J Sched (2017) 20:545–555
DOI 10.1007/s10951-016-0484-y

Minimizing worst-case and average-case makespan over scenarios

Esteban Feuerstein[1] · **Alberto Marchetti-Spaccamela**[2] · **Frans Schalekamp**[3] · **René Sitters**[4] · **Suzanne van der Ster**[5] · **Leen Stougie**[4] · **Anke van Zuylen**[6]

Published online: 17 June 2016

Abstract We consider scheduling problems over scenarios where the goal is to find a single assignment of the jobs to the machines which performs well over all scenarios in an explicitly given set. Each scenario is a subset of jobs that must be executed in that scenario. The two objectives that we consider are minimizing the maximum makespan over all scenarios and minimizing the sum of the makespans of all scenarios. For both versions, we give several approximation algorithms and lower bounds on their approximability. We also consider some (easier) special cases. Combinatorial optimization problems under scenarios in general, and scheduling problems under scenarios in particular, have seen only limited research attention so far. With this paper, we make a step in this interesting research direction.

Keywords Job scheduling · Approximation algorithm · Makespan · Scenarios

1 Introduction

We introduce a new setting of scheduling over *scenarios*, where the goal is to find one solution for the scheduling problem that performs well for each scenario in a predefined set. In particular, we are given a set J of jobs where job j has processing time p_j, and a set of k scenarios $\mathcal{S} = \{S_1, \ldots, S_k\}$, where each scenario is specified as a subset of jobs in J that must be executed if that scenario occurs. The goal is to find an assignment of jobs to machines that is the same *for all scenarios* and optimizes a function of the scheduling objective *over all scenarios*.

We focus on the problem of minimizing the makespan of a schedule. The two objectives that we consider are minimizing the *maximum* makespan over all scenarios and minimizing the *sum* of makespans of all scenarios, or, equivalently, the *average* makespan over all scenarios.

The more *egalitarian* objective of minimizing the maximum scheduling objective is a special case of *robust* optimization (see for instance Ben-Tal and Nemirovski 2002). The premise of robust optimization is that not all parameters of a problem instance are fully known, and the goal is to find the best solution with respect to the worst-case realization of the uncertain parameters. This uncertainty is often parameterized using ranges of possible values for the input parameters, and a common complication is finding a solution that is feasible with respect to all possible outcomes of the uncertain parameters.

In our particular setting, we are hedging against a *finite set of scenarios*, and feasibility is not an issue: any solution that assigns every job to exactly one machine is feasible for every scenario. In the field of scheduling, examples of robust optimization are mainly found in applied literature, for example, in the area of project scheduling, where many uncertain parameters exist. A small change in realizations of these parameters can render a nominal solution completely infeasible, so robustness is an essential requirement (see for example Lin et al. 2004; Wang and Chan 2015).

The more *utilitarian* objective of minimizing the sum of objective functions is a special case of *a priori* optimization (introduced by Jaillet 1985 and Jaillet 1988), which is

✉ Suzanne van der Ster
ster@in.tum.de

1 Universidad de Buenos Aires, Buenos Aires, Argentina

2 Sapienza Università di Roma, Rome, Italy

3 Cornell University, Ithaca, NY, USA

4 Vrije Universiteit & CWI, Amsterdam, The Netherlands

5 Technische Universität München, Munich, Germany

6 College of William and Mary, Williamsburg, VA, USA

a special case of *stochastic* optimization (see for instance the textbook Birge and Louveaux 2011). In stochastic optimization, part of the problem is assumed to be coming from a random process, the distribution of which is generally assumed to be known completely, but in some cases is assumed to be only accessible by "black-box" sampling. After the realization of the random process is observed, further decisions (so called "recourse" decisions) can be made. The goal is to optimize the expected objective value. The premise of a priori optimization is that there are no recourse decisions.

The setting in this paper of minimizing the sum of objective functions can be seen as a stochastic optimization problem with a uniform distribution over a finite set of fully specified scenarios, where reoptimization (recourse) after observing the scenario is not allowed. An example of finite sets of scenarios in stochastic optimization (programming) is the sampling-based approximation methods in Kleywegt et al. (2002), where the assumption is that these scenarios can be accessed by black-box sampling. Scheduling in the stochastic optimization framework is studied in (Pinedo 2012, Appendix F), where, in contrast to the setting of this paper, the *processing times* are assumed to be stochastic variables.

In this paper, we mainly restrict ourselves to two machines. We will denote the problem of minimizing the maximum makespan over scenarios on two machines by MM2 and the problem of minimizing the sum of makespans over scenarios on two machines by SM2.

To see that even for very small problems the optimal solution for the MM2 problem differs from the optimal solution for the SM2 problem, consider the following example:

Example 1 Let set J contain three jobs, numbered 1, 2 and 3, that are executed on two machines. The processing times are $p_1 = 2$, $p_2 = p_3 = 1$ and the given scenarios are $S_1 = \{1, 2, 3\}$, $S_2 = S_3 = \{2, 3\}$.

For the MM2 problem, it is optimal to assign job 1 to the first machine and jobs 2 and 3 to the second machine. This yields a makespan of 2 in every scenario, and it is clear that anything better is impossible.

However, for the SM2 problem, an optimal solution is to assign jobs 1 and 2 to the first machine and job 3 to the second machine. In scenario S_1, this yields a makespan of 3, but in scenarios S_2 and S_3, the makespan equals 1. Summing these makespans yields an objective of 5, whereas the previously given solution would yield an objective of 6.

Motivation From a practical point of view, the type of problems we study in this paper emerges in situations where some training or investment is needed to obtain skills that are needed before machines (or workers) can perform particular jobs. In such situations, one should decide on an assignment of all possible jobs to the workers, such that the workers can train for the jobs assigned to them ahead of time. The goal is to assign specializations to machines such that the workload of the machines is optimized, either on average or in the worst case.

Suppose, for example, that the machines represent secretarial employees and each day a subset of tasks from a larger set of possible tasks needs execution and the specializations are partitioned over the employees. The objective of minimizing the maximum makespan would ascertain that every day all employees can go home before 5 PM, while the objective of minimizing the sum of makespans regulates the total hours in, for example, a month, that any employee will have to work (assuming that all scenarios occur with the same frequency).

Other examples of such settings are assignment of clients to lawyers, households to power sources, compile-time assignment of computational tasks to processors.

From a theoretical point of view, it is interesting to find out how the complexity of a problem changes by simply having several scenarios to take into account. Is the shift in complexity for solving a problem (exactly or approximately) dramatic? So, would computation times remain polynomial if the single-scenario version is polynomial-time solvable? And, in case of approximations, are problems raised to another approximation class or do they only have higher lower bounds on approximability while remaining in the same class?

Moreover, the problem of optimizing the sum of objective functions over a finite set of scenarios appears as the first-stage problem in a boosted sampling approach to two-stage stochastic optimization problems Gupta et al. (2011). This approach is used in a setting where uncertainty is defined as a black-box model, i.e., information about distributions can only be learnt by sampling from this black box. The boosted sampling approach then samples a finite set of scenarios from the black box and solves a deterministic problem, where a single solution should be found, that minimizes the sum of the objective values over the set of drawn scenarios. Using this approach, some results have been found for combinatorial optimization problems over scenarios, for problems like VERTEX COVER, STEINER TREE, and UNCAPACITATED FACILITY LOCATION Gupta et al. (2011).

Problem formulation Given is a job set J, together with a set of k scenarios $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$, where $S_i \subseteq J$ for all i. Note that we allow repetitions of the same subset in the set of scenarios. Unless specified otherwise, k is assumed to be an arbitrary number (specified by the input), not a constant. Each job j in J has processing time p_j.[1] For a set S, we

[1] An alternative way of viewing the scenarios would be that every scenario specifies a $|J|$-tuple of processing times, where the processing time of job j in any scenario S_i equals either p_j or 0.

denote the total processing time of jobs in that set by $p(S)$. We consider two machines and are interested in minimizing the *makespan* of a schedule. The following two variations of this scheduling problem over scenarios are considered:

MM2 Assign the jobs in J to two machines such that the maximum makespan over all scenarios is minimized. Formally, we are searching for a partition $A, \bar{A}$ of J that minimizes $\max_{i=1,\ldots,k} \max\{p(A \cap S_i), p(\bar{A} \cap S_i)\}$.

SM2 Assign the jobs in J to two machines in such a way that it minimizes the sum of makespans over all scenarios. Formally, we seek to partition J into $A, \bar{A}$ in order to minimize $\sum_{i=1}^{k} \max\{p(A \cap S_i), p(\bar{A} \cap S_i)\}$.

Related work To the best of the authors' knowledge, there is only very little work that considers scheduling over scenarios in the same manner as defined here. Kasperski et al. (2012) also consider the makespans of schedules, but define different objective functions over the scenarios. The objective they consider is called the *ordered weighted averaging aggregation operator* (OWA), which is a weighted average over the ordered costs of the scenarios. A special case, for example, is the Hurwicz criterion which is a convex combination of the costs of the scenario with maximum cost and the scenario with minimum cost.

Giving a unit weight to the most costly scenario translates to a min-max problem, comparing (for two machines) to our MM2 problem. For two machines, the authors give a 3/2 lower bound on the approximability of the problem Kasperski et al. (2012), whereas we present a lower bound of 2. For an arbitrary number of machines m, they observe the same relationship between the problem and the VECTOR SCHEDULING problem that we do (see later), and give a randomized algorithm that gives with high probability an $O\big(\log(km)/\log\log(km)\big)$-approximate schedule.

For the Hurwicz criterion, they give a 3/2 lower bound for the problem on 2 machines and observe that it can always be approximated within a factor of 2. For the general OWA operator on two machines, a matching lower and upper bound of 2 are given. For general m, it is shown that the problem can be solved in $O(n^{mk+1}m(p_{\max})^{mk})$ time (n being the total number of jobs and $p_{\max}$ being the processing time of the largest job), which is pseudopolynomial if m and k are constant.

Note that the min-sum problem that we consider can also be cast as an OWA objective by giving a weight of $1/k$ to each scenario. However, this special case is not considered by the authors.

In a different paper Kasperski et al. (2013), the same set of authors also considers optimizing over scenarios and studies the SELECTING ITEMS problem. From a set of n items, a subset of size p must be selected at minimum cost. The costs, however, are uncertain, and k cost scenarios are defined.

Finally, in very recent work, Kasperski and Zieliński (2016) consider single-machine scheduling problems under uncertainty defined by scenarios. They consider the problems of minimizing the maximum weighted tardiness and minimizing the sum of weighted completion times, both with and without precedence constraints. With respect to the scenarios, the very general OWA operator is considered, as well as many special cases of that objective. Inapproximability results are given for the most general cases, while some of the special cases can be solved in polynomial time. For a few of the harder cases, approximation algorithms are given.

Note that our research has been done independently of the works mentioned here, and although these authors study the same setting as we do, they study slightly different objectives and there is not much overlap between their results and ours.

Scenarios in terms of sets of jobs, as we propose here, appear also in Chen et al. (2015) in which two-stage scheduling problems are studied, most prominently in a black-box setting for a sample average approximation method of the stochastic version of the problem. In that paper, averages or robustness are measured with respect to first-stage decisions on time slots acquired for having machines available; the second-stage scheduling of jobs leads to different schedules in different scenarios.

Results Both problems that we study are NP-hard, since the single-scenario version (the well-known MAKESPAN MINIMIZATION problem) is NP-hard. However, the single-scenario version is only weakly NP-hard for two machines and an FPTAS exists Ausiello et al. (1999), whereas the problems defined here are strongly NP-hard. We will give various approximability and inapproximability results for several versions and special cases of the problem.

In Sect. 2, we study the problem MM2. We start by showing that the problem cannot be approximated to within a ratio of $2-\epsilon$, even in the case that $p_j = 1$ for all j. This is a remarkable result in two ways. First of all, notice that *any* assignment of the jobs, even assigning all jobs to one of the two machines, is at most 2-approximate. Secondly, the single-scenario version of the problem is trivial. Further, for the case that $p_j = 1$ for all j and $|S_i| \leq 3$ for all i, we show by a reduction from SET SPLITTING that the problem cannot be approximated to within ratio 3/2.

Inspired by the tight lower and upper bound on approximability of the general problem, we considered special cases, and we present a polynomial-time algorithm solving the problem for the special case that every scenario contains only two jobs.

We also note that both the MM2 and the SM2 problem are polynomial-time solvable if the number of scenarios is constant and all jobs have unit processing time.

Further, we observe a direct relationship between the MM2 problem and the VECTOR SCHEDULING problem (see

Sect. 2 for the definition of VECTOR SCHEDULING). By this relationship, combined with results from Chekuri and Khanna (2004), we find that if the number of scenarios is constant, there exists a PTAS for the MM2 problem and for an arbitrary number k of scenarios (and any number of machines) there exists an $O(\log^2 k)$-approximation.

The problem SM2 is studied in Sect. 3. Assuming $P \neq NP$, we prove inapproximability to within 1.0196, and to within 1.0404 under the *Unique Games Conjecture* Khot (2002). We present a very simple randomized algorithm that can be used for any number of machines and show that for the two-machine case, it gives a 3/2-approximate solution. For special cases with small scenarios, we can do better using deterministic algorithms. For instances where each scenario has size at most three, we use a reduction to MAX CUT to obtain a 1.12144-approximate algorithm. For cases where for all i, $|S_i| \leq r$ for some fixed r, we show a reduction to WEIGHTED MAX NOT-ALL-EQUAL r-SAT. For $r = 4$, this gives a deterministic 3/2-approximation algorithm, but for larger r, it does not give approximation ratios better than 2.

2 Minimizing the maximum makespan

First, we obtain inapproximability of MM2 by a reduction from SET SPLITTING, which was shown NP-complete in Garey and Johnson (1990) even if all sets have size three.

Theorem 1 *It is NP-hard to approximate MM2 with scenarios of size at most three to within a factor of 3/2.*

Proof It is shown Garey and Johnson (1990) that SET SPLITTING is NP-complete, even if all sets have size three. In SET SPLITTING, a set of elements is given together with a number of subsets. The SET SPLITTING problem is to decide whether the superset can be partitioned into two parts such that *each* of the subsets is split, i.e., such that no subset is entirely on one side of the partition. For the inapproximability result, consider an instance of SET SPLITTING. For each element in the instance, we introduce a job with processing time 1. For each set in the input, we introduce a scenario with the corresponding three jobs. Then, there is a partition of the jobs into two sets A and $\bar{A}$ such that the objective value of MM2 is 2 if and only if there is a YES answer to the SET SPLITTING instance.

In particular, the NP-completeness with sets of size three implies a 3/2 inapproximability result. The makespan for every scenario is either 2 or 3. Approximating this problem to within a factor better than 3/2 implies being able to solve the decision problem SET SPLITTING. □

In the general case, we obtain a stronger inapproximability of MM2 using a recent result Austrin et al. (2014) on the hardness of HYPERGRAPH BALANCING.

Theorem 2 (Theorem I.2 in Austrin et al. 2014) *For $g \geq 1$, given a $(2g+1)$-uniform hypergraph that admits a 2-coloring under which each hyperedge is evenly balanced (g elements of one color and $g+1$ of the other), it is NP-hard to find a 2-coloring that avoids creating a monochromatic hyperedge.*

Corollary 1 *It is NP-hard to approximate MM2 with unitary jobs to within ratio $2-\epsilon$ for any $\epsilon > 0$.*

Proof Let $g = \lceil 1/\epsilon \rceil$, so $\epsilon \geq 1/g$. Given a $(2g+1)$-uniform hypergraph that admits a 2-coloring in which each hyperedge is balanced, we can define an MM2 instance containing a job with unit processing time for each vertex of the hypergraph, and a scenario for each hyperedge. Because the hypergraph admits a 2-coloring under which each hyperedge is evenly balanced, there exists a solution to this MM2 instance where the makespan of every scenario is exactly $g+1$.

A $(2-\epsilon)$-approximation to MM2 for this instance would therefore give a solution for which the maximum makespan is at most $(2-\epsilon)(g+1) \leq (2-(1/g))(g+1) < 2g+1$, which corresponds to a 2-coloring of the hypergraph that has no monochromatic edge. In other words, a $(2-\epsilon)$-approximation to MM2 would solve a problem that is NP-hard. □

This bound is remarkable since, for the problem on two machines, *any* solution is 2-approximate (even for arbitrary job sizes). Also notice that the single-scenario version of the problem with unit-size jobs is trivial on any number of machines.

On the positive side, we show for MM2 that, if the number of jobs per scenario is at most two, the problem can be solved in polynomial time. Since single-job scenarios will always have makespan 1, we concentrate on the case in which all scenarios consist of two jobs.

Theorem 3 *The problem MM2 with $|S_i| = 2$ for all $S_i \in \mathcal{S}$ can be solved in time $O(|\mathcal{S}| \log |\mathcal{S}|)$.*

Proof We create a graph with a vertex for each job and connect by an edge the jobs that appear together in a scenario. We define the weight of edge (j,k) to be $p_j + p_k$, i.e., the sum of the processing times of the jobs associated to the incident vertices. Note that a solution for the MM2 scheduling problem is a partitioning of the job set, and can be associated with a coloring of the vertices in this graph with two color classes. The objective value is then equal to the maximum of the highest weight of any monochromatic edge and the largest processing time of any job. In other words, we should find a 2-coloring of the vertices of this graph, such that the maximum weight of a monochromatic edge is minimized.

Our algorithm is based on the observation that the weight of an edge of lowest weight in any odd cycle in this graph is a lower bound on the objective value.

Starting with all vertices being part of their own singleton component, and having color 1, we grow components by inserting edges, and label the vertices with the component they belong to, and with a color that can assume two values: 1 and 2. A color inversion of a vertex changes the color of the vertex (i.e., if it is colored 1, the color is changed to 2, and vice versa). We consider the edges in order of descending weight. If the next edge has weight less than $p_{\max}$, the optimal value is $p_{\max}$ and the present coloring is an optimal solution. Otherwise, when considering the next edge, say (j, k), the following 3 cases can occur:

Case 1. Vertices j and k have the same color and are in the same component. We end the algorithm. An optimal partitioning of the job set is given by the two color classes of the jobs, where jobs that have color 1 (respectively, 2) are assigned to machine 1 (respectively, 2) and the objective is equal to the weight of edge (j, k).
Case 2. Vertices j and k have different colors. If the vertices are in different components, we update the component label for all nodes of the smaller component (breaking ties arbitrarily), so that all nodes have the same label. We then proceed to the next edge.
Case 3. Vertices j and k have the same color and are in different components. In this case, we invert the color of all nodes in the smaller component (breaking ties arbitrarily) and then proceed as in Case 2.

By construction, two nodes of the same color in the same component are joined by an even-length path. Therefore, when the algorithm terminates in Case 1, we have found an odd cycle in the graph, of which this last edge has the lowest weight. As we argued, this is a lower bound on the optimal value. Since all next edges do not have higher weight, it will be the best lower bound. Since it is higher than $p_{\max}$, it will in fact be the optimal value.

The running time follows from the observation that any time we invert the color and/or update the label of a vertex, it ends up in a component of at least twice the size of the component it belonged to before. Hence, the label of a vertex can be updated at most $\log |J|$ times. The total time can thus be bounded by $|\mathcal{S}| \log |\mathcal{S}|$ time for sorting the edges by weight, plus $|J| \log |J|$ time for updating the vertex colors and labels. Finally, we may assume without loss of generality that each job appears in at least one scenario, so $|\mathcal{S}| \geq |J|/2$. □

Another special case is the case of a constant number of scenarios. For jobs with unit processing times, the problem can be solved exactly.

Theorem 4 *The MM2 problem with a constant number of scenarios and unit processing times can be solved exactly in polynomial time.*

Proof Since all processing times are equal, jobs appearing in the same set of scenarios are identical for any scheduling algorithm. Thus, we define each job to be of a certain *type*, by the scenarios it appears in. A constant number k of scenarios yields a constant number 2^k of possible job types. Then, finding an assignment consists of determining how many jobs of each type are assigned to the first machine. Since there can be no more jobs per type than the total number of jobs, say n, there exist at most n^{2^k} assignments, which is a polynomial in input size n for constant k. For each of these assignments, the minimum makespan over all scenarios can be found in polynomial time, and thus, the whole problem can be solved in polynomial time. □

Note that this proof also works for the SM2 problem with a constant number of scenarios and unit processing times. The proof can also be extended to a larger (but constant) number of machines.

We conclude this section by noticing that if we consider any number of machines, the problem of minimizing the maximum makespan reduces to the VECTOR SCHEDULING problem, where each coordinate corresponds to a scenario.

Definition 1 In the VECTOR SCHEDULING problem, we are given a set V of n rational d-dimensional vectors $v_1, \ldots, v_n$ from $[0, \infty)^d$ and a number m. A valid solution is a partition of V into m sets $A_1, \ldots, A_m$. The objective is to minimize $\max_{1 \leq i \leq m} || \sum_{j \in A_i} v_j ||_\infty$.

This problem is a d-dimensional generalization of the MAKESPAN MINIMIZATION problem, where each job is a d-dimensional vector and the machines are d-dimensional objects as well. In our setting, d equals k, the number of scenarios. Each coordinate of job j equals its processing time in the corresponding scenario (either 0 or p_j). Results of Chekuri and Khanna (2004) on VECTOR SCHEDULING can directly be translated into our setting.

Theorem 5 (Chekuri and Khanna 2004) *For the problem of minimizing the maximum makespan over scenarios $S_1, \ldots, S_k$ on m machines,*

1. *there exists a PTAS when k is constant;*
2. *there exists a polynomial-time $O(\log^2 k)$-approximation for arbitrary k; and*
3. *unless $NP = ZPP$, there exists no constant-approximation algorithm for arbitrary k.*

Note that our results improve on those from Theorem 5 for the special cases we consider.

3 Minimizing the sum of makespans

We now turn our attention to SM2, the problem of minimizing the sum of the makespans over all scenarios, in the case of two machines.

We start this section by noting that SM2 is MAX SNP-hard even with unitary processing times and scenarios containing two jobs each.

Theorem 6 *The problem SM2 is NP-hard to approximate to within a factor of* 1.0196 *and UGC-hard to approximate to within a factor of* 1.0404*, even if all jobs have unit length, and all scenarios contain two jobs.*

Proof We use a reduction from MAX CUT Garey and Johnson (1990). Given an (unweighted) MAX CUT instance G, we create a job with processing time 1 for each vertex, and for each edge we create a scenario consisting of the two jobs corresponding to the vertices incident to that edge. A cut in G is induced by a partition of the vertices, which corresponds to a partition $A, \bar{A}$ of the jobs. Scenario (j, k) contributes 1 to the objective of the scheduling problem if and only if $|\{j, k\} \cap A| = 1$, i.e., edge (j, k) is in the cut, and it contributes 2 otherwise (because if the two jobs are assigned to the same machine, the makespan of that scenario is 2). Thus, the objective value of the scheduling problem is equal to twice the total number of edges in the graph minus the size of the cut.

Now, let $OPT(CUT)$ be the optimal MAX CUT objective, let ℓ be the number of edges in the instance, and assume we have an $(1+\alpha)$-approximation algorithm for our problem, i.e., a solution with sum of makespans at most $(1+\alpha)(2\ell - OPT(CUT))$. Hence, at least $(1+\alpha)OPT(CUT) - \alpha 2\ell$ scenarios have a makespan of 1, i.e., the size of the corresponding cut in G is at least $(1+\alpha)OPT(CUT) - \alpha 2\ell$. Now, note that $OPT(CUT) \geq \ell/2$, and hence, the size of the cut is at least $(1-3\alpha)OPT(CUT)$.

The lower bound on the approximability of MAX CUT proven by Håstad (2001) gives us that $1 - 3\alpha \geq 0.941176$ unless $P = NP$, and by a result of Khot et al. (2007), $1 - 3\alpha \geq 0.878567$ under the Unique Games Conjecture. □

In the remainder of this section, we will give approximation results for SM2. As for MM2 in the previous section, we notice that also for this problem any solution is a trivial 2-approximation. We will first present and analyze the most straightforward randomized algorithm for any number of machines that assigns the jobs to each of the machines independently with equal probability. For two machines, it gives a 3/2-approximation. Finally, for the two-machine case, we present two deterministic approximation algorithms, which have good approximation guarantees if the number of jobs in each scenario is small.

3.1 Randomized approximation algorithm

We analyze the approximation ratio of the randomized algorithm that randomly assigns the jobs to each of m machines independently with equal probability.

Lemma 1 *Given is a scenario S and a partition of the scenario into m sets $A^1, \ldots, A^m$. When assigning each job in the scenario to one of the m machines independently at random with equal probability, the expected load of the least loaded machine is at least $(m!/m^m) \min_{i=1,\ldots,m} p(A^i)$.*

Proof Assigning each job in the sets A^i to one of the m machines induces partitions of each of the sets A^i into $A^i_1, A^i_2 \ldots, A^i_m$, where the jobs in the same set of the partition are assigned to the same machine (where these sets are not necessarily all non-empty).

We will fix such partitions $A^i_1, A^i_2 \ldots, A^i_m$ for all i, and prove that the expected load of the least loaded machine is at least $(m!/m^m) \min_{i=1,\ldots,m} p(A^i)$ conditioned on the fact that the jobs in each set of the partition $A^i_1, A^i_2, \ldots, A^i_m$ are all on the same machine, and no two sets of the partition of A^i are assigned to the same machine, for every i. Note that the conditional probability that A^i_j is on machine ℓ equals $1/m$ for all i, j, and ℓ, and that these probabilities for different i are independent.

For each i, there is a subset A^i_j ($j \in \{1, \ldots, m\}$) of A^i occurring least frequently on the machine with the lowest load, say Q^i. Denote this frequency by α_i. We will now argue that $\sum_{i=1}^m \alpha_i \geq m!/m^m$.

Consider the sets Q^i. If all sets Q^i are on distinct machines, then one of them is on the least loaded machine, and therefore, $\sum_{i=1}^m \alpha_i$ is lower bounded by the probability that all sets Q^i are on distinct machines. The probability that m sets are assigned to m distinct machines, if they are assigned to machines independently at random with probability $1/m$, is $m!/m^m$. There are m^m ways of assigning m sets to m machines, all of which have the same (conditional) probability of happening. Then, $m!$ of these assignments are such that each machine gets assigned exactly one set. Therefore, $\sum_{i=1}^m \alpha_i \geq m!/m^m$.

Denoting by $L_{\min}$ the load on the least loaded machine, the bound on $\sum_{i=1}^m \alpha_i$ implies the claim in the lemma, because (conditioned on the partitioning)

$$E[L_{\min}] \geq \sum_{i=1}^m \alpha_i p(A^i) \geq \left(\sum_{i=1}^m \alpha_i\right) \min_i p(A^i) = \frac{m!}{m^m} \min_{i=1,\ldots,m} p(A^i).$$

This is true for every partitioning and hence also holds unconditionally. □

Theorem 7 *Randomly assigning all jobs to one of m machines independently with equal probability is a* $m-(m-1)\frac{m!}{m^m}$*-approximation for the SMm problem.*

Proof Consider a scenario S, and fix a schedule of minimum makespan. For each machine i, let A^i be the set of the jobs in S on machine i in this optimal schedule. Denote by OPT_S the optimal makespan in scenario S and denote by $E(ALG_S)$ the expected makespan in scenario S of the schedule resulting from the randomized algorithm.

Now we note the following bounds, where we use $L_{\min}$ to denote the load of the least loaded machine:

$$OPT_S \geq \frac{1}{m-1}\left(\sum_{i=1}^{m} p(A^i) - \min_i p(A^i)\right),$$

$$E(ALG_S) \leq \sum_{i=1}^{m} p(A^i) - (m-1)E[L_{\min}].$$

Combining these bounds and the bound on $E[L_{\min}]$ following from Lemma 1 gives that

$$\begin{aligned}
\frac{E(ALG_S)}{OPT_S} &\leq \frac{\sum_{i=1}^{m} p(A^i) - (m-1)E[L_{\min}]}{\frac{1}{m-1}\left(\sum_{i=1}^{m} p(A^i) - \min_i p(A^i)\right)} \\
&\leq (m-1)\left(\frac{\sum_{i=1}^{m} p(A^i) - \frac{(m-1)m!}{m^m}\min_i p(A^i)}{\sum_{i=1}^{m} p(A^i) - \min_i p(A^i)}\right) \\
&= (m-1)\left(1 + \frac{\left(1-\frac{(m-1)m!}{m^m}\right)\min_i p(A^i)}{\sum_{i=1}^{m} p(A^i) - \min_i p(A^i)}\right) \\
&\leq (m-1)\left(1 + \left(1-\frac{(m-1)m!}{m^m}\right)\cdot\frac{\frac{1}{m}\sum_{i=1}^{m} p(A^i)}{\frac{m-1}{m}\sum_{i=1}^{m} p(A^i)}\right) \\
&= (m-1)\left(1 + \left(1-\frac{(m-1)m!}{m^m}\right)\cdot\frac{1}{m-1}\right) \\
&= m - \frac{(m-1)m!}{m^m}.
\end{aligned}$$

Hence, the expected makespan for scenario S is at most $m-\frac{(m-1)m!}{m^m}$ times the optimal makespan for scenario S, which implies (by linearity of expectations) that the sum over all scenarios of the expected makespans is at most $m-\frac{(m-1)m!}{m^m}$ times the optimal summed makespan of all scenarios. □

Corollary 2 *Randomly assigning each job to the two machines independently with equal probability is a* 3/2*-approximation for SM2.*

We notice that the proof of the previous lemma bounds the objective value by comparing the load on a machine in a given scenario to the load for the optimal schedule for that scenario, rather than the optimal schedule for our problem. We conjecture that $m-(m-1)\frac{m!}{m^m}$ is not a tight ratio for this algorithm.

However, it is easy to see that the analysis of the simple randomized algorithm is tight for SM2. Consider an instance of two jobs $\{1,2\}$ with unitary processing time and one scenario $S_1=\{1,2\}$. The optimal solution is to assign one job to each machine, whereas the randomized algorithm either assigns both jobs to the same machine with probability $\frac{1}{2}$, or one job to each machine with probability $\frac{1}{2}$.

3.2 Deterministic approximation algorithms

We now show how to get improved approximation guarantees for the problem on two machines, for instances, in which each scenario has a small number of jobs. We show that the SM2 problem can be reduced to the WEIGHTED MAX NOT-ALL-EQUAL SATISFIABILITY problem that we will abbreviate as MAX-NAE SAT.

Definition 2 In MAX-NAE SAT, a boolean expression is given, and a weight for each clause. A clause in the expression is satisfied if it contains both true and false literals. The problem is to find an assignment of true/false values to the variables, such as to maximize the total weight of the satisfied clauses.

Note that if r is such that $|S_i| \leq r$ for all $S_i \in \mathcal{S}$, then by adding dummy jobs of processing time 0, we can assume that every scenario contains exactly the same number of jobs, i.e., $|S_i| = r$ for all $S_i \in \mathcal{S}$. We will reduce the SM2 problem with scenarios of size at most r to the MAX-NAE SAT problem with clauses of length r (MAX-NAE r-SAT).

Theorem 8 *A* $(1-\gamma_r)$*-approximation for* MAX-NAE r-SAT *implies a* $(1+2^{r-2}\gamma_r)$*-approximation for the SM2 problem with* $|S| \leq r$ *for all scenarios* $S \in \mathcal{S}$.

Proof We start by formulating the SM2 problem as a MAX-NAE SAT problem. Each job j corresponds to a variable x_j in the MAX-NAE SAT instance. An assignment of the variables in the MAX-NAE SAT instance corresponds to an assignment in SM2 as follows: machine 1 is assigned all jobs for which the corresponding variable is set to true, and machine 2 processes all jobs for which the corresponding variable is set to false.

We now construct a set of weighted clauses for each scenario such that the weight of the satisfied clauses for a given assignment is equal to the load of the least loaded machine in the scenario. Hence, maximizing the weight of the satisfied clauses will maximize the weight of the least loaded machine, and it will thus minimize the weight of the machine with the heaviest load, i.e., the makespan.

For a given scenario S of SM2 with r jobs, we construct 2^{r-1} clauses of length r as follows: For each partitioning of S into two sets A and $\bar{A}$, we create a clause denoted by $C_S(\{A,\bar{A}\})$. In clause $C_S(\{A,\bar{A}\})$, all variables corresponding to jobs in one set appear negated, all variables corresponding to jobs in the other set appear non-negated.

Note that $C_S(\{A, \bar{A}\})$ has the same truth table as $C_S(\{\bar{A}, A\})$ (namely a clause is false if and only if all its literals are false, or all its literals are true). This means that if A is assigned to the first machine and $\bar{A}$ is assigned to the second machine, then all clauses *except* $C_S(\{A, \bar{A}\})$ are satisfied.

Denote by $w_S(\{A, \bar{A}\})$ the weight on the clause $C_S(\{A, \bar{A}\})$. To ensure the weight of the satisfied clauses is equal to the weight of the least loaded machine in SM2, we define weights on the clauses to be so that

$$\sum_{\substack{B,\bar{B}:\\ B\cup\bar{B}=S,\\ B\cap\bar{B}=\emptyset}} \Big(w_S(\{B, \bar{B}\})\Big) - w_S(\{A, \bar{A}\}) = \min\{p(A), p(\bar{A})\}.$$

Let $N = 2^{r-1}$, i.e., N is the number of clauses corresponding to each scenario. The solution to this system of equations is to set

$$w_S(\{A, \bar{A}\}) = \frac{1}{N-1} \sum_{\substack{B,\bar{B}:\\ B\cup\bar{B}=S,\\ B\cap\bar{B}=\emptyset}} \Big(\min\{p(B), p(\bar{B})\}\Big) - \min\{p(A), p(\bar{A})\}.$$

The weights thus defined are not necessarily non-negative: consider a scenario S that contains $r = 4$ jobs of unit length. There are four ways of partitioning S into one set of size one and one set of size three, and there are $\binom{4}{2}/2 = 3$ ways of partitioning S into two sets of size two. Therefore, $\sum_{B,\bar{B}} \min\{p(B), p(\bar{B})\} = 10$, but that means that for a partitioning into sets $A, \bar{A}$ of size two $w_S(\{A, \bar{A}\}) = \frac{1}{7}(10) - 2 < 0$.

To use approximation algorithms for MAX- NAE SAT, we need to make sure that all weights are non-negative. We accomplish this by adding a constant $K(S)$ to all weights of clauses corresponding to scenario S, where we set $-K(S)$ equal to a lower bound on the weights. We derive a lower bound on the weights by noting that

1. $\frac{1}{N}\sum_{B,\bar{B}} \min\{p(B), p(\bar{B})\}$ is the expected value of the least loaded machine when all jobs are assigned to a machine with probability $\frac{1}{2}$ independently. By Lemma 1, its value is lower bounded by $\frac{1}{2}\min\{p(B), p(\bar{B})\}$ for any partition $B, \bar{B}$; and hence it is also lower bounded by $\frac{1}{2}\max_{B,\bar{B}} \min\{p(B), p(\bar{B})\}$; and
2. trivially, for any S, $\max_{B,\bar{B}} \min\{p(B), p(\bar{B})\} \le \frac{1}{2}p(S)$.

Therefore,

$$\begin{aligned}
w_S(\{A, \bar{A}\}) &= \tfrac{1}{N-1} \sum_{\substack{B,\bar{B}:\\ B\cup\bar{B}=S,\\ B\cap\bar{B}=\emptyset}} \Big(\min\{p(B), p(\bar{B})\}\Big) \\
&\quad - \min\{p(A), p(\bar{A})\} \\
&= \tfrac{N}{N-1}\tfrac{1}{N} \sum_{\substack{B,\bar{B}:\\ B\cup\bar{B}=S,\\ B\cap\bar{B}=\emptyset}} \Big(\min\{p(B), p(\bar{B})\}\Big) \\
&\quad - \min\{p(A), p(\bar{A})\} \\
&\ge \tfrac{N}{N-1}\tfrac{1}{2} \max_{B,\bar{B}} \min\{p(B), p(\bar{B})\} \\
&\quad - \min\{p(A), p(\bar{A})\} \\
&\ge \tfrac{\frac{1}{2}N-(N-1)}{N-1} \max_{B,\bar{B}} \min\{p(B), p(\bar{B})\} \\
&= -\tfrac{1}{2}\tfrac{N-2}{N-1} \max_{B,\bar{B}} \min\{p(B), p(\bar{B})\} \\
&\ge -\tfrac{1}{4}\tfrac{N-2}{N-1} p(S).
\end{aligned}$$

Thus, we set $K(S) = \frac{1}{4}\frac{N-2}{N-1}p(S)$, such that $\tilde{w}_S(\{A, \bar{A}\}) = w_S(\{A, \bar{A}\}) + K(S) \ge 0$ for all partitionings $A, \bar{A}$ of S into two sets.

A solution to the MAX- NAE SAT instance is now mapped to a solution of SM2, by assigning the jobs for which the variable is set to true to machine 1, and scheduling the other jobs on machine 2. We note that the w-weights of the clauses corresponding to scenario S were chosen so that the sum of the weights of the clauses that are satisfied is exactly equal to the load on the least loaded machine in scenario S. Also, $N-1$ clauses of scenario S are satisfied in any solution to the MAX- NAE SAT instance. Therefore, the total $\tilde{w}$-weight of the clauses for scenario S that are satisfied in any MAX- NAE SAT solution is equal to the load on the least loaded machine in scenario S plus an additional $(N-1)K(S)$.

We let $L = \sum_S p(S)$ and denote by $L^*_{\min}$ the sum over all scenarios of the load of the least loaded machine in an optimal solution and by $L^*_{\max}$ the sum over all scenarios of the load of the most loaded machine in an optimal solution, so that $L^*_{\min} + L^*_{\max} = L$. Note that the additional term $K(S)$ in the $\tilde{w}$-weights of the MAX- NAE SAT solution causes an increase of the objective value with respect to the w-weights solution by adding an additional $\sum_S (N-1)K(S) = \sum_S \frac{1}{4}(N-2)p(S) = \frac{1}{4}(N-2)L$ to each solution.

In particular, an optimal solution to the MAX- NAE SAT instance has objective value $L^*_{\min} + \frac{1}{4}(N-2)L$ and a $(1-\gamma)$-approximation algorithm for the MAX- NAE SAT instance therefore has objective value at least $(1-\gamma)\left(L^*_{\min} + \frac{1}{4}(N-2)L\right)$. Let us denote by $ALG(L_{\min})$ and $ALG(L_{\max})$ the sum over all scenarios of the least and most loaded machines in the corresponding job assignment. Note that

$$\begin{aligned}
ALG(L_{\min}) &\ge (1-\gamma)\left((L^*_{\min} + \tfrac{1}{4}(N-2)L\right) - \tfrac{1}{4}(N-2)L \\
&= (1-\gamma)L^*_{\min} - \tfrac{1}{4}\gamma(N-2)L.
\end{aligned}$$

Therefore,

$$\begin{aligned} ALG(L_{\max}) &= L - ALG(L_{\min}) \\ &\leq L - \left((1-\gamma)L^*_{\min} - \tfrac{1}{4}\gamma(N-2)L\right) \\ &= (1-\gamma)(L - L^*_{\min}) + \gamma L + \tfrac{1}{4}\gamma(N-2)L \\ &= (1-\gamma)L^*_{\max} + \tfrac{1}{4}\gamma(N+2)L. \end{aligned}$$

Noting that $L \leq 2L^*_{\max}$ gives $ALG(L_{\max}) \leq (1-\gamma)L^*_{\max} + \frac{1}{2}\gamma(N+2)L^*_{\max} = (1+\frac{1}{2}\gamma N)L^*_{\max}$, which proves the theorem, since $N = 2^{r-1}$. □

For $r = 3$, Zwick (1999) gives a 0.90871-approximation for MAX- NAE 3- SAT. By the previous lemma, this gives a 1.18258-approximation for SM2 with scenarios of size at most three. For $r = 4$, Karloff and Zwick (1997) give a $\frac{7}{8}$-approximation for MAX- NAE 4- SAT. By our lemma, this implies a $\frac{3}{2}$-approximation for SM2 with scenarios of size four. Note that this matches the guarantee we proved for the algorithm that randomly assigns each job to one of the two machines.

Corollary 3 *For the SM2 problem with scenarios containing at most three jobs, there exists a 1.18258-approximation algorithm, and for the problem with scenarios of size at most four, there exists a deterministic 3/2-approximation algorithm.*

For general r, the best approximation factor known for MAX- NAE SAT is 0.74996 due to Zhang et al. (2004), and the implied approximation guarantees for our problem are worse than the trivial bound of 2.

If every scenario has at most three jobs, we can obtain an even better approximation guarantee by reducing SM2 to the MAX CUT problem.

Theorem 9 *A $(1-\gamma)$-approximation algorithm for* MAX CUT *implies a $(1+\gamma)$-approximation algorithm for the SM2 problem with scenarios containing at most three jobs.*

Proof If every scenario has exactly two jobs, SM2 can be reduced to MAX CUT as follows: we create a vertex for every job, and add an edge between j and k of weight $\min\{p_j, p_k\}$ for every scenario that contains jobs j and k. For any cut, the weight of the edges crossing the cut is then exactly the sum over all scenarios of the load of the least loaded machine. Since the makespan for a scenario S is $p(S)$ minus the load of the least loaded machine, maximizing the load of the least loaded machine, summed over all scenarios, is equivalent to minimizing the sum of the makespans.

If every scenario has three jobs, we can also reduce SM2 to MAX CUT, but the reduction is slightly more involved. We again create a vertex for every job, and for a scenario containing jobs j, k, and h, we add edges $\{j,k\}$, $\{k,h\}$ and $\{j,h\}$. If multiple scenarios contain jobs j and k, the corresponding edge will have the same multiplicity in the constructed graph. Note that a cut will either have zero or two of the edges corresponding to a given scenario crossing the cut. We now set the weight of the edges in such a way that if two edges cross the cut, then the sum of the weights of the two edges is equal to the load of the least loaded machine. In order to do this, we first define b_j to be the load of the least loaded machine in the scenario, if j is on one machine, and k and h are on the other machine, i.e., $b_j = \min\{p_j, p_k + p_h\}$. We similarly define $b_k = \min\{p_k, p_j + p_h\}$ and $b_h = \min\{p_h, p_j + p_k\}$. Then we want to set the weights $w(e)$ such that

$$\begin{aligned} w(j,k) + w(j,h) &= b_j; \\ w(j,k) + w(k,h) &= b_k; \\ w(j,h) + w(k,h) &= b_h. \end{aligned}$$

This is a system of three linearly independent equations with three unknowns, which has the (unique) solution $w(e) = \frac{1}{2}(b_j + b_k + b_h) - b_v$, where $e \in \{\{j,k\},\{k,h\},\{j,h\}\}$, and $v = \{j,k,h\}\backslash e$. Note that for any cut, the contribution of the edges of a scenario to the weight of the cut is exactly equal to the load on the least loaded machine if we assign the jobs on one side of the cut to one machine and the jobs on the other side of the cut to the other machine.

We now show that the weights thus defined are non-negative. Substituting the expressions for b_j, b_k, b_h, we get that

$$\begin{aligned} w(j,k) &= \tfrac{1}{2}b_j + \tfrac{1}{2}b_k - \tfrac{1}{2}b_h \\ &= \tfrac{1}{2}\left(\min\{p_j, p_k + p_h\} + \min\{p_k, p_j + p_h\} \right. \\ &\quad \left. - \min\{p_h, p_j + p_k\}\right). \end{aligned}$$

Now, noting that either $\min\{p_j, p_k + p_h\} + \min\{p_k, p_j + p_h\} = p_j + p_k$ or $\min\{p_j, p_k + p_h\} + \min\{p_k, p_j + p_h\} \geq p_h$, we get that $\min\{p_j, p_k + p_h\} + \min\{p_k, p_j + p_h\} \geq \min\{p_h, p_j + p_k\}$, and thus, $w(j,k) \geq 0$.

Let $L = \sum_{S\in\mathcal{S}} p(S)$, and for the optimal solution to SM2, let $L^*_{\min}$ be the sum over all scenarios of the load of the least loaded machine, and let $L^*_{\max}$ be the sum over all scenarios of the load of the most loaded machine, i.e., the sum of makespans. Then $L = L^*_{\min} + L^*_{\max}$. We denote by $ALG(L_{\min})$ and $ALG(L_{\max})$ the sums over all scenarios of the loads on the least loaded and most loaded machines defined by the cut. A $(1-\gamma)$-approximation to MAX CUT gives us an assignment of jobs to machines such that the sum over all scenarios of the least loaded machine is at least $ALG(L_{\min}) \geq (1-\gamma)L^*_{\min} = (1-\gamma)(L - L^*_{\max})$. The sum of the makespans over all scenarios is $ALG(L_{\max}) \leq L - (1-\gamma)(L - L^*_{\max}) = \gamma L + (1-\gamma)L^*_{\max}$. Now, note that the makespan for any scenario is at least half of the sum of the processing times, and hence, $L \leq 2L^*_{\max}$. So, a $(1-\gamma)$-approximation for MAX CUT implies a $(1+\gamma)$-

approximation for SM2 in the case where all scenarios have at most three jobs. □

Since the weights of the edges in the graph constructed in the proof of Theorem 9 are non-negative, we can use the 0.87856-approximation for MAX CUT of Goemans and Williamson (1995). Let $\gamma = 1 - 0.87856$. Then, the $(1+\gamma)$-approximation for SM2 gives us the following corollary:

Corollary 4 *There exists a* 1.12144*-approximation algorithm for the SM2 problem with scenarios containing at most three jobs.*

4 Epilogue

Complexity analysis of combinatorial optimization problems under a set of fully specified scenarios poses theoretically interesting questions as we hope to have shown with this paper. Explicitly stated as such, these type of problems have not been addressed often in the literature. The min-sum version has emerged as a result of a sampling algorithm for stochastic network optimization problems Gupta et al. (2011) and as such, several results in this direction already exist. In the context of scheduling, Kasperski et al. (2012, 2013); Kasperski and Zieliński (2016) have worked on variations of scheduling under scenarios. Apart from this, we are not aware of any other work in this direction.

In this paper, we have seen that the complexity of the scheduling problems can change dramatically from their, sometimes trivial, single-scenario version to their multiple-scenario version. However, in all cases that we studied here, the rise in complexity required a number of scenarios that is part of the input. In network design, there are examples of problems in which the complexity changes from being in P to being NP-complete from a single-scenario version to a two-scenario version Oriolo et al. (2013).

Apart from posing interesting theoretical questions, problems of this type enhance our ability to model scheduling problems where a learning aspect for performing jobs prohibits that job assignments can be adjusted on a day-by-day basis but merely require a fixed assignment whose quality then necessarily differs over the various instances.

In relation to the min-max version of the problem, we also like to mention a version of combinatorial optimization which has become known under the name universal optimization, e.g., in Epstein et al. (2012), a universal scheduling problem is studied. In such a problem, the scenarios are not explicitly specified but can be seen to be chosen by an adversary. The quality of an algorithm is then measured by comparing its solution to the solution of a clairvoyant optimal algorithm.

We finish this section by posing some questions emerging from our multiple-scenario scheduling problem. We believe that the analysis of the randomized algorithm for SMm given in Sect. 3.1 is far from tight, although it gives a tight result for SM2. Apart from giving a tighter analysis, it would also be interesting to find out if our randomized algorithm can be derandomized.

For the SM2 version, it would be interesting to see how the gap between the 3/2-approximation ratio of the randomized algorithm for the general case on two machines and the 1.0404 lower bound under the Unique Games Conjecture can be closed. We have no conjecture on which of the bounds is closest to the best attainable ratio.

Finally, the very strong bound on inapproximability for the MM2 problem raises the question whether such a strong lower bound will also hold for the case of m machines.

References

Ausiello, G., Crescenzi, P., Kann, V., Gambosi, G., Marchetti-Spaccamela, A., & Protasi, M. (1999). *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Berlin: Springer.

Austrin, P., Håstad, J., & Guruswami, V. (2014). $(2+\epsilon)$-SAT is NP-hard. In: *Proceedings of 55th Annual IEEE Symposium on Foundations of Computer Science* (pp. 1–10).

Ben-Tal, A., & Nemirovski, A. (2002). Robust optimization—methodology and applications. *Mathematical Programming*, *92*(3), 453–480.

Birge, J. R., & Louveaux, F. (2011). *Introduction to stochastic programming*. New York: Springer Science & Business Media.

Chekuri, C., & Khanna, S. (2004). On multidimensional packing problems. *SIAM Journal on Computing*, *33*(4), 837–851.

Chen, L., Megow, N., Rischke, R., & Stougie, L. (2015). Stochastic and robust scheduling in the cloud. In: *Proceedings of the 18th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems* (pp. 175–186).

Epstein, L., Levin, A., Marchetti-Spaccamela, A., Megow, N., Mestre, J., Skutella, M., & Stougie, L. (2012). Universal sequencing on an unreliable machine. *SIAM Journal on Computing*, *41*(3), 565–586.

Garey, M. R., & Johnson, D. S. (1990). *Computers and intractability: A guide to the theory of NP-completeness*. New York: W. H. Freeman & Co.

Goemans, M. X., & Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, *42*(6), 1115–1145.

Gupta, A., Pál, M., Ravi, R., & Sinha, A. (2011). Sampling and cost-sharing: Approximation algorithms for stochastic optimization problems. *SIAM Journal on Computing*, *40*(5), 1361–1401.

Håstad, J. (2001). Some optimal inapproximability results. *Journal of the ACM*, *48*(4), 798–859.

Jaillet, P. (1985). Probabilistic traveling salesman problems. Technical Report 185, Operations Research Center, MIT.

Jaillet, P. (1988). A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, *36*, 929–936.

Karloff, H. J., & Zwick, U. (1997). A 7/8-approximation algorithm for MAX 3SAT? In: *Proceedings of 38th Annual IEEE Symposium on Foundations of Computer Science* (pp. 406–415).

Kasperski, A., Kurpisz, A., & Zieliński, P. (2012). Parallel machine scheduling under uncertainty. In: *Proceedings of 14th Interna-*

tional Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU) (pp. 74–83).

Kasperski, A., Kurpisz, A., & Zieliński, P. (2013). Approximating the min-max (regret) selecting items problem. *Information Processing Letters*, *113*, 23–29.

Kasperski, A., & Zieliński, P. (2016). Single machine scheduling problems with uncertain parameters and the OWA criterion. *Journal of Scheduling*, *19*, 177–190.

Khot, S. (2002). On the power of unique 2-prover 1-round games. In: *Proceedings of 34th ACM Symposium on Theory of Computing* (pp. 767–775).

Khot, S., Kindler, G., Mossel, E., & O'Donnell, R. (2007). Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM Journal on Computing*, *37*(1), 319–357.

Kleywegt, A., Shapiro, A., & de Mello, T. H. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, *12*(2), 479–502.

Lin, X., Janak, S., & Floudas, C. (2004). A new robust optimization approach for scheduling under uncertainty: I. bounded uncertainty. *Computers and Chemical Engineering*, *28*, 1069–1085.

Oriolo, G., Sanità, L., & Zenklusen, R. (2013). Network design with a discrete set of traffic matrices. *Operations Research Letters*, *41*(4), 390–396.

Pinedo, M. (2012). *Theory, scheduling algorithms and systems*. Berlin: Springer.

Wang, Z., & Chan, F. (2015). A robust replenishment and production control policy for a single-stage production/inventory system with inventory inaccuracy. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *45*(2), 326–337.

Zhang, J., Ye, Y., & Han, Q. (2004). Improved approximations for max set splitting and max NAE SAT. *Discrete Applied Mathematics*, *142*(1–3), 133–149.

Zwick, U. (1999). Outward rotations: A tool for rounding solutions of semidefinite programming relaxations, with applications to MAX CUT and other problems. In: *Proceedings of 31st ACM Symposium on Theory of Computing* (pp. 679–687).