Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 359 (2006) 400–417
Theoretical Computer Science
www.elsevier.com/locate/tcs

Approximation schemes for scheduling and covering on unrelated machines

Pavlos S. Efraimidis[a,*], Paul G. Spirakis[b,1]

[a] *Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece*
[b] *Computer Technology Institute, Riga Feraiou 61, 26221 Patras, Greece*

Received 23 December 2004; received in revised form 23 December 2005; accepted 12 May 2006

Communicated by G. Ausiello

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 a new rounding approach that yields approximation schemes for multi-objective minimum makespan scheduling with a fixed number of linear cost constraints. The same approach gives approximation schemes for covering problems like maximizing the minimum load on any machine, and for assigning specific or equal loads to the machines.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Randomized rounding; Multi-objective scheduling; Covering; Approximation schemes; 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 a new rounding procedure that yields approximation schemes for the following job assignment problems: multi-objective scheduling with an arbitrary constant number of linear cost constraints, maximizing the minimum load on any machine, and assigning specific or equal loads to unrelated machines. More precisely, we present approximation schemes for the following problems:

- *MkSCH*: Multi-objective scheduling with the makespan objective and an arbitrary constant number of cost constraints. The makespan of a schedule is the maximum load on any machine. A cost constraint a is defined in the following way: the assignment of a job j to a machine i has, besides the processing time p_{ij}, a cost c^a_{ij}. The objective is to find a schedule of bounded makespan and costs.

* Corresponding author. Tel.: +30 25410 79756; fax: +30 25410 79764.
E-mail addresses: pefraimi@ee.duth.gr (P.S. Efraimidis), spirakis@cti.gr (P.G. Spirakis).
[1] Part of this work is funded under the EU IP project DELIS in the proactive initiative on Complex Systems.

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.05.025

  - *SCHED* (*or M0SCH*): The special case of MkSCH with no cost constraints. SCHED is the common scheduling problem with the objective to minimize the makespan.
  - *GAP* (*or M1SCH*): The case of MkSCH with 1 cost constraint. This problem, which is known as the generalized assignment problem, has the objective to find a schedule of bounded makespan and cost.
  - *M2SCH*: Similar to M1SCH but with 2 cost constraints.
- *COV*: The symmetric problem of SCHED, with the objective to maximize the minimum load of any machine.
- *EXACT*: The problem of assigning a specific load to every machine.
- *EQUAL*: The problem of minimizing the makespan when all machines must have equal load.

The approximation algorithms of this work are based on one of the foremost paradigms in the design of approximation algorithms: a 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 specific algorithmic approach that we present, develops on the following principle, which we consider as the most important contribution of this work: *Before applying randomized rounding we identify the most critical decisions that must be taken to solve the problem, and then each critical decision is either neutralized or a backup choice is provided for it. If necessary, we correct after the rounding step any bad choices taken by the randomized procedure.* We use this principle to build rounding procedures, which we call combinatorial randomized rounding (CRR) procedures. The CRR procedures achieve strong approximation ratios by neutralizing large coefficients that can make the deviation bounds weaker.

The importance of using randomness not only in a blind but also in an intelligent way has recently been stressed in the fields of heuristics [5] and approximation algorithms. A characteristic example is the use of randomization with two or more choices that has been studied in several recent works [21,27]. Even though conventional randomized rounding cannot be considered a completely 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 carefully guiding the rounding process.

The CRR rounding approach combines conventional randomized rounding with combinatorial and algorithmic techniques. Linear programming techniques and standard combinatorial arguments are used to support the randomized rounding procedure. Decision procedures are used to achieve tighter LP relaxations. Appropriate Hoeffding–Chernoff bounds are applied to bound deviations and it is shown how to exploit their properties within a general randomized rounding procedure. Finally, all CRR-based randomized algorithms of this work are derandomized with the method of conditional probabilities [23], and yield deterministic algorithms of equivalent time complexity.

We would like to note that our work is reminiscent but different from the related work of Jansen and Porkolab in [17]. The CRR-based algorithms of this work focus on a carefully guided randomized rounding procedure. From [17], which was the original motivation for this work, we draw some algorithmic machinery and the following critical idea: it is possible to tolerate a number of corrections to the final approximate solution by applying an appropriate preprocessing step at the start of the algorithm. While this idea is used in [17] in a deterministic context, we use it here to derive a new way to amplify the performance of randomized rounding.

1.1. Preliminaries

Given a minimization (respectively, maximization) problem and a constant $\varepsilon > 0$, a $(1+\varepsilon)$ (respectively, $1-\varepsilon$) approximate solution x is a solution such that $|OPT - Obj(x)| \leqslant \varepsilon OPT$, where OPT is the optimal objective value and $Obj(x)$ is the objective value of x. A polynomial time approximation scheme (PTAS) for a minimization (respectively, maximization) problem is an algorithm that for each $\varepsilon > 0$, finds a $(1+\varepsilon)$ (respectively, $1-\varepsilon$) approximate solution in time polynomial in the problem size N. A PTAS with running time polynomial on $1/\varepsilon$, is a fully polynomial time approximation scheme (FPTAS). A relaxed decision procedure (RDP) is a polynomial time algorithm that accepts as input a minimization (respectively, maximization) problem instance and a value T for the objective function, and either finds a feasible solution with objective value at most $(1+\varepsilon)T$ (respectively, at least $(1-\varepsilon)T$), or decides that there is no solution with objective value at most (respectively, at least) T. A RDP that is polynomial on $1/\varepsilon$ is fully polynomial. A randomized RDP (RRDP) is a randomized RDP, such that the probability that it fails to find a relaxed solution for a feasible problem instance is strictly less than a given value $\rho : \; 0 < \rho \leqslant 1$. Detailed definitions of approximation schemes and a thorough presentation of their application in scheduling problems are given in [24].

Problem	This work	Known result
MkSCH	RDP: $n \cdot O(E^{m+k}\, M \ln(M))$ $M = (m+k)\epsilon^{-1}$	-
SCHED	RDP: Like MkSCH for $k = 0$ FPTAS: $O(log(m/\epsilon)$ times the RDP	$n\,(m/\epsilon)^{O(m)}$ FPTAS (17)
GAP	RDP: Like MkSCH for $k = 1$ FPTAS: $O(log(m\epsilon^{-1} \cdot n)$ times the RDP	$n\,(m/\epsilon)^{O(m)}$ RDP (17)
COV	RDP: $n \cdot O(E^{m}\, m\, \epsilon^{-2})$ FPTAS: $O(log(m\epsilon^{-1} \cdot n)$ times the RDP	m identical machines (28) m related machines (4)
EXACT	RDP: $O(m\, \mu^{m}\, \epsilon^{-2}\, n^{m\mu+1})$, for $\mu = 2\epsilon^{-2} \ln m$	-
EQUAL	PTAS: $O(\log(m\epsilon^{-1} \cdot n))$ times EXACT	-
$E = O(\log(1/\epsilon)/\log(1+\epsilon)+1)$ and k = number of cost constraints in MkSCH.		

Fig. 1. Results of this work and related results.

1.2. Results and related work

Scheduling is an active, widely studied field [12,1,18,25,20]. Bicriteria scheduling has also been studied for example in [26] and in more recent works like [2,3,20]. Problem SCHED is known to be NP-hard even for $m = 2$ [9]. The proof is based on a reduction from Partition [9, Problem SP12]. Similar reductions can be used to show that GAP, MkSCH, GOV, EXACT and EQUAL are also NP-hard for $m = 2$. A FPTAS for SCHED was given in [16], and later, an interesting PTAS for SCHED was presented in [19]. The best known results for SCHED and GAP, are a linear time FPTAS for SCHED and a linear time RDP decision procedure for GAP, given by Jansen and Porkolab in [17]. We present the first, to our knowledge, FPTAS for the general MkSCH problem. While being simpler and more general, the proposed algorithm matches the best known results for the specific cases of 0 (SCHED) or 1 (GAP) cost constraints. The main results of this work and related known results are summarized in Fig. 1.

Problem COV or else maximizing the minimum machine completion time is a well known problem with applications in sequencing of maintenance actions for modular gas turbine aircraft engines [8] and generally in systems that are alive only when all machines are alive and the systems should be maintained alive as long as possible [4]. PTAS's for COV are presented in [28] for *identical* machines and in [4] for *related* machines. COV and on-line versions of COV are studied in [6,14,13] for identical machines. The above algorithms are polynomial on m. We present the first, to our knowledge, approximation of COV for (a fixed number of) unrelated machines. Scheduling unrelated machines is a more general version of COV and we consider it interesting since it can handle collections of machines that are related in an arbitrary, not necessarily linear, way. The complexity of the proposed algorithm is linear for the RDP and quasi-linear for the FPTAS.

Finally we define problems EXACT and EQUAL, two hard combinatorial problems, which have more strict requirements on the machine loads than MkSCH and COV. We present a RDP for EXACT and use it to build a PTAS for EQUAL. The complexity of the presented algorithms is a high degree polynomial. However these approximation schemes are, to our knowledge, the first PTAS's for these problems.

1.3. Assumptions

We assume that the number of machines m is $m \geqslant 2$. For maximization problems the error ratio parameter of the approximation schemes $\varepsilon > 0$ is necessarily always less than 1. We will assume that this also holds in the case of minimization problems. This assumption is wlog and is done to simplify the calculation of certain error-related coefficients that are used in the algorithms. *Hence, we assume that in all approximation schemes the overall error ratio parameter ε is* $0 < \varepsilon < 1$. In the approximation algorithms, the overall approximation ratio ε is the aggregation of several smaller approximation ratios ε_i. In order to be precise, we distinguish in the analysis each smaller ratio ε_i with an index i. Fig. 2 provides a brief explanation of the error ratios ε_i in the different algorithms.

Error Ratio	MkSCH	COV	EXACT and EQUAL
ϵ_1	Technique $\mathcal{T}_1$	Technique $\mathcal{T}_1$	A-Young
ϵ_2	Technique $\mathcal{T}_2$	Technique $\mathcal{T}_2$	Rounding
ϵ_3	LogPDD	A-Young	Binary Search (EQUAL)
ϵ_4	Rounding	Rounding	
ϵ_5	Unlucky Jobs	Unlucky Machines	
ϵ_6	Binary Search	Binary Search	

Fig. 2. Error ratios.

The rest of this work is organized in the following way. Multi-objective scheduling MkSCH is discussed in Section 2. The problem of maximizing the minimum load (COV) is discussed in Section 3. Problems EXACT and EQUAL are discussed in Section 4. We conclude in Section 5 with a discussion.

2. Multi-objective scheduling

In this section, we consider problem MkSCH and show how a CRR-based fully linear time RDP can be built for it. For simplicity, we illustrate our approach on problem M2SCH, the special case of MkSCH with $k = 2$ cost constraints, and present algorithm A-M2SCH, a fully linear time RDP for M2SCH. Adapting A-M2SCH to handle problem SCHED (M0SCH), GAP (M1SCH) or any other case of MkSCH is straightforward. We shortly discuss these adaptations and compare the algorithms with the related results of [17] in Section 2.3.

Algorithm A-M2SCH is a fully linear time RRDP decision procedure for M2SCH. Given an instance of M2SCHED, values $\varepsilon : 0 < \varepsilon \leqslant 1$ and $\rho : 0 < \rho < \frac{1}{2}$ and values T, C^1, C^2 then, if the combination T, C^1, C^2 is feasible, with probability of success strictly larger than $(1-\rho)$, A-M2SCH produces a schedule of makespan at most $(1+\varepsilon)T$ and costs at most $(1+\varepsilon)C^1$ and $(1+\varepsilon)C^2$, respectively. The IP formulation of M2SCH is

$$\text{Find } [x_{ij}] \text{ s.t. : } \begin{cases} \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}^{a} x_{ij} \leqslant C^a \;\; (a = 1, 2), \\ \sum_{j=1}^{m} p_{ij} x_{ij} \leqslant T \;\; (i = 1, \ldots, m), \\ \sum_{i=1}^{m} x_{ij} = 1 \;\; (j = 1, \ldots, n), \\ x_{ij} \in \{0, 1\} \;\; (i = 1, \ldots, m;\; j = 1, \ldots, n). \end{cases}$$

Algorithm A-M2SCH.
Input: An instance of M2SCH and values $\varepsilon, \rho, T, C^1$ and C^2.
Step 0: *Normalization.* For $i = 1, \ldots, 6$ let $\varepsilon_i = \varepsilon/9 = \Theta(\varepsilon)$. Let $\mu = 3\ln((m+2)/\rho)\varepsilon_4^{-2}$ and scale the problem so that $T = C^1 = C^2 = \mu$.
Step 1: *Initializations.* Let $q = \lceil 3\,(m+2)^2\,\varepsilon_5^{-1}\,\mu \rceil$ and $\forall j,\ d_j = \min_i\{p_{ij} + c_{ij}^1 + c_{ij}^2\}$. Check that $\sum_j d_j \leqslant (m+2)\,\mu$, else ($T$,$C$1,$C$2) is not feasible.
Step 2: *Initial filtering.* $\forall i, j$: if $max\{p_{ij}, c_{ij}^1, c_{ij}^2\} > \mu$ *then* $x_{ij} = 0$.
Step 3: *Large jobs.* Let J_ℓ be the set of large jobs $J_\ell = \{j \mid d_j$ belongs to the q largest d_j (ties are resolved arbitrarily)$\}$. Let Φ be the set of all possible assignments of the large jobs and let Φ_2 be an appropriate subset $\Phi_2 \subseteq \Phi$.
Step 4: *Feasible fractional schedule.*
$\forall$ assignment $\varphi \in \Phi_2$:
(1) Formulate the integer program IP-M2SCH(φ).
(2) Relax IP-M2SCH(φ) to the linear program LP-M2SCH(φ).
(3) Find a fractional $(1+\varepsilon_3)$-approximate solution to LP-M2SCH(φ).
(4) If the fractional solution satisfies the relaxed problem, then select it and abandon the loop.
Let $[x_{ij}]$ be the feasible fractional solution that has been selected.

Step 5: *Combinatorial randomized rounding.*

(1) If $p_{ij} > 1$ set $p_{ij} = 1$ and mark the coefficient p_{ij}. The same is done for large coefficients $c^1_{ij} > 1$ and $c^2_{ij} > 1$.
(2) Round the schedule $[x_{ij}]$ with exclusive randomized (XRR).
(3) *Filtering*: If a job j is randomly assigned to a marked coefficient p_{ij} or c^1_{ij} or c^2_{ij}, then call it "unlucky" and remove it from the rounded schedule.
(4) Assign each unlucky job j to a machine $m_j = \text{argmin}_i\{p_{ij} + c^1_{ij} + c^2_{ij}\}$.

Analysis of algorithm A-M2SCH: Given an instance of M2SCH, the machine load constraints are scaled with the factor μ/T and the two cost constraints are scaled with $\mu/C1$ and $\mu/C2$, respectively. This is done to simplify the analysis. Now, the rhs of all constraints is μ, and the objective is to find a schedule of makespan and costs at most μ. Note that a necessary condition for the combination of values $(T, C1, C2)$ to be feasible is that in the normalized problem:

$$\sum_j d_j \leqslant (m+2)\,\mu. \tag{1}$$

Step 2: *Initial filtering.* $\forall i, j$: if $p_{ij} > \mu$ *then* $x_{ij} = 0$. This step makes the LP relaxation more tight and has no impact on the feasibility of the problem.

Step 3: *Large jobs.* The number of large jobs is $q = \lceil 3\,(m+2)^2\,\mu\,\varepsilon_5^{-1}\rceil = \mathrm{O}((m+2)^2 \ln((m+2)/\rho)\varepsilon^{-3})$. Algorithm A-M2SCH has to guess the assignment φ^* of the large jobs in an optimal solution. Since φ^* is not known in advance, the algorithm examines possible assignments of the large jobs.

Approximate enumeration: The cardinality of the set Φ of all possible assignments of the large jobs to the machines is m^q, a constant that depends exponentially on $1/\varepsilon$. However, it is sufficient to examine only a substantially smaller subset $\Phi_2 \subseteq \Phi$, with cardinality polynomial on $1/\varepsilon$. We use a combination of two techniques $\mathcal{T}_1$ and $\mathcal{T}_2$ for approximate enumeration of job assignments. A grouping technique of [16], which we call $\mathcal{T}_1$, is used to generate a number of representative assignments exactly as it is used in [17]. Technique $\mathcal{T}_1$ reduces the maximum number of large job assignments that have to be examined from m^q to $((m+2)q\varepsilon_1^{-1})^{m+2} = (m\log(m/\rho)\varepsilon^{-1})^{\mathrm{O}(m)}$. The running time for $\mathcal{T}_1$ is $(m\log(m/\rho)\varepsilon^{-1})^{\mathrm{O}(m)}$. For technique $\mathcal{T}_1$, the reader is referred to the enumeration technique in [17].

Technique $\mathcal{T}_2$: Technique $\mathcal{T}_2$ is a geometric grouping technique that partitions the interval $[0, \mu]$ into geometrically increasing sub-intervals: $[0, \varepsilon_2\mu]$, $(\varepsilon_2\mu, \varepsilon_2(1+\varepsilon_2)\mu]$, $(\varepsilon_2(1+\varepsilon_2)\mu, \varepsilon_2(1+\varepsilon_2)^2\mu], \ldots, (\varepsilon_2(1+\varepsilon_2)^{E-1}\mu, \mu]$, where

$$E = \left\lceil \frac{\log(1/\varepsilon_2)}{\log(1+\varepsilon_2)} + 1 \right\rceil. \tag{2}$$

Any two assignments $\varphi, \chi \in \Phi$ are considered to be in the same group, if for each machine i, their respective loads φ_i and χ_i on machine i are in the same sub-interval. The outcome of $\mathcal{T}_2$ are at most E^{m+2} different assignments.

Since the quality of the approximations given by the algorithms in this work is measured with multiplicative error factors, the geometric grouping technique $\mathcal{T}_2$ is by definition more effective than technique $\mathcal{T}_1$. However, while $\mathcal{T}_1$ can be used to incrementally build assignments of large jobs, it seems that $\mathcal{T}_2$ can only be applied on complete assignments of all large jobs. For this reason we first use $\mathcal{T}_1$ to incrementally generate a representative set $\Phi_1 \subseteq \Phi$ of large job assignments, and then we apply $\mathcal{T}_2$ to select the final set $\Phi_2 \subseteq \Phi_1$. Applying $\mathcal{T}_2$ is very simple: for each assignment φ produced by $\mathcal{T}_1$, we check if no other assignment of the same $\mathcal{T}_2$ group has been produced earlier, and only then we process φ, while else we reject it. This check can be done in O(1) time using constant space $\mathrm{O}(E^{m+2})$. Using $\mathcal{T}_2$ on top of $\mathcal{T}_1$, reduces the number of assignments that have to be examined by algorithm A-M2SCH to at most E^{m+2} without any additional cost to the running time.

Lemma 1. *The cardinality of Φ_2 and the running time for generating it are* $\mathrm{O}(E^{m+2})$ and $(m\log(m/\rho)/\varepsilon)^{\mathrm{O}(m)}$, *respectively. Using Φ_2 instead of Φ in algorithm A-M2SCH introduces at most an arbitrary small constant error factor of* $(1+\varepsilon_1)\,(1+\varepsilon_2)$ *to the final solution.*

Since each assignment φ will cost algorithm A-M2SCH linear time to process it, the application of $\mathcal{T}_1$ and $\mathcal{T}_2$ significantly reduces the coefficient of the linear time term n in the running time of A-M2SCH. With $\mathcal{T}_2$ we do not avoid the cost of running technique $\mathcal{T}_1$, however, this cost is not introduced into the coefficient of the linear time term n in the overall complexity of A-M2SCH.

Step 4: *Fractional schedule.* Given an assignment φ of the large jobs J_ℓ, the problem of assigning the remaining jobs in an optimal way (to minimize the makespan and the costs) can be formulated as the following integer program IP-M2SCH(φ). Let λ_i be the load on machine i due to φ and λ_c^a the cost on cost constraint a due to φ.

$$\min \tau \text{ s.t. } : \begin{cases} \lambda_c^a + \sum_{j=1}^n \sum_{i=1}^m c_{ij}^a x_{ij} \leqslant \tau \ (a = 1, 2), \\ \lambda_i + \sum_{j \in [n]-J_\ell} p_{ij} x_{ij} \leqslant \tau \ (i = 1, \ldots, m), \\ \sum_{i=1}^m x_{ij} = 1 \ (j \in [n] - J_\ell), \\ x_{ij} \in \{0, 1\} \ (i = 1, \ldots, m; \ j \in [n] - J_\ell). \end{cases}$$

Relaxing the integrality constraints on the binary variables x_{ij} to $x_{ij} \geqslant 0$, gives a corresponding linear program LP-M2SCH(φ). Let τ^* be the optimal objective value of LP-M2SCH(φ^*) for the optimal assignment φ^*. Assuming that the combined values T, $C1$ and $C2$ are feasible for the problem instance gives that $\tau^* \leqslant OPT \leqslant \mu$. The linear program LP-M2SCH has m variables for every job j, a packing constraint for the load of every machine i, and two packing constraints for the costs. Hence the variables are grouped into n independent m-dimensional simplexes (blocks) and there is a constant number of positive packing constraints (coupling constraints). As shown in [17], these properties can be exploited by the logarithmic-potential-based price directive decomposition algorithm (LogPDD) of Grigoriadis and Khachiyan [10], to efficiently approximate LP-M2SCH(φ) within any constant factor $(1+\varepsilon_3)$. The following Theorem follows from [10,17]:

Theorem 2. *The linear program LP-M2SCH can be approximated with algorithm LogPDD within any constant ratio* $(1+\varepsilon_3)$ *in* $O(n((m+2)/\varepsilon_3)^2 \ln((m+2)/\varepsilon_3))$ *time.*

Let $[\hat{x}_{ij}]$ be the approximate fractional solution found for the optimal assignment φ^* by algorithm LogPDD with approximation ratio $(1+\varepsilon_3)$. If τ_3 is the objective value of $[\hat{x}_{ij}]$, then:

$$\begin{aligned} &\lambda_c^{*a} + \sum_{j=1}^n \sum_{i=1}^m c_{ij}^a \hat{x}_{ij} \leqslant \tau_3 \quad (a = 1, 2), \\ &\lambda_i^* + \sum_{j \in [n]-J_\ell} p_{ij} \hat{x}_{ij} \leqslant \tau_3 \quad (i = 1, \ldots, m), \\ &\sum_{i=1}^m \hat{x}_{ij} = 1 \quad (j \in [n] - J_\ell), \\ &\hat{x}_{ij} \geqslant 0 \quad (i = 1, \ldots, m; \ j \in [n] - J_\ell). \end{aligned}$$

By the approximation guarantee of LogPDD and since $\tau^* \leqslant \mu$, we get: $\tau_3 \leqslant (1+\varepsilon_3)\,\tau^* \leqslant (1+\varepsilon_3)\,\mu$. Since the assignment φ^* is not known, the algorithm starts to calculate fractional schedules for assignments $\varphi \in \Phi$ given by the enumeration techniques of Lemma 1. Given any $\varphi \in \Phi$, the LogPDD algorithm finds an approximate fractional solution for LP-M2SCH(φ). The process continues until a fractional solution $[x_{ij}]$ is found, such that its objective value τ_3 satisfies:

$$\tau_3 \leqslant (1+\varepsilon_1)\,(1+\varepsilon_2)\,(1+\varepsilon_3)\,\mu. \tag{3}$$

If no such fractional solution is found, the combination of input values T, $C1$ and $C2$ is infeasible for the problem instance.

Step 5: *Rounding.* The fractional solution $[x_{ij}]$ is rounded to an approximate integer schedule $[X_{ij}]$ with the following standard RR procedure, which we call *exclusive RR (XRR)*.

Definition 3 (*XRR*). 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 $[X_{ij}]$ be the rounded schedule and let τ_4 be the objective value of $[X_{ij}]$. Let J be set of all jobs and let J_s be the set of all jobs except the large jobs $J_s = J \backslash J_\ell$. 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 $E[X_{ij}] = x_{ij}$. *Now we can show that, since at this step of the CRR-based algorithm all coefficients are equal or less to* 1 *and the rhs is at least* μ, *the deviation of the rounded solution can be bounded with a constant factor* $(1 + \varepsilon_4)$.

Rounded machine load constraints: For each machine i, the load Ψ_i of i in the rounded schedule is equal to the sum of a given positive value λ_i and the weighted sum of independent Bernoulli trials: $\Psi_i = \lambda_i + \sum_{j \in J_s} p_{ij} X_{ij}$. The probability that Ψ_i is larger than $(1+\varepsilon_4)\,\tau_3$ can be bounded with the following Hoeffding–Chernoff bounds that follow from standard deviation bounds given for example in [23, Theorem 1, 11, p. 200, Theorem 2.3], and other references given in [11].

Proposition 4. *Let* $\lambda \geqslant 0$ *be an arbitrary real, and let* $X_1, X_2, \ldots, X_n$ *be independent random variables, such that, for each* $j = 1, \ldots, n$, $0 \leqslant X_j \leqslant 1$ *and* $E[X_j] = p_j$. *Let* $\Psi = \lambda + \sum_{j=1}^{r} X_j$. *Then* $E[\Psi] = \lambda + \sum_{j=1}^{r} p_j = \tau$. *Let* $\varepsilon > 0$, $\delta \in (0, 1)$, *and* $\tau > 0$. *Then*:

$$P[\Psi > (1+\varepsilon)\tau] < \mathrm{e}^{-\varepsilon^2\tau/2(1+\varepsilon/3)} \quad \textit{and} \quad P[\Psi > (1+\delta)\tau] < \mathrm{e}^{-(1/3)\delta^2\tau}, \tag{4}$$

$$P[\Psi < (1-\delta)\tau] < \mathrm{e}^{-(1/2)\delta^2 S}. \tag{5}$$

Proof. For $\lambda = 0$ the above bounds follow directly from [11, p. 200, Theorem 2.3]. The fact that a part of Ψ of size λ is deterministic can make the deviation bounds only stronger and hence we can ignore this fact. For completeness we provide a proof: we first consider deviations above the mean value, and define $B(S, \varepsilon) = \mathrm{e}^{-\varepsilon^2\tau/2(1+\varepsilon/3)}$. Let $\lambda \geqslant 0$, $\Psi' = \sum_j X_j$ and $\tau = \tau' + \lambda$. If Ψ would contain no deterministic part, then for deviation ε the bound would be $B(S, \varepsilon)$. Assume a deviation ratio ε' for $\Psi' = \Psi - \lambda$ such that the absolute deviations for Ψ' and Ψ (with the assumption that Ψ has no deterministic part) are equal $\varepsilon'\tau' = \varepsilon\tau$. Then it is easy to show that $B(\tau', \varepsilon') \leqslant B(\tau, \varepsilon)$ by first expanding and then simplifying the expressions. Similarly, if we define $C(\tau, \varepsilon) = \mathrm{e}^{-(1/2)\delta^2\tau}$ for deviations below the mean value, it is easy to show that $C(\tau', \varepsilon') \leqslant C(\tau, \varepsilon)$. □

Proposition 5. *The probability that the load on any specific machine in the rounded solution is larger than* $(1+\varepsilon_4)\,\tau_3$, *is strictly less than* $\rho/(m+2)$.

Proof. Assume a specific machine i. For every job $j = 1, \ldots, n$, the load due to job j on machine i, is $X_j = p_{ij} X_{ij}$. The total load of machine i is the sum $\Psi_i = \sum_{j=1}^{n} X_j$. The random variables X_j satisfy the conditions of Proposition 4 and hence we can apply the corresponding bound for deviations above the mean value for $\varepsilon_4 < 1$:

$$\forall i\colon P_i = \mathrm{P}[\Psi_i > (1+\varepsilon_4)\tau_3] < \mathrm{e}^{-(1/3)\varepsilon_4{}^2 3\ln((m+2)/\rho)\varepsilon_4^{-2}} = \frac{\rho}{m+2}. \qquad \square \tag{6}$$

Rounded cost constraints: We have to address separately the deviation of the rounded cost constraints. The reason is that the cost constraints are the weighted sum of all variables X_{ij}. However, by definition, the variables X_{ij} of the XRR procedure that correspond to the same job j are not independent. Hence, the deviation bounds of Proposition 4 cannot be directly applied. We overcome this issue and obtain results equivalent to the results on the deviation of the machine load constraints.

Proposition 6. *The probability that the cost on any specific cost constraint in the rounded solution is larger than* $(1+\varepsilon_4)\tau_3$, *is strictly less than* $\rho/(m+2)$.

Proof. For cost constraint a and for every job j, let X_j^a be the cost induced by job j: $X_j^a = \sum_{i=1}^{m} c_{ij}^a X_{ij}$ on constraint a. Note that $X_j^a \in [0, 1]$ in the rounded schedule. For each constraint a, the variables X_j^a are independent discrete random variables and the total cost of the random schedule on constraint a is $X^a = \sum_{j=1}^{n} X_j^a$. Hence the cost for each rounded cost constraint a is the sum of independent discrete random variables satisfying the conditions of Proposition 4.

We apply the corresponding deviation bound as in Proposition 5:

$$\forall a\colon P_c^a = \mathrm{P}[\Psi_c^a > (1+\varepsilon_4)\tau_3] < \frac{\rho}{m+2}. \qquad \square \tag{7}$$

Proposition 7. *The probability that the objective value τ_4 of the rounded solution is $\tau_4 > (1+\varepsilon_4)\tau_3$, is strictly less than ρ.*

Proof. The sum of all probabilities P_i, $i = 1, \ldots, m$, and P_c^a, $a = 1, 2$, is a sufficient bound on the probability that the load of at least one rounded constraint in the rounded schedule is greater than $(1+\varepsilon_4)\tau_3$:

$$\begin{aligned} P[\tau_4 > (1+\varepsilon_4)\,\tau_3] &= P\left[\exists i : \Psi_i > (1+\varepsilon_4)\,\tau_3 \text{ OR } \exists a : \Psi_c^a > (1+\varepsilon_4)\,\tau_3\right] \\ &\leqslant \sum_i P_i + \sum_a P_c^a \; < \; m\rho/(m+2) + 2\rho/(m+2) = \rho. \qquad \square \end{aligned}$$

Unlucky jobs: Let J_u be the set of all jobs $j \in J_s$ that have been randomly assigned to marked coefficients p_{ij}. *We call these jobs "unlucky" and remove them from the rounded schedule*. To obtain the final schedule we assign every unlucky job $j \in J_u$ to a machine $m_j = \operatorname{argmin}_i\{p_{ij} + c_{ij}^1 + c_{ij}^2\}$. Note the following very critical fact, which is exploited by CRR: *The number of unlucky jobs can bounded with a constant and hence their overhead is tolerable*.

Proposition 8. *If τ_4 is the objective value of the rounded solution, and $\tau_4 \leqslant (1+\varepsilon_4)\,\tau_3$, then reassigning the unlucky jobs costs at most a factor $(1+\varepsilon_5)$ to the objective value.*

Proof. Since the objective value of the rounded solution is τ_4 and every unlucky job contributes at least 1 unit to at least 1 problem constraint, the number $|J_u|$ of unlucky jobs cannot exceed $(m+2)\,\tau_4$.

Each unlucky job is assigned to a machine $i = \operatorname{argmin}_i\{p_{ij} + c_{ij}^1 + c_{ij}^2\}$. We provide a combinatorial argument similar to an argument used in [17] to show that $\sum_{j\in J_u} d_j \leqslant \varepsilon_5\,(1+\varepsilon_1)\,(1+\varepsilon_2)\,(1+\varepsilon_3)\,(1+\varepsilon_4)\,\mu$. Assume to the contrary that $\sum_{j\in J_u} d_j > \varepsilon_5\,(1+\varepsilon_1)\,(1+\varepsilon_2)\,(1+\varepsilon_3)\,(1+\varepsilon_4)\,\mu$. Then, the largest d_j of a job $j \in J_u$ is at least: $\max_{j\in J_u}\{d_j\} > \varepsilon_5\,/\,(m+2)$. However, then the sum of the d_j's of the q largest jobs J_ℓ is: $\sum_{j\in J_\ell} d_j \geqslant q\cdot\max_{j\in J_u}\{d_j\} > q\,\varepsilon_5\,/\,(m+2) = 3\,(m+2)\,\mu$. This is a contradiction with Eq. (1). $\square$

Overall approximation ratio: Let τ_5 be the objective value after reassigning the unlucky jobs. Then:

$$\tau_5 \leqslant \mu \prod_{i=1}^{5} (1+\varepsilon_i) \leqslant (1+\varepsilon)\,(1+\varepsilon_6)^{-1}\,\mu. \tag{8}$$

Complexity: From Lemma 1, the number of assignments φ of large jobs that have to be examined is $\mathrm{O}(E^{m+2})$ for $E = \log(1/\varepsilon)/\log(1+\varepsilon)+1$. The assignments can be generated in constant $(m\log(m/\rho)\varepsilon^{-1})^{\mathrm{O}(m)}$ time. From Theorem 2, we know that for each φ, the corresponding LP can be solved in linear $\mathrm{O}(nM^2\,\ln M)$ time, where $M = (m+2)\varepsilon_3^{-1}$. Rounding and corrections can be done in linear $\mathrm{O}(mn)$ time. Therefore, the overall complexity of A-M2SCH is:

$$(m\log(m/\rho)\varepsilon^{-1})^{\mathrm{O}(m)} + n\mathrm{O}(E^{m+2}\,M\ln(M)), \quad \text{for } M = (m+2)\varepsilon_3^{-1}. \tag{9}$$

Proposition 9. *Algorithm A-M2SCH is a linear time RRDP for M2SCH.*

2.1. Derandomization

We show that algorithm A-M2SCH can be derandomized by applying the method of conditional probabilities with the use of a pessimistic estimator. We first give a brief description of the general method and then we present a specific pessimistic estimator for M2SCH. For a detailed description of the method of conditional probabilities and the use of pessimistic estimators the reader is referred to [23,22].

In algorithm A-M2SCH the probability that the randomly rounded solution violates the approximation ratio is less than a given constant ρ, and therefore less than 1. This fact is an existence proof for at least one rounded solution that satisfies the required approximation ratio. The method of conditional probabilities mimics this existence proof to actually build such a rounded solution in a deterministic manner. More precisely, given a fractional solution for M2SCH, the method assigns all jobs one by one to the machines. Each job assignment is done in such a way that, after the assignment, the probability of failure for the remaining jobs, if they were assigned with XRR, 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 is a rounded solution that satisfies the approximation guarantee.

A pessimistic estimator: Instead of calculating the exact probability of failure at each step of the method of conditional probabilities, it is sufficient to show that this probability is always strictly less than 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 that is visited during the rounding procedure, is called a *pessimistic estimator*. 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 an appropriate pessimistic estimator.

Let $[x_{ij}]$ be a feasible fractional solution for an instance of A-M2SCH (for simplicity, we ignore the fact that, at this step in algorithm A-M2SCH, a constant number of jobs, the large jobs, are already assigned):

$$\begin{aligned}
&\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^{a} x_{ij} \leqslant \tau \quad (a = 1, 2).\\
&\sum_{j=1}^{n} p_{ij} x_{ij} \leqslant \tau \quad (i = 1, \ldots, m),\\
&\sum_{i=1}^{m} x_{ij} = 1 \quad (j = 1, \ldots, n),\\
&x_{ij} \geqslant 0 \quad (i = 1, \ldots, m;\; j = 1, \ldots, n).
\end{aligned}$$

Let $[X_{ij}]$ be the randomly rounded solution obtained from $[x_{ij}]$, where the variables X_{ij} are Bernoulli trials. In the rounded solution, let Ψ_i be the load of machine i, and let Ψ_C^a be the cost for cost constraint a. Now, the probability that the makespan and the costs of the rounded schedule do not exceed $(1+\varepsilon)\tau$ can be bounded with the following pessimistic estimator:

$$U = P\left[\bigvee_{a=1}^{2}\left(\Psi_C^a > (1+\varepsilon)\tau\right) \vee \bigvee_{i=1}^{m}\left(\Psi_i > (1+\varepsilon)\tau\right)\right] \leqslant \sum_{a=1}^{2} \mathrm{P}[\Psi_C^a > (1+\varepsilon)\tau] + \sum_{i=1}^{m} \mathrm{P}[\Psi_i > (1+\varepsilon)\tau]. \quad (10)$$

We first examine the machine load constraints. We will use Eq. (12), an intermediate result of the proof of Chernoff bounds as it is given in [23, Theorem 1]. For completeness we include the proof of Eq. (12). For any machine i and any positive real t:

$$P[\Psi_i > (1+\varepsilon)\tau] = P\left[\mathrm{e}^{t\Psi_i} > \mathrm{e}^{t(1+\varepsilon)\tau}\right] < \mathrm{e}^{-t(1+\varepsilon)\tau} E\left[\mathrm{e}^{t\Psi_i}\right], \quad (11)$$

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

$$P[\Psi_i > (1+\varepsilon)\tau] < \mathrm{e}^{-t(1+\varepsilon)\tau} \prod_{j=1}^{n} (p_j \mathrm{e}^{t\alpha_j} + 1 - p_j). \quad (12)$$

Using Eq. (12) gives for each $i = 1, \ldots, m$ and for any positive real t

$$P_i = P[\Psi_i > (1+\varepsilon)\tau] < \mathrm{e}^{-t(1+\varepsilon)\tau} \prod_{j=1}^{n} (x_{ij} \mathrm{e}^{tp_{ij}} + 1 - x_{ij}) \quad (13)$$

and this gives, for example, for machine $i = 1$

$$P_1 < \mathrm{e}^{-t(1+\varepsilon)\tau} \prod_{j=1}^{n} (x_{1j} \mathrm{e}^{tp_{1j}} + 1 - x_{1j}). \quad (14)$$

When the procedure examines the first job $j = 1$

$$P_1 < (x_{11}\mathrm{e}^{tp_{i1}} + x_{21} + \cdots + x_{m1})\mathrm{e}^{-t(1+\varepsilon)\tau} \prod_{j=2}^{n} (x_{1j}\mathrm{e}^{tp_{1j}} + 1 - x_{1j}). \tag{15}$$

Similarly, we process the cost constraints. Using the arguments of Proposition 6 to obliterate the dependency issue of the binary random variables x_{ij} we get for any cost constraint a and any positive real t:

$$P_C^a = P[\Psi_C^a > (1+\varepsilon)\tau] = P\left[\mathrm{e}^{t\Psi_C^a} > \mathrm{e}^{t(1+\varepsilon)\tau}\right] < \mathrm{e}^{-t(1+\varepsilon)\tau} E\left[\mathrm{e}^{t\Psi_C^a}\right].$$

Since the variables $X_j^a = \sum_{i=1}^m c_{ij} X_{ij}$ are independent

$$P_C^a = P[\Psi_C^a > (1+\varepsilon)\tau] < \mathrm{e}^{-t(1+\varepsilon)\tau} \prod_{j=1}^{n} \left(\sum_{i=1}^{m} x_{ij}\mathrm{e}^{tc_{ij}^a} \right). \tag{16}$$

When focusing on the first cost constraint and on job $j = 1$ this gives

$$P_C^1 < (x_{11}\mathrm{e}^{tc_{11}^1} + \cdots + x_{m1}\mathrm{e}^{tc_{m1}^1})\mathrm{e}^{-t(1+\varepsilon)\tau} \prod_{j=2}^{n} \left(\sum_{i=1}^{m} x_{ij}\mathrm{e}^{tc_{ij}^1} \right). \tag{17}$$

Hence at the start of the derandomized rounding procedure

$$U = U(\mathit{root}) = x_{11}B_1 + x_{21}B_2 + \cdots + x_{m1}B_m, \tag{18}$$

where B_i, for $i = 1, \ldots, m$, are numbers independent from all variables x_{i1}, for $i = 1, \ldots, m$. Since $x_{11} + x_{21} + \cdots + x_{m1} = 1$, the pessimistic estimator U(root) is a convex combination of the numbers B_i. Consequently, the minimum of the numbers B_i must satisfy $\min_i(B_i) \leqslant U(\mathit{root})$. We assign job $j = 1$ to a machine m_j such that $m_j = \mathrm{argmin}_i\{B_i\}$.

The rounding procedure continues in the same way, by assigning one by one all unassigned 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.

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:

- $\prod_{j=l}^{n} \left(x_{ij}\mathrm{e}^{tp_{ij}} + 1 - x_{ij} \right)$ for $i = 1, \ldots, m$ and $l = 1, \ldots, n$, and
- $\prod_{j=l}^{n} \left(\sum_{i=1}^{m} x_{ij}\mathrm{e}^{tc_{ij}^a} \right)$ for $l = 1, \ldots, n$ and $a = 1, 2$.

We can for example calculate and store the values of all the above products. This requires the storage of a linear number $\mathrm{O}(mn + kmn)$ of results, where k is the number of cost constraints. 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 M2SCH (and all other algorithms presented in this work), rounding with a pessimistic estimator costs $\mathrm{O}(mn + kmn)$ time. This time is clearly dominated in algorithm A-M2SCH by the time to calculate the appropriate fractional solution (Steps 0–4). Thus, the overall complexity of the derandomized algorithm is equivalent to the complexity of the corresponding randomized algorithm.

Proposition 10. *The derandomization of algorithm A-M2SCH yields algorithm A_{det}-M2SCH, a deterministic RDP for M2SCH with the same asymptotic complexity as A-M2SCH.*

2.2. Optimization versions

The RDPs for MkSCH problems can be used to build FPTAS for optimization versions of MkSCH. A common type of MkSCH optimization problem is to minimize the makespan or one of the cost constraints for given bounds on the other constraints. Another option is to minimize a linear combination of makespan and costs [26,17]. A recent approach is to study the tradeoff between the different problem constraints [2,3]. We focus on problem M2SCH and discuss the case of minimizing makespan for given bounds on the two cost constraints. We will call this problem M2SCH-T. It is straightforward to adapt this discussion for other cases of MkSCH optimization problems. We will use the following property of combinatorial optimization problems:

Definition 11. 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 12. *M2SCH-T has the poly-bottleneck property.*

Proof. Let $p_{\max}$ be the maximum processing time that appears in an optimal solution to M2SCH-T. Then the makespan of the optimal schedule is at least $p_{\max}$ and at most $n\, p_{\max}$. There are at most $m\, n$ different possibilities for $p_{\max}$, since $p_{\max}$ has to be one of the specified p_{ij}. □

Given a RDP decision procedure for a problem having the poly-bottleneck property, a two-phase binary search procedure can efficiently solve corresponding optimization problems. In the first phase, a weight of the problem instance that dominates the objective value is identified, and this weight is then used in the second phase to approximate the objective value. Assume an instance of problem M2SCH-T and two given values $C1$ and $C2$ for the two cost constraints. Then:

Binary Search Procedure (BS) for M2SCH-T

Weights: Let W be the set of numbers that contains all distinct values p_{ij} of the instance description. The cardinality of W is $w = |W| \leqslant mn$. Sort the items of W in $O(mn \log(mn))$ time and let the sorted list be: $w_1 < w_2 < \cdots < w_w$.

Phase 1: *Indexed binary search.*

Find the minimum index $x : 1 \leqslant x \leqslant w$ such that $(T = nw_x, C1, C2)$ is feasible for M2SCH-T. This requires at most $\lceil log(mn) \rceil$ search steps.

Phase 2: *Standard binary search.*

Find the minimum value $t : w_x \leqslant t < n\, w_x$ such that $(T = t, C1, C2)$ is feasible and $t \leqslant OPT(1 + \varepsilon_6)$. This requires at most $\lceil log(mn/\varepsilon_6) \rceil$ search steps.

The binary search needs $O(\log(m\varepsilon_6^{-1} n))$ binary search steps and at each step it calls the RDP algorithm A_{det}-M2SCH. Hence:

Corollary 13. *Applying algorithm* A_{det}-M2SCH $O(\log(m\varepsilon_6^{-1} n))$ *times within an appropriate binary search procedure gives a quasi-linear time FPTAS for M2SCH-T.*

2.3. Problems SCHED, GAP and MkSCH

Algorithm A-M2SCH can be adapted for problems SCHED, GAP and generally for any MkSCH problem. We provide information for these adaptations and compare the results with results of [17]. Let $E = O(\log(1/\varepsilon) / \log(1+\varepsilon) + 1)$.

Problem SCHED: For SCHED, the appropriate parameters are $d_j = \min_i p_{ij}$, $\mu = 3\ln(m/\rho)\varepsilon_4^{-2}$, and the number of large jobs is $q = \lceil m^2\, \varepsilon_5^{-1}\, \mu \rceil$. The complexity of the RDP A-SCHED is $n \cdot O(E^m\, M\, \ln(M))$ for $M = m\varepsilon^{-1}$. The optimization version of SCHED appears to be easier than the optimization versions of other MkSCH problems, because the optimal makespan OPT can be bounded within the constant factor m. More precisely, it is easy to show that for $D = \sum_j d_j$, the optimal objective value OPT satisfies: $D/m \leqslant OPT \leqslant D$. Hence a binary search procedure using A-SCHED can find an approximate solution after a constant number $O(\log(m/\varepsilon_6))$ of binary search steps.

Problem GAP: The parameters are $d_j = \min_i\{p_{ij} + c_{ij}\}$, $\mu = 3\ln((m+1)/\rho)\varepsilon_4^{-2}$ and $q = \lceil 2\,(m+1)^2\, \varepsilon_5^{-1}\, \mu \rceil$. The complexity of the RDP A-GAP is $n \cdot O(E^{m+1}\, M\, \ln(M))$, for $M = (m+1)\varepsilon^{-1}$. Optimization versions can be handled similarly to optimization versions of problem M2SCH.

Problem MkSCH: Let k be the number of cost constraints. Then the parameters are $d_j = \min_i\{p_{ij} + \sum_a c_{ij}^a\}$, $\mu = 3\ln((m+k)/\rho)\varepsilon_4^{-2}$ and $q = \lceil (k+1)\,(m+1)^2 \varepsilon_5^{-1}\, \mu \rceil$. The complexity of the RDP A-MkSCH is $n \cdot O(E^{m+k}\, M\, \ln(M))$, for $M = (m+k)\varepsilon^{-1}$. Optimization versions can be handled similarly to optimization versions of problem M2SCH.

[2] The term "bottleneck" has been used by Hochbaum and Schmoys in [15] 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.

Discussion: The CRR approach provides uniform FPTAS's for the whole class of MkSCH problems. Fig. 1 shows that the asymptotic complexity of SCHED and GAP is better than the corresponding results of [17], because the CRR-based algorithms have to examine a smaller number of large job assignments. This holds, even though for problem SCHED, the CRR-approach has to enumerate a greater number of large jobs than the corresponding algorithm in [17]. However, this superiority of the CRR approach is achieved with technique $\mathcal{T}_2$ which can also be applied to the results of [17]. The most important advantage of the CRR approach is the rounding step, which

- is *simple*, since it is a standard randomized rounding procedure or the corresponding pessimistic estimator method, and which
- is *general*, since it applies to the whole class of MkSCH problems.

Hence, the rounding step is simpler and more general then the rounding steps of the algorithms in [17], and this is especially evident for the involved linear time rounding scheme for GAP in [17].

3. The covering (COV) problem

Given an instance of COV and a value *T* for the minimum makespan, the IP formulation of the decision version of COV is:

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

We first present algorithm A-COV, a RRDP for COV, and then show how to build a FPTAS for COV. Algorithm A-COV accepts an instance of COV, a value *T* and constants $\varepsilon : 0 < \varepsilon < 1$ and $\rho :\ 0 < \rho \leqslant 1$, and if the value *T* is feasible, then with probability of success strictly larger than $(1 - \rho)$, it generates a schedule of minimum machine load at least $(1 - \varepsilon)T$.

Algorithm A-COV

Input: An instance of COV and values ε, ρ and T.

Step 0: *Normalization.* For $i = 1, \ldots, 6$, let $\varepsilon_i = \varepsilon/9 = \Theta(\varepsilon)$. Let $\mu = 2\ln(m/\rho)\varepsilon_4^{-2}$ and scale the problem with the factor μ/T.

Step 1: *Initializations.* Let $q = \lceil 2m^2\varepsilon_5^{-1}\mu \rceil$.

Step 2: *Initial filtering.* For each $p_{ij} > \mu$ set $p_{ij} = \mu$.

Step 3: *Large coefficients and large jobs.* Define an appropriate set J_ℓ^I of large jobs and let Φ_2 be an appropriate set of assignments of the large jobs.

Step 4: *Feasible fractional assignment.*

$\forall$ assignment $\varphi \in \Phi_2$:

(1) Formulate the corresponding covering problem as IP-COV(φ).

(2) Relax IP-COV(φ) to a linear program LP-COV(φ).

(3) Find an approximate fractional schedule with algorithm A-Young.

(4) If the fractional schedule is feasible, then select it and abandon the loop.

Let $[x_{ij}]$ be the feasible fractional solution that has been selected.

Step 5: *Combinatorial randomized rounding.*

(1) Round $[x_{ij}]$ with XRR.

(2) With probability strictly larger than $1 - \rho$, no constraint of type I is violated by a factor less than $(1 - \varepsilon_4)$.

(3) Correct any constraint of type II that is violated, by moving at most μ large jobs to the corresponding machine.

Analysis of algorithm A-COV: For $\mu = 2\ln(m/\rho){\varepsilon_4}^{-2}$, the problem is normalized by multiplying it with the factor μ/T. The next step is to handle the very large coefficients $p_{ij} > \mu$. While in packing problems these coefficients are wiped out, in covering problems we set them equal to $p_{ij} = \mu$. This has no impact on the feasibility of the covering problem for objective value μ. After this step, all coefficients $p_{ij} > 1$ are considered large. Let $q = \lceil 2m^2\varepsilon_5^{-1}\mu \rceil$, let *J* be the set of all jobs, and for each constraint *i*, let J_ℓ^i be the set of jobs that correspond to the large coefficients $p_{ij} > 1$ of the constraint. We distinguish *two types* of constraints:

- *Type* I: Constraints that have strictly less than *q* large coefficients.

• *Type* II: Constraints with at least q large coefficients.

Step 3: Let J_ℓ^I be the union of all sets J_ℓ^i that correspond to constraints of type I. Hence J_ℓ^I is the set of all jobs that correspond to large coefficients in constraints of type I. The cardinality of J_ℓ^I is $|J_\ell^I| \leqslant m\,q$, a constant. Let Φ be the set of all possible assignments φ of the jobs in J_ℓ^I. Like in algorithm A-M2SCH, we examine only the assignments $\varphi \in \Phi_2$, a subset of Φ. The set Φ_2 can be generated with adapted versions of techniques $\mathcal{T}_1$ and $\mathcal{T}_2$ of Section 2. Using Φ_2 instead of Φ_1 introduces at most a factor $(1-\varepsilon_1)(1-\varepsilon_2)$ to the objective function.

Step 4: Let λ_i be the load on machine i due to assignment $\varphi \in \Phi_2$. Given any φ, the problem of assigning the remaining jobs is formulated as an integer program IP-COV(φ):

$$\text{Find } x, \text{ s.t.:} \begin{cases} \sum_{j \in J \setminus J_\ell^I} p_{ij} x_{ij} \geqslant \mu - \lambda_i & (i = 1, \dots, m), \\ \sum_{i=1}^m x_{ij} = 1 & (j \in J \setminus J_\ell^I), \\ x_{ij} \in \{0, 1\} & (i = 1, \dots, m;\ j \in J \setminus J_\ell^I). \end{cases}$$

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(φ):

$$\text{Find } x, \text{ s.t.:} \begin{cases} \sum_{j \in J \setminus J_\ell^I} p_{ij} x_{ij} \geqslant \mu - \lambda_i & (i = 1, \dots, m), \\ 1 \leqslant \sum_{i=1}^m x_{ij} \leqslant 1 & (j \in J \setminus J_\ell^I), \\ x_{ij} \geqslant 0 & (i = 1, \dots, m;\ j \in J \setminus J_\ell^I). \end{cases}$$

The linear program LP-COV(φ) 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 [29] (we call it algorithm A-Young):

Corollary 14. *Follows from Corollary* 1 *of* [29]. *The linear program LP-COV*(φ) (*and LP-EXACT*(φ) *in algorithm A-EXACT of Section* 4) *can be approximated with algorithm A-Young within any constant ratio* ε *in* $O(\varepsilon^{-2}mn)$ *time*.

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

$$[\tilde{x}_{ij}] : \begin{cases} \sum_{j \in J \setminus J_\ell^I} p_{ij} x_{ij} \geqslant \mu - \lambda_i & (i = 1, \dots, m), \\ 1 \leqslant \sum_{i=1}^m x_{ij} \leqslant 1 + \varepsilon_3 & (j \in J \setminus J_\ell^I), \\ x_{ij} \geqslant 0 & (i = 1, \dots, m;\ j \in J \setminus J_\ell^I). \end{cases}$$

Algorithm A-COV starts to calculate fractional solutions for each assignment $\varphi \in \Phi_2$ 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. For each job j, the variables $[\tilde{x}_{ij}]$ are scaled appropriately so that $\sum_{i=1}^m x_{ij} = 1$. The outcome is a feasible fractional job assignment $[x_{ij}]$ of all jobs, large and not-large, that satisfies:

$$[x_{ij}] : \begin{cases} \sum_{j=1}^n p_{ij} x_{ij} \geqslant \mu (1 + \varepsilon_3)^{-1} & (i = 1, \dots, m), \\ \sum_{i=1}^m x_{ij} = 1 & (j = 1, \dots, n), \\ x_{ij} \in [0, 1] & (i = 1, \dots, m;\ j = 1, \dots, n). \end{cases}$$

Step 5: *Rounding with corrections*. The solution $[x_{ij}]$ is rounded with standard XRR of Definition 3 to an integer solution $[X_{ij}]$.

Type I: Applying the deviation bound of Proposition 4 for deviations below the mean value gives that, with probability strictly larger than $1 - \rho$, none of the constraints of type I in $[X_{ij}]$ is violated by a factor larger than $(1 - \varepsilon_4)$.

Type II: Given a rounded solution $[X_{ij}]$, let i be a constraint of type II that is not satisfied, i.e., $\sum_{j=1}^{n} p_{ij}x_{ij} < \mu$. Then we call machine i an "unlucky" machine. By definition a constraint of type II has at least q large coefficients. From all jobs $j \in J_\ell^i$ that are not assigned to i we select the jobs that have the smallest load in their current placement in the rounded solution. These jobs are sufficient to cover the violated constraint. The parameters μ and q are defined so that for every such correction of a constraint of type II, the possible cost to any other constraint, due to removing jobs from it, does not exceed $\varepsilon_5\, \mu\, /\, m$.

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

$\mathbf{Z}_\ell^{\mathbf{i}} \leqslant \varepsilon_5\, \mu\, /\, m \Rightarrow$ In this case we move the jobs to constraint i. The total cost for correcting all constraints of type II that are violated is $\leqslant \varepsilon_5\, \mu$.

$\mathbf{Z}_\ell^{\mathbf{i}} > \varepsilon_5\, \mu\, /\, m \Rightarrow$ In this case the total load due to the initial assignment of all jobs in J_ℓ^i is larger then $2m\mu$ and hence we can move a sufficient number of (at most μ) jobs to the violated constraint i without causing any violation to other constraints.

Hence, the cost for correcting all constraints of type II cannot not exceed $\varepsilon_5\mu$.

Approximation ratio: From the above discussion, if the input value T is feasible, the final objective value τ satisfies: $\tau \geqslant T\,(1-\varepsilon_1)\,(1-\varepsilon_2)\,(1+\varepsilon_3)^{-1}\,(1-\varepsilon_4)\,(1-\varepsilon_5)\,T \geqslant (1-\varepsilon)\,(1-\varepsilon_6)^{-1}\,T$.

Complexity: The number of assignments φ of large jobs that have to be examined is $\mathrm{O}(E^m)$, for $E = \mathrm{O}(\log(1/\varepsilon)/\log(1+\varepsilon)+1)$. The assignments can be generated in constant $\mathrm{O}((m\log(m/\rho)\varepsilon^{-1})^{\mathrm{O}(m)})$ time. For each assignment φ the corresponding LP can be approximated in $\mathrm{O}(\varepsilon_3^{-2}mn)$ time. Rounding and corrections can be done in linear $\mathrm{O}(mn)$ time. Hence the overall complexity of A-COV is: $\mathrm{O}((m\log(m/\rho)\varepsilon^{-1})^{\mathrm{O}(m)}) + n\mathrm{O}(E^m\,\varepsilon_3^{-2}m)$.

Proposition 15. *Algorithm A-COV is a linear time RRDP for COV.*

Similar to algorithm A-M2SCH in Section 2, algorithm A-COV can be derandomized with use of a pessimistic estimator to reveal a corresponding deterministic linear time algorithm A_{det}-COV. Furthermore, it is easy to show that COV has the poly-bottleneck property and hence algorithm A_{det}-COV can be used to build a quasi-linear time FPTAS for COV.

Corollary 16. *Applying algorithm A_{det}-COV* $\mathrm{O}(\log(m\varepsilon_6^{-1}n))$ *times within a binary search procedure gives a quasi-linear time FPTAS for COV.*

4. Problems EQUAL and EXACT

In this section we present approximation schemes for problems EQUAL and EXACT. First we present algorithm A-EXACT, a PTAS for problem EXACT, and then we use A-EXACT as a RDP to build a PTAS for EQUAL. The algorithms apply brute-force processing for enumerating possible assignments to large coefficients and hence their complexity is a high degree polynomial on n. Given an instance of EXACT and a vector $[t_i]$ for the machine loads, the IP formulation of EXACT is:

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

We consider the relaxed version of EXACT where the equality constraints can be violated by at most a given constant factor. More precisely, algorithm A-EXACT accepts as input an instance of EXACT, constants $\varepsilon : 0 < \varepsilon \leqslant 1$ and $\rho : 0 < \rho \leqslant 1$, and a vector $[t_i]$. If the values $[t_i]$ are feasible, then with probability of success larger than $(1-\rho)$, A-EXACT generates a schedule with load Ψ_i on machine i: $(1-\varepsilon/2)t_i \leqslant \Psi_i \leqslant (1+\varepsilon/2)t_i$, for $i = 1, \ldots, m$.

Algorithm A-EXACT

Input: An instance of EXACT, values ε, ρ and a vector $[t_i]$.

Step 0: *Normalization.* Let $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \varepsilon/8 = \Theta(\varepsilon)$. Let $\mu = 2\ln(m/\rho)\varepsilon_2^{-2}$ and scale each machine load constraint of the problem with the factor μ/t_i.

Step 1: *Filtering*. For each $p_{ij} > \mu$, set $x_{ij} = 0$.
Step 2: *Large coefficients*. Enumerate the feasible assignments (L^1, L^0) of jobs to large coefficients.
Step 3: *Feasible fractional schedule*. For every assignment (L^1, L^0):
(1) Formulate the integer program IP-EXACT(L^1, L^0) for assigning the remaining variables $x_{ij} : (i, j) \notin L$.
(2) Relax IP-EXACT(L^1, L^0) to the linear program LP-EXACT(L^1, L^0).
(3) Calculate an approximate fractional solution with algorithm A-Young.
(4) If the solution $[\tilde{x}_{ij}]$ is feasible, then select it and abandon the loop.
Step 4: *Scaling*. Scale the fractional solution $[\tilde{x}_{ij}]$, so that for each $j = 1, \ldots, n$, the corresponding constraint $\sum_{i=1}^{m} x_{ij} = 1$ is satisfied.
Step 5: *Randomized rounding*. Round $[x_{ij}]$ with XRR.

Analysis of algorithm A-EXACT: The problem is first normalized with the factors $\mu/t_i,\ i = 1, \ldots, m$, so that the objective is to find a schedule with load (approximately) μ on all machines. In the initial filtering step, all very large coefficients $p_{ij} > \mu$ are excluded from the solution by setting the corresponding x_{ij} equal to 0. This has obviously no impact on any solution of objective value μ.

Step 2: *Large coefficients* L_i. We focus on individual coefficients p_{ij} that are considered large, and enumerate the possible assignments of values to the variables x_{ij} that correspond to these coefficients. The goal is that the rounding step (Step 5 of A-EXACT) must concern only variables x_{ij} with coefficients $p_{ij} \leqslant 1$. For each machine i, let L_i be the set of all pairs (i, j) that correspond to large coefficients $p_{ij} : 1 < p_{ij} \leqslant \mu$. Assume an assignment of binary values $\{0, 1\}$ to the variables x_{ij} that correspond to large coefficients L_i, and let $L_i^1 \subseteq L_i$ 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 μ 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

$$\binom{n}{r} \leqslant \left(\frac{n^r}{r!}\right) = p_{i,r}(n) = \mathrm{O}(n^r) \tag{19}$$

sets $L_i^1 \subseteq L_i$, when the cardinality of L_i^1 is $|L_i^1| = r$ for $0 \leqslant r \leqslant \mu$. 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 larger than the number of all possible sets L_i^1 with cardinality $0 \leqslant |L_i^1| \leqslant \mu$:

$$\sum_{r=0}^{\mu} \binom{n}{r} \leqslant (\mu + 1) \left(\frac{n^r}{r!}\right) = p_i(n) = \mathrm{O}(\mu n^{\mu}). \tag{20}$$

The total number $p(n)$, of all feasible assignments of jobs to large coefficients for all machines, can be bounded with a high degree polynomial on n:

$$p(n) = \prod_{i=1}^{m} p_i(n) = \mathrm{O}(\mu^m n^{m\mu}). \tag{21}$$

Step 3: *Fractional solution*: For every possible assignment (L^1, L^0) 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 IP-EXACT(L^1, L^0):

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

The integer program IP-EXACT(L^1, L^0) is relaxed to a mixed packing and covering program LP-EXACT(L^1, L^0). Like in Section 3, an approximately feasible fractional solution is found for the linear program LP-EXACT(L^1, L^0)

with algorithm A-Young of [29]. Let λ_i be the load on machine i due to the assignment (L^1, L^0). If the value μ is feasible for the problem, then for at least one of the assignments (L^1, L^0) the corresponding fractional solution $[\tilde{x}_{ij}]$ satisfies:

$$[\tilde{x}_{ij}] : \begin{cases} \mu \leqslant \lambda_i + \sum_{j=1}^{n} p_{ij}\tilde{x}_{ij} \leqslant (1+\varepsilon_1)\mu & (i = 1, \ldots, m), \\ 1 \leqslant \sum_{i=1}^{m} \tilde{x}_{ij} \leqslant (1+\varepsilon_1) & (j = 1, \ldots, n), \\ \tilde{x}_{ij} \in [0, 1] & (i = 1, \ldots, m;\ j = 1, \ldots, n). \end{cases}$$

Algorithm A-EXACT 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.

Step 4: *Scaling*. The solution $[\tilde{x}_{ij}]$ is scaled so that all jobs are completely assigned. This introduces at most a factor $(1+\varepsilon_1)^{-1}$ to the approximation ratio. Let $[x_{ij}]$ be the scaled fractional solution. Then:

$$[x_{ij}] : \begin{cases} (1+\varepsilon_1)^{-1}\mu \leqslant t_i + \sum_{j=1}^{n} p_{ij}x_{ij} \leqslant (1+\varepsilon_1)\mu & (i = 1, \ldots, m), \\ \sum_{i=1}^{m} x_{ij} = 1 & (j = 1, \ldots, n), \\ x_{ij} \in [0, 1] & (i = 1, \ldots, m;\ j = 1, \ldots, n). \end{cases}$$

Step 5: *Rounding*. The fractional solution $[x_{ij}]$ is rounded with XRR to an integer schedule $[X_{ij}]$. Since the rounding concerns only small coefficients $p_{ij} \leqslant 1$, applying the deviation bounds of Proposition 4 gives that with probability strictly larger than $1-\rho$, XRR does not introduce deviations by factors larger than $(1-\varepsilon_2)$ and $(1+\varepsilon_2)$, below and above the fractional load, respectively. Consequently, if μ is feasible, then, with probability strictly larger than $1-\rho$, the final solution $[X_{ij}]$ satisfies:

$$[X_{ij}] : \begin{cases} \sum_{j=1}^{n} p_{ij}X_{ij} \leqslant (1+\varepsilon_1)(1+\varepsilon_2)\mu & (i = 1, \ldots, m), \\ \sum_{j=1}^{n} p_{ij}X_{ij} \geqslant (1+\varepsilon_1)^{-1}(1-\varepsilon_2)\mu & (i = 1, \ldots, m), \\ \sum_{i=1}^{m} X_{ij} = 1 & (j = 1, \ldots, n), \\ X_{ij} \in \{0, 1\} & (i = 1, \ldots, m;\ j = 1, \ldots, n). \end{cases}$$

Complexity: Algorithm A-EXACT examines $O(\mu^m n^{m\mu})$ assignments. The time to process each assignment is dominated by algorithm A-Young for the LP, which is $O(m\varepsilon_1^{-2}n)$. The overall complexity is $O(m\,\mu^m\,\varepsilon^{-2}\,n^{m\mu+1})$.

Proposition 17. *Algorithm A-EXACT is a RRDP for EXACT.*

Algorithm A-EXACT can be derandomized with the use of a pessimistic estimator to produce a corresponding deterministic polynomial time algorithm A_{det}-EXACT. The derandomization is similar to the approach used in Section 2.1 for algorithm A-M2SCH.

Problem EQUAL: It is easy to show that problem EQUAL has the poly-bottleneck property. Using this fact it should be possible to apply a two-phase binary search procedure like the procedure in Section 2.2 to find an approximately feasible solution with approximately minimum makespan. However, there is an important issue that needs to be addressed: the range of objective values for which (the decision version of) EQUAL is feasible, might not be continuous or it might even be empty.

Feasibility: Assume an instance of EQUAL and a specific objective value T that is examined during binary search. If the objective value T is found to be not feasible, then this fact alone is not sufficient for the binary search procedure to decide that the optimal objective value must be larger than T. We overcome this issue by adding a positive slack variable s to all machine load constraints in the relaxed LP formulation of the decision version of EQUAL. The modified linear program LPs-EQUAL is:

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

If there is an optimal objective value for the instance of EQUAL, let *OPT* be this value. The binary search procedure searches for the smallest feasible makespan value *T*. The introduction of the slack parameter *s* to the LP formulation causes any input value *T*, such that $T \geqslant OPT$, to be feasible for the above LP. More precisely, assume an objective value $T < OPT$. Even if *T* is not feasible for the decision version of EQUAL, the linear program LPs-EQUAL is feasible, since the positive slack variable *s* can take the value $s = OPT - T$. Consequently, the binary search procedure can approximately converge to the optimal objective value. During binary search, the RDP procedure A_{det}-EXACT is executed $O(\log(mn/\varepsilon_3))$ times. If after $O(\log(mn/\varepsilon_3))$ steps the binary procedure finds no feasible value *T* for problem EQUAL, or if LPs-EQUAL is infeasible for a large enough value $T = n \max_{i,j} p_{ij}$, then the specific instance of EQUAL is infeasible.

Corollary 18. *Applying algorithm A_{det}-EXACT within an appropriate binary search procedure gives a PTAS for EQUAL.*

5. Discussion

We present a 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 carefully guided randomized rounding procedures. The CRR approach does not depend on any particular problem structure and can find further applications in:

- Problems with a constant number of constraints, where it is tractable to enumerate possible value assignments to variables with large coefficients.
- Problems that admit fractional values for the variables with the largest coefficients. In this case, we can omit the costly enumeration step and apply the remaining part of the rounding procedures. If, for example, it would be acceptable for the largest jobs in MkSCH to be fractionally assigned in the final solution, then a CRR-based algorithm could handle efficiently MkSCH for any, not necessarily fixed, number of machines *m*.

We would also like to note, that the randomized algorithms of this work can be efficiently implemented in parallel and distributed settings.

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

Acknowledgements

The authors wish to thank the anonymous referees for valuable comments that improved substantially the presentation of the paper.

References

[1] F.N. Afrati, E. Bampis, C. Chekuri, D.R. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Sartinutella, C. Stein, M. Sviridenko, Approximation schemes for minimizing average weighted completion time with release dates, in: IEEE FOCS, 1999, pp. 32–44.

[2] E. Angel, E. Bampis, A. Kononov, A fptas for approximating the unrelated parallel machines scheduling problem with costs, in: ESA'01, Lecture Notes in Computer Science, Vol. 2161, 2001, pp. 194–205.

[3] Preliminary results of this work were given in [7].

[3] E. Angel, E. Bampis, A. Kononov, On the approximate tradeoff for bicriteria batching and parallel machine scheduling problems, Theoret. Comput. Sci. 306 (2003) 319–338.
[4] Y. Azar, L. Epstein, Approximation schemes for covering and scheduling on related machines, in: Proc. of First APPROX, 1998, pp. 39–47.
[5] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.
[6] J. Csirik, H. Kellerer, G. Woeginger, The exact lpt-bound for maximizing the minimum completion time, Oper. Res. Lett. 11 (1992) 281–287.
[7] P.S. Efraimidis, P.G. Spirakis, Randomized approximation schemes for scheduling unrelated parallel machines, Electron. Colloq. on Computational Complexity (ECCC) (007), 2000.
[8] D.K. Friesen, B. L Deuermeyer, Analysis of greedy solutions for a replacement part sequencing problem, Math. Oper. Res. 6 (1981) 74–87.
[9] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, 1979.
[10] M. Grigoriadis, L. Khachiyan, Coordination complexity of parallel price-directive decomposition, Math. Oper. Res. 21 (1996) 317–327.
[11] M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, B. Reed (Eds.), Probabilistic Methods for Algorithmic Discrete Mathematics, Springer, Berlin, 1998.
[12] L. Hall, Approximation algorithms for scheduling, in: D. Hochbaum (Ed.), Approximation Algorithm for NP-hard Problems, PWS Publishing Company, 1997(Chapter 1).
[13] Y. He, Y. Jiang, Optimal semi-online preemptive algorithms for machine covering on two uniform machines, Theoret. Comput. Sci. 339 (2005) 293–314.
[14] Y. He, Z. Tan, Ordinal on-line scheduling for maximizing the minimum machine completion time, J. Combin. Optim. 6 (2) (2002) 199–206.
[15] D. Hochbaum, D.B. Shmoys, A unified approach to approximation algorithms for bottleneck problems, J. ACM 3 (33) (1986) 533–550.
[16] E. Horowitz, S. Sahni, Exact and approximate algorithms for scheduling nonidentical processors, J. ACM 23 (1976) 317–327.
[17] K. Jansen, L. Porkolab, Improved approximation schemes for scheduling unrelated parallel machines, in: ACM STOC, 1999, pp. 408–417.
[18] K. Jansen, M.I. Sviridenko, Polynomial time approximation schemes for the multiprocessor open and flow shop scheduling problem, Lecture Notes in Computer Science, Vol. 1770, Springer, Berlin, 2000, pp. 455–465.
[19] J.K. Lenstra, D.B. Shmoys, É. Tardos, Approximation algorithms for scheduling unrelated parallel machines, Math. Programming 46 (1990) 259–271.
[20] J. Leung (Ed.), Handbook of Scheduling: Algorithms, Models and Performance Analysis, Chapman & Hall/CRC, Boca Raton, FL, London, 2004.
[21] M. Mitzenmacher, A. Richa, R. Sitaraman, The power of two random choices: a survey of techniques and results, in: P.M. Pardalos, S. Rajasekaran, J. Reif, J.D.P. Rolim (Eds.), Handbook of Randomized Algorithms, Kluwer Academic Publishers, Dordrecht, 2001.
[22] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, Cambridge, 1995.
[23] P. Raghavan, Probabilistic construction of deterministic algorithms: approximating packing integer programs, J. Comput. System Sci. 37 (1988) 130–143.
[24] P. Schuurman, G.J. Woeginger, Approximation schemes—a tutorial, in: R.H. Moehring, C.N. Potts, A.S. Schulz, G.J. Woeginger, L.A. Wolsey (Eds.), Lectures on Scheduling, to appear, http://www.win.tue.nl/∼egwoegi/papers/ptas.ps.
[25] P. Schuurman, G.J. Woeginger, Polynomial time approximation algorithms for machine scheduling: ten open problems, J. Scheduling 2 (1999) 203–213.
[26] D. Shmoys, É. Tardos, An approximation algorithm for the generalized assignment problem, Math. Programming A62 (1993) 461–474.
[27] B. Vöcking, How asymmetry helps load balancing, J. ACM 50 (4) (2003) 568–589.
[28] G.J. Woeginger, A polynomial-time approximation scheme for maximizing the minimum machine completion time, Oper. Res. Lett. 20 (4) (1997) 149–154.
[29] N.E. Young, Sequential and parallel algorithms for mixed packing and covering, in: IEEE FOCS, 2001, pp. 538–546.