Real-time Scheduling with a Budget

Joseph (Seffi) Naor[1], Hadas Shachnai[2*], and Tami Tamir[3]

[1] Computer Science Dept., Technion, Haifa 32000, Israel
E-mail: naor@cs.technion.ac.il,
[2] Bell Labs, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974.
E-mail: hadas@research.bell-labs.com,
[3] Dept. of Computer Science and Eng., Box 352350, Univ. of Washington, Seattle, WA 98195.
E-mail: tami@cs.washington.edu

Abstract. Suppose that we are given a set of jobs, where each job has a processing time, a non-negative weight, and a set of possible time intervals in which it can be processed. In addition, each job has a processing cost. Our goal is to schedule a feasible subset of the jobs on a single machine, such that the total weight is maximized, and the cost of the schedule is within a given budget. We refer to this problem as *budgeted real-time scheduling (BRS)*. Indeed, the special case where the budget is *unbounded* is the well-known real-time scheduling problem. The second problem that we consider is *budgeted real-time scheduling with overlaps (BRSO)*, in which several jobs may be processed simultaneously, and the goal is to maximize the time in which the machine is utilized. Our two variants of the real-time scheduling problem have important applications, in vehicle scheduling, linear combinatorial auctions and QoS management for Internet connections. These problems are the focus of this paper.
Both BRS and BRSO are strongly NP-hard, even with unbounded budget. Our main results are $(2+\varepsilon)$-approximation algorithms for these problems. This ratio coincides with the best known approximation factor for the (unbudgeted) real-time scheduling problem, and is slightly weaker than the best known approximation factor of $e/(e-1)$ for the (unbudgeted) real-time scheduling with overlaps, presented in this paper. We show that better ratios (or simpler approximation algorithms) can be derived for some special cases, including instances with unit-costs and the budgeted *job interval selection problem (JISP)*. Budgeted JISP is shown to be APX-hard even when overlaps are allowed and with unbounded budget. Finally, our results can be extended to instances with multiple machines.

1 Introduction

In the well-known *real-time scheduling* problem (also known as the *throughput maximization* problem), we are given a set of n jobs; each job J_j has a processing time p_j, a non-negative weight w_j, and a set of time intervals in which it can be processed (given as either a window with release and due-dates or as a discrete set of possible processing intervals). The goal is to schedule a feasible subset of the jobs on a single machine, such that the overall weight of the scheduled jobs is maximized. In this paper we consider two variants of this problem.

In the *budgeted real-time scheduling (BRS)* problem, each job J_j has a processing cost c_j. A budget B is given, and the goal is to find a maximum weight schedule, among the feasible schedules whose total processing cost is at most B. In *real-time scheduling with overlaps (RSO)*, the jobs are scheduled on a single *non-bottleneck* machine, which can process simultaneously several jobs. The goal is to maximize the overall time in which the machine is utilized (i.e., processes at least one

* On leave from the Computer Science Dept., Technion, Haifa 32000, Israel.

job).[4] In the budgeted case ($BRSO$), each job J_j has a processing cost c_j. The goal is to maximize the time in which the machine is utilized, among the schedules with total processing cost at most B.

In our study of BRS, RSO and BRSO, we distinguish between *discrete* and *continuous* instances. In the discrete case, each job J_j can be scheduled to run in one of a given set of n_j intervals $I_{j,\ell}$ ($\ell = 1, \ldots, n_j$). The special case where each job has at most k intervals, i.e, $\forall j, n_j \leq k$, is called $JISP_k$. In the continuous case, job J_j has release date r_j, due date d_j, and a processing time p_j. It is possible to schedule J_j in any interval $[s_j, e_j]$ such that $s_j \geq r_j,\ e_j \leq d_j$, and $e_j = s_j + p_j$. We consider also $JISP_1$, where each job can be processed in the single interval $I_{j,1} = [r_j, d_j]$, and $p_j = d_j - r_j$.

We consider general (discrete and continuous) instances, where each job has processing time p_j, a weight w_j, and a processing cost c_j. For some variants we study also classes of instances in which (i) jobs have unit-costs (that is, $c_j = 1\ \forall j$), or (ii) for all the jobs $w_j = p_j$.

The BRS and BRSO problems extend the classic real-time scheduling problem to model the natural goal of gaining the maximum available service for a given budget. In particular, the following practical scenarios yield instances of our problems.[5]

Multi-Vehicle Scheduling on a Path: The vehicle scheduling problem arises in many applications, including robot handling in manufacturing systems and secondary storage management in computer systems (see e.g. [KN-01]). Suppose that a fleet of vehicles needs to service requests on a path. There is an operation cost to each vehicle, and a segment on the line in which the vehicle can provide service. Our objective is to assign the vehicles to service requests on line segments such that the total length of the union of the line segments, i.e., the part of the line which is covered, is maximized, yet the overall cost is within some budget constraints.

Combinatorial Auctions: In auctions used in e-commerce, a buyer needs to complete an order for a given set of goods. There is a collection of *sellers*, each offers a subset (or *bundle*) of the goods at some cost. Each of the goods g_i is associated with a weight w_i, which indicates its priority in the order. The buyer needs to satisfy a fraction of the order of maximum weight, by selecting a subset of the offers, such that the total cost is bounded by the buyer's budget, B. In auctions for *linear* goods (see, e.g., in [T-00]), we have an ordered list of m goods $g_1, \ldots, g_m$, and the offers should refer to bundles of the form $g_i, g_{i+1}, \ldots, g_{j-1}, g_j$. Note that while selecting a subset of the offers we allow overlaps, i.e., the buyer may acquire more than the needed amount from some good; however, this does not decrease the cost of any of the offers. Thus, we get an instance of the $BRSO$ problem, where any job J_j can be processed in one possible time interval.

QoS Upgrade in a Network: Consider an end-to-end connection between s and t that uses several Internet service providers (ISP). Each ISP provides a basic service (for free) and to upgrade the service one needs to pay; that is, an ISP can decrease the delay in its part of the path for a certain cost. (See, e.g., [LORS-00,LO-02].) The end-to-end delay is additive (over all ISP-s). We have a budget and we need to decide on how to distribute it between the ISP-s. In certain scenarios, an ISP may need to choose to upgrade only a portion of the part of the $s-t$ path that it controls, however, it has the freedom to choose which portion. In this problem instance, "jobs" (upgraded segments) are allowed to overlap.

[4] Note that job weights have no effect on the objective function.

[5] Other applications, including transmission of continuous-media data and crew scheduling, are given in [NST-03].

1.1 Our Results

We give hardness results and approximation algorithms for BRS, RSO, and $BRSO$. Specifically, we show that continuous RSO is strongly NP-hard.[6] In the discrete case, both BRS and $BRSO$ are shown to be APX-hard, already for instances where $\forall j, n_j \leq k$ ($JISP_k$), and where all the intervals corresponding to a job have the same length, for any $k \geq 3$. In Section 3, we present a $(2+\varepsilon)$-approximation algorithm for BRS (both discrete and continuous). We build on the framework of Jain and Vazirani [Va-01] for using Lagrangian relaxation in developing approximation algorithms. Our algorithm is based on a novel combination of Lagrangian relaxation with efficient search on the set of feasible solutions. We show that a simple Greedy algorithm yields a 4-approximation for BRS with unit costs, where $w_j = p_j \; \forall j$.

In Section 4, we give a $(2+\varepsilon)$-approximation algorithm for continuous inputs of $BRSO$, and a $(3+\varepsilon)$-approximation for discrete inputs, using the Lagrangian relaxation technique. For RSO we present a Greedy algorithm that achieves the ratio of 2. An improved ratio of $e/(e-1)$ is obtained by a randomized algorithm (where e denotes the base of the natural logarithm). For $JISP_1$, we obtain an optimal solution for instances of $BRSO$ with unit costs, and a *fully polynomial time approximation scheme (FPTAS)* for arbitrary costs. (Note that $JISP_1$ is weakly NP-hard. This can be shown by reduction from Knapsack [GJ-79].) Finally, in Section 5 our results are shown to extend to instances of BRS and $BRSO$ in which the jobs can be scheduled on multiple machines.

The approximation technique that we use for deriving our $(2+\varepsilon)$-approximation results (see in Section 2) is shown to apply to a fundamental class of budgeted maximization problems, including *throughput maximization in a system of dependent jobs*, which generalizes the BRS problem (see in [NST-03]). We show that, using the technique, any problem in the class which has an LP based ρ-approximation with unbounded budget, can be approximated within factor $\rho+\varepsilon$ in the budgeted case, for any $B \geq 1$.

Due to space constraints we state some of the results without proofs.[7]

1.2 Related Work

To the best of our knowledge, the *budgeted* real-time scheduling problem is studied here for the first time. There has been extensive work on real-time scheduling, both in the discrete and the continuous models. Garey and Johnson (cf. [GJ-79]) showed that the continuous case is strongly NP-hard, while the discrete case, JISP, was shown by Spieksma [S-99] to be APX-hard, already for instances of $JISP_k$, where $k \geq 2$. Bar-Noy et al. [BG$^+$99,BB$^+$00] and independently Berman and DasGupta [BD-00] presented 2-approximation algorithms for the discrete case,[8] and $(2+\varepsilon)$ ratio in the continuous case. As shown in [BB$^+$00], this ratio holds for arbitrary number of machines. While none of the existing techniques has been able to improve upon the 2 and $(2+\varepsilon)$ ratios for general instances of the real-time scheduling problem, improved bounds were obtained for some special cases. In particular, Chuzhoy et al. [COR-01] considered the unweighted version, for which they gave an $(e/(e-1)+\varepsilon)$-approximation algorithm, where ε is any constant. For other special cases, they developed polynomial time approximation schemes. Finally, some special cases of JISP were shown to be polynomially solvable (see, e.g., in [AS-87,B-99]).

We are not aware of previous work on the RSO and $BRSO$ problems. Since overlaps are allowed and the goal is to maximize the overall time in which the machine is utilized, these problems

[6] The continuous real-time scheduling problem (with no overlaps) is known to be strongly NP-hard [GJ-79].

[7] The detailed proofs are given in [NST-03].

[8] A 2-approximation for *unweighted* JISP is obtained by a Greedy algorithm, as shown in [S-99].

can be viewed as maximum coverage problems. In previous work on budgeted covering (see, e.g., [KMN-99]), the covering items are *sets*; once a set is selected, the covered elements are uniquely defined. In contrast, in RSO (and $BRSO$) the covering items are *jobs*, and we can choose the time segments (= elements) that will be covered by a job, by determining the time interval in which this job is processed.

2 Approximation via Lagrangian Relaxation

We describe below the general approximation technique that we use for deriving our results for BRS and $BRSO$. Our approach builds on the framework developed by Jain and Vazirani [Va-01][pp. 250-251] (see also [Ga-96]), for using Lagrangian relaxations in approximation algorithms. Our approach applies to the following class of *subset selection* problems. The input for any problem in the class consists of a set of elements $A = \{a_1, \ldots, a_n\}$; each element $a_j \in A$ is associated with a weight w_j, and the cost of adding a_j to the solution set is $c_j \geq 1$. We have a budget $B \geq 1$. The goal is to find a subset of the elements $A' \subseteq A$ satisfying a given set of constraints (including the budget constraint), such that the total weight is maximized. We assume that any problem Π in the class satisfies the following property. **(P1)** Let A' be a feasible solution for Π; then, any subset $A'' \subseteq A'$ is also a feasible solution.

Denote by $x_j \in \{0,1\}$ the indicator variable for the selection of a_j. The integer program for Π has the following form.

$$\begin{array}{lll} (\Pi) & \text{maximize} & \displaystyle\sum_{a_j \in A} w_j x_j \\ & \textit{subject to}: & \text{Constraints}: C_1, \ldots, C_r \\ & & \displaystyle\sum_j c_j x_j \leq B. \end{array}$$

In the linear relaxation we have $x_j \in [0,1]$. The Lagrangian relaxation of this program is

$$\begin{array}{lll} (L-\Pi(\lambda)) & \text{maximize} & \lambda \cdot B \ + \displaystyle\sum_{a_j \in A} (w_j - c_j \lambda) x_j \\ & \textit{subject to}: & \text{Constraints}: C_1, \ldots, C_r \end{array}$$

Assume that $\mathcal{A}_\pi$ is a ρ-approximation algorithm to the optimal integral solution for $L-\Pi(\lambda)$, for any value of $\lambda > 0$. Thus, there exist values $\lambda_1 < \lambda_2$ such that $\mathcal{A}_\pi$ finds integral ρ-approximate solutions $\mathbf{x}_1, \mathbf{x}_2$ for $L-\Pi(\lambda_1), L-\Pi(\lambda_2)$, respectively, and the budgets used in these solutions are B_1, B_2, where

$$B_2 < B < B_1. \tag{1}$$

Let W_1, W_2 denote the weights of the solutions $\mathbf{x}_1, \mathbf{x}_2$, then $W_i = \lambda_i B + \sum_{a_j \in A}(w_j - c_j\lambda_i)\mathbf{x}_{ij}$, $i \in \{1,2\}$, $1 \leq j \leq n$. W.l.o.g, we assume that $W_1, W_2 \geq 1$.

Following the framework of [Va-01], we require that $\mathcal{A}_\pi$ satisfies the following property. Let $\alpha = (B - B_2)/(B_1 - B_2)$, then the convex combination of the solutions $\mathbf{x}_1, \mathbf{x}_2$, namely, $\mathbf{x} = \alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2$ is a (fractional) ρ-approximate solution that uses the budget B. This is indeed the case if, for example, the solutions $\mathbf{x}_1, \mathbf{x}_2$ are obtained from a primal-dual algorithm. In this case, a convex combination of the dual solutions corresponding to $\mathbf{x}_1$ and $\mathbf{x}_2$ can be used to prove this property. This will be heavily used in our algorithms for the BRS and $BRSO$ problems. Our goal is to find a feasible integral solution whose weight is close to the weight of $\mathbf{x}$. We show that for the class of subset selection problem that we consider here, by finding 'good' values of λ_1, λ_2, we obtain an integral solution that is within factor $\rho + \varepsilon$ from the optimal. The running time of our algorithm is dominated by the complexity of the search for λ_1, λ_2 and the running time of $\mathcal{A}_\pi$.

We now summarize the steps of the algorithm, $\mathcal{A}_L$, which gets as input the set of elements $a_1, \ldots, a_n$, an accuracy parameter $\varepsilon > 0$ and the budget $B \geq 1$. Let $c = \sum_j c_j$ denote the total cost of the instance.

1. Let $\varepsilon' = \varepsilon/c$.
2. Define the *modified weight* of an element a_j to be $w'_j = w_j/c_j$.
   Let $\omega_1 \leq \cdots \leq \omega_R$ be the set of R distinct values of modified weights.
3. Find in $(0, \omega_R)$ the values of $\lambda_1 < \lambda_2$
   satisfying (1), such that $\lambda_2 - \lambda_1 \leq \varepsilon'$.
4. Output the (feasible) integral solution found by $\mathcal{A}_\pi$ for $L - \Pi(\lambda_2)$.

Analysis: The heart of our approximation technique is the following theorem.

Theorem 1. *For any $0 < \varepsilon'$ and λ_1, λ_2 satisfying (1), if $0 < \lambda_2 - \lambda_1 < \varepsilon'$, then $W_2 \geq W_1 - \varepsilon' c$, where c is the total cost of the instance.*

Proof. We note that for a fixed value of λ we can omit from the input elements a_j for which $w'_j = w_j/c_j \leq \lambda$. We denote by S_i the *feasible* set of modified weights for λ_i, i.e., the set of values ω_ℓ satisfying $\omega_\ell \geq \lambda_i$; then $S_2 \subseteq S_1$. Let $A_i \subseteq S_i$ be the set of elements selected by $\mathcal{A}_\pi$ for the solution, for the given value λ_i. Then, $W_1 = \lambda_1 B + \sum_{A_1}(w_j - c_j\lambda_1)$ and $W_2 = \lambda_2 B + \sum_{A_2}(w_j - c_j\lambda_2)$. We handle two cases separately.

(i) The feasible sets for λ_1, λ_2 are identical, that is, $S_1 = S_2$. Then

$$W_2 \geq \lambda_2 B + \sum_{A_1}(w_j - c_j\lambda_2) \geq \lambda_1 B + \sum_{A_1}(w_j - c_j(\lambda_1 + \varepsilon')) = W_1 - \varepsilon' \sum_{A_1} c_j \geq W_1 - \varepsilon' c$$

The leftmost inequality follows from the fact that all the elements in A_1 were feasible also with λ_2. (Note that we can guarantee that the inequality is satisfied, by comparing W_2 with the weight of A_1 at λ_2, and by taking the subset which gains the maximum of these two weights.)

(ii) The feasible set for λ_1 contains some elements whose modified weights that are not contained in S_1, that is, $S_2 \subset S_1$. For simplicity, we assume that $S_1 = \{\omega_{\ell+1}, \omega_{\ell+2}, \ldots, \omega_R\}$, while $S_2 = \{\omega_{\ell+2}, \ldots, \omega_R\}$, that is, for some $1 \leq \ell < R$, $\omega_{\ell+1} \in S_1$ and $\omega_{\ell+1} \notin S_2$. In general, several modified weight values may be contained in S_1 but not in S_2. A similar argument can be applied for this case. Denote by $\hat{A}_1$ the subset of elements in A_1 whose modified weights are equal to $\omega_{\ell+1}$. Then,

$$W_2 \geq \lambda_2 B + \textstyle\sum_{A_1 \setminus \hat{A}_1}(w_j - c_j\lambda_2) \geq \lambda_1 B + \sum_{A_1 \setminus \hat{A}_1}(w_j - c_j(\lambda_1 + \varepsilon'))$$

$$= \lambda_1 B + \textstyle\sum_{A_1}(w_j - \lambda_1 c_j) - \sum_{\hat{A}_1}(w_j - \lambda_1 c_j) - \varepsilon' \sum_{A_1 \setminus \hat{A}_1} c_j$$

$$= W_1 - \textstyle\sum_{\hat{A}_1} c_j(\omega_{\ell+1} - \lambda_1) - \varepsilon' \sum_{A_1 \setminus \hat{A}_1} c_j \geq W_1 - \varepsilon' \sum_{A_1} c_j \geq W_1 - \varepsilon' c$$

The first inequality is due to the fact that the set of elements $A_1 \setminus \hat{A}_1$ was available with λ_2, and that Π satisfies property **(P1)**; the second inequality follows from the difference $(\lambda_2 - \lambda_1)$ being bounded by ε'; the last inequality follows from $(\omega_{\ell+1} - \lambda_1) < \varepsilon'$. This completes the proof. □

Let $0 < \varepsilon < 1$ be an input parameter, then taking $\varepsilon' = \varepsilon/c$, we get from Theorem 1 that

$$W_2 \geq (W_1 - \varepsilon' c)\alpha + W_2(1-\alpha) \geq (W_1\alpha + W_2(1-\alpha)) - \varepsilon' c \geq (W_1\alpha + W_2(1-\alpha))(1-\varepsilon).$$

Finally, since $\mathbf{x}$ gives a ρ-approximation to the optimal, we get

Theorem 2. *Algorithm $\mathcal{A}_L$ achieves an approximation factor of $(\rho + \varepsilon)$ for Π.*

Implementation: Note that to obtain a $(\rho + \varepsilon)$-approximation, we need to find the values of $\lambda_1, \lambda_2 \in (0, \omega_R)$ that satisfy (1), such that $(\lambda_2 - \lambda_1) < \varepsilon/c$. As $\omega_R = \max_j w_j/c_j$ may be arbitrarily large, a naive search may require exponential number of steps. We show that by allowing a small increase (of ε) in the approximation ratio, we can implement this search in polynomial time. (i) Initially, we guess the weight of an optimal integral solution, W^*, to within factor $(1-\varepsilon)$. This can be done in $O(\lg(n/\varepsilon))$ steps, since $\max_j w_j \leq W^* \leq n \cdot \max_j w_j$. We then omit from the input any element a_j whose weight is smaller than $\varepsilon W^*/n$. We scale the weights of the remaining elements, so that all the weights are in the range $[1, n/\varepsilon]$. (ii) For any element a_j with $c_j < \varepsilon B/n$, we round up c_j to $\varepsilon B/n$. We scale the other costs, such that all costs are in $[1, n/\varepsilon]$. (iii) We scale accordingly the size of the interval $(0, \omega_R)$.

Now, we argue that in the above scaling and rounding we only slightly decrease the weight of the solution. Indeed, by omitting elements with 'small' weights, we decrease the total weight of the elements selected by $\mathcal{A}_\pi$ at most by factor of ε. Also, by rounding up the 'small' costs to $\varepsilon B/n$, we get that the total weight obtained by $\mathcal{A}_\pi$ at λ_2 is at least $\lambda_2 B + \sum_{a_j \in A_2} \left(w_j - (c_j + \frac{\varepsilon B}{n})\,\lambda_2\right) \geq W(1-\varepsilon)$. Thus, overall we lose a factor of 2ε in the approximation ratio. The overall running time of our search procedure is $O(\lg(n/\varepsilon) \cdot \lg(n^3/\varepsilon^3)) = O(\lg^2(n/\varepsilon^3))$. It follows that the running time of $\mathcal{A}_L$ is $O(\lg^2(n/\varepsilon^3))$ times the running time of algorithm $\mathcal{A}_\pi$.

3 Approximating Bounded Real-time Scheduling

3.1 A Greedy Algorithm

Consider first the special case of unit cost jobs, where $w_j = p_j \ \forall\, j$. Suppose that the budget is $B = k$; thus, we need to select a subset of k non-overlapping jobs, such that machine utilization is maximized. For such instances, we can obtain a constant approximation using an $O(n \log n)$ greedy algorithm. The algorithm $\mathcal{A}_G$ (formulated for continuous inputs) first sorts the jobs in non-increasing order by their processing times. Then, $\mathcal{A}_G$ schedules at most k jobs by scanning the sorted list; that is, while there is available budget, the next job J_j is scheduled in the earliest available time interval in $[r_j, d_j]$.

Theorem 3. *$\mathcal{A}_G$ is a $(4+\varepsilon)$-approximation for BRS with unit costs where $w_j = p_j \ \forall\, j$. In the discrete case, $\mathcal{A}_G$ achieves the ratio of 4.*

As we show below, better ratio can be achieved, for *general* instances, by using the Lagrangian relaxation technique.

3.2 A $(2+\varepsilon)$-approximation Algorithm

In the following we derive a $(2+\varepsilon)$-approximation for discrete instances of BRS. A similar result can be obtained for the continuous case, by discretizing the instance. Recall that in the discrete case, any job J_j can be scheduled in the intervals $I_{j,1}, \ldots, I_{j,n_j}$. We define a variable $x(j,\ell)$ for each interval $I_{j,\ell}$, $1 \leq j \leq n, 1 \leq \ell \leq n_j$. Then the integer program for the problem is:

$$
\begin{array}{llll}
(BRS) & \text{maximize} & \displaystyle\sum_{j=1}^{n}\sum_{\ell=1}^{n_j} w_j x(j,\ell) & \\
& \textit{subject to}: & \forall j: & \displaystyle\sum_{\ell=1}^{n_j} x(j,\ell) \le 1 \\
& & \forall t: & \displaystyle\sum_{t \in I_{j,\ell}} x(j,\ell) \le 1 \\
& & \displaystyle\sum_{j=1}^{n}\sum_{\ell=1}^{n_j} c_j x(j,\ell) \le B. &
\end{array}
$$

In the linear program $x_j \in [0,1]$. Taking the Lagrangian relaxation, we get an instance of the *throughput maximization* problem. As shown in [BB$^+$00], an algorithm based on the local ratio technique yields a 2-approximation for this problem, in $O(n^2)$ steps. This algorithm has a primal-dual interpretation; thus, we can apply the technique in Section 2 to obtain an algorithm, $\mathcal{A}$, which uses the algorithm for throughput maximization as a procedure.[9]

Theorem 4. *Algorithm $\mathcal{A}$ yields a $(2+\varepsilon)$-approximation for BRS, in $O(n^2 \lg^2(n/\varepsilon^2))$ steps.*

4 Approximation Algorithms for *RSO* and *BRSO*

In this section, we present approximation algorithms for the *RSO* and *BRSO* problems. In Section 4.1 we consider *RSO*. We first give a randomized $e/(e-1)$-approximation algorithm for discrete inputs; then, we describe a greedy algorithm that achieves a ratio of $(2-\epsilon)$ for continuous inputs, and $(3-\epsilon)$ for discrete inputs. In Section 4.2, we show that the greedy algorithm can be interpreted equivalently as a primal-dual algorithm. This allows us to apply the Lagrangian relaxation framework (Section 2) and to achieve a $(3+\varepsilon)$-approximation for *BRSO* in the discrete case, where all the intervals corresponding to a job have the same length. For continuous inputs we obtain a $(2+\varepsilon)$-approximation.

4.1 The *RSO* Problem

In the RSO problem, we may select *all* the jobs, and the problem reduces to scheduling the jobs optimally so as to maximize the coverage of the line. Clearly, when $\forall j, p_j = d_j - r_j$, i.e., each job has only one possible interval, the schedule in which all the jobs are selected is optimal. When $\forall j, p_j \le d_j - r_j$, the problem becomes hard to solve (see in [NST-03]).

Theorem 5. *The RSO problem is strongly NP-hard.*

[9] Note that since $W_1, W_2 \ge \max_j w_j$, in our search for λ_1, λ_2 we can take $\varepsilon' = \varepsilon/n$.

A Randomized $e/(e-1)$-approximation algorithm We start with a linear programming formulation of RSO. Assume that the input is given in a discrete fashion, and let $b_0, \ldots, b_m$ denote the set of start and end points (in sorted order), called *breakpoints*, of the time intervals $I_{j,\ell}$, $j = 1, \ldots, n$, $\ell = 1, \ldots, n_j$. We have a variable $x(j, \ell)$ for each interval $I_{j,\ell}$. For any pair of consecutive breakpoints b_{i-1} and b_i, the objective function gains $(b_i - b_{i-1})$ times the "coverage" of the interval $[b_{i-1}, b_i]$. Note that we take the minimum between 1 and the total cover, since we gain nothing if some interval is covered by more than one job.

$$(L-RSO) \ \text{maximize} \quad \sum_{i=1}^{m} \min\left(\sum_{j=1}^{n} \sum_{I_{j,\ell} \ni [b_{i-1}, b_i]} x(j,\ell), 1\right) \cdot (b_i - b_{i-1})$$

$$\textit{subject to}: \qquad \text{For all jobs } J_j\text{:} \quad \sum_{\ell=1}^{n_j} x(j,\ell) \leq 1$$

$$\text{For all } (j,\ell), \ell = 1, \ldots, n_j\text{:} \ \ x(j,\ell) \geq 0.$$

We compute an optimal (fractional) solution to $(L-RSO)$. Clearly, the value of this solution is an upper bound on the value of an optimal integral solution. To obtain an integral solution, we apply randomized rounding to the optimal fractional solution. That is, for every job J_j, the probability that J_j is assigned to interval $I_{j,\ell}$ is equal to $x(j,\ell)$. If $\sum_{\ell=1}^{n_j} x(j,\ell) < 1$, then with probability $1 - \sum_{\ell=1}^{n_j} x(j,\ell)$ job J_j is not assigned to any interval.

We now analyze the randomized rounding procedure. Consider two consecutive breakpoints b and b'. Define for each job J_j, $y_j = \sum_{I_{j,\ell} \ni [b,b']} x(j,\ell)$. Clearly, $\sum_{j=1}^{n} y_j = \sum_{j=1}^{n} \sum_{I_{j,\ell} \ni [b,b']} x(j,\ell)$. W.l.o.g., since each job J_j is assigned to a single interval, we can think of all the intervals of J_j that cover $[b, b']$ as a single (virtual) interval that is chosen with probability y_j. The probability that none of the virtual intervals is chosen is $P_0 = \prod_{j=1}^{n}(1 - y_j)$. Let $r = \min(\sum_{j=1}^{n} y_j, 1)$. Then,

$$P_0 \leq \prod_{j=1}^{n} \left(1 - \frac{\sum_{i=1}^{n} y_i}{n}\right) = \left(1 - \frac{\sum_{i=1}^{n} y_i}{n}\right)^n < e^{-\sum_{i=1}^{n} y_i} \leq e^{-r}.$$

Hence, the probability that $[b, b']$ is covered is $1 - P_0 \geq 1 - e^{-r} \geq \left(1 - \frac{1}{e}\right) \cdot r \geq \left(1 - \frac{1}{e}\right) \cdot \min\left(\sum_{j=1}^{n} y_j, 1\right)$. Therefore, the expected contribution to the objective function of any interval $[b_{i-1}, b_i]$ is $(1 - 1/e) \cdot \min(\sum_{j=1}^{n} y_j, 1) \cdot (b_i - b_{i-1})$. By linearity of expectation, the expected value of the objective function after applying randomized rounding is

$$\left(1 - \frac{1}{e}\right) \cdot \sum_{i=1}^{m} \min\left(\sum_{j=1}^{n} \sum_{I_{j,\ell} \ni [b_{i-1}, b_i]} x(j,\ell), 1\right) \cdot (b_i - b_{i-1}),$$

yielding an approximation factor of $1 - 1/e \approx 0.63212$.

A Greedy Approximation Algorithm We now describe a greedy algorithm, which yields a $(2-\varepsilon)$-approximation for continuous instances of RSO, and $(3-\varepsilon)$-approximation for discrete instances. Assume that $\min_j r_j = 0$, and let $T = \max_j d_j$. Let I be the set of all the jobs in the instance; U is the set of unscheduled jobs. Denote by s_j, e_j the start-time and completion time of the job J_j in the greedy schedule, respectively. Given a partial schedule, we say that J_ℓ is *redundant* if we can remove J_ℓ from the schedule without decreasing the machine utilization. Algorithm Greedy proceeds in the interval $[0, T]$. At time t, we select an arbitrary job, among the jobs J_i with $r_i < t$ and $d_i > t$. We schedule this job such that its contribution to the utilization, starting at time t, is maximized. The following is a pseudocode of the algorithm.

1. $U = I$, $t = 0$;
2. Let $J_j \in U$ be a job having $d_j > t$ and $r_j \leq t$.
   Schedule J_j such that its completion time, e_j, is $\min(t + p_j,\ d_j)$.
   Remove J_j from U.
   For any redundant job J_ℓ, omit J_ℓ from the schedule and return it to U.
3. Let $F \subseteq U$ be the set of unscheduled jobs, i, having $d_i > e_j$.
   Let $t_F = \min_{J_i \in F} r_i$, and let $t = \max(e_j,\ t_F)$.
   If $F \neq \emptyset$ and $t < T$ go to step 2.

We use in the analysis the following properties of the greedy schedule.

Property 1. Once an interval $[x_1, x_2] \in [0, T]$ is covered by Greedy, it remains covered until the end of the schedule.

Property 2. When the algorithm considers time t, some job will be selected and scheduled such that for some $\varepsilon > 0$, the machine is utilized in the interval $[t, t + \varepsilon]$.

Property 3. Consider the set U of non-scheduled jobs at the end of the schedule. For any $J_j \in U$ the machine is utilized in the time interval $[r_j, d_j)$.

Theorem 6. *The Greedy algorithm yields a $(2 - \varepsilon)$-approximation for the RSO problem.*

Proof. Let $S = I \setminus U$ denote the set of jobs scheduled by Greedy, and let $O \subseteq S$ denote the set of scheduled job such that $J_j \in O$ iff J_j overlaps with another scheduled job, J_k, and $e_j > e_k$. (i) By Property 3, for any $J_j \in U$ the machine is utilized in the time interval $[r_j, d_j]$. (ii) For any $J_j \in S$ the machine is utilized in the time interval $[r_j, s_j]$, otherwise Greedy would have scheduled it earlier. (iii) For any $J_j \in O$ the machine is utilized in the time interval $[r_j, d_j]$. This follows from (ii) and from the fact that $e_j = d_j$, otherwise J_j would not overlap with a job with an earlier completion time.

Given the schedule of Greedy, we allow OPT to add jobs of U and to shift the scheduled jobs of S in any way that increases the utilization. Consider the three *disjoint* sets of jobs, $U, O, S \setminus O$. By the above discussion, the utilization can be increased only by shifting to the left (i.e., scheduling earlier) the jobs of $S \setminus O$. Note that in any time $t \in [0, T]$ there is at most one job, J_j, of $S \setminus O$ (if two or more jobs overlap, then only the one with the earliest completion time is in $S \setminus O$). Let $0 < \varepsilon_j \leq 1$ be such that J_j overlaps in the Greedy schedule in $(1 - \varepsilon_j)$ of its length. Then OPT can shift J_j into an interval in which it does not overlap at all. Hence, OPT can increase the amount of time the machine is utilized in the greedy schedule at most by factor of $(2 - \varepsilon)$. □

The analysis for discrete inputs is similar, only that in this case, by selecting for two jobs overlapping at time t different intervals, and by adding a non-scheduled job to run at t, OPT may triple the utilization obtained by Greedy, thus we get the bound of $(3 - \varepsilon)$.

4.2 The *BRSO* Problem

As *BRSO* generalizes the *RSO* problem, Theorem 5 implies that it is strongly NP-hard. For discrete inputs we show the following (see in [NST-03]).

Theorem 7. *The discrete BRSO is APX-hard, already for instances where $\forall j, n_j \leq k$ ($JISP_k$), for any $k \geq 3$.*

A Primal-Dual Algorithm We first present a primal-dual algorithm for RSO, and show that an execution of the Greedy algorithm given in Section 4.1 can be equivalently interpreted as an execution of the primal-dual algorithm. Thus, the primal-dual algorithm finds a 3-approximate solution to RSO. The primal LP is equivalent to $L-RSO$, given in Section 4.1. We have a variable $x(j,\ell)$ for each interval $I_{j,\ell}$, and a variable $z_i,\ i=1,\ldots,m$ for each interval $[b_{i-1},b_i]$ defined by consecutive breakpoints. In the dual LP we have a variable y_j for each job J_j, and two variables, p_i and q_i for each interval $[b_{i-1},b_i]$ defined by consecutive breakpoints.

$$
\begin{array}{llll}
(L-RSO-Primal) & \text{maximize} & \displaystyle\sum_{i=1}^{m} z_i \cdot (b_i - b_{i-1}) & \\
\textit{subject to}: & \text{For all jobs } J_j\text{:} & \displaystyle\sum_{\ell=1}^{n_j} x(j,\ell) \le 1 & \\
 & \text{For all } i=1,\ldots,m & : z_i \le 1 & \\
 & \text{For all } i=1,\ldots,m & : z_i - \displaystyle\sum_{I_{j,\ell}\ni[b_{i-1},b_i]} x(j,\ell) \le 0 & \\
 & \text{For all } j,\ell,i\text{:} & x(j,\ell), z_i \ge 0. &
\end{array}
$$

$$
\begin{array}{llll}
(L-RSO-Dual) & \text{minimize} & \displaystyle\sum_{j=1}^{n} y_j + \sum_{i=1}^{m} p_i & \\
\textit{subject to}: & \text{For all } (j,\ell), \ell=1,\ldots,n_j\text{:} & y_j - \displaystyle\sum_{I_{j,\ell}\ni[b_{i-1},b_i]} q_i \ge 0 & \\
 & \text{For all } i=1,\ldots,m\text{:} & p_i + q_i \ge (b_i - b_{i-1}) & \\
 & \text{For all } j,i\text{:} & y_j, p_i, q_i \ge 0. &
\end{array}
$$

Given an integral solution for $L-RSO-Primal$, we say that an interval I belongs to it if there is a job that is assigned to I. An integral solution for $L-RSO-Primal$ is *maximal*, if it cannot be extended and if no interval belonging to it is contained in the union of other intervals belonging to it.

Lemma 1. *Any maximal integral solution (x,z) to $L-RSO-Primal$ is a 3-approximate solution.*

Proof. If $[b_{i-1},b_i]$ is covered by (x,z), then set $p_i=b_i-b_{i-1}$, otherwise set $q_i=b_i-b_{i-1}$. Clearly, this defines a feasible dual solution in which $\sum_{i=1}^{m} p_i = \sum_{i=1}^{m} z_i \cdot (b_i-b_{i-1})$. Thus, it remains to bound $\sum_{j=1}^{n} y_j$ in this solution.

For each job J_j that is not assigned to any interval in (x,z), i.e., its intervals are contained in intervals of other jobs, we can set $y_j=0$. Suppose that for job J_j, $x(j,\ell)=1$. Consider, for example, an interval $I_{j,\ell'}$, $\ell \ne \ell'$, that contains two consecutive breakpoints b_{i-1} and b_i such that $[b_{i-1},b_i]$ is not covered by any job. In this case $q_i=b_i-b_{i-1}$ and $y_j \ge q_i$. Thus, in order to bound $\sum_{j=1}^{n} y_j$, we say that the values of q_i-s that determine the y_j-s "charge" the p_i values corresponding to the breakpoints covered by $I_{j,\ell}$. This can be done since all the intervals in which J_j can be scheduled have the same length. Since our primal solution is maximal, any point is covered by at most two intervals to which jobs are assigned, and therefore any variable p_i can be "charged" by intervals belonging to at most two different jobs. Thus, $\sum_j y_j \le 2\sum_i p_i$, proving that $\sum_{i=1}^{m} p_i = \sum_{i=1}^{m} z_i \cdot (b_i-b_{i-1}) \le \sum_{j=1}^{n} y_j + \sum_{i=1}^{m} p_i \le 3 \cdot \sum_{i=1}^{m} p_i$, meaning that (x,z) is a 3-approximate solution. □

For continuous instances, we can discretize time such that each time slot is of size ϵ. This will incur a $(1+\epsilon')$ degradation in the objective function where $\epsilon' = \text{poly}(\epsilon, n)$. We can show that for a discrete input obtained from a discretization of a continuous input instance, the primal-dual algorithm yields a 2-approximate solution. Applying the Lagrange relaxation technique (presented in Section 2), we get the following.

Theorem 8. *BRSO can be approximated within factor $(2+\varepsilon)$ in the continuous case, and $(3+\varepsilon)$ in the discrete case, in $O(n^2 \cdot \lg^2(n/\varepsilon^2))$ steps.*

4.3 An FPTAS for $JISP_1$

For instances of $BRSO$ where $p_j = d_j - r_j$ ($JISP_1$), we use a reduction to the *budgeted longest path problem* in acyclic graph, to obtain an optimal polynomial time algorithm for unit costs, and an FPTAS for general instances. In the budgeted longest path problem, we are given an acyclic graph, $G(V,E)$; each edge $e \in E$ has a length $\ell(e)$, and a cost $c(e)$. Our goal is to find the longest path in G connecting two given vertices s,t, whose price is bounded by a given budget B. The problem is polynomially solvable for unit edge costs, and has an FPTAS for arbitrary costs [Ha-92].

Given an instance of $BRSO$ where $\forall j,\ p_j = d_j - r_j$, we construct the following graph, G. Each job j is represented by a vertex; there is an edge $e = (i,j)$ iff $d_i < d_j$ and $r_i \le r_j$. The length of the edge is $\ell(e) = \min(d_j - d_i, p_j)$, and its cost is c_j. Note that $\ell(e)$ reflects the machine utilization gained if the deadlines of J_i, J_j are adjacent to each other in the schedule. In addition, each vertex j is connected to the source, s, where $\ell(s,j) = p_j$, $c(s,j) = c_j$, and to a sink t, where $\ell(j,t) = 0$ and $c(s,j) = 0$.

Theorem 9. *There is a schedule achieving utilization of u time units and having cost $b \le B$ if and only if G contains a path of length u and price b.*

Proof. For a given schedule, sort the jobs in the schedule such that $d_{j_1} \le d_{j_2} \le \ldots \le d_{j_w}$. W.l.o.g, the schedule does not include two jobs J_i, J_j such that $r_j < r_i$ and $d_i < d_j$, since in such a schedule we gain nothing from processing J_i. Thus, we can assume that $r_{j_i} \le r_{j_{i+1}}$, $\forall 1 \le i \le w$. This implies that the graph G contains the path $s, j_1, j_2, \ldots, j_w, t$ (the first and last edges in this path exist by the structure of G). Suppose that the utilization of the schedule is u and its cost is b. We show that the length of the corresponding path in G is u and its cost is b. Recall that the edge (j_w, t) has length 0 and costs nothing, thus, we consider only the first w edges in this path. The utilization we gain from scheduling j_i is p_{j_i} if $i = 1$ and $\min(p_{j_i}, d_{j_i} - d_{j_{i-1}})$ if $1 < i \le w$. This is exactly $\ell(j_{i-1}, j_i)$ (or $\ell(s, j_1)$ for the first vertex in the path). Also, the cost of the schedule is the total processing cost of the scheduled jobs, which is identical to the total cost of edges in the path.

For a given directed path in G, we schedule all the jobs whose corresponding vertices appear on the path. Note that the price of the path consists of the price of the participating vertices, thus, b is also the price of the schedule. Also, as discussed above $\ell(i,j)$ reflects the contribution of the corresponding job to the utilization, thus the path induces a schedule with the correct utilization and cost. □

5 Multiple Machines

Suppose that we have m machines, and a budget B, which can be distributed in any way among the machines. It can be shown that this model is equivalent to the single machine case, by concatenating the schedules on the m machines to a single schedule in the interval $[0, mT]$, on a single machine.

Thus, all of our results carry over to this model. When we have a budget specified for each machine, we show that any approximation algorithm $\mathcal{A}$ for a single machine can be run iteratively on the machines and the remaining jobs, to obtain a similar approximation ratio. Denote this algorithm $\mathcal{A}^*$.

Theorem 10. *If $\mathcal{A}$ is an r-approximation then the iterative algorithm $\mathcal{A}^*$ is an $(r+1)$-approximation.*

Note that in most cases $\mathcal{A}^*$ performs better. For example, when we iterate Greedy (Section 4.1) for the *RSO* problem, it can be seen that the proof for a single machine is valid also for multiple machines, thus Greedy is $(2-\varepsilon)$-approximation. In the full version of the paper, we show that our results can be extended to apply also for the case where the processing costs of the jobs are machine dependent, that is, the cost of processing J_j on the k-th machine is c_{jk}, $1 \leq j \leq n$, $1 \leq k \leq m$.

Acknowledgments. We thank Shmuel Zaks for encouraging us to work on RSO and its variants. We also thank Magnús Halldórsson and Baruch Schieber for valuable discussions.

References

[AS-87] E.M. Arkin and E.B. Sliverberg. "Scheduling jobs with fixed start and end times". *Discrete Applied Math.*, 18:1–8, 1987.

[B-99] P. Baptiste. Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times. *J. of Scheduling*, 2:245–252, 1999.

[BB+00] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *J. of the ACM*, 48:1069-1090, 2001.

[BG+99] A. Bar-Noy, S. Guha, J. Naor and B. Schieber Approximating the throughput of real-time multiple machine scheduling. *SIAM J. on Computing*, 31:331-352, 2001.

[BD-00] P. Berman and B. DasGupta. "Multi-phase algorithms for throughput maximization for real-time scheduling." *J. of Combinatorial Optimization*, **4**:307–323, 2000.

[COR-01] J. Chuzhoy, R. Ostrovsky, and Y. Rabani. "Approximation algorithms for the job interval selection problem and related scheduling problems." *FOCS*, 2001.

[GJ-79] M.R. Garey and D.S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* W.H. Freeman, 1979.

[Ga-96] N. Garg, A 3-approximation for the minimum tree spanning k vertices, Proceedings of FOCS, 1996.

[Ha-92] R. Hassin, Approximation schemes for the restricted shortest path problem. In *Mathematics of Operations Research*, 17(1): 36–42, 1992.

[K-91] V. Kann. Maximum bounded 3-dimensional matching is max SNP-complete. *Information Processing Letters*, 37:27–35, 1991.

[KN-01] Y. Karuno and H. Nagamochi, A 2-approximation algorithm for the multi-vehicle scheduling problem on a path with release and handling times. *ESA*, 2001.

[KMN-99] S. Khuller, A. Moss, J. Naor, The budgeted maximum coverage problem. *Information Processing Letters* 70(1): 39-45, 1999.

[LO-02] D. H. Lorenz and A. Orda, Optimal partition of QoS requirements on unicast paths and multicast trees, IEEE/ACM Trans. on Networking, 10(1), 102–114, 2002.

[LORS-00] D. H. Lorenz, A. Orda, D. Raz, Y. Shavitt, Efficient QoS partition and routing of unicast and multicast, *8th Int. Workshop on Quality of Service*, Pittsburgh, 2000.

[NST-03] J. Naor, H. Shachnai and T. Tamir Real-time Scheduling with a Budget, `http://www.cs.technion.ac.il/~hadas/PUB/rtbudget.ps`.

[S-99] F. C. R. Spieksma. "On the approximability of an interval scheduling problem." *J. of Scheduling,* **2**:215–227, 1999.

[T-00] M. Tennenholtz. Some tractable combinatorial auctions. *AAAI/IAAI*, 98-103, 2000.

[Va-01] V. Vazirani, Approximation algorithms, Springer Verlag, 2001.