Approximation Schemes for Scheduling and Covering on Unrelated Machines

Pavlos S. Efraimidis and Paul G. Spirakis

2004
Approximation Schemes for Scheduling and Covering on Unrelated Machines

Pavlos S. Efraimidis\textsuperscript{a,*}, Paul G. Spirakis\textsuperscript{b,1}

\textsuperscript{a}Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece
\textsuperscript{b}Computer Technology Institute, Riga Feraioi 61, 26221 Patras, Greece

Abstract

We examine the problem of assigning \( n \) independent jobs to \( m \) unrelated parallel machines, so that each job is processed without interruption on one of the machines, and at any time, every machine processes at most one job. We focus on the case where \( m \) is a fixed constant, and present an algorithmic framework that reveals, in a uniform way, approximation schemes for the following assignment problems: Minimizing the maximum load on any machine, the generalized assignment problem, multi-objective scheduling with a constant number of linear cost functions, maximizing the minimum load on any machine, and assigning equal load to all machines.

\textit{Key words:} Randomized Rounding, Approximation Schemes, Multi-Objective Scheduling, Covering, Derandomization

1 Introduction

We examine the problem of assigning \( n \) independent jobs to \( m \) unrelated parallel machines, so that each job is processed without interruption on one of the machines, and at any time, every machine processes at most one job. We focus on the case where \( m \) is a fixed constant, and present an algorithmic

\* Corresponding author.

\textit{Email addresses:} pefraimi@ee.duth.gr (Pavlos S. Efraimidis), spirakakis@cti.gr (Paul G. Spirakis).

\textsuperscript{1} Part of this work is funded under the EU IP project DELIS in the proactive initiative on Complex Systems.
framework that reveals, in a uniform way, approximation schemes for the following assignment problems: Minimizing the maximum load on any machine, the generalized assignment problem, multi-objective scheduling with a constant number of linear cost functions, maximizing the minimum load on any machine, and assigning equal load to all machines.

1.1 The Technique

Each problem is first formulated as an integer program (IP), then a fractional solution is found for the corresponding relaxed linear program (LP), and finally, the fractional solution is rounded to a feasible solution. The rounding step is based on randomized rounding, a probabilistic method to convert a solution of a relaxed combinatorial optimization (CO) problem into an approximate solution to the original problem. Randomized rounding has been introduced by Raghavan and Thompson in (17) and has found many applications in approximation algorithms.

The heart of our algorithms is the Combinatorial Randomized Rounding (CRR) technique, a class of carefully guided randomized rounding procedures that preprocess the CO problems, perform randomized rounding and then, if necessary, correct any bad choices made during the randomized rounding step.

In CRR rounding procedures, the most critical decisions that must be taken are identified, and then each critical decision is either neutralized or a backup choice is provided for it. In order to be able to tolerate a number of backup choices after the rounding step, the problems are preprocessed and appropriate combinatorial arguments are applied.

We show that CRR, when it applies, achieves approximation ratios up to any given constant ratio, while conventional randomized rounding typically guarantees only approximation factors that are logarithmic on \( m \). This is an important improvement even for the case where \( m \) is a fixed constant. Moreover, all CRR-based randomized algorithms derived in this work are derandomized with the method of conditional probabilities (16) and produce deterministic algorithms of equivalent time complexity.

The importance of using randomness not only in a blind but also in an intelligent form has recently been stressed in the fields of heuristics (3) and approximation algorithms. A characteristic example is the use of randomization with two or more choices that has been studied in several recent works (eg. (14; 20)). Even though conventional randomized rounding cannot be considered a blind rounding procedure, since it uses the fractional values of the variables to guide the rounding process, we show with CRR that, for certain problem classes, improved approximations are achieved by enhanced rounding
processes.

1.2 Definitions

All problems considered in this work concern the assignment of \( n \) independent jobs to \( m \) parallel machines, where \( m \) is a fixed constant. The processing time of job \( j \) on machine \( i \) is \( p_{ij} \). A summary of the notation used in this paper is given in Appendix A. We consider the following problems:

**SCHED:** The scheduling problem with the objective to minimize the maximum load on any machine, i.e. the makespan of the schedule.

**GAP:** The generalized assignment problem, where the assignment of each job \( j \) to a machine \( i \) has, besides the processing time \( p_{ij} \), a cost \( c_{ij} \). The objective is to find a schedule of bounded makespan and cost.

**MSCHED:** Multi-objective scheduling, the generalization of SCHED and GAP, which admits an arbitrary constant number of linear ”cost” functions.

**COV:** The symmetric problem of SCHED where the objective is to maximize the minimum load of any machine.

**EQUAL:** The problem of minimizing the makespan when all machines must have equal load.

**Preliminaries.** A polynomial time approximation scheme (PTAS) is an algorithm that for each \( \epsilon > 0 \), finds a \((1 + \epsilon)\)-approximate solution in time polynomial in the problem size \( N \). A PTAS with running time polynomial on \( \epsilon \) is a fully polynomial time approximation scheme (FPTAS). A relaxed decision procedure (RDP), accepts as input a problem instance, and a value \( T \) for the objective function, and either finds a feasible solution with objective value at most \((1 + \epsilon)T\), or decides that there is no solution with objective value at most \( T \). A randomized RDP (RRDP) is a randomized relaxed decision procedure, such that the probability that it may fail to find a relaxed solution for a feasible problem instance is at most a given value \( \rho : 0 < \rho < 1/2 \). Similarly we define randomized PTAS (RPTAS) and randomized FPTAS (RFPTAS). Detailed definitions of approximation schemes and a thorough presentation of their application in scheduling problems are given in (18).

1.3 Results

We present RDP decision procedures and FPTAS approximation schemes for SCHED, GAP, MSCHED, and COV and a RDP decision procedure and a PTAS approximation scheme for EQUAL. All algorithms have linear or quasi-linear time complexities, except the algorithms for EQUAL which require a high degree polynomial time. The results are summarized in the following
Figure 1. The constant values $m$ and $\epsilon$ introduce a significant constant factor into the time complexities.

<table>
<thead>
<tr>
<th>The Problem</th>
<th>This work</th>
<th>Known result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHED</td>
<td>$O(n)$ FPTAS</td>
<td>$O(n)$ FPTAS (11)</td>
</tr>
<tr>
<td>GAP</td>
<td>$O(n)$ RDP $O(n \log n)$ FPTAS</td>
<td>$O(n)$ RDP (11)</td>
</tr>
<tr>
<td>MSCHED</td>
<td>$O(n)$ RDP $O(n \log n)$ FPTAS</td>
<td>–</td>
</tr>
<tr>
<td>COV</td>
<td>$O(n)$ RDP $O(n \log n)$ FPTAS</td>
<td>m Identical Machines (21) m Related Machines (2)</td>
</tr>
<tr>
<td>EQUAL</td>
<td>$Poly(n)$ RDP $Poly(n \log n)$ PTAS</td>
<td>–</td>
</tr>
</tbody>
</table>

Fig. 1. Results.

**Related Work:** Scheduling is an active, widely studied field (8; 1; 12; 19). For the SCHED problem it is known that it is NP-hard, even when $m = 2$. A FPTAS for SCHED was given in (10), and later, an interesting PTAS for SCHED was given in (13). The best known results for SCHED and GAP, are a linear time FPTAS for SCHED and and a linear time RDP decision procedure for GAP, given by Jansen and Porkolab in (11). For COV, PTAS approximation schemes are presented in (21) for identical machines and in (2) for related machines. Both algorithms are polynomial on $m$. We are not aware of algorithms for the case of unrelated machines, even when $m$ is a constant. EQUAL is a hard CO problem, since the problems partition and subset sum are special cases of it (see problems SP12 and SP13 in (5)), when $m = 1$ or $m = 2$ and the machines are identical. We are not aware of algorithms for EQUAL.

We consider the CRR approach as the most important contribution of this work. CRR is an innovative, conceptually simple way to amplify the performance of randomized rounding. Its main achievement is the way it provides strong approximation ratios, by neutralizing the bad events that can make the deviation bounds weaker. The CRR approach is supported by a set of interesting algorithmic techniques. We derive appropriate Chernoff and Hoeffding-Chernoff bounds and show how to exploit their properties within a general randomized rounding procedure. Linear programming techniques and standard combinatorial arguments are used to support the randomized rounding procedure. Decision procedures are used to achieve tighter LP relaxations. An interesting property of CO problems, the poly-bottleneck property, is defined
and applied for building approximation schemes from relaxed decision procedures. Finally, from (11), which was the original motivation for our work, we draw the idea to distinguish between large and small jobs. The outcome is a general algorithmic framework that produces in a uniform way randomized approximation schemes for an important class of assignment problems. Finally, we convert all randomized approximation schemes to deterministic approximation schemes with the method of conditional probabilities and in particular with the use of pessimistic estimators. We provide efficient pessimistic estimators that produce deterministic approximation schemes of equivalent time complexity with the corresponding randomized schemes.

The CRR-based approach teamed up with the method of conditional probabilities, combines the elegance, the generality and the simplicity of a randomized technique with the advantages of deterministic algorithms. It handles in a uniform way a class of interesting job assignment problems and produces, in spite of its generality, efficient approximation schemes. For SCHED and GAP, the CRR-based algorithms match the best known results of (11). In our view, the CRR-based algorithms are not only more general, but also simpler and more elegant than the results of (11). The advantages of CRR become especially evident on GAP, which is handled by CRR in a simple, general way like the other assignment problems in contrast to the involved rounding scheme of (11). Besides SCHED and SCHED with a linear cost constraint (i.e. GAP), the CRR-based approach can handle SCHED with any fixed number of linear cost constraints. To our knowledge, this is the most general result known for this class of problems. Furthermore, the same CRR-approach reveals equivalent approximation schemes for the symmetric problem of scheduling, the machine covering problem. The known results for machine covering concern identical and related machines, and hence this is the first result for this class of covering problems. Finally we show how CRR can handle problems with both packing and covering constraints like the equi-distribution of load to unrelated machines (EQUAL).

The rest of this work is organized in the following way. The basic concepts of the CRR approach are presented in Section 2. The SCHED problem is considered in Section 3. Algorithms for GAP and MSCHED are discussed in Section 4. The problems COV and the EQUAL are discussed in Section 5 and Section 6, respectively. The derandomization of all randomized algorithms is described in Section 7. Appropriate Chernoff and Hoeffding-Chernoff bounds for the randomized algorithms and their derandomization are derived in Section 8. The enumeration techniques for assigning large jobs to machines are given in Section 9. Conclusions are given in Section 10. A summary of the notation used in this work is given in Appendix A.
2 Combinatorial Randomized Rounding

In this Section, we illustrate the basic concepts of Combinatorial Randomized Rounding (CRR) on the SCHED problem. The IP formulation IP–SCHED of SCHED is:

\[
\min \tau \quad \text{s.t.:} \\
\sum_{j=1}^{n} p_{ij} x_{ij} \leq \tau \quad (i = 1, \ldots, m) \\
\sum_{i=1}^{m} x_{ij} = 1 \quad (j = 1, \ldots, n) \\
x_{ij} \in \{0, 1\} \quad (i = 1, \ldots, m; \; j = 1, \ldots, n)
\]

In IP–SCHED, \( \tau \) is the makespan, \( p_{ij} \) is the processing time of job \( j \) on machine \( i \), and \( x_{ij} \) is a binary variable that indicates if job \( j \) is assigned to machine \( i \). Let \( [x_{ij}] \) be the fractional solution to the LP relaxation of IP–SCHED, and \( [X_{ij}] \) the rounded solution, obtained from \( [x_{ij}] \) with standard randomized rounding (RR): For each job \( j \) independently, exactly one of the corresponding \( x_{ij} \)'s is set to 1 and the rest to 0. The probability of each \( x_{ij} \) to be rounded to 1 or 0, is determined by its fractional value:

\[
X_{ij} = \begin{cases} 
1, & \text{with probability } x_{ij} \\
0, & \text{with probability } 1 - x_{ij}
\end{cases}
\]

Let \( S_i = \sum_{j=1}^{n} p_{ij} x_{ij} \) and \( \Psi_i = \sum_{j=1}^{n} p_{ij} X_{ij} \) be the load of machine \( i \) in the fractional and the rounded schedule, respectively. Note that for each machine \( i \), the corresponding \( X_{ij} \)'s are random variables corresponding to independent Bernoulli trials and \( \Psi_i \) is the weighted sum of Bernoulli trials, with \( E[\Psi_i] = S_i \). The deviation of the random variables \( \Psi_i \) above or below their mean value can be bounded with Chernoff-like bounds. The proof of the following Chernoff bounds is given in Section 8:

Let \( \alpha_1, \alpha_2, \ldots, \alpha_r \) be real numbers in \((0,1]\). Let \( \lambda \geq 0 \) be an arbitrary real and let \( X_1, X_2, \ldots, X_n \) be independent Bernoulli trials with \( E[X_j] = p_j \). Let \( \Psi = \lambda + \sum_{j=1}^{r} \alpha_j X_j \). Then \( E[\Psi] = \lambda + \sum_{j=1}^{r} \alpha_j p_j = T \). Let \( \epsilon \geq 0, \; \delta \in (0,1) \), and \( T = E[\Psi] > 0 \). Then:

\[
Pr[\Psi > (1 + \epsilon)T] < e\left(\frac{-\epsilon^2 T}{2(1+\epsilon)}\right) \quad \text{and} \quad Pr[\Psi > (1 + \delta)T] < e\left(\frac{-\delta^2 T}{4}\right), \quad (1)
\]

\[
Pr[\Psi < (1 - \delta)T] < e\left(\frac{-\delta^2 T}{2}\right). \quad (2)
\]
2.1 The Principle

Let $i$ be a particular machine, let $\delta > 0$ be the maximum acceptable deviation, and let $\rho : 0 < \rho < 1/2$ the maximum acceptable probability of failure (i.e. a deviation ratio larger than $1 + \delta$). Then, for $\mu = \frac{3\ln(m/\rho)}{\delta^2}$, we normalize the constraint by multiplying all coefficients with the factor $\mu / S_i$. An examination of the Chernoff-bound of Equation 1, reveals that the bound on the deviation above the mean value can be made arbitrary small, if the ratio of the mean value to the maximum coefficient $\mu / (\max_i p_{ij})$ is large enough. For $T = \mu$ and for probability of failure at most $\rho/m$, the effect of the size of the coefficients on the Chernoff bound is given in Figure 2.

<table>
<thead>
<tr>
<th>case</th>
<th>$\max p_{ij}$</th>
<th>bound on the deviation factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>not bounded</td>
<td>no direct bound on the deviation</td>
</tr>
<tr>
<td>2</td>
<td>$\leq \mu$</td>
<td>$O(\sqrt{\log(m/\rho)})$</td>
</tr>
<tr>
<td>3</td>
<td>$\leq 1$</td>
<td>$(1 + \delta)$</td>
</tr>
</tbody>
</table>

Fig. 2. The effect of the coefficients on the deviation bound.

The Boosting Effect. CRR achieves to neutralize the effect of large coefficients in the rounding procedure. The argument is straightforward: The existence of large coefficients makes the bounds on the deviations weaker. Using standard RR we show that the number of ”unlucky” jobs that are assigned to large coefficients is bounded by a constant, depending on $m$. Hence we handle these unlucky jobs in a special way and boost the deviation bounds and the corresponding approximation guarantees from a factor logarithmic on $m$ (case 2 in Figure 2) to any constant factor $(1 + \delta)$ (case 3 in Figure 2). This is an important and, to some extend, surprising result for a randomized rounding based technique.

3 The Scheduling (SCHED) problem

In this Section, we present algorithm A–SCHED, a fully linear time RRDP decision procedure for SCHED and then use its derandomized version to build a FPTAS approximation scheme for SCHED.
3.1 Algorithm A–SCHED

Given an instance of SCHED and a makespan value $T$, algorithm A–SCHED first selects a set $J_\ell$ with the largest jobs, and then guesses their assignment $\varphi^*$ in an optimal solution. Since $\varphi^*$ is not known in advance, the algorithm tries all possible assignments of the large jobs. Assuming that $\varphi^*$ has been found, the problem of assigning all remaining jobs is formulated as an IP. A fractional solution is found with a LP technique. The fractional solution is rounded with randomized rounding to an integer solution.

So far, the rounding procedure corresponds to case 2 in Figure 2 and hence the Chernoff bounds can guarantee logarithmic on $m$ upper bounds on the deviation of the rounded schedule. However, the next step will change this: *All jobs that are randomly assigned to a large coefficient are removed. Applying Chernoff bounds again, gives a $1 + \epsilon$ upper bound on the deviation (case 3 in Figure 2)!

Interestingly, applying conventional randomized rounding reasoning, reveals that the number of jobs that are unlucky and have to be removed are, with probability at least $1 - \rho/2$, not more than an appropriate constant number. A combinatorial argument shows that these unlucky jobs can be reassigned in the final schedule in a greedy manner, each job to its optimal machine $i = \arg\min_i \{p_{ij}\}$, by introducing at most an arbitrary constant factor to the approximation ratio.

Algorithm A–SCHED

**Input:** An instance of SCHED, constants $\epsilon : 0 < \epsilon \leq 1$ and $\rho : 0 < \rho < 1/2$, and a makespan value $T$.

**Output:** If $T$ is feasible, then with probability of success at least $(1 - \rho)$, a schedule of makespan at most $(1 + \epsilon)T$.

**Step 0:** *Normalization.* Let $\epsilon_i = \epsilon/9 = \Theta(\epsilon), i = 1, \ldots, 6,$ and $\mu = \frac{3\ln\frac{2m}{\rho}}{\epsilon^2}$. The problem is scaled with the factor $\mu/T$.

**Step 1:** *Initializations.* $\forall j$, $d_j = \min_i \{p_{ij}\}$, $D = \sum_j d_j$, $\xi = 2 \ln \left(\frac{2m}{\rho}\right)$, and $k = \left\lceil \xi \cdot \mu \cdot \frac{m^2}{\epsilon^3} \right\rceil$.

**Step 2:** *Initial Filtering.* $\forall i, j : \text{If } p_{ij} > \mu \text{ then } x_{ij} = 0$

**Step 3:** *Large jobs.* Let $J_\ell$ be the set of large jobs $J_\ell = \{ j \mid d_j \text{ belongs to the } k \text{ largest } d_j \text{ (ties are resolved arbitrarily)} \}$. Let $\Phi$ be the set of all possible assignments of the large jobs and let $\Phi_f$ be an appropriate subset $\Phi_f \subseteq \Phi$ ($\Phi_f$ is described in the analysis of the algorithm).

**Step 4:** *Feasible Fractional Schedule.*

$\forall$ assignment $\varphi \in \Phi_f$:

1. The assignment of the remaining jobs is formulated as an integer program IP–SCHED($\varphi$).
(2) IP–SCHED(ϕ) is relaxed to the linear program LP–SCHED(ϕ).
(3) A \((1 + \epsilon_2)\)-approximate solution to LP–SCHED(ϕ) is found.
The first fractional schedule \([x_{ij}]\) that satisfies the relaxed problem is selected.

**Step 5: Combinatorial Randomized Rounding.**

1. The fractional schedule \([x_{ij}]\) is rounded with randomized rounding (RR).
2. Filtering: If a job \(j\) has been randomly assigned to a \(p_{ij} > 1\) then this job \(j\) is called "unlucky" and it is removed from the rounded schedule.
3. Each unlucky job \(j\) is assigned to a machine \(i = \arg\min_i\{p_{ij}\}\)

**3.2 Analysis of algorithm A–SCHED**

Let OPT be the optimal makespan of SCHED and let \(\tau^*\) be the optimal makespan when an arbitrary set of jobs can be fractionally assigned. Assigning every job \(j\) to the machine \(i\) that achieves its minimum processing time \(d_j = \min_i p_{ij}\), gives a feasible schedule of makespan at most \(D = \sum_j d_j\). Since, on the other hand, the total work of the machines is at least \(D\), the optimal makespan is at least \(D/m\). Hence:

\[
\frac{D}{m} \leq \tau^* \leq \text{OPT} \leq D.
\]

Given an instance of SCHED, the problem is scaled with the factor \(\mu/T\), where \(\mu = 3 \ln\left(\frac{2m}{\rho}\right)\epsilon_4^{-2}\). This is done to simplify the analysis. The objective is now to find a schedule of makespan at most \(\mu\).

**Step 2: Initial Filtering.** \(\forall i, j: \text{If } p_{ij} > \mu \text{ then } x_{ij} = 0\). This step makes the LP relaxation more tight and has no impact on the feasibility of the problem. After this step, all active \(p_{ij}\) are \(p_{ij} \leq \mu\).

**Step 3: Large Jobs.** The number of large jobs is \(k = \lceil \xi \cdot \mu \cdot m^2 \cdot \epsilon_3^{-1} \rceil = O(m^2 \ln^2(2m/\rho) \epsilon^{-3})\). The cardinality of the set \(\Phi\) of all possible assignments of the large jobs to the machines is \(m^k\), a constant that depends exponentially on \(\epsilon\). In Section 9 it is shown that it is sufficient to examine only a substantially smaller subset \(\Phi_f \subseteq \Phi\), with cardinality polynomial on \(\epsilon\). The appropriate enumeration techniques for generating \(\Phi_f\) are also given in Section 9. Using \(\Phi_f\) instead of \(\Phi\) in algorithm A–SCHED improves the complexity of A–SCHED to fully polynomial at the cost of an arbitrary small constant error factor of \((1 + \epsilon_5) \cdot (1 + \epsilon_6)\) to the final solution (Lemma 9.2).

We continue the analysis with the assumption that A–SCHED examines all assignments \(\varphi \in \Phi\), and relax this assumption at the end, by introducing the extra factor to the final approximation ratio (Proposition 3.4).
Step 4: Fractional Schedule. Given an assignment \( \varphi \) of the large jobs \( J_\ell \), the problem of assigning the remaining jobs in an optimal way (to minimize the makespan) can be formulated as the following integer program \( IP-SCHED(\varphi) \).

Let \( \varphi_i \) be the load on machine \( i \) due to \( \varphi \).

\[
\begin{align*}
\min \tau & \quad \text{s.t. :} \\
\varphi_i + \sum_{j \in [n] - J_\ell} p_{ij} x_{ij} & \leq \tau \quad (i = 1, \ldots, m) \\
\sum_{i=1}^m x_{ij} & = 1 \quad (j \in [n] - J_\ell) \\
x_{ij} & \in \{0, 1\} \quad (i = 1, \ldots, m; \ j \in [n] - J_\ell)
\end{align*}
\]

Relaxing the integrality constraints on \( x_{ij} \) to \( x_{ij} \geq 0 \) gives a corresponding linear program \( LP-SCHED(\varphi) \). Let \( \varphi^* \) be the assignment of the large jobs in an optimal schedule and let \( \tau^* \) be the optimal objective value of the corresponding \( LP-SCHED(\varphi^*) \). Assuming that the input value \( T \) is feasible for the problem instance gives that \( \tau^* \leq OPT \leq \mu \). The linear program \( LP-SCHED \) has \( m \) variables for every job \( j \) and a packing constraint for the load of every machine \( i \). Hence the variables are grouped into \( n \) independent \( m \)-dimensional simplices (blocks) and there is a constant number of positive packing constraints (coupling constraints). As shown in (11), these properties can be exploited by the logarithmic-potential based price directive decomposition algorithm (LogPDD) of Grigoriadis and Khachiyan (6), to efficiently approximate \( LP-SCHED(\varphi) \) within any constant factor \( 1 + \epsilon_2 \). The following Theorem follows from (6; 11):

**Theorem 3.1** The linear program \( LP-SCHED \) (and the linear programs \( LP-GAP \) and \( LP-MSCHED \)) can be approximated with algorithm LogPDD within any constant ratio \( 1 + \epsilon_2 \) in \( O(n) \) time.

Let \( [\hat{x}_{ij}] \) be the approximate fractional solution found for the assignment \( \varphi^* \) by algorithm LogPDD with approximation ratio \( (1 + \epsilon_2) \). If \( \tau_1 \) is the makespan of \( [\hat{x}_{ij}] \), then:

\[
\begin{align*}
\varphi^*_i + \sum_{j \in [n] - J_\ell} p_{ij} \hat{x}_{ij} & \leq \tau_1 \quad (i = 1, \ldots, m) \\
\sum_{i=1}^m \hat{x}_{ij} & = 1 \quad (j \in [n] - J_\ell) \\
\hat{x}_{ij} & \geq 0 \quad (i = 1, \ldots, m; \ j \in [n] - J_\ell)
\end{align*}
\]

By the approximation guarantee of LogPDD and since \( \tau^* \leq \mu \) we get: \( \tau_1 \leq \tau^* \cdot (1 + \epsilon_2) \leq \mu \cdot (1 + \epsilon_2) \). For each \( \varphi \in \Phi \), the LogPDD algorithm finds an approximate fractional solution to \( LP-SCHED(\varphi) \). Since the assignment \( \varphi^* \) is not known, the algorithm starts to calculate fractional schedules for each possible assignments of the large jobs. The process continues until a fractional solution \( [x_{ij}] \) is found, such that its makespan \( \tau_2 \) satisfies:

\[
\tau_2 \leq \mu \cdot (1 + \epsilon_2) \quad .
\]
If no such fractional solution is found, the input value $T$ is infeasible for the problem instance.

**Step 5: Rounding.** The fractional solution $[x_{ij}]$ is rounded to an approximate integer schedule with the standard RR procedure of Section 2. Let $[X_{ij}]$ be the rounded schedule and let $\tau_3$ be the makespan of $[X_{ij}]$. Let $J$ be set of all jobs and let $J_s$ be the set of all except the large jobs $J_s = J \setminus J_\ell$. The rounding procedure is equivalent to replacing all variables $x_{ij}$ that correspond to jobs $j \in J_s$, with a corresponding Bernoulli trial $X_{ij}$, such that $E[X_{ij}] = x_{ij}$. The load $\Psi_i$ of each machine $i$ in the rounded schedule is equal to the sum of a given positive value $\varphi_i$ and the weighted sum of independent Bernoulli trials:

$$\Psi_i = \varphi_i + \sum_{j \in J_s} p_{ij} X_{ij}.$$ 

Let $\xi = 2 \ln(\frac{2m}{\rho})$ be an appropriate deviation ratio and let $E_1$ be the event:

$$E_1 = \{ \text{The makespan } \tau_3 \text{ of the rounded solution is } \tau_3 > \xi \cdot \tau_2 \}.$$  

(5)

**Proposition 3.1** The probability of event $E_1$ is at most $\rho/2$.

**Proof:** For each machine $i$, the probability that its load $\Psi_i$ in the rounded schedule is larger than $\xi \cdot \tau_2$ can be bounded with the Chernoff bound of Theorem 8.1:

$$\forall i : P_i = \Pr\{\Psi_i > \xi \cdot \tau_2\} \leq \frac{\rho}{2m}.$$  

(6)

The sum of all probabilities $P_i$ is a sufficient bound on the probability that at least one machine in the rounded schedule has load more than $\xi \cdot \tau_2$. Hence:

$$\Pr\{\tau_3 > \xi \cdot \tau_2\} = \Pr\{\exists i : \Psi_i > \xi \cdot \tau_2\} \leq \sum_i P_i \leq \frac{\rho}{2}.$$  

Filtering. Let $J_u$ be the set of all jobs $j \in J_s$ that have been randomly assigned to large coefficients $p_{ij} : p_{ij} > 1$. We call the jobs of $J_u$ "unlucky" and remove them from the rounded schedule. The remaining schedule is called the filtered rounded schedule. For each machine $i$, let $\Psi'_i$ be the random variable:

$$\Psi'_i = \varphi_i + \sum_{j \in J_s \& p_{ij} \leq 1} p_{ij} X_{ij}.$$ 

$\Psi'_i$ corresponds to the load of the machine $i$ due to all the remaining jobs, if unlucky jobs are excluded. The random part of the random variables $\Psi'_i$ is a weighted sum of Bernoulli trials, where each weight is at most 1. Let $\tau_4$ be the makespan of the filtered rounded schedule and let $E_2$ be the event:

$$E_2 = \{ \text{In the filtered rounded schedule } \tau_4 > (1 + \epsilon_4) \cdot \tau_2 \}.$$  

(7)

The following Proposition is proved similarly to Proposition 3.1:
Proposition 3.2  The probability of event $\mathcal{E}_2$ is at most $\rho/2$.

The unlucky jobs are handled with the following combinatorial argument:

Lemma 3.1  Let $d_1 \geq d_2 \geq \ldots \geq d_n > 0$ be a sorted sequence of real numbers and let $D = \sum_{j=1}^{n} d_j$. Let $p \geq 0$ be a non-negative integer and $\epsilon_3 > 0$ a constant. Let $k = \lceil p/\epsilon_3 \rceil$. Any set $S$ of at most $|S| \leq p$ reals $d_i$ that contains none of the $k$ largest reals $\{d_i \mid i < k\}$, satisfies $\sum_{d_i \in S} d_i \leq \epsilon_3 \cdot D$.

Proof:  The total number of jobs $n$ is assumed to be larger than the constant $k$ (else SCHED can be solved by a brute force method in constant time). The real $d_k$ satisfies $d_k \leq \frac{\epsilon_3}{p} D$, because else $\sum_{i=1}^{k} d_i > D$ (contradiction). Since $\forall d_i \in S \Rightarrow i > k$ this implies that $\forall d_i \in S : d_i \leq d_k \leq \frac{\epsilon_3}{p} D$. Hence: $\sum_{d_i \in S} d_i \leq p \cdot \frac{\epsilon_3}{p} \cdot D \leq \epsilon_3 \cdot D$.

Corollary 3.1  For any set $J_p$ of at most $p = m \cdot \xi \cdot \mu$ jobs that do not belong to the large jobs $J_\ell$ ($J_p \cap J_\ell = \emptyset$), the sum of their minimum processing times $d_j$ is at most $\sum_{j \in J_p} d_j \leq \epsilon_3 \cdot \frac{D}{m} \leq \epsilon_3 \cdot \text{OPT}$.

Let $\mathcal{E}_3$ be the event that the sum of the minimum processing times $d_j$ of all unlucky jobs $j \in J_u$ is larger than $\epsilon_3 \cdot D/m$:

$$\mathcal{E}_3 = \{ \sum_{j \in J_u} d_j > \epsilon_3 \cdot D/m \}.$$  \hspace{1cm} (8)

Proposition 3.3  The probability of event $\mathcal{E}_3$ is at most $\rho/2$.

Proof:  By Proposition 3.1 the probability that the non-filtered rounded schedule has makespan larger than $\xi \cdot \mu$ is at most $\rho/2$. Hence the probability that the total load on all machines in the rounded schedule exceeds $m \cdot \xi \cdot \mu$ is at most $\rho/2$. Since every unlucky job has processing time larger than 1, the probability that the total number of unlucky jobs in the rounded schedule is larger than $m \cdot \xi \cdot \mu$ is at most $\rho/2$, and hence, from Corollary 3.1 with the same probability the sum of their minimum processing times $d_j$ is at most $\epsilon_3 \cdot D/m \leq \epsilon_3 \cdot \mu$. Assigning each unlucky job to the machine where its processing time is minimized, does not increase the makespan of the schedule by more than $\epsilon_3 \cdot D/m$, even if in the worst case all unlucky jobs end up on the same machine.

Final Schedule. The final schedule is obtained from the filtered rounded schedule, by assigning every unlucky job $j \in J_p$ to a machine $i$, where $p_{ij} = d_j$. Let $\tau_5$ be the makespan of the final schedule.

Proposition 3.4  Let $\mathcal{E}_4$ be the event $\mathcal{E}_4 = \mathcal{E}_2 \cup \mathcal{E}_3$. Then:

(1) The probability of event $\mathcal{E}_4$ is at most $\rho$, and
(2) If NOT($\mathcal{E}_4$), then $\tau_5 \leq \mu(1 + \epsilon_2)(1 + \epsilon_4) + \epsilon_3 \cdot D/m$.
Proof: 1. \( \Pr\{E_2 \cup E_3\} \leq \Pr\{E_2\} + \Pr\{E_3\} \leq \rho \).

2. If event \( E_4 \) is NOT True then both events \( E_2 \) and \( E_3 \) are NOT True, and hence by Propositions 3.2 and 3.3 the makespan \( \tau_5 \) of the final schedule is:

\[
\tau_5 \leq \tau_4 + \epsilon_3 \cdot \tau_2 \leq \tau_2 \cdot (1 + \epsilon_4) + \epsilon_3 \cdot D/m .
\] (9)

Substituting \( \tau_2 \leq \mu \cdot (1 + \epsilon_2) \) completes the proof. \( \blacksquare \)

The use of the enumeration techniques \( T_1 \) and \( T_2 \) introduces a factor \((1 + \epsilon_5)(1 + \epsilon_6)\) to the first term of the right hand side of Equation 9:

\[
\tau_5 \leq \mu \cdot (1 + \epsilon_2)(1 + \epsilon_4)(1 + \epsilon_5)(1 + \epsilon_6) + \epsilon_3 \cdot D/m \leq (1 + \epsilon) \cdot \mu .
\] (10)

Combining Theorem 3.1 and Equation 10 gives:

**Proposition 3.5** Algorithm A–SCHED is a \( O(n) \)–time RRDP for SCHED.

In Section 7 it is shown how algorithm A–SCHED can be derandomized with the method of conditional probabilities. The outcome is a deterministic algorithm \( A_{det} \)–SCHED that is a linear time RDP for SCHED.

From Equation 3 it is known that the optimal makespan \( OPT \) of SCHED is always in the interval \([D/m, D]\). Using algorithm \( A_{det} \)–SCHED within a bisection search in the interval \([D/m, D]\), gives after at most a constant number \( O(\log(m/\epsilon_1)) \) of steps, a makespan value \( T \) such that \( T \leq OPT \cdot (1 + \epsilon_1) \). When \( A_{det} \)–SCHED is executed with this input value \( T \leq OPT \cdot (1 + \epsilon_1) \) the outcome will satisfy:

\[
\tau_5 \leq OPT \cdot (1 + \epsilon_1)(1 + \epsilon_2)(1 + \epsilon_4)(1 + \epsilon_5)(1 + \epsilon_6) + \epsilon_3 \cdot OPT ,
\] (11)

and this gives:

\[
\tau_5 \leq OPT \cdot (1 + \epsilon) .
\] (12)

Hence, using \( A_{det} \)–SCHED within a binary search procedure gives an approximation scheme for the optimization version of problem SCHED:

**Corollary 3.2** Applying algorithm \( A_{det} \)–SCHED within an appropriate binary search procedure gives a \( O(n) \)–time FPTAS for SCHED.
The Generalized Assignment (GAP) problem

We present algorithm A–GAP, a fully linear time RRDP decision procedure for GAP, and then use its derandomized version to build an FPTAS approximation scheme for GAP. Given the maximum makespan value $T$ and the maximum cost value $C$, the IP formulation of the decision version of GAP is:

$$\begin{align*}
\text{Find } x, \text{ s.t.:} \\
\sum_{j=1}^{n} \sum_{i=1}^{m} c_{ij} x_{ij} &\leq C \\
\sum_{j=1}^{n} p_{ij} x_{ij} &\leq T \quad (i = 1, \ldots, m) \\
\sum_{i=1}^{m} x_{ij} &\leq 1 \quad (j = 1, \ldots, n) \\
x_{ij} &\in \{0, 1\} \quad (i = 1, \ldots, m; j = 1, \ldots, n)
\end{align*}$$

4.1 Algorithm A–GAP

Given an instance of GAP and values $T$ and $C$, an $\epsilon$–relaxed decision procedure for GAP returns a schedule of cost at most $(1 + \epsilon)C$ and makespan at most $(1 + \epsilon)T$, or decides that there is no schedule of cost and makespan, at most $C$ and $T$, respectively. The probability of failure is at most a given constant $\rho$. The structure of algorithm A–GAP is almost identical to that of algorithm A–SCHED, with a small number of adaptations:

- The measures $d_j$ and $D$ are extended to cover the two objectives of GAP:
  $$d_j = \min_i \{p_{ij} + c_{ij}\} \text{ and } D = \sum_{j=1}^{n} d_j,$$
- in several expressions $m$ is replaced with $m + 1$, and
- a job $j \notin J_\ell$ is unlucky if it is randomly assigned to a machine $i$, such that $p_{ij} > 1$ or $c_{ij} > 1$.

Algorithm A–GAP starts by selecting a constant number of jobs, the large jobs. For each possible assignment of the large jobs, a fractional schedule is calculated with algorithm LogPDD. The first feasible fractional solution is rounded randomly to an integer schedule. All unlucky jobs of the rounded schedule are removed and then each unlucky job $j$ is reassigned to a machine $i = \arg\min_i \{p_{ij} + c_{ij}\}$.

Algorithm A–GAP

Input: An instance of GAP, the constants $\epsilon : 0 < \epsilon \leq 1$, and $\rho : 0 < \rho < 1/2$, and the values $T$ for makespan and $C$ for cost.

Output: If the values $T$ and $C$ are feasible, then with probability at least $(1 - \rho)$, a schedule of makespan at most $(1 + \epsilon)T$ and cost at most $(1 + \epsilon)C$.

Step 0: Normalization. Let $\mu = \frac{3 \ln 2 \ln m + 1}{(\epsilon \rho)^2}$. The processing times $p_{ij}$ are scaled
with the factor $\mu/T$ and the costs $c_{ij}$ with the factor $\mu/C$. Now the problem is to decide if there is a schedule of makespan and cost bounded by $T = C = \mu$.

**Step 1: Initializations.**

Let $\epsilon_i = \epsilon/9 = \Theta(\epsilon)$, $i = 1, \ldots, 6$.  
\[ d_j = \min_{i} \{p_{ij} + c_{ij}\}, \quad D = \sum_{j=1}^{n} d_j, \]
\[ \xi = 2 \ln \left( \frac{2m+2}{n} \right), \quad k = \left\lceil \xi \cdot \mu \cdot \frac{(m+1)^2}{\epsilon^3} \right\rceil. \]

**Step 2: First Filtering.**

\[ \forall i, j : \text{If } (p_{ij} > \mu \text{ OR } c_{ij} > \mu) \text{ then } x_{ij} = 0 \]

**Step 3: Large jobs.**

Let $J_k = \{ j \mid d_j \text{ belongs to the k largest } d_j \}$ be the set of large jobs and let $\Phi$ be the set of all possible assignments of the large jobs to the machines. Let $\Phi_f$ be an appropriate subset of $\Phi$.

**Step 4: Feasible Fractional Schedule $[x_{ij}]$.**

\[ \forall \text{ assignment } \varphi \in \Phi_f \text{ do } \]
\[ (1) \text{ Formulate the corresponding scheduling problem as an integer program } \text{ILP–GAP}(\varphi). \]
\[ (2) \text{ Relax ILP–GAP}(\varphi) \text{ to the linear program LP–GAP}(\varphi). \]
\[ (3) \text{ Find the approximate fractional schedule with algorithm LogPDD.} \]

The first feasible fractional schedule $[x_{ij}]$ is selected.

**Step 5: Combinatorial Randomized Rounding.**

\[ (1) \text{ The fractional schedule } [x_{ij}] \text{ is rounded with RR.} \]
\[ (2) \text{ Filtering } : \text{ If a job } j \text{ has been randomly assigned to a } p_{ij} > 1 \text{ or a } c_{ij} > 1 \]
\[ \text{ then job } j \text{ is called "unlucky" and it is removed from the schedule.} \]
\[ (3) \text{ Every unlucky job } j \text{ is assigned to machine } i = \arg\min_{i} \{p_{ij} + c_{ij}\} \]

The proof of the following proposition, that A–GAP is RRDP for GAP is similar to the corresponding proof for algorithm A–SCHED (Proposition 3.5).

**Proposition 4.1** Algorithm A–GAP is a $O(n)$–time RRDP for GAP.

An extra issue in A–GAP is the deviation of the rounded cost constraint. Note that the cost constraint is the weighted sum of all variables $X_{ij}$. However, by definition of the RR procedure, the variables $X_{ij}$ that correspond to the same job $j$ are not independent. Hence, the Chernoff bounds of Theorem 8.1 do not apply. However we obtain equivalent results in the following way: For every job $j$, let $X_j^c$ be the cost induced by job $j$: $X_j^c = \sum_{i=1}^{m} c_{ij} X_{ij}$. Note that $X_j^c \in (0, \mu)$ in the rounded schedule and $X_j^c \in (0, 1)$ in the filtered rounded schedule. The variables $X_j^c$ are independent random variables and the total cost of the random schedule is $X^c = \sum_{j=1}^{n} X_j^c$. The deviation of $X^c$ above and below its mean value can be bounded with the following Hoeffding-Chernoff bound proved in Section 8:

Let $X_1^c, X_2^c, \ldots, X_n^c$ be independent discrete random variables such that $X_j^c \in [0, 1]$, for $1 \leq j \leq n$. Also let $X^c = \sum_{j} X_j^c$, $\Psi^c = \sum_{j} E[X_j^c]$, $\epsilon \geq 0$, and
δ ∈ (0, 1). Then:

\[ \Pr[\Psi^c > (1 + \epsilon)S] < e \left( \frac{-\frac{\epsilon^2 S}{2}}{2(1 + \epsilon)} \right) \quad \text{and} \quad \Pr[\Psi^c < (1 + \delta)S] < e \left( \frac{-\delta^2 S}{3} \right), \quad (13) \]

\[ \Pr[\Psi^c < (1 - \epsilon)S] < e \left( \frac{-\epsilon^2 S}{2} \right). \quad (14) \]

In Section 7 we show how algorithm A–GAP can be derandomized with the method of conditional probabilities to produce algorithm \( A_{det} \)-GAP, a linear time RDP for GAP.

### 4.2 Optimization versions of GAP

Since the GAP problem has two separate objectives, the makespan \( T \) and the cost \( C \), it admits more than one optimization problems. Two natural cases are GAP–T and GAP–C, which are obtained by specifying an upper bound on one objective and then optimizing the other. Another case is when the objective function is a linear function of the makespan and the cost. We present an approximation scheme for GAP–T, where the objective is to optimize the makespan \( T \) without exceeding a maximum cost \( C \). It is straightforward to adapt this algorithm for other cases.

We will use the following property of combinatorial optimization problems:

**Definition 4.1** A combinatorial optimization problem has the poly-bottleneck\(^2\) property, if its optimal objective value is always within a polynomial factor of one of its input items (weights).

**Proposition 4.2** GAP has the poly-bottleneck property.

**Proof:** Let \( p_{\text{max}} \) be the maximum processing time that appears in an optimal solution to GAP. Then the makespan of the optimal schedule is at least \( p_{\text{max}} \) and at most \( n \cdot p_{\text{max}} \). There are at most \( m \cdot n \) different possibilities for \( p_{\text{max}} \), since \( p_{\text{max}} \) has to be one of the specified \( p_{ij} \). In the same way let \( c_{\text{max}} \) be the maximum cost that appears in an optimal solution to GAP. Then the cost of the optimal schedule is at least \( c_{\text{max}} \) and at most \( n \cdot c_{\text{max}} \). There are at most \( m \cdot n \) different possibilities for \( c_{\text{max}} \), since \( c_{\text{max}} \) has to be one of the specified \( c_{ij} \). ■

\(^2\) The term "bottleneck" has been used by Hochbaum and Schmoys in (9) for a class of graph optimization problems, where the value of the optimal solution is always one of the (edge) weights in the original specification of the instance of the problem.
Given a RDP decision procedure for a problem having the poly-bottleneck property, a two-phase binary search procedure can solve corresponding optimization problems.

**Binary Search Procedure (BS)**

**Step 1:** *Weights.* Let \( W \) be the set of numbers that contains for each pair \((i, j)\), the values \( p_{ij} \) and \( n \cdot p_{ij} \). The cardinality of \( W \) is \( w = |W| \leq 2 \cdot m \cdot n \).

**Step 2:** *Sorting.* Sort the items \( w_i \). Let the sorted list of the items of \( W \) be: \( w_1 \geq w_2 \geq \ldots \geq w_w \).

**Step 3:** *Indexed binary search.*

Find the maximum index \( x : 1 \leq x \leq w \) such that \( T = C = w_x \) is feasible.

This requires at most \( \lceil \log(2mn) \rceil \) binary search steps.

**Step 4:** *Standard binary search.*

Find the maximum value \( t : w_x \leq t < \min\{w_{x+1}, n \cdot w_x\} \)

such that \( T = C = t \) is feasible and \( t \leq \text{OPT}(1 + \epsilon_1) \).

This requires at most \( \lceil \log(mn/\epsilon_1) \rceil \) binary search steps.

The derandomized version of algorithm \( \text{A–GAP} \) is a RDP for problem GAP. This RDP procedure can be used within a binary search framework like BS to build a FPTAS approximation scheme for optimization versions of GAP. The FPTAS calls algorithm \( \text{A–GAP} \) at most \( O(\log n/\epsilon_1) \) times.

**Corollary 4.1** Applying algorithm \( \text{A}_{\text{det}–\text{GAP}} \) within an appropriate binary search procedure gives a \( O(n \log n) \) time FPTAS for \( \text{GAP–T} \).

### 4.3 Algorithm \( \text{A–MSCHED} \)

In the same way that algorithm \( \text{A–SCHED} \) is extended to handle an extra cost function in algorithm \( \text{A–GAP} \), it can be extended to handle any constant number of linear cost functions. This gives a linear-time RDP and a quasi-linear time FPTAS algorithm for MSCHED. We omit the description of algorithm \( \text{A–MSCHED} \).

### 5 The Covering (COV) problem

We first present a RRDP decision procedure and then show how to build a FPTAS approximation scheme. Given an instance of COV and a value \( T \) for the minimum makespan, the IP formulation of the decision version of COV is:
Find x, s.t.: \[
\begin{align*}
\sum_{j=1}^{n} p_{ij} x_{ij} &\geq T & (i = 1, \ldots, m) \\
\sum_{i=1}^{m} x_{ij} &\geq 1 & (j = 1, \ldots, n) \\
x_{ij} &\in \{0, 1\} & (i = 1, \ldots, m; \ j = 1, \ldots, n)
\end{align*}
\]

5.1 Algorithm A–COV

For \( \mu = \frac{2 \ln \frac{m}{\epsilon^3}}{m^3} \), the problem is normalized by multiplying it with the factor \( \mu/T \). The next step is to handle the very large coefficients \( p_{ij} > \mu \). While in packing problems these large coefficients are zeroed out, in covering problems we set them equal to \( p_{ij} = \mu \). This has no impact on the feasibility of the covering problem for makespan \( \mu \). After this step all coefficients \( p_{ij} > 1 \) are considered large. Let \( k = \frac{2m^2 \mu}{\epsilon^3} \). Let \( J \) be the set of all jobs and for each constraint \( i \) let \( J_i^I \) be the set of jobs that correspond to the large coefficients of the constraint. A constraint \( i \) is of

- **Type I**: If it has less than \( k \) large coefficients.
- **Type II**: If it has at least \( k \) large coefficients.

Let \( J_i^I \) be the union of all sets \( J_i^I \) that correspond to constraints of type I, i.e. the set of all jobs that correspond to large coefficients in constraints of type I. The cardinality of \( J_i^I \) is \( |J_i^I| \leq m \cdot k \), a constant. Let \( \Phi \) be the set of all possible assignments \( \varphi \) of the large jobs. Like in algorithm A–SCHED, we examine only the assignments \( \varphi \in \Phi_J \), a subset of \( \Phi \) generated with adapted versions of techniques \( T_1 \) and \( T_2 \) of Section 9. Let \( \varphi_i \) be the load on machine \( i \) due to assignment \( \varphi \). Given any \( \varphi \), the problem of assigning the remaining jobs is formulated as an integer program IP–COV(\( \varphi \)):

Find x, s.t.: \[
\begin{align*}
\sum_{j \in J \setminus J_i^I} p_{ij} x_{ij} &\geq \mu - \varphi_i & (i = 1, \ldots, m) \\
\sum_{i=1}^{m} x_{ij} &\geq 1 & (j \in J \setminus J_i^I) \\
\sum_{i=1}^{m} x_{ij} &\leq 1 & (j \in J \setminus J_i^I) \\
x_{ij} &\in \{0, 1\} & (i = 1, \ldots, m; \ j \in J \setminus J_i^I)
\end{align*}
\]

Relaxing the integrality constraints on the variables \( x_{ij} \) and replacing every equality constraint with a pair of a packing and a covering constraint gives the following linear program LP–COV(\( \varphi \)).
Find \( x \), s.t.:

\[
\begin{align*}
\sum_{j \in J \setminus J_i^I} p_{ij} x_{ij} &\geq \mu - \varphi_i \quad (i = 1, \ldots, m) \\
\sum_{i=1}^{m} x_{ij} &\geq 1 \quad (j \in J \setminus J_i^I) \\
\sum_{i=1}^{m} x_{ij} &\leq 1 \quad (j \in J \setminus J_i^I) \\
x_{ij} &\geq 0 \quad (i = 1, \ldots, m; \ j \in J \setminus J_i^I)
\end{align*}
\]

The linear program \( \text{LP–COV}(\varphi) \) has only positive coefficients and all its constraints are either packing or covering constraints. This LP can be approximately solved with the RDP decision procedure for mixed packing and covering problems given in (22) (we call it algorithm A–Young):

**Corollary 5.1** *(From Corollary 1 of (22)) The linear program \( \text{LP–COV}(\varphi) \) (and \( \text{LP–EQUAL}(\varphi) \) in algorithm A–EQUAL of Section 6) can be approximated with algorithm A–Young within any constant ratio \( \epsilon \) in \( O(N/\epsilon^2) \) time, where \( N \) is the number of non-zero entries in the problem matrices.*

If the input value \( T \) is feasible, then algorithm A–Young returns an approximately feasible fractional solution \( \tilde{x}_{ij} \):

\[
\text{Solution } \tilde{x} : \quad \begin{align*}
\sum_{j=1}^{n} p_{ij} \tilde{x}_{ij} &\geq \mu \quad (i = 1, \ldots, m) \\
\sum_{i=1}^{m} \tilde{x}_{ij} &\geq 1 \quad (j = 1, \ldots, n) \\
\sum_{i=1}^{m} \tilde{x}_{ij} &\leq 1 \cdot (1 + \epsilon_2) \quad (j = 1, \ldots, n) \\
\tilde{x}_{ij} &\in \{0, 1\} \quad (i = 1, \ldots, m; \ j = 1, \ldots, n)
\end{align*}
\]

Algorithm A–COV starts to calculate fractional solutions for each possible assignment \( \varphi \) of the large jobs, until the first approximately feasible fractional solution \( [\tilde{x}_{ij}] \) is found. If no such solution is found, then the input value \( T \) is infeasible for the instance of COV. Let \( [\tilde{x}_{ij}] \) be an approximately feasible fractional solution. By scaling the variables \( [\tilde{x}_{ij}] \) appropriately for each job \( j \), we get the following feasible fractional job assignment \( [x_{ij}] \):

\[
\text{Solution } x : \quad \begin{align*}
\sum_{j=1}^{n} p_{ij} x_{ij} &\geq \mu \cdot (1/(1 + \epsilon_2)) \quad (i = 1, \ldots, m) \\
\sum_{i=1}^{m} x_{ij} &= 1 \quad (j = 1, \ldots, n) \\
x_{ij} &\in \{0, 1\} \quad (i = 1, \ldots, m; \ j = 1, \ldots, n)
\end{align*}
\]

**Rounding with corrections.** The solution \( [x_{ij}] \) is rounded with standard RR to an integer solution \( [X_{ij}] \). With probability at least \( 1 - \rho \), none of the constraints of type \( I \) in \( [X_{ij}] \) is violated by a factor larger than \( (1 - \epsilon_3) \).

Given a rounded solution \( [X_{ij}] \), let \( i \) be a constraint of type \( II \) that is not satisfied, i.e. \( \sum_{j=1}^{n} p_{ij} x_{ij} < \mu \). By definition a constraint of type \( II \) has at least
$k$ large coefficients. From all jobs $j \in J_i^\ell$ that are not assigned to $i$ we select the jobs that have the smallest load in their placement in the rounded solution. These jobs are sufficient to cover the violated constraint. A combinatorial argument shows that, for every correction of a constraint of type $II$, the possible cost to any other constraint, due to removing jobs from it, does not exceed $\epsilon_4^4 \mu$. Hence, the cost of all corrections of all constraints of type $II$ cannot exceed $\epsilon_4 \cdot \mu$.

In particular, for any violated constraint $i$ of type $II$, let $Z_i^\ell$ be the minimum total cost to the rounded solution due to moving jobs to machine $i$. We distinguish two cases:

$Z_i^\ell \leq \epsilon_4^4 \mu \Rightarrow$ In this case we move the jobs to the violated constraint $i$. The total cost for correcting all constraints of type $II$ that are violated is $\leq \epsilon_3 \mu$.

$Z_i^\ell > \epsilon_4^4 \mu \Rightarrow$ In this case the total load due to the initial assignment of all jobs in $J_i^\ell$ is larger then $2m \mu$ and hence we can move a sufficient number (at most $\mu$) of jobs to the violated constraint $i$ without causing any violation to other constraints.

**Proposition 5.1** Algorithm $A-COV$ is a $O(n)$ time RRDP decision procedure for COV.

The derandomization technique of Section 7 can be used on algorithm $A-COV$ to produce a corresponding deterministic linear time algorithm $A_{det-COV}$.

**Algorithm A–COV**

**Input:** A COV instance, constants $\epsilon : 0 < \epsilon \leq 1$ and $\rho : 0 < \rho < 1/2$, and a value $T$.

**Output:** If the value $T$ is feasible, then with probability of success at least $(1 - \rho)$, a schedule of makespan at least $(1 - \epsilon) T$.

**Step 0:** Normalization. Let $\epsilon_i = \epsilon/9 = \Theta(\epsilon)$, for $i = 1, \ldots, 6$, and $\mu = \frac{2 \ln m}{\rho (\epsilon_3)^2}$. To simplify the algorithm, the problem is scaled with the factor $\mu/T$.

**Step 1:** Initial Filtering. For each $p_{ij} > \mu$ set $p_{ij} = \mu$.

**Step 2:** Constraints. Let $k = \frac{2 m^2 \mu}{\epsilon_3}$.

**Step 3:** Large coefficients and large jobs. Let $\Phi_f \subseteq \Phi$ be an appropriate set of assignments of the large jobs $J_i^\ell$.

**Step 4:** Feasible Fractional Assignment. $\forall$ assignment $\varphi \in \Phi_f$ do

1. The corresponding covering problem is formulated as IP–COV($\varphi$).
2. IP–COV($\varphi$) is relaxed to an linear program LP–COV($\varphi$).
3. An approximate fractional schedule with algorithm A–Young. The first feasible fractional schedule $[x_{ij}]$ is selected.

**Step 5:** Combinatorial Randomized Rounding.

1. $[x_{ij}]$ is rounded with randomized rounding (RR)
2. With probability at least $1 - \rho$, no constraint of type I is violated by more than $(1 - \epsilon_3)$. 

20
Any constraint of type II that is violated can be corrected by moving at most $\mu$ large jobs to the corresponding machine.

### 5.2 Maximum COV

The COV problem has the poly-bottleneck property and hence algorithm $A_{det-COV}$ can be used within a binary search procedure BS to give a quasi-linear time FPTAS for COV.

**Corollary 5.2** Applying algorithm $A_{det-COV}$ within a binary search procedure BS gives a quasi-linear time FPTAS for COV.

### 6 The Equal Loads (EQUAL) problem

In this Section we present a RRDP decision procedure and a PTAS approximation scheme for problem EQUAL. The complexity of the algorithms is a high degree polynomial on $n$ and hence the results are considered mainly of theoretical interest. Given an instance of EQUAL and a makespan value $T$ for the makespan, the IP formulation of the decision version of EQUAL is

$$\text{Find } x, \text{s.t.:} \begin{cases} \sum_{j=1}^{n} p_{ij} x_{ij} = T & (i = 1, \ldots, m) \\ \sum_{i=1}^{m} x_{ij} = 1 & (j = 1, \ldots, n) \\ x_{ij} \in \{0, 1\} & (i = 1, \ldots, m; j = 1, \ldots, n) \end{cases}$$

### 6.1 Algorithm $A_{EQUAL}$

We consider the relaxed version of EQUAL where the equality constraints can be violated by at most a given constant factor. Hence, given a problem instance of EQUAL and a makespan value $T$, the objective is to calculate a schedule where each machine has a machine load $\Psi_i : T(1 - \epsilon/2) \leq \Psi_i \leq T \cdot (1 + \epsilon/2)$. Let $\epsilon_i = \epsilon/8 = \Theta(\epsilon)$, for $i = 1, \ldots, 5$. For $\mu = \frac{3 \ln 2m}{(\epsilon_3)^2}$ the problem is normalized with the factor $\mu/T$.

**Initial Filtering.** For all $p_{ij} > \mu$, the corresponding $x_{ij}$ is set to 0.

**Large Coefficients.** For each machine $i$, let $L_i$ be the set of all pairs $(i, j)$ that correspond to large coefficients $p_{ij} : 1 < p_{ij} \leq \mu$. Assume an assignment of binary values $\{0, 1\}$ to the variables $x_{ij}$ that correspond to large coefficients...
Let \( L_i \) and \( L_i^1 \subseteq L_i \) be the set of pairs \((i, j)\), such that \( x_{ij} = 1 \), and \( L_i^0 \subseteq L_i \) be the set of pairs \((i, j)\), such that \( x_{ij} = 0 \). At most \( \mu = T \) jobs can be assigned to large coefficients of machine \( i \) in a feasible solution. For each machine \( i \), even if all its coefficients are large, there are not more than

\[
\binom{n}{r} \leq \frac{n^r}{r!} = p_i(r)(n) \quad (15)
\]

sets \( L_i^1 \subseteq L_i \) when the cardinality of \( L_i^1 \) is restricted to be exactly \( |L_i^1| = r \), where \( 0 \leq r \leq \mu \). The total number of sets \( L_i^1 \subseteq L_i \) that correspond to all feasible assignments to large coefficients for a specific machine, cannot be more than the number of all possible sets \( L_i^1 \) with cardinality \( 0 \leq |L_i^1| \leq \mu \):

\[
\sum_{r=0}^{\mu} \binom{n}{r} \leq \mu \cdot \frac{n^r}{r!} = p_i(n) \quad . \quad (16)
\]

The total number of all feasible assignments of jobs to large coefficients for all machines cannot exceed \( p(n) = \prod_{i=1}^{m} p_i(n) \), a high degree polynomial on \( n \).

**Fractional Solution.** For every possible assignment \( L^1 \) of jobs to large coefficients, the problem of assigning values to the remaining variables \( x_{ij} : (i, j) \notin L \), is formulated as an integer program \( \text{IP–EQUAL}(L^1) \):

\[
\text{Find } x, \text{ s.t.:} \begin{cases} 
\sum_{j=1}^{n} p_{ij} x_{ij} \geq \mu & (i = 1, \ldots, m) \\
\sum_{j=1}^{n} p_{ij} x_{ij} \leq \mu & (i = 1, \ldots, m) \\
\sum_{i=1}^{m} x_{ij} \geq 1 & (j = 1, \ldots, n) \\
\sum_{i=1}^{m} x_{ij} \leq 1 & (j = 1, \ldots, n) \\
x_{ij} \in \{0, 1\} & (i = 1, \ldots, m; \ j = 1, \ldots, n) \\
x_{ij} = 1, (i, j) \in L_i^1 \ x_{ij} = 0, (i, j) \in L_i^0 
\end{cases}
\]

The integer program \( \text{IP–EQUAL}(L^1) \) is relaxed to a mixed packing covering program \( \text{LP–EQUAL}(L^1) \). Like in Section 5, an approximately feasible fractional solution is found for the linear program \( \text{LP–EQUAL}(L^1) \) with algorithm \( A–Young \) of (22). Let \( t_i \) be the load on machine \( i \) due to the assignment \( L^1 \). If the value \( \mu \) is feasible for the problem, then at least one fractional solution \([\tilde{x}_{ij}]\) satisfies:
Algorithm A–EQUAL iteratively examines each possible assignment of jobs to large coefficients $L^1$ and calculates the corresponding fractional solution. The iteration continues, until the first fractional solution that satisfies the same constraints as $\tilde{x}_{ij}$ is found. This fractional solution is then scaled with a factor less of equal to $1/(1 + \epsilon_2)$ so that all jobs are completely assigned. This introduces at most a factor $1 - \epsilon_3 = 1/(1 + \epsilon_2)$ to the approximation ratio. Let $[x_{ij}]$ be the scaled fractional solution. Then:

$$t_i + \sum_{j=1}^n p_{ij} \tilde{x}_{ij} \geq \mu (1 - \epsilon_3) \quad (i = 1, \ldots, m)$$
$$t_i + \sum_{j=1}^n p_{ij} \tilde{x}_{ij} \leq \mu (1 + \epsilon_2) \quad (i = 1, \ldots, m)$$
$$\sum_{i=1}^m \tilde{x}_{ij} = 1 \quad (j = 1, \ldots, n)$$
$$\tilde{x}_{ij} \in [0, 1] \quad (i = 1, \ldots, m; \ j = 1, \ldots, n)$$

**Rounding.** The fractional solution $[x_{ij}]$ is rounded with randomized rounding to an integer schedule $[X_{ij}]$. Since the rounding concerns only small coefficients $p_{ij} \leq 1$, with probability at least $1 - \rho$, it does not introduce deviations by factors larger than $(1 - \epsilon_4)$ and $(1 + \epsilon_5)$, below and above the fractional load, respectively. Consequently, if $T = \mu$ is feasible, then, with probability at least $1 - \rho$, the final solution $[X_{ij}]$ satisfies:

$$\sum_{j=1}^n p_{ij} X_{ij} \geq \mu (1 - \epsilon_5)(1 - \epsilon_4) \quad (i = 1, \ldots, m)$$
$$\sum_{j=1}^n p_{ij} X_{ij} \leq \mu (1 + \epsilon_2)(1 + \epsilon_5) \quad (i = 1, \ldots, m)$$
$$\sum_{i=1}^m X_{ij} = 1 \quad (j = 1, \ldots, n)$$
$$X_{ij} \in \{0, 1\} \quad (i = 1, \ldots, m; \ j = 1, \ldots, n)$$

Hence:

**Proposition 6.1** Algorithm A–EQUAL is a RRDP decision procedure for EQUAL.

The derandomization technique of Section 7 can be used on algorithm A–EQUAL to produce a corresponding deterministic polynomial time algorithm $A_{\text{det}}$–EQUAL.
Algorithm A–EQUAL

Input: An instance of EQUAL, constants \( \epsilon : 0 < \epsilon \leq 1 \) and \( \rho : 0 < \rho < 1/2 \), and a value \( T \).

Output: If the value \( T \) is feasible, then with probability of success at least \( 1 - \rho \), a schedule such that, for the load \( \Psi_i \) of machine \( i = 1, \ldots, m \):

\[
(1 - \epsilon/2)T \leq \Psi_i \leq (1 + \epsilon/2)T.
\]

Step 0: Normalization. Let \( \epsilon_i = \epsilon/8 = \Theta(\epsilon), i = 1, \ldots, 5 \), and \( \mu = \frac{2 \ln m}{(\epsilon_3)^2} \). To simplify the algorithm, the problem is scaled with the factor \( \mu/T \).

Step 1: Initial Filtering. For each \( p_{ij} > \mu \) set \( x_{ij} = 0 \).

Step 2: Large Jobs. For every possible assignment \( L^1 \) of jobs to large coefficients, a fractional assignment of the remaining variables \( x_{ij} : (i, j) \notin L \) is found with an appropriate LP technique (algorithm A–Young). The first feasible fractional schedule \( [\tilde{x}_{ij}] \) is selected.

Step 3: The fractional solution \( [\tilde{x}_{ij}] \) is scaled, so that for each \( i = 1, \ldots, n \), the corresponding constraint \( \sum_j x_{ij} = 1 \) is satisfied.

Step 4: Randomized Rounding.

1. The fractional solution \( [x_{ij}] \) is rounded with randomized rounding (RR)
2. With probability at least \( 1 - \rho \), the rounding does not introduce deviations by a factor larger than \( 1 - \epsilon_4 \) below the fractional load and deviations by a factor larger than \( 1 + \epsilon_5 \) above the fractional load.

6.2 Optimization Version

EQUAL has the poly-bottleneck property and hence we can apply a two-phase binary search procedure to find an approximately feasible solution with approximately minimum makespan \( T \). An important issue is that in EQUAL the range of feasible objective values may not be continuous. We overcome this issue within the binary search procedure by adding a positive slack variable \( s \) to the machine load constraints in the relaxed LP formulation of EQUAL:

Find \( x \), s.t.: \[
\begin{align*}
\sum_{j=1}^n p_{ij}x_{ij} + s &\geq \mu \quad (i = 1, \ldots, m) \\
\sum_{j=1}^n p_{ij}x_{ij} + s &\leq \mu \quad (i = 1, \ldots, m) \\
\sum_{i=1}^m x_{ij} &\geq 1 \quad (j = 1, \ldots, n) \\
\sum_{i=1}^m x_{ij} &\leq 1 \quad (j = 1, \ldots, n) \\
x_{ij} &\in \{0, 1\} \quad (i = 1, \ldots, m; j = 1, \ldots, n) \\
x_{ij} = 1, (i, j) \in L^1_i &\quad x_{ij} = 0, (i, j) \in L^0_i
\end{align*}
\]

Let \( OPT \) be the optimal objective value for the COV problem. The introduction of the slack parameter \( s \) makes any input value \( T \), such that \( T \geq OPT \), feasible for the above linear program. This makes it possible for
the binary search procedure to (approximately) converge to the optimal objective value. The binary search procedure searches for the smallest feasible makespan value $T$. During this search it executes the RDP procedure $A_{\text{det-EQUAL}}$ $O(\log(n/\epsilon))$ times.

**Corollary 6.1** Applying algorithm $A_{\text{det-EQUAL}}$ within a binary search procedure BS gives a PTAS approximation scheme for EQUAL.

**Note 1** By scaling each machine load constraint appropriately it is possible with algorithm $A$–EQUAL to approximately solve the more general problem of assigning specific loads to the machines.

## 7 Derandomization

In this Section, we show how all randomized algorithms of this work can be efficiently derandomized to generate deterministic algorithms of equivalent time complexity. The derandomization is based on the method of conditional probabilities (16).

### 7.1 Modified Versions of $A$–SCHED and $A$–GAP

We first note that algorithms $A$–SCHED and $A$–GAP (and $A$–MSCHED) in their original description, perform two concurrent randomized roundings:

- An explicit rounding with all problem coefficients, and
- an implicit rounding where large coefficients are neutralized.

Due to this fact, these algorithms are not directly compatible with the method of conditional probabilities. We overcome this issue with a modified but equivalent version of the algorithms that is more appropriate for derandomization. The modification concerns the rounding step, i.e. Step 5, of algorithms $A$–SCHED and $A$–GAP, where a feasible fractional solution $[x_{ij}]$ is rounded to an integer solution.

**Step 5:** *Modified Version for $A$–SCHED and $A$–GAP.*

- In the feasible fractional solution $[x_{ij}]$ all large coefficients $p_{ij} > 1$ (and $c_{ij} > 1$ for $A$–GAP) are set equal to 1 and marked as ”fixed” coefficients. Clearly, the fractional solution with the fixed coefficients remains feasible.
- We apply RR (or the method of pessimistic estimators) to obtain a rounded solution. The rounding concerns only small coefficients and hence the rounded solution satisfies (for RR, with probability at least $1 - \rho/2$) the tight deviation bound of $1 + \epsilon$.
In the rounded solution at most a constant number of jobs can be assigned to coefficients with value 1. Some of these coefficients may belong to the "fixed" coefficients. The jobs that have been assigned to "fixed" coefficients, correspond to the unlucky jobs of the original algorithm descriptions in Sections 3 and 4. We remove all such jobs and assign them greedily as in the original versions of the algorithms.

7.2 The Method of Conditional Probabilities

In the CRR-based algorithms of this work, the probability that the randomly rounded solution violates the approximation ratio is less than a given constant $\rho < 1$ and hence less than 1. This fact is a proof for the existence of at least one rounded solution that satisfies the required approximation ratio. The method of conditional probabilities mimics the probability existence proof to actually build such a rounded solution in a deterministic manner. The computation can be modelled with a decision tree, with one level for each job $j$. Level $j$ of the tree represents the decision of assigning job $j$ to a machine $i = 1, 2, \ldots, m$. The probability of selecting the descendant $i$ at level $j$ is equivalent to the probability of assigning job $j$ to machine $i$. The existence proof guarantees that there is always at least one good leaf in the decision tree.

Our task is to walk down the tree to a good leaf in deterministic polynomial time. P. Raghavan (16).

When the procedure starts, the probability of failure at the root node is less than $\rho$. As shown in Equation 20, for at least one of the descendants of the root node the probability of failure is at most the probability of failure of its parent (the root). We walk down to this child node and repeat the procedure for the next level.

In particular, we show that for all job assignment problems considered in this work, we can deterministically generate a rounded solution that satisfies the desired approximation ratio. The derandomized procedure assigns deterministically all jobs one by one to the machines, in such a way that, after each job assignment, the probability of failure for the remaining jobs (if they were randomly assigned) does not increase. The outcome of this procedure is a rounded solution with probability of failure (i.e. a violation of the approximation ratio) less than 1, and hence it must be a rounded solution satisfying the approximation guarantee.
7.3 The Pessimistic Estimator

A sufficient condition for the deterministic rounding procedure to work, is that the probability of failure is always less than 1. Hence, it is not necessary to know the exact probability of failure at each of the procedure, as long as this probability can be bounded from above with 1. A function $U$ that is efficiently computable and that provides an upper bound on the probability of failure for each node of the decision tree is called a pessimistic estimator. For an exact definition of pessimistic estimators and further details on the method of pessimistic estimators, the reader is referred to (16).

Assume that the first $j-1$ jobs have been assigned to the machines $m_1, m_2, \ldots, m_{j-1}$, respectively. Then the pessimistic estimator $U_j(m_1, m_2, \ldots, m_{j-1})$ is an upper bound on the probability of failure for the subproblem of randomly assigning the remaining jobs. Thus, derandomization with the method of conditional probabilities is essentially reduced to providing the appropriate pessimistic estimator.

We use pessimistic estimators to derandomize all CRR-based algorithms of this work. Since the pessimistic estimators of all problems (of this work) are very similar, we select the A–GAP algorithm and present a pessimistic estimator $U$ (Equation 7.3) for it. Problem GAP has both, machine load constraints and a cost constraint and hence it is straightforward to adapt the given pessimistic estimator to the other problems.

Let $[x_{ij}]$ be a feasible fractional solution for an instance of A–GAP (for simplicity, we ignore the fact that, at this step in algorithm A–GAP, a constant number of jobs, the large jobs, are already assigned):

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \leq S$$
$$\sum_{j=1}^{n} p_{ij} x_{ij} \leq S \quad (i = 1, \ldots, m)$$
$$\sum_{i=1}^{m} x_{ij} = 1 \quad (j = 1, \ldots, n)$$
$$x_{ij} \geq 0 \quad (i = 1, \ldots, m; \ j = 1, \ldots, n)$$

Let $[X_{ij}]$ be the randomly rounded solution obtained from $[x_{ij}]$, where the variables $X_{ij}$ are Bernulli trials. For the cost constraint, let $X^C_j = \sum_{i=1}^{m} c_{ij} X_{ij}$ be the cost due to assignment of job $j$. The random variables $X^C_j$ for $j = 1, \ldots, n$ are independent discrete variables. In the rounded solution let $\Psi_i$ be the load of machine $i$ and let $\Psi_C$ be the cost of the schedule. Now, the probability that the makespan and the cost of the rounded schedule do not exceed $S(1 + \epsilon)$ can be bounded with the pessimistic estimator:

$$U = Pr[(\Psi_C > S(1 + \epsilon)) \lor (\forall_{i=1}^{m} \Psi_i > S(1 + \epsilon))]$$

27
\[
\leq \Pr[\Psi_C > S(1 + \epsilon)] + \sum_{i=1}^{m} \Pr[\Psi_i > S(1 + \epsilon)]
\]

Using Equation 24 gives for each \(i = 1, \ldots, m\) and for any positive real \(t\)
\[
P_i = \Pr[\Psi_i > S(1 + \epsilon)] < e^{-t(1+\epsilon)S} \prod_{j=1}^{n} (x_{ij}e^{tp_{ij}} + 1 - x_{ij})
\]

and this gives, for example for machine \(i = 1\)
\[
P_1 < e^{-t(1+\epsilon)S} \prod_{j=2}^{n} (x_{1j}e^{tp_{ij}} + 1 - x_{1j}) .
\]

When the procedure examines the first job \(j = 1\)
\[
P_1 < (x_{11}e^{tp_{11}} + x_{21} + \ldots + x_{m1}) e^{-t(1+\epsilon)S} \prod_{j=2}^{n} (x_{1j}e^{tp_{ij}} + 1 - x_{1j}) .
\]

Using Equation 29 gives for any positive real \(t\)
\[
P_C = \Pr[\Psi_C > S(1 + \epsilon)] < e^{-t(1+\epsilon)S} \prod_{i=1}^{m} (\sum_{j=1}^{\infty} x_{ij}e^{tc_{ij}})
\]

and when focusing on job \(j = 1\) this gives
\[
P_C < (x_{11}e^{tc_{11}} + \ldots + x_{m1}e^{tc_{m1}}) e^{-t(1+\epsilon)S} \prod_{j=2}^{n} (\sum_{i=1}^{m} x_{ij}e^{tc_{ij}})
\]

Hence at the start of the derandomized rounding procedure
\[
U = U(root) = x_{11}B_1 + x_{21}B_2 + \ldots + x_{m1}B_m
\]

where \(B_i\), for \(i = 1, \ldots, m\) , are numbers independent from all variables \(x_{i1}\), for \(i = 1, \ldots, m\). Since
\[
x_{11} + x_{21} + \ldots + x_{m1} = 1 ,
\]

the pessimistic estimator \(U(root)\) is a convex combination of the numbers \(B_i\). Consequently, the minimum of the numbers \(B_i\) must satisfy \(\min_i(B_i) \leq U(root)\). We assign job \(j = 1\) to a machine \(i\) such that \(i = \arg\min_i\{B_i\}\).
The rounding procedure continues in the same way, by assigning one by one all remaining jobs. It examines in turn each job $j$ (i.e. each set of variables $\{x_{1j}, x_{2j}, \ldots, x_{mj}\}$) and assigns it to the machine (i.e. it sets the corresponding $x_{ij} = 1$ and the remaining $m - 1$ variables $x_{ij} = 0$), with the criterion to minimize (or at least not to increase) the value of the function $U$. The outcome is a deterministic rounded schedule that satisfies the approximation guarantees of the randomized rounding procedure.

Note that when the rounding is made with the method of conditional probabilities we can use $\rho = 1$ for the bound on the probability of failure.

7.4 Complexity

Assuming infinite precision for the computation of the exponentials, it is easy to show that the rounding step with the method of conditional probabilities can be implemented in linear time. During the rounding procedure we use the products

- $\prod_{j=k}^{n} (x_{ij}e^{tp_{ij}} + 1 - x_{ij})$ for $i = 1, \ldots, m$ and $k = 1, \ldots, n$, and
- $\prod_{j=k}^{n} (\sum_{i=1}^{m} x_{ij}e^{tc_{ij}})$ for $k = 1, \ldots, n$.

We can for example calculate and store the values of all the above products for every $k = 1, \ldots, n$. This requires the storage of a linear number of results.

Using the stored results, the value of the pessimistic estimator at each step of the rounding procedure can be calculated in constant time. Consequently, for all problems considered in this work, rounding with pessimistic estimators requires linear time, and the overall complexity of the derandomized algorithms is equivalent to the complexity of the corresponding randomized algorithms.

**Proposition 7.1** The derandomization of algorithm $A$–GAP with the method of pessimistic estimators generates algorithm $A_{\text{det}}$–GAP, a deterministic linear time relaxed decision procedure for GAP.

8 Chernoff bounds

In this Section we prove Chernoff bounds and Hoeffding-Chernoff bounds for the deviations, above and below the mean value, of

- weighted sums of independent Bernulli trials, and
8.1 Deviation above the mean value

The following standard Chernoff-like bound is given in (16):

**Theorem 8.1** (16, Theorem 1). Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be real numbers in $(0,1]$. Let $X_1, X_2, \ldots, X_n$ be independent Bernoulli trials with $E[X_j] = p_j$. Let $\Psi = \sum_{j=1}^{n} \alpha_j X_j$. Then $E[\Psi] = \sum_{j=1}^{n} \alpha_j p_j = S$. Let $\delta > 0$, and $S = E[\Psi] > 0$. Then:

\[
Pr[\Psi > (1 + \delta)S] < \left[ \frac{e^{\delta}}{(1 + \delta)^{(1+\delta)}} \right]^S .
\]  

**Proof:** We include the proof given in (16) because we use an intermediate result (Equation 24) of the proof in the method of conditional probabilities.

For any positive real $t$:

\[
Pr[\Psi > (1 + \delta)S] = Pr[e^{t\Psi} > e^{t(1+\delta)S}] < e^{-t(1+\delta)S} E[e^{t\Psi}]
\]  

where the right inequality follows from the Markov inequality. Now, since the variables are independent

\[
Pr[\Psi > (1 + \delta)S] < e^{-t(1+\delta)S} \prod_{j=1}^{n} (p_j e^{t\alpha_j} + 1 - p_j),
\]  

and using that $1 + x \leq e^x$, for $x \geq 0$, gives

\[
Pr[\Psi > (1 + \delta)S] < e^{-t(1+\delta)S} \prod_{j=1}^{n} \exp[p_j (e^{t\alpha_j} - 1)].
\]  

Substituting $t = \ln(1 + \delta)$ and using that $(1 + \delta)^\alpha \leq 1 + \delta \alpha$, for $0 \leq \delta \leq 1$ and $0 \leq \alpha \leq 1$ gives:

\[
(1 + \delta)^{-(1+\delta)m} \exp\left[\sum_{j=1}^{n} (p_j \delta \alpha_j)\right] \leq \left[ \frac{e^{\delta}}{(1 + \delta)^{(1+\delta)}} \right]^m .
\]  

Consider the more general definition:
Definition 8.1 For \( j = 1, \ldots, n \), let \( X_j \)'s be independent discrete random variables such that

\[
X_j = \begin{cases} 
  c_{1j}, & \text{with probability } x_{1j} \\
  c_{2j}, & \text{with probability } x_{2j} \\
  \vdots \\
  c_{mj}, & \text{with probability } x_{mj} 
\end{cases}
\]

where \( \sum_{i=1}^{m} x_{ij} = 1 \), for \( j = 1, \ldots, n \) and \( 0 \leq c_{ij} \leq 1 \), for \( i = 1, \ldots, m \) and \( j = 1, \ldots, n \).

Proposition 8.1 Let \( X_j \) be discrete random variables as defined in Definition 8.1. Let \( \Psi = \sum_{j=1}^{n} X_j \). Then \( E[\Psi] = \sum_{j=1}^{n} p_j = S \). Let \( \delta > 0 \), and \( S = E[\Psi] > 0 \). Then:

\[
Pr[\Psi > (1 + \delta)S] < \left[ \frac{e^{\delta}}{(1 + \delta)^{(1+\delta)}} \right]^S .
\]

Proof: We start as in the proof of Theorem 8.1. For any positive real \( t \):

\[
Pr[\Psi > (1 + \delta)S] = Pr[e^{t\Psi} > e^{t(1+\delta)S}] < e^{-t(1+\delta)S} E[e^{t\Psi}]
\]

Since the variables \( X_j \) are independent

\[
P = Pr[\Psi > (1 + \delta)S] < e^{-t(1+\delta)S} \prod_{j=1}^{n} \left( \sum_{i=1}^{m} x_{ij} e^{t c_{ij}} \right) .
\]

Substituting \( t = \ln(1 + \delta) \) and using that \((1 + \delta)^{\alpha} \leq 1 + \delta \alpha \), for \( 0 \leq \delta \leq 1 \) and \( 0 \leq \alpha \leq 1 \) gives:

\[
P < (1 + \delta)^{-(1+\delta)S} \prod_{j=1}^{n} \left( \sum_{i=1}^{m} x_{ij} (1 + \delta c_{ij}) \right) = (1 + \delta)^{-(1+\delta)S} \prod_{j=1}^{n} (1 + \delta \sum_{i=1}^{m} x_{ij} c_{ij}) ,
\]

and, finally, using that \( 1 + x \leq e^x \), for \( x \geq 0 \), gives

\[
P < (1 + \delta)^{-(1+\delta)S} \prod_{j=1}^{n} \exp\left( \delta \sum_{i=1}^{m} x_{ij} c_{ij} \right) = \left[ \frac{e^{\delta}}{(1 + \delta)^{(1+\delta)}} \right]^S .
\]

The following technical Lemma, which is easy to be shown, is used to replace the right hand side of Equation 22 with a simpler expression.
Lemma 8.1 ((7, Page 200, Lemma 2.4)). For all $x \geq 0$,

$$(1 + x) \ln(1 + x) - x \geq 3x^2/(6 + 2x). \quad (31)$$

Let $B(S, \delta) = e^{-\delta^2 S / (2(1 + \delta)^2)}$. From Theorem 8.1 and Lemma 8.1 we get:

Lemma 8.2 For $\delta > 0$ and $S \geq 0$

$$\left[ \frac{e^{\delta}}{(1 + \delta)^{(1 + \delta)}} \right]^S \leq B(S, \delta), \text{ and for } \delta < 1 : B(S, \delta) \leq e^{\left( -\frac{\delta^2 S}{4} \right)} \quad (32)$$

8.2 Deviation below the mean value

Chernoff and Hoeffding-Chernoff bounds for deviations below the mean value can be shown in the same way as the bounds for deviations above the mean value. The following Chernoff bound is given in (16, Theorem 2):

Theorem 8.2 Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be real numbers in $(0, 1]$. Let $X_1, X_2, \ldots, X_n$ be independent Bernulli trials with $E[X_j] = p_j$. Let $\Psi = \sum_{j=1}^{n} \alpha_j X_j$. Then $E[\Psi] = \sum_{j=1}^{n} \alpha_j p_j = S$. For $\delta \in (0, 1)$, and $S = E[\Psi] > 0$. Then:

$$\Pr[\Psi < (1 - \delta)S] < \left[ \frac{e^{(-\delta)}}{(1 - \delta)^{(1 - \delta)}} \right]^S. \quad (33)$$

The proof of the following Hoeffding-Chernoff bound is similar to the proof of Proposition 8.1:

Proposition 8.2 Let $X_j$ be discrete random variables as defined in Definition 8.1. Let $\Psi = \sum_{j=1}^{n} X_j$. Then $E[\Psi] = \sum_{j=1}^{n} p_j = S$. Let $\gamma \in (0, 1]$, and $S = E[\Psi] > 0$. Then:

$$\Pr[\Psi < (1 - \gamma)S] < \left[ \frac{\gamma}{(1 - \gamma)^{(1 - \gamma)}} \right]^S. \quad (34)$$

We use McLaurin’s expansion (as it is used in (15, Theorem 4.2)) for $\ln(1 - \delta)$ to replace the right hand side of the bound:

$$\left[ \frac{e^{(-\gamma)}}{(1 - \gamma)^{(1 - \gamma)}} \right]^S < e^{-\frac{\gamma^2 S}{4}}. \quad (35)$$

32
Finally, it easy to show that all bounds (for deviations below and above the mean value) of this Section hold when the sum $\Psi$ includes an arbitrary constant $\lambda \geq 0$, i.e. $\Psi = \lambda + \sum_{j=1}^{n} \alpha_j X_j$.

9 Approximate Enumeration

In this Section, we present techniques $T_1$ and $T_2$ for approximate enumeration of job assignments. Technique $T_1$, a grouping technique of (10), is used to generate a number of representative assignments (as it is used in (11)). Technique $T_2$, is a geometric grouping technique, which selects the final set of assignments. Finally, algorithm $A_L$ that combines techniques $T_1$ and $T_2$, is presented. The techniques are illustrated for algorithm A–SCHED, and can be adapted for algorithms A–GAP, A–MSCHED and A–COV.

Recall from Section 3 that $J_k$ is the set of $k$ large jobs and $\Phi$ is the set of all possible assignments of the large jobs to the machines. The cardinality of $\Phi$ depends exponentially on $1/\epsilon$. The techniques of this Section are used to generate a substantially smaller subset $\Phi_f \subseteq \Phi$, with cardinality polynomial on $\epsilon$. This improves the time complexity of A–SCHED to fully polynomial, at the cost of introducing an arbitrarily small constant error factor to the approximation guarantee of the final solution.

9.1 Technique $T_1$ (10; 11)

For every valid assignment $\varphi$, the load $\varphi_i$ on machine $i$ is $0 \leq \varphi_i \leq D$. We partition the interval $[0, \mu]$ into $N = k/\epsilon_5$ sub-intervals each of size at most $\mu \epsilon_5/k$. Two assignments $\varphi, \chi \in \Phi$, are considered to be in the same group, if for each machine $i$, their respective loads $\varphi_i$ and $\chi_i$ are in the same sub-interval. From all assignments that belong to the same group, the algorithm considers only one arbitrary of them. Let $\Phi_f \subseteq \Phi$ be a set with at most one assignment of each group. The cardinality of $\Phi_f$ is $O((m \log (m/\rho) \epsilon^{-1})^{O(m)})$. The set $\Phi_f$ can be generated in time $O((m \log (m/\rho) \epsilon^{-1})^{O(m)})$ (11, Section 2.1).

Let $\varphi^* \in \Phi$ be an assignment that corresponds to an optimal schedule. Then there is a corresponding assignment $\bar{\varphi}^*$ in $\Phi_f$, such that $\forall i$, $\bar{\varphi}^*_i \leq \varphi^*_i \cdot (1 + \epsilon_5)$. Hence enumerating the assignments in $\Phi_f$ instead of $\Phi$ improves the complexity of algorithm A–SCHED to fully polynomial at the cost of an extra error ratio of at most $(1 + \epsilon_5)$ to the final solution.
9.2 Technique $T_2$

Technique $T_2$ is a geometric grouping technique that is applied on top of technique $T_1$. The interval $[0, \mu]$ is again partitioned, this time to geometrically increasing sub-intervals: $[0, \epsilon_6\mu], (\epsilon_6\mu, \epsilon_6(1+\epsilon_6)\mu], (\epsilon_6(1+\epsilon_6)\mu, \epsilon_6(1+\epsilon_6)^2\mu], \ldots, (\epsilon_6(1+\epsilon_6)^L\mu, \mu]$, where $L = \left\lceil \frac{\log(1+\epsilon_6)}{\log(1+\epsilon_6)} \right\rceil$. Any two assignments $\varphi, \chi \in \Phi$ are considered to be in the same group, if for each machine $i$, their respective loads $\varphi_i$ and $\chi_i$ on machine $i$ are in the same sub-interval. Technique $T_2$ partitions the assignments $\varphi \in \Phi$ into at most $L^m$ groups. Let $\Phi_2 \subseteq \Phi_1$ be the set of all assignments of $\Phi_1$ with at most one assignment in each of the groups of $T_2$.

Lemma 9.1 For every $x \in [0, 1]: \frac{1}{2}x \leq \ln(1+x)$.

Proof: It is sufficient to show that $f(x) = \ln(1+x) - \frac{1}{2}x \geq 0$ for $x \in [0, 1]$. This is true since $f(0) = 0$ and $f'(x) = \frac{1}{1+x} - \frac{1}{2} \geq 0$ for $x \in [0, 1]$. □

The cardinality of $\Phi_2$ is $O(L^m) = O\left(\left(\frac{\log(1/\epsilon_6)}{\log(1+\epsilon_6)}\right)^m\right) = O\left(\left(\frac{2\log(1/\epsilon_6)}{\epsilon_6}\right)^m\right)$. The last equality is due to Lemma 9.1. Using technique $T_2$ introduces an extra error ratio of at most $(1 + \epsilon_6)$ to the final solution. The following algorithm $A_L$ implements techniques $T_1$ and $T_2$ and generates the set $\Phi_f$ from $\Phi$.

Algorithm $A_L$:

$\Phi(0) = \{(0, 0, \ldots, 0)\}$;
Loop 1: For each large job $j = 1, \ldots, k$:
  Loop 2: For each $n$-tuple $t$ in $\Phi(j-1)$:
    Loop 3: For each of the $i = 1, \ldots, m$ machines:
      Generate the $n$-tuple
      $t' = t + (0, \ldots, p_{ij}, \ldots, 0)$ (Add $p_{ij}$ to position $i$ of tuple $t$.)
      Insert $t'$ into $\Phi(j)$, unless there is already an assignment
      of the same group of technique $T_1$.
End Loop 3, 2, 1
$\Phi_1 = \Phi(k)$
The final set $\Phi_f = \Phi_2 \subseteq \Phi_1$ is the set of all assignments of $\Phi_1$
with at most one assignment in each of the groups of technique $T_2$.

Algorithm $A_L$ uses technique $T_1$ to generate a set $\Phi_1$ of representative assignments (at most one assignment for each group of $T_1$) and then applies technique $T_2$ to select the final set of assignments $\Phi_f \subseteq \Phi_1 \subseteq \Phi$ with at most one assignment from each group in $T_2$. If $\Phi_f$ is used by algorithm A–SCHED, the number of iterations of the main part of A–SCHED is $O(\min(|\Phi_1|, |\Phi_2|))$. The cost is the extra factor $(1+\epsilon_5) \cdot (1+\epsilon_6)$ in the approximation guarantee of the final solution.
Lemma 9.2 The cardinality of $\Phi_f$ and the running time for generating it are $O\left(\min\left\{\left(\frac{m\log(m/\rho)}{\epsilon}\right)^{O(m)}, \left(\frac{2\log(1/\epsilon)}{\epsilon}\right)^{m}\right\}\right)$ and $\left(\frac{m\log(m/\rho)}{\epsilon}\right)^{O(m)}$ respectively. Using $\Phi_f$ instead of $\Phi$ in algorithms $A$–SCHED and $A$–GAP introduces at most an arbitrary small constant error factor of $(1 + \epsilon_5) \cdot (1 + \epsilon_6)$ to the final solution.

10 Discussion

We present a general framework for approximation schemes for a class of job assignment problems\(^3\). The key ingredient of our approach is the combinatorial randomized rounding (CRR) technique, i.e. a class of rounding procedures that exploit Chernoff and Hoeffding-Chernoff bounds, take advantage of decision procedures, distinguish between large and small problem coefficients and enhance the conventional randomized rounding procedure with combinatorial arguments. The CRR approach does not depend on any particular problem structure and should find further applications. For example, in assignment problems where the number $m$ of machines is part of the input, our approach produces approximation schemes for mixed solutions, where a number of large jobs is assigned fractionally. Furthermore, the randomized algorithms of this work can be efficiently implemented in parallel and distributed settings.

CRR reduces the gap between fractional and integer solutions from a logarithmic factor on $m$ of standard RR to an arbitrary constant factor. An interesting question is, if this important improvement is due to a much better integer solution or mainly due to a more tight fractional solution (or both).

The success of the algorithmic techniques of this work provides theoretical evidence for the following sound approach for heuristics: Given a hard problem, first identify a small number of critical decisions, and then solve for each possible combination of these decisions, a corresponding, much easier, subproblem.

References


\(^3\) Preliminary results of this work were given in (4).


APPENDIX

A. Notation

A.1 General notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>The standard randomized rounding procedure (Section 2)</td>
</tr>
<tr>
<td>CRR</td>
<td>The combinatorial randomized rounding procedure</td>
</tr>
<tr>
<td>CO</td>
<td>Combinatorial Optimization</td>
</tr>
<tr>
<td>LP</td>
<td>Linear Programming</td>
</tr>
<tr>
<td>$T_1$ and $T_2$</td>
<td>The grouping techniques described in Section 9</td>
</tr>
<tr>
<td>Quasi-linear</td>
<td>The product of a linear and a polylogarithmic function</td>
</tr>
<tr>
<td>LogPDD</td>
<td>The logarithmic-potential based price directive decomposition algorithm (LogPDD) of Grigoriadis and Khachiyan (6)</td>
</tr>
<tr>
<td>A–Young</td>
<td>The relaxed decision procedure for mixed packing and covering problems of Young (22)</td>
</tr>
</tbody>
</table>

A.2 Error ratios $\epsilon_i$ in the approximation algorithms

<table>
<thead>
<tr>
<th>Error Ratio</th>
<th>SCHED, GAP</th>
<th>COV</th>
<th>EQUAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon_1$</td>
<td>Binary Search</td>
<td>Binary Search</td>
<td>Binary Search</td>
</tr>
<tr>
<td>$\epsilon_2$</td>
<td>LogPDD</td>
<td>A–Young</td>
<td>A–Young</td>
</tr>
<tr>
<td>$\epsilon_3$</td>
<td>Unlucky Jobs</td>
<td>Rounding</td>
<td>Corrections</td>
</tr>
<tr>
<td>$\epsilon_4$</td>
<td>Filtered Rounded</td>
<td>Corrections</td>
<td>Deviation Below</td>
</tr>
<tr>
<td>$\epsilon_5$</td>
<td>Technique $T_1$</td>
<td>Technique $T_1$</td>
<td>Deviation Above</td>
</tr>
<tr>
<td>$\epsilon_6$</td>
<td>Technique $T_2$</td>
<td>Technique $T_2$</td>
<td></td>
</tr>
</tbody>
</table>