Techniques for Scheduling with Rejection

Daniel W. Engels*, David R. Karger**, Stavros G. Kolliopoulos***, Sudipta Sengupta**, R. N. Uma†, and Joel Wein‡

Dartmouth College, MIT, Polytechnic University

Abstract. We consider the general problem of scheduling a set of jobs where we may choose not to schedule certain jobs, and thereby incur a penalty for each rejected job. More specifically, we focus on choosing a set of jobs to reject and constructing a schedule for the remaining jobs so as to optimize the sum of the weighted completion times of the jobs scheduled plus the sum of the penalties of the jobs rejected.

We give several techniques for designing scheduling algorithms under this criterion. Many of these techniques show how to reduce a problem with rejection to a (potentially more complex) scheduling problem without rejection. Some of the reductions are based on general properties of certain kinds of linear-programming relaxations of optimization problems, and therefore are applicable to problems outside of scheduling; we demonstrate this by giving an approximation algorithm for a variant of the facility-location problem.

In the last section of the paper we consider a different notion of rejection in the context of scheduling: scheduling jobs with due dates so as to maximize the number of jobs that complete by their due dates, or equivalently to minimize the number of jobs that do not complete by their due date and that thus can be considered "rejected." We investigate the approximability of a simple version of this problem, giving approximation algorithms and characterizing integrality gaps of a class of linear-programming relaxations.

* dragon@glenfiddich.lcs.mit.edu. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. Research supported by NSF Contract MIP-9612632.

** {karger, sudipta}@theory.lcs.mit.edu. Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. Research supported in part by ARPA contract N00014-95-1-1246 and NSF contract CCR-9624239, as well as grants from the Alfred P. Sloane and David and Lucille Packard foundations.

*** stavros@cs.dartmouth.edu. Department of Computer Science, Dartmouth College, Hanover, NH 03755-3510. Research partially supported by NSF Award CCR-9308701 and NSF Career Award CCR-9624828.

† ruma@tiger.poly.edu. Department of Computer Science, Polytechnic University, Brooklyn, NY, 11201. Research partially supported by NSF Grant CCR-9626831.

‡ wein@mem.poly.edu. Department of Computer Science, Polytechnic University, Brooklyn, NY, 11201. Research partially supported by NSF Grant CCR-9626831 and a grant from the New York State Science and Technology Foundation, through its Center for Advanced Technology in Telecommunications.

1 Introduction

Most of traditional scheduling theory begins with a set of n jobs to be scheduled in a particular machine environment so as to optimize a particular optimality criterion. At times, however, a higher-level decision has to be made: given a set of tasks, and limited available capacity, choose only a subset of these tasks to be scheduled, while perhaps incurring some penalty for the jobs that are not scheduled. Knapsack is a "pure" instance of the problem, where we care only about what subset to "accept." In scheduling, however, the set of accepted jobs must also be scheduled at some cost.

In this paper we contribute to the understanding of this problem by studying several scheduling models in which the scheduler can choose to *reject* (not schedule) certain jobs at a penalty. We give several techniques for designing exact and approximation algorithms in this paradigm.

1.1 Problem Definition. For much of this paper we study variants of the following basic scheduling problem. We are given n jobs $j = 1, \ldots, n$, each with a nonnegative processing time p_j, a weight w_j, and a *rejection penalty* e_j. For each job we must decide either to schedule that job (on a machine that can process at most one job at a time) or to reject it. If we schedule job j, we denote its completion time by C_j. If we reject job j, we pay its rejection penalty e_j. Our goal is to choose a subset S of the n jobs to schedule on the machine so as to minimize the sum of the weighted completion times of the scheduled jobs and the penalties of the rejected jobs. We denote the set of rejected jobs by $\overline{S}$, and thus our overall optimality criterion may be denoted as $\sum_{j \in S} w_j C_j + \sum_{j \in \overline{S}} e_j$.

At times we consider scheduling models in which the processing of the jobs is constrained by either *release date* constraints (job j cannot begin processing before a specified release date r_j) or *precedence* constraints ($j \prec k$ is a precedence constraint stating that job k can begin its processing no earlier than job j completes its processing). Precedence constraints impose a partial ordering on the jobs. We also consider models with more than one machine available for processing and other related optimality criteria; we leave their details to be introduced as appropriate in the body of the paper.

Our interest in these models arises primarily from two considerations. First, it is becoming widely understood that in actual manufacturing settings a scheduling algorithm is only one element of a larger decision analysis tool that takes into account a variety of factors such as inventory, potential machine disruptions, etc. [13]. Furthermore, some practitioners [12] have identified as a critical element of this larger decision picture the "pre-scheduling negotiation" in which one considers the capacity of your production environment and then agrees to schedule certain jobs at the requested quality of service (e.g. job turnaround time) and other jobs at a reduced quality of service, while rejecting other jobs altogether.

Typically, the way that current systems handle the integration of objective functions is to have a separate component for each problem element and to integrate these components in an ad-hoc heuristic fashion. Therefore, models that integrate more than one element of this decision process while remaining amenable to solution by algorithms with performance guarantees have the potential to be very useful elements in this decision process.

Secondly, our work is directly inspired by that of Bartal, Leonardi, Marchetti-Spaccamela, Sgall and Stougie [1] who studied a multiprocessor scheduling problem with the objective of trading off between schedule *makespan* (length) and job rejection cost. Makespan and average weighted completion time are perhaps the two most basic and well-studied of all scheduling optimality criteria; therefore, it is of interest to understand the impact of the "rejection option" on the $\sum w_j C_j$ objective as well.

Since we will be dealing with a number of scheduling models, we will use the scheduling notation introduced by Graham, Lawler, Lenstra and Rinnooy Kan [6] to denote the model being considered.

1.2 Relevant Previous Work. There has been much work on scheduling without rejection to minimize $\sum w_j C_j$. For the simplest variant of the problem, $1||\sum w_j C_j$ (scheduling jobs with no side constraints on one machine to minimize $\sum w_j C_j$) Smith [19] proved that an optimal schedule could be constructed by ordering the jobs by non-decreasing p_j/w_j ratios. More complex variants are typically $\mathcal{NP}$-hard, and recently there has been a great deal of work on the development of approximation algorithms for them (e.g. [14, 8, 5, 2, 17, 16]). A ρ*-approximation algorithm* is a polynomial-time algorithm that always finds a solution of objective function value within a factor of ρ of optimal (ρ is also referred to as the *performance guarantee*).

To the best of our knowledge the only previous work on scheduling models that include rejection in our manner is that of Bartal et. al., who seem to have formally introduced the notion. They considered the problem of scheduling with rejection in a multiprocessor setting with the aim of minimizing the makespan of the scheduled jobs and the sum of the penalties of the rejected jobs [1]. They give a $(1+\phi)(\approx 2.618)$-competitive algorithm for the on-line version, where ϕ is the golden ratio.

1.3 Our Results. Our work differs from that of Bartal et. al. in that we focus in $\sum w_j C_j$ rather than the makespan. More importantly, we focus less on a particular problem and more on giving general techniques that are applicable to a number of problems. We address only the offline setting (the problem is fully specified in advance).

We begin in Section 2 by studying the simplest version of the problem, $1||\sum_S w_j C_j + \sum_{\overline{S}} e_j$ and give a simple and intuitive greedy algorithm that solves certain special cases exactly. We show that the general problem $1||\sum_S w_j C_j + \sum_{\overline{S}} e_j$ is weakly $\mathcal{NP}$-Complete, and give a pseudopolynomial-time algorithm based on dynamic programming that solves it exactly. Our dynamic program can be interpreted as capturing the option of assigning a job to a "rejection machine". We conclude Section 2 by giving a direct reduction (using the idea of a "rejection machine") of certain scheduling problems with rejection to more complex problems without.

In Sections 3 and 4 we give another example of how techniques that are used for scheduling problems without rejection can be adapted to handle problem variants with rejection. In contrast to a direct problem-to-problem reduction, however, we examine the structure of certain linear-programming relaxations of such scheduling problems, show how they can be augmented to include a rejection variable to which a job can be assigned, and then give a thresholding technique that allows us to obtain an approximation algorithm for a problem with rejection. This technique is in fact quite general and is applicable to many sorts of optimization problems; we demonstrate its generality by giving an approximation algorithm for a variant of the facility location problem with rejection.

Finally, in Section 5, we move to a different and more traditional notion of rejection in scheduling, which is nonetheless not well understood. In this setting each job has a due date, and a job is essentially "rejected" if it is not scheduled by its due date. Specifically, we consider the problem $1|r_j|\sum U_j$: scheduling jobs with release dates on one machine so as to minimize the number of jobs that miss their deadline or, equivalently, maximize the number of jobs that meet their deadline. This is similar to the problem of scheduling jobs with rejection because the jobs that do not meet their deadline can be considered rejected. Nothing is known about approximating these strongly $\mathcal{NP}$-hard scheduling problems. We give integrality gaps for both maximization and minimization versions of the problem and also give an approximation algorithm for the maximization problem.

2 Scheduling with Rejection

In this section, we focus on the problem of scheduling jobs on a single machine to minimize weighted completion time. If rejection is not considered, the problem is solvable in polynomial time using Smith's rule: schedule the jobs in increasing order of p_j/w_j. We show that adding the option of rejection makes the problem weakly $\mathcal{NP}$-complete. Such problems cannot admit polynomial-time solutions unless $\mathcal{P} = \mathcal{NP}$. But they can admit *pseudo-polynomial* time algorithms that run fast when the input numbers are small. We give such an algorithm, based on dynamic programming, for our scheduling problem.

Intuitively, the dynamic program can be seen as solving a scheduling problem on two machines, one a "rejection machine" that takes care of all of the rejected jobs and has an unusual cost function for doing so. It is therefore a variant of Rothkopf [15] and Lawler and Moore's [10] pseudo-polynomial algorithms for scheduling to minimize weighted completion time on a fixed number of parallel machines. Indeed, our algorithm also generalizes to schedule with rejection on any fixed number of parallel machines. We use the "rejection machine" intuition again to devise a 3/2-approximation algorithm for the problem of scheduling with rejection on an arbitrary number of machines.

Although we use the two-machine intuition, the rejection cost that the second machine pays is in some sense a simpler function than the weighted completion time. So for certain special cases, we are able to give simple and efficient polynomial time algorithms based on greedy methods.

2.1 Complexity of Scheduling with Rejection. For any fixed number of processors $m > 1$, $Pm||(\sum_S w_j C_j + \sum_{\bar{S}} e_j)$ is trivially seen to be $\mathcal{NP}$-complete by restricting the problem to $Pm||\sum w_j C_j$, a known weakly $\mathcal{NP}$-complete problem [4]. However, we can also prove that adding rejection makes the *single* machine problem $\mathcal{NP}$-complete.

Theorem 1. $1||(\sum_S w_j C_j + \sum_{\bar{S}} e_j)$ *is weakly* $\mathcal{NP}$*-complete.*

Proof. We reduce the Partition Problem [4] to $1||(\sum_S w_j C_j + \sum_{\bar{S}} e_j)$ in a straightforward manner. Each of the n elements a_i in the Partition Problem correspond to a job J_i in $1||(\sum_S w_j C_j + \sum_{\bar{S}} e_j)$ with weight and processing time equal to a_i and rejection cost equal to $ba_i + \frac{1}{2}a_i^2$, where $b = \frac{1}{2}\sum_{i=1}^n a_i$.

2.2 Dynamic Programming. We give an $O(n\sum_{j=1}^n w_j)$ time algorithm using dynamic programming to solve $1||(\sum_S w_j C_j + \sum_{\bar{S}} e_j)$. In the full version of the paper, we will show how to modify it to obtain an FPAS for $1||(\sum_S w_j C_j + \sum_{\bar{S}} e_j)$.

To solve our problem, we set up a dynamic program for a harder problem: namely, to find the schedule that minimizes the objective function when the total weight of the scheduled jobs is given. We number the jobs in ascending order of p_j/w_j. Let $\phi_{w,j}$ denote the optimal value of the objective function when the jobs in consideration are $j, j+1, \ldots, n$, and the total weight of the scheduled jobs is w. Note that $\phi_{w,n} = 0$ for any $w \neq w_n$ and $\phi_{w_n,n} = w_n p_n$, forming the boundary conditions for the dynamic program.

Now, consider any optimal schedule for the jobs $j, j+1, \ldots, n$ in which the total weight of the scheduled jobs is w. In any such schedule, there are two possible cases — either job j is rejected or job j is scheduled. So we have $\phi_{w,j} = \min\,(\phi_{w,j+1} + e_j, \phi_{w-w_j,j+1} + wp_j)$.

Now, observe that the weight of the scheduled jobs can be at most $\sum_{j=1}^n w_j$, and the answer to our original problem is $\min\{\phi_{w,1} : 0 \leq w \leq \sum_{j=1}^n w_j\}$. Thus, we need to compute exactly $n\sum_{j=1}^n w_j$ values $\phi_{w,j}$. We remark that a similar dynamic program solves $1||(\sum_S w_j C_j + \sum_{\bar{S}} e_j)$ when the processing times are equal.

Theorem 2. *Dynamic programming yields an* $O(n\sum w_j)$*-time (or an* $O(n\sum p_j)$*-time) algorithm for exactly solving* $1||(\sum_S w_j C_j + \sum_{\bar{S}} e_j)$*.*

2.3 Special cases. We consider two special cases for which simple greedy algorithms exist to solve $1||(\sum_S w_j C_j + \sum_{\bar{S}} e_j)$. We give an algorithm for the case when all weights are equal. An algorithm for the case when all processing times are equal is analogous and not presented. Our greedy algorithm, which we call `SCHREJ`, is as follows. We start with all jobs scheduled, i.e., the scheduled set $S = \{1, 2, \ldots, n\}$. (Note that we can optimally schedule the jobs in any subset

S using Smith's ordering.) We then reject jobs greedily until we arrive at an optimal schedule. Note that when we reject a previously scheduled job j, there is a change (positive or negative) in the objective function. We determine a job k that causes the maximum decrease in the the objective function. If there is no such job, then (as we will argue below) we have reached an optimal solution. Otherwise, we remove job k from S (i.e. reject it), and iterate on the remaining jobs in S.

As this algorithms runs, we maintain the set S of currently scheduled jobs. Let $\Delta_j(S)$ denote the amount by which the cost of our schedule changes if we (pay to) reject job $j \in S$. Let $k \in S$ be such that $\Delta_k(S)$ is minimum.

Lemma 1. *During any iteration of* SCHREJ*, for any $j \in S$, we have*

$$\Delta_j(S) = -[w_j \sum_{i \leq j, i \in S} p_i \; + \; p_j \sum_{i > j, i \in S} w_i] + e_j$$

The following lemma is an immediate consequence of Lemma 1.

Lemma 2. *The value of $\Delta_j(S)$ increases across every iteration of* SCHREJ*, so long as job j remains in S.*

The following lemma argues that the algorithm can terminate when $\Delta_k(S) \geq 0$.

Lemma 3. *For a set S of jobs, if $\Delta_j(S)$ is non-negative for each $j \in S$, then there is an optimal schedule in which all the jobs in S are scheduled.*

Proof. (sketch) Consider any non-empty set of jobs $R \subset S$. Start with the schedule in which all jobs in S are scheduled, and start rejecting the jobs in R one by one. Observe that the objective function value does not decrease during every such rejection.

For the above lemmas, we have not used the fact that the weights w_j are equal. But this fact is used by the next lemma, which proves that we are justified in moving job k from S to $\overline{S}$ when $\Delta_k(S) < 0$.

Lemma 4. *For a set S of jobs with equal weights, if $\Delta_k(S)$ (the minimum of the $\Delta_j(s)$ for $j \in S$, as computed in* SCHREJ*) is negative, then there is an optimal schedule for the set of jobs in S in which job k is rejected.*

Proof. (sketch) Consider an optimal schedule Γ in which job k is scheduled and the set of rejected jobs is R. Clearly, R is non-empty, otherwise, since $\Delta_k < 0$, we can get a better schedule by rejecting job k. We show that we can improve the schedule Γ by rejecting job k and instead scheduling one of the jobs in R (the one immediately preceding or following job k according to Smith's rule). We compare the objective function value for the two schedules by starting from a schedule in which all jobs are scheduled, and then rejecting the set R of jobs in a particular order. This order is slightly different for the two cases considered.

Theorem 3. *Algorithm* SCHREJ *outputs an optimal schedule for $1||(\sum_S w_j C_j + \sum_{\bar{S}} e_j)$ with equal weights in $O(n^2)$ time.*

Proof. (sketch) By induction. According to the previous lemmas, $SCHREJ$ only rejects job it is safe to reject, and terminates when all remaining jobs must be in the optimal schedule.

2.4 Worst-Case Performance Guarantees. We now turn to several NP-hard variants of the problem and give small-constant-factor approximation algorithms; we do this by reducing the problem with rejection to a scheduling problem without rejection.

First we consider the single machine version of the problem $1||(\sum_S w_jC_j + \sum_{\bar{S}} e_j)$. An instance I can be reduced in an approximation-preserving manner to an instance I' of $R||\sum w_jC_j$ with $n+1$ unrelated machines. Let the machines be indexed $1, 2, \ldots, (n+1)$. Machine $(n+1)$ is the *original* machine with $p_{n+1,j} = p_j$ for job j. Job j has $p_{jj} = \frac{e_j}{w_j}$ on machine j and $p_{ij} = \infty$ on machine i where $i = 1, \ldots, n$ and $i \neq j$. We omit further details. A similar reduction can be obtained from $1|r_j|(\sum_S w_jC_j + \sum_{\bar{S}} e_j)$ to $R|r_{ij}|\sum w_jC_j$ using the observation that without loss of generality $e_j > w_jr_j$ for all j. Inspection of the argument above reveals that it can be extended to parallel identical machines and parallel unrelated machines. Using the approximation algorithms of Schulz and Skutella [17] for $R||\sum w_jC_j$ and $R|r_{ij}|\sum w_jC_j$ we obtain the following theorem.

Theorem 4. *There exists a $\frac{3}{2}$-approximation algorithm for $R||\sum_S w_jC_j + \sum_{\bar{S}} e_j$ and a 2-approximation algorithm for $R|r_j|\sum_S w_jC_j + \sum_{\bar{S}} e_j$, both in expectation.*

3 A General Problem $\mathcal{P}$ with Rejection

In this section we give a rather general method of converting certain kinds of approximation algorithms for optimization problems without rejection to approximation algorithms for problem variants with rejection. In contrast to the last result in the previous section, however, which was a direct reduction to a scheduling problem in which the solution to that scheduling problem was used as a black box, in this case we exploit the common structure of different linear-programming relaxations used in approximation for scheduling and other optimization problems.

We begin by stating a general "assignment" optimization problem and gaining some understanding of the structure of solutions to two relaxations of this problem. Then we use this structure in a general algorithm for the optimization problem augmented to include the possibility of rejection.

Let us consider the following general problem $\mathcal{P}$. We have two sets of objects $\mathcal{O}_1$ and $\mathcal{O}_2$. We want to assign each object in $\mathcal{O}_1$ to an object in $\mathcal{O}_2$ such that a given set of constraints are satisfied by the assignment and some objective function of the assignment is minimized. We will formulate this problem as an integer program which uses the 0-1 decision variables x_{ij}, where x_{ij} is 1 if object $i \in \mathcal{O}_1$ is assigned to object $j \in \mathcal{O}_2$ and 0 otherwise. So $\mathcal{P}$ is formulated by the following integer program (IP). Let $\mathbf{c}, \mathbf{d} \geq \mathbf{0}$.

$$
\begin{array}{lrcll}
\text{minimize} & \mathbf{c^T x} + \mathbf{d^T y} & & & (3.1)\\
\text{subject to} & \sum_j x_{ij} & = & 1 \quad \forall i \in \mathcal{O}_1 & (3.2)\\
& \sum_{i,k \in S(j) \subseteq \mathcal{O}_2} x_{ik} & \leq & \kappa_j \quad \forall j \in \mathcal{O}_2 & (3.3)\\
& \mathbf{A}(\mathbf{x} \circ \mathbf{y}) & \geq & \mathbf{0} & (3.4)\\
& x_{ij} & \in & \{0,1\} \quad \forall i \in \mathcal{O}_1, \forall j \in \mathcal{O}_2 & (3.5)\\
& x_{ij} & = & 0 \quad (i,j) \in M \subseteq \mathcal{O}_1 \times \mathcal{O}_2 & (3.6)
\end{array}
$$

Here, $\mathbf{x} \circ \mathbf{y}$ denotes the concatenation of vectors $\mathbf{x}, \mathbf{y}$. $\mathbf{A}(\mathbf{x} \circ \mathbf{y}) \geq \mathbf{0}$ is a set of additional constraints that have to be satisfied by any feasible assignment. We call the constraints (3.3) *capacity constraints.*

We can replace the integrality constraint on x_{ij} by the constraint $0 \leq x_{ij} \leq 1$. This essentially means that an object $i \in \mathcal{O}_1$ can be fractionally assigned to objects in $\mathcal{O}_2$. We will let $(LP1)$ denote the resulting relaxation to (IP).

We will now consider the following additional relaxation to $(LP1)$. Let β be a positive constant that is at most 1. We will further relax the constraint $\sum_j x_{ij} = 1$ to the constraint $\beta \leq \sum_j x_{ij} \leq 1$. This says that at least a β fraction of object $i \in \mathcal{O}_1$ must be assigned to some object(s) in $\mathcal{O}_2$. We will call this linear program $(LP2)$.

Let us denote their corresponding optimal solutions by IP^*, $LP1^*$ and $LP2^*$ respectively. Since $(LP1)$ is a relaxation of (IP) and $(LP2)$ is a relaxation of $(LP1)$, we have the inequality $LP2^* \leq LP1^* \leq IP^*$.

Let $\mathcal{A}$ be an algorithm that accepts a feasible solution to $(LP1)$ as input and returns an output with value at most ρ times the input's objective function value (it thus acts as a ρ-approximation algorithm for $(IP1)$). We will show that if problem $\mathcal{P}$ satisfies certain Conditions 1 and 2 (to be stated later), then we can use $\mathcal{A}$ in an approximation algorithm (with somewhat worse performance) for a version of $\mathcal{P}$ that allows rejection. We now state Condition 1.

Condition 1 *There exists a polynomial-time algorithm to convert a feasible solution* $\mathbf{q}$ *to* $(LP2)$*, of value* $v(\mathbf{q})$*, to a feasible solution to* $(LP1)$ *of value within at most* $f(\beta) \cdot v(\mathbf{q})$.

`Case a:` $f(\beta) = \frac{1}{\beta}$, *if* $\mathcal{P}$ *does not have capacity constraints.*

`Case b:` $f(\beta) = \frac{1}{\beta^2}$, *if* $\mathcal{P}$ *has capacity constraints.*

We now modify the original problem $\mathcal{P}$ to include rejection, and use our structural insights to give an approximation algorithm. We modify $\mathcal{P}$ so that each object $i \in \mathcal{O}_1$ can either be assigned to an object $j \in \mathcal{O}_2$ or can be rejected for a certain penalty. Let us call this modified problem $\mathcal{P}_R$. The problem $\mathcal{P}_R$ can be formulated similar to IP. But now we will have an additional 0-1 decision variable z_i which is 1 if object $i \in \mathcal{O}_1$ is rejected and 0 otherwise. The objective function to be minimized is now given by $\mathbf{c^T x} + \mathbf{d^T y} + \mathbf{g^T z}$ and the constraint $\sum_j x_{ij} = 1$ in the (IP) is replaced by $\sum_j x_{ij} + z_i = 1$. Let the corresponding linear relaxation be $(LP1_R)$. We introduce a second "decomposability" condition that is natural for problems involving rejection. Let $\mathcal{P}(S)$ denote the restriction of the assignment problem $\mathcal{P}$ above, such that x_{ij} is defined only for $i \in S \subseteq \mathcal{O}_1$.

Intuitively, we want to capture the notion that the set $\mathcal{O}_1 - S$ of objects to be rejected is determined by the objective but not the constraints of $\mathcal{P}_R$.

Condition 2 *Let* $(\mathbf{x}, \mathbf{y})$ *be a feasible solution for* $\mathcal{P}(\mathcal{O}_1)$. *If* x_{ij} *with* $i \in \mathcal{O}_1 - S$, *is removed from* $\mathbf{x}$ *to obtain* $\mathbf{x}'$, *the pair of vectors* $(\mathbf{x}', \mathbf{y})$ *is feasible for* $\mathcal{P}(S)$.

Consider the following algorithm $\mathcal{B}$ to get a solution for $\mathcal{P}_R$. First solve $(LP1_R)$. If $\sum_j x_{ij} < \beta$, then include $i \in \mathcal{O}_1$ in the set $\bar{S}$ of rejected objects. Else include $i \in S$. Trivially, $S \cup \bar{S} = \mathcal{O}_1$ and $S \cap \bar{S} = \emptyset$. $\beta \in (0, 1]$ will be chosen later. The resulting fractional solution is a solution to (LP2). Convert it to a solution to (LP1) (as described in Condition 1) and follow algorithm $\mathcal{A}$ to assign objects in S to $\mathcal{O}_2$.

Theorem 5. *Let* $\mathcal{P}$ *satisfy Conditions 1(a) and 2. If there is a* ρ*-approximation algorithm for* $\mathcal{P}$ *w.r.t* $(LP1)$, *then* $\mathcal{B}$ *is a* $(1 + \rho)$*-approximation algorithm for* $\mathcal{P}_R$ *w.r.t* $(LP1_R)$.

Proof. Let $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ be an optimal solution to $(LP1_R)$ with objective value $LP1_R^* = \gamma + \delta$, where $\gamma = \mathbf{c}^{\mathbf{T}}\mathbf{x}^* + \mathbf{d}^{\mathbf{T}}\mathbf{y}^*$, and $\delta = \mathbf{g}^{\mathbf{T}}\mathbf{z}^*$. The cost of rejecting objects in $\bar{S}$ is at most $\frac{\delta}{1-\beta}$. The term γ can be written as $\gamma^S + \gamma^{\bar{S}} + \mathbf{d}^{\mathbf{T}}\mathbf{y}^*$, where γ^S is the optimal cost of fractionally assigning objects in S to objects in $\mathcal{O}_2$ and $\gamma^{\bar{S}}$ is the optimal cost of fractionally assigning objects in $\bar{S}$ to objects in $\mathcal{O}_2$. Denote by x_S^* the vector x^* restricted to elements x_{ij}^* such that $i \in S$. By Condition 2, $\mathbf{q} = (\mathbf{x}_{\mathbf{S}}^*, \mathbf{y}^*)$ is a feasible solution to $(LP2)(S)$ of $\mathcal{P}(S)$, with value $v = \gamma^S + \mathbf{d}^{\mathbf{T}}\mathbf{y}^*$. Since Condition 1(a) is satisfied by $\mathcal{P}$, $\mathbf{q}$ can be converted to a feasible solution to $(LP1)(S)$ of value at most $\frac{\rho}{\beta}v$. Therefore $\mathcal{B}$ is a $(\max\{\frac{\rho}{\beta}, \frac{1}{1-\beta}\})$-approximation algorithm for $\mathcal{P}_R$ w.r.t $(LP1_R)$. Note note that $(\max\{\frac{\rho}{\beta}, \frac{1}{1-\beta}\})$ is minimized for $\beta = \frac{\rho}{\rho+1}$ yielding a performance guarantee of $(1 + \rho)$.

Theorem 6. *Let* $\mathcal{P}$ *satisfy Conditions 1(b) and 2. If there is a* ρ*-approximation algorithm for* $\mathcal{P}$ *w.r.t* $(LP1)$, *then* $\mathcal{B}$ *is a* $(2/(\rho+2-\sqrt{\rho(\rho+4)}))$*-approximation algorithm for* $\mathcal{P}_R$ *w.r.t* $(LP1_R)$.

4 Applications

Scheduling Problem. We consider the problem of scheduling jobs on a single machine subject to release dates and precedence constraints, $1|r_j, prec|$-$(\sum_S w_j C_j + \sum_{\bar{S}} e_j)$. Hall, Schulz, Shmoys and Wein [8] have given a 3-approximation algorithm for the corresponding problem without rejection $1|r_j, prec|$-$\sum w_j C_j$ based on a time-indexed linear-programming relaxation [20] of the problem. It can be shown that this relaxation of $1|r_j, prec|(\sum_S w_j C_j + \sum_{\bar{S}} e_j)$ satisfies Conditions 1(b) and 2. So for the rejection variant of this problem, by Theorem 6 with $\rho = 3$ and $\beta = 0.791$, we have

Theorem 7. *There is a* 4.79*-approximation algorithm for* $1|r_j, prec|(\sum_S w_j C_j + \sum_{\bar{S}} e_j)$, *under the assumption that* $\max_j r_j + \sum_{j=1}^n p_j$ *is polynomial in* n.

A polynomial-size interval-indexed formulation [16, 20] can be used to obtain an improved (4.496)-approximation. This formulation is not of the form (IP) but can be shown to satisfy an extension to Theorem 6.

Uncapacitated Facility Location Problem. In this problem, we are given a set of locations $N = \{1, \ldots, n\}$ and subsets $F, D \subseteq N$. We may open a facility at location $i \in F$ for a non-negative cost f_i. For each location $j \in D$, there is a positive integral demand d_j that must be shipped to it. The cost of shipping a unit of demand from a facility at location i to a location j is given by c_{ij} where the c_{ij}'s are non-negative. Since we consider only the *metric* variant of the problem, the c_{ij}'s are symmetric and satisfy the triangle inequality.

We now add the additional constraint that each location $j \in D$ can either be assigned to some facility $i \in F$ in which case it incurs a cost of c_{ij} for each unit of demand shipped or it may be rejected incurring a penalty of e_j.

A sequence of papers [18], [7], [3] give approximation algorithms for the basic problem that are based on a linear programming relaxation. We show that the linear program can be modified and that Conditions 1(a),2 apply, yielding

Theorem 8. *There is a* 2.736*-approximation algorithm for the uncapacitated facility location problem with rejection.*

5 Integrality gaps and approximations for $1|r_j|\sum U_j$

We now move to a more traditional notion of rejection in scheduling, which is nonetheless not well understood. We have one machine and each job has a release date r_j and a due date d_j; a job j is essentially "rejected" ($U_j = 1$) if it cannot be scheduled to complete by its due date. An optimal solution minimizes $\sum U_j$.

5.1 The quality of fractional relaxations. Solving $1|r_j|\sum U_j$ to optimality is strongly $\mathcal{NP}$-hard, [9] therefore we are interested in approximations. We distinguish between two optimization metrics. In the minimization problem one seeks to minimize $\sum_{j=1}^{n} U_j$. In the maximization problem one seeks to maximize the number of jobs that meet their deadline, i.e. $\sum_{j=1}^{n}(1 - U_j)$. A heavily used technique in approximation algorithms for scheduling problems is that of solving first a *fractional relaxation.* Typically, a relaxation in a single-machine setting allows jobs to be processed in a preemptive manner once released. We call the resulting schedule S_f, *fractional.* For $1|r_j|\sum U_j$, the contribution U_j of job j to the objective of a fractional relaxation is equal to $1 - z_j$, where $0 \leq z_j \leq 1$ is the fraction of job j that is processed before the deadline d_j. The optimum objective value of a fractional schedule would typically be superoptimal for the problem at hand. Hence if an efficient rounding algorithm is invoked to convert the fractional schedule into a feasible one, it will incur some degradation to the fractional objective. We study gaps between the fractional and the true optima. We use $p_{\max}$ and $p_{\min}$ to denote the maximum and minimum processing times among the input jobs.

Lemma 5. *There is an instance J of $1||\max \sum_{j=1}^{n}(1-U_j)$ on which the value of a fractional schedule is $\Omega(\ln(p_{\max}/p_{\min}))$ times the true optimum.*

Theorem 9. *There is an instance J of $1|r_j, p_j = p|\min \sum_{j=1}^{n} U_j$ on which the value of a fractional schedule is at most $2/n$ times the true optimum.*

5.2 An IP formulation and approximation algorithms. The results in the previous section indicate that fractional relaxations may give weak bounds for approximation. We show that nonetheless they can yield solutions of good quality for the maximization problem; we give an IP formulation and an approximation algorithm associated with its linear relaxation. The formulation we propose uses interval-indexed variables and thus has polynomial size. We define the set I to contain all disjoint time intervals of the form $(a, b]$ where a, b are consecutive release times or deadlines, i.e. for no other release time or deadline c is it the case that $a < c < b$. Therefore $|I| = O(n)$. The intervals in I are numbered so we refer unambiguously to $i \in I$. If $i = (a, b]$ the *length* $l(i)$ of i is $b - a$. The set $I(j)$ of *legal* intervals for job j is the set of intervals $(a_{j_k}, b_{j_k}]$ in I such that $r_j \leq a_{j_k}$ and $b_{j_k} \leq d_j$. In the ensuing formulation we use the variable x_{ij} to denote the time spent processing job j during interval $i \in I$.

$$
\begin{array}{lrcll}
\text{minimize} & \sum_{j=1}^{n} U_j & & & \qquad (5.1)\\
\text{subject to} & \sum_{i\in I(j)} x_{ij} & \leq & p_j \quad j = 1, \ldots, n & \qquad (5.2)\\
 & \sum_{j=1}^{n} x_{ij} & \leq & l(i) \quad i \in I & \qquad (5.3)\\
 & 1 - \frac{\sum_{i\in I(j)} x_{ij}}{p_j} & \leq & U_j \quad j = 1, \ldots, n & \qquad (5.4)\\
 & x_{ij} & = & 0 \quad i \notin I(j) \quad j = 1, \ldots, n & \qquad (5.5)\\
 & x_{ij} & \in & Z^{+} & \qquad (5.6)\\
 & U_j & \in & \{0, 1\} \quad j = 1, \ldots, n & \qquad (5.7)
\end{array}
$$

It can be seen that the integer program $IP1$ formed by equations (5.1) through (5.7) is a valid relaxation of the problem. Observe that even in an integer solution to $IP1$ a job j may be processed in a preemptive manner. Call LP_{min} the linear relaxation of $IP1$. Let LP_{max} be the linear relaxation of $IP1$ with (1) replaced with "maximize $\sum_{j=1}^{n}(1-U_j)$". It can be shown that the gaps in Lemma 5 and Theorem 9 apply also for the fractional optima of LP_{min}, LP_{max}.

Lemma 6. *There is an algorithm that outputs in polynomial time a solution to $1|r_j, p \leq p_j \leq 2p|max \sum_{j=1}^{n}(1-U_j)$ of value at least $\lceil z^*/10\rceil$, where z^* is the optimum of LP_{max} for the problem.*

Theorem 10. *There is an algorithm that outputs in polynomial time a solution to $1|r_j|max \sum_{j=1}^{n}(1-U_j)$ of value at least $\lceil z^*/(10\lceil\log(p_{\max}/p_{\min})\rceil)\rceil$, where z^* is the optimum of LP_{max} for the problem.*

Acknowledgments. We are thankful to David Shmoys for providing pointers to the literature and giving us access to [11] and to David Williamson for helpful discussions.

References

1. Y. Bartal, S. Leonardi, A. Marchetti-Spaccamela, J. Sgall, and L. Stougie. Multiprocessor scheduling with rejection. In *Proc. 7th SODA*, 95–103, 1996.
2. C. Chekuri, R. Motwani, B. Natarajan, and C. Stein. Approximation techniques for average completion time scheduling. In *Proc. 8th SODA*, 609–618, 1997.
3. F. A. Chudak. Improved approximation algorithms for uncapacitated facility location. In Proc. 6th IPCO 1998. To appear.
4. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.
5. M. Goemans. Improved approximation algorithms for scheduling with release dates. In *Proc. 8th SODA*, 591–598, 1997.
6. 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. *Annals of Discrete Mathematics*, 5:287–326, 1979.
7. S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. In *Proc. 9th SODA*, 1998.
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*, (3):513–544, August 1997.
9. E. L. Lawler. Scheduling a single machine to minimize the number of late jobs. Preprint, Computer Science Division, Univ. of California, Berkeley, 1982.
10. E. L. Lawler and J. M. Moore. A functional equation and its application to resource allocation and sequencing problems. In *Manag. Sci.*, volume 16, 77–84, 1969.
11. E. L. Lawler and D. B. Shmoys. Weighted number of late jobs (preliminary version). To appear in: J.K. Lenstra and D.B. Shmoys (eds.) Scheduling, Wiley.
12. Maxwell. Personal communication. 1996.
13. I. M. Ovacik and R. Uzsoy. *Decomposition Methods for Complex Factory Scheduling Problems.* Kluwer Academic Publishers, 1997.
14. C. Phillips, C. Stein, and J. Wein. Scheduling jobs that arrive over time. In *Proc. of 4th WADS, LNCS, 955*, 86–97, Berlin, 1995. Springer-Verlag. To appear in Mathematical Programming B.
15. M. H. Rothkopf. Scheduling independent tasks on parallel processors. In *Manag. Sci.*, volume 12, 437–447, 1966.
16. A. S. Schulz and M. Skutella. Random–based scheduling: New approximations and LP lower bounds. In J. Rolim, editor, *Randomization and Approximation Techniques in Computer Science*, *LNCS, 955*, 119 – 133. Springer, Berlin, 1997.
17. A. S. Schulz and M. Skutella. Scheduling–LPs bear probabilities: Randomized approximations for min–sum criteria. In R. Burkard and G. Woeginger, editors, *Algorithms – ESA'97*, *LNCS, 1284*, 416 – 429. Springer, Berlin, 1997.
18. D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proc. of the 29th ACM STOC*, 265–274, 1997.
19. W.E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.
20. M. E. Dyer and L. A. Wolsey. Formulating the single machine sequencing problem with release dates as a mixed integer program. In *Discrete Applied Mathematics*, 26, 255-270, 1990.