\big) && \text{(Using Lemma 6)} \\
&\leq \sum_{k\in J} \beta_k \sum_{j\leq k} p_j \big(\max_{i\leq k}\{r_i\} + p(S_j)\big) \\
&\leq \kappa \sum_{k\in J} \beta_k p(S_k) \sum_{j\leq k} p_j + \sum_{k\in J} \beta_k \sum_{j\leq k} p_j p(S_j) && \text{(Using (b) of Lemma 4)} \\
&= \kappa \sum_{k\in J} \beta_k p(S_k)^2 + \sum_{k\in J} \beta_k \Big(\frac{p(S_k)^2 + p^2(S_k)}{2}\Big) \\
&\leq (2\kappa+1) \sum_{k\in J} \beta_k \Big(\frac{p(S_k)^2 + p^2(S_k)}{2}\Big) && (3)
\end{aligned}$$

Combining (1), (2) and (3) we get

$$Cost \leq (1+\frac{1}{\kappa}) \sum_{j\in J} \alpha_j (r_j + p_j) + (2\kappa+1) \sum_{k\in J} \beta_k \Big(\frac{p(S_k)^2 + p^2(S_k)}{2}\Big)$$

To get the best approximation guarantee we optimize κ. κ will be optimal when $1+\frac{1}{\kappa} = 2\kappa+1$, which gives us that $\kappa = \frac{\sqrt{2}}{2}$. This lets us derive the approximation guarantee.

$$\begin{aligned}
Cost &\leq (1+\sqrt{2})\Big(\sum_{j\in J} \alpha_j (r_j+p_j) + \sum_{k\in J} \beta_k \Big(\frac{p(S_k)^2 + p^2(S_k)}{2}\Big)\Big) \\
&\leq (1+\sqrt{2}) \cdot OPT \qquad \square
\end{aligned}$$

References

1. R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, *Optimization and approximation in deterministic sequencing and scheduling: A survey*, Ann. Discrete Math., 5 (1979), pp. 287326.
2. J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker, *Complexity of machine scheduling problems*, Ann. Discrete Math., 1 (1977), pp. 343362.
3. F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko, *Approximation schemes for minimizing average weighted completion time with release dates*, in Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, New York, NY, 1999, pp. 3243.
4. M. W. P. Savelsbergh, R. N. Uma, and J. M. Wein, *An experimental study of LP-based approximation algorithms for scheduling problems*, in Proceedings of the 9th Annual ACMSIAM Symposium on Discrete Algorithms, San Francisco, CA, 1998, ACM, New York, 1998, pp. 453462.
5. H. Belouadah, M. E. Posner, and C. N. Potts, *Scheduling with release dates on a single machine to minimize total weighted completion time*, Discrete Appl. Math., 36 (1992), pp. 213231.
6. M. E. Dyer and L. A. Wolsey,*Formulating the single machine sequencing problem with release dates as a mixed integer program*, Discrete Appl. Math., 26 (1990), pp. 255270
7. M. Queyranne, *Structure of a simple scheduling polyhedron*, Math. Programming, 58 (1993), pp. 263285.
8. M. Queyranne and A. S. Schulz, *Polyhedral Approaches to Machine Scheduling*, Preprint 408, Department of Mathematics, Technische UniversitatBerlin, Germany, 1994.

9. 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*, Math. Oper. Res., 22 (1997), pp. 513544.
10. L. A. Hall, D. B. Shmoys, and J. Wein, *Scheduling to minimize average completion time: Off-line and on-line algorithms*, in Proceedings of the 7th Annual ACMSIAM Symposium on Discrete Algorithms, Atlanta, GA, 1996, ACM, New York, 1996, pp. 142151.
11. E. J. Anderson and C. N. Potts, *On-line scheduling of a single machine to minimize total weighted completion time*, in Proceedings of the 13th Annual ACMSIAM Symposium onDiscrete Algorithms, San Francisco, CA, 2002, pp. 548557.
12. A. S. Schulz,*Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 1084, W. H. Cunningham, S. T. McCormick, and M. Queyranne, eds., Springer, Berlin, 1996, pp. 301315.
13. S. Chakrabarti, C. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein, *Improved scheduling algorithms for minsum criteria*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 1099, F. Meyer auf der Heide and B. Monien, eds., Springer, Berlin, 1996, pp. 646657.
14. Michel X. Goemans, Maurice Queyranne, Andreas S. Schulz, Martin Skutella, and Yaoguang Wang, *Single machine scheduling with release dates*, in SIAM J. Discrete Math Vol. 15, No. 2, 2002, pp. 165192.
15. R. N. Uma and J. M. Wein, *On the relationship between combinatorial and LP-based approaches to NP-hard scheduling problems*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 1412, R. E. Bixby, E. A. Boyd, and R. Z. Ros-Mercado, eds., Springer, Berlin, 1998, pp. 394408.
16. Chandra Chekuri, Sanjeev Khanna, *Approximation Algorithms for Minimizing Average Weighted Completion Time*, in Handbook of Scheduling: Algorithms, Models, and Performance Analysis, edited by Joseph Leung, CRC Press, 2004
17. C. Chekuri, R. Motwani, B. Natarajan, and C. Stein, *Approximation techniques for average completion time scheduling*, SIAM J. Comput., 31 (2001), pp. 146166.
18. A. S. Schulz and M. Skutella, *The power of α-points in preemptive single machine scheduling*, J. Sched., 5 (2002).
19. A. S. Schulz and M. Skutella, *Scheduling-LPs bear probabilities: Randomized approximations for min-sum criteria*, in AlgorithmsESA 97, Lecture Notes in Comput. Sci. 1284, Springer, Berlin, 1997, pp. 416429
20. Rajiv Gandhi, Julin Mestre, *Combinatorial Algorithms for Data Migration to Minimize Average Completion Time* Algorithmica 54(1): 54-71 (2009)