Chapter

A Brief Look at Multi-Criteria Problems: Multi-Threshold Optimization versus Pareto-Optimization

Nodari Vakhania and Frank Werner

Abstract

Multi-objective optimization problems are important as they arise in many practical circumstances. In such problems, there is no general notion of optimality, as there are different objective criteria which can be contradictory. In practice, often there is no unique optimality criterion for measuring the solution quality. The latter is rather determined by the value of the solution for each objective criterion. In fact, a practitioner seeks for a solution that has an acceptable value of each of the objective functions and, in practice, there may be different tolerances to the quality of the delivered solution for different objective functions: for some objective criteria, solutions that are far away from an optimal one can be acceptable. Traditional Pareto-optimality approach aims to create all non-dominated feasible solutions in respect to all the optimality criteria. This often requires an inadmissible time. Besides, it is not evident how to choose an appropriate solution from the Pareto-optimal set of feasible solutions, which can be very large. Here we propose a new approach and call it multi-threshold optimization setting that takes into account different requirements for different objective criteria and so is more flexible and can often be solved in a more efficient way.

Keywords: multi-critneria optimization, optimal solution, Pareto-optimization, multi-threshold optimization, scheduling algorithm, time complexity

1. Introduction

Multi-objective optimization problems are important as they arise in many practical circumstances. In such problems, there is no general notion of optimality, as there are different objective criteria which are often contradictory: an optimal solution for one criterion may be far away from an optimal one for some other criterion. Thus for many such real-life problems, there is no unique optimality criterion for measuring the solution quality. The latter is rather determined by the value of the solution for each objective criterion. In fact, a practitioner is not interested, generally, in optimizing a particular objective criterion, but he rather seeks for a solution that has an acceptable value of each of the objective functions. Furthermore, in practice, there may exist different tolerances to the quality of the

delivered solution for different objective functions. In particular, for some objective criteria, solutions far away from an optimal one can be acceptable. Such solutions can often be obtained by relatively low computational efforts even for intractable problems.

Taking into account these considerations, here we propose a new approach and call it multi-threshold optimization setting that takes into account different requirements for different objective criteria, in contrary to a traditional Pareto-optimality approach. The Pareto-optimality concept, named after an Italian scientist Vilfredo Pareto, is a traditionally used compromise to address a complicated multi-objective scenario. It looks for a so-called Pareto-optimal frontier of the feasible solutions consisting of those solutions that are not dominated by any other feasible solution (with respect to any of the given objective functions). This often requires an inadmissible time: finding the Pareto-optimal frontier often remains an intractable (NP-hard) problem. This is always the case if at least one of the corresponding single-criterion problems is NP-hard. Finding the Pareto-optimal set of solutions may be NP-hard even if none of the single-criterion problem is NP-hard. Besides, it is not evident how to choose an appropriate solution from the Pareto-optimal set of feasible solutions, which can be very large. The multi-threshold optimization approach is more flexible since it takes into account different requirements for different objective criteria: in practice, some objective criteria can be more critical than the other ones, and hence there may exist different degrees of tolerance for the deviation of the objective value of different criteria from the optimal objective value of the corresponding single-criterion problems.

The multi-threshold optimization problem seeks for a feasible schedule whose objective values are acceptable for a given particular application for all objective functions; in particular, they do not exceed (for minimization problems) or are no smaller (for maximization problems) than the components of a threshold vector specified by the practitioner whose ith component is some threshold value for the ith objective function. As we observe, depending on the components of the above vector, it might be possible to solve the multi-threshold optimization problem in a low-degree polynomial time even if all the corresponding single-criterion problems are NP-hard. A threshold vector with specific threshold values for each objective function is supposed to have a direct practical meaning. For practically useful values of the threshold vector, the multi-threshold optimization problem might be solved in a low-degree polynomial time by a kit of heuristic algorithms, each one being designed for one of the corresponding single-criterion problems. If the kit of heuristic algorithms fails to find a feasible solution respecting the threshold vector, then the heuristics for NP-hard single-criterion problems can be replaced by implicit enumeration algorithms. In fact, the replacement can be accomplished step by step, starting from the most critical heuristics. This kind of approach may be more practical since some objective criteria can be optimized easier than other ones. Besides, as already noted, the practitioner may not be interested, in general, in the minimization of each objective function but rather in a solution of an acceptable quality for every objective function: in practice, there may be different tolerances to the quality of the delivered solution for each objective function, and different objective functions might be optimized with quite different costs.

Thus our approach may lead to more efficient and practical solution of a multi-criteria problem than the corresponding Pareto-optimal setting. In the following sections, we give a brief comparative analysis of the Pareto-optimization approach with our multi-threshold optimization approach illustrating its advantage on single-machine scheduling problems.

2. Multi-criteria optimization problems

For an extensive description of multi-criteria optimization problems and the solution methods, the reader may have a look on a book by T'kindt and Billaut [1] and a survey paper [2] by the same authors.

Discrete *optimization* problems have emerged in the late 1940s of the last century due to the rapid growth of the industry and new rising demands in efficient solution methods. Modeled in mathematical language, such an optimization problem has a finite set of so-called feasible solutions; each feasible solution is determined by a set of mathematically formulated restrictions that naturally arise in practice. The quality of a feasible solution is measured by an objective function, whose domain is the whole set of feasible solutions. Ideally, one aims to determine a feasible solution that gives an extremal (minimal or maximal) value to the objective function, a so-called optimal solution. Since the number of feasible solutions is typically finite, theoretically, finding an optimal solution is trivial: just enumerate all the feasible solutions calculating for each of the value of the objective function and select any one with the optimal objective value. The main issue here is that a complete enumeration of all feasible solutions is mostly impossible in practice.

There are two distinct classes of combinatorial optimization problems, the class P of polynomially solvable ones and the intractable NP-hard problems. For a problem from the class P, there exists an efficient (polynomial in the size of the problem) algorithm, whereas no such algorithm exists for an NP-hard problem (the number of feasible solutions of an NP-hard optimization problem grows exponentially with the size of the input). It is widely believed that it is very unlikely that an NP-hard problem can be solved in polynomial time. Hence, it is natural to develop approximation solution methods.

Multi-criteria optimization problems are optimization problems with two or more different objective criteria. For the majority of such problems, there exists no single solution which optimizes (minimizes or maximizes) all the objective functions. In this sense, different objectives are contradictory, and hence, it is not straightforward to understand which feasible solution to the problem is optimal: a multi-criteria optimization problem typically has no optimal solution. In this situation, one may look for a solution which attains an acceptable value for each objective function or a solution which is not dominated by any other solution, in the sense that there is no other feasible solution which attains better objective values for all objective functions. We shall refer to the first and second versions of the multi-criteria optimization problem as *multi-threshold optimization* and *Pareto-optimization* versions and define them more formally below.

Let the k objective functions over the set $\mathcal{F}$ of feasible solutions of a given multi-criteria optimization problem be $f_1, \ldots, f_k$. Since these functions might be mutually contradictory, there may exist no feasible solution minimizing/maximizing all objective functions simultaneously. Without loss of generality, let us consider from now on the minimization version of our multi-criteria optimization problem.

Let F_i^* be the optimal value for a single-criterion problem with the objective to minimize function f_i, and let A^i be some threshold value for the objective function f_i, $i = 1, \ldots, k$.

In the multi-threshold optimization version, we look for a feasible solution σ such that $f_i(\sigma) \leq A^i$ for each $i = 1, \ldots, k$.

A commonly used dominance relation for the Pareto-optimization version is defined as follows.

A solution $\sigma_1 \in \mathcal{F}$ *dominates* a solution $\sigma_2 \in \mathcal{F}$ if $f_i(\sigma_1) < f_i(\sigma_2)$ for $i = 1, \ldots, k$; in fact, we allow $\leq$ instead of $<$ for all values of i except one requiring to have at least one strict inequality.

Now $\sigma \in \mathcal{F}$ is a *Pareto-optimal solution* if no other solution from the set $\mathcal{F}$ dominates the solution σ. We shall refer to the set of all such feasible solutions as *Pareto-optimal set*. Forming a Pareto-optimal set of feasible solutions may be not easy. For instance, let, for $k = 2, f_1(\sigma_1) \leq f_1(\sigma_2)$; then solution σ_2 is dominated by solution σ_1 if $f_2(\sigma_1) < f_2(\sigma_2)$. This condition can be verified in polynomial time for any pair of solutions σ_1 and σ_2 (given that the corresponding optimization problem is from the class NP). However, whenever the number of feasible solutions grows exponentially with the length of the input (which is the case for NP-hard problems), the explicit evaluation of all possible pairs of feasible solutions (which is unavoidable for finding a dominant solution) would lead us to an exponential-time performance. In particular, if one of the single-criterion problems is NP-hard, finding a Pareto-optimal set for the multi-objective setting will take an exponential time.

Theorem 1 The problem of finding a Pareto-optimal set of feasible solutions for a multi-objective optimization problem with the objective functions $f_1, \ldots, f_k$ is NP-hard if one of the corresponding single-criterion problems is NP-hard.

Proof. We basically reformulate the above reasoning. Consider a bi-criteria optimization problem with $k = 2$. Consider the set S_A of feasible solutions with $f_1(\sigma) = A$ for all $\sigma \in S_A$ and some threshold value A of function f_1 (without loss of generality assume that $S_A \neq \emptyset$). A Pareto-optimal solution from the set S_A must attain the minimum possible value of function f_2 as otherwise it will be dominated by one that attains this value. Then we arrive at a single-criterion optimization problem with the objective function f_2, which is NP-hard.

From the first glance, the multi-threshold optimization version of a multi-criteria optimization problem may seem to be easier than the Pareto-optimality version. This is, in part, correct, but considering a threshold vector with arbitrary components, in general, we will also arrive at an intractable problem as the decision version of an NP-hard single-criterion optimization problem is NP-complete. In particular, suppose that we are given a single-criterion optimization problem with the objective to minimize the function f_i ($i \in \{1, \ldots, k\}$). If this problem is NP-hard, then its decision version, given a threshold value A of function f_i, if there is a feasible solution $\sigma \in \mathcal{F}$ with $f_i(\sigma) \leq A$, is NP-complete. Hence, if one of the single-criterion optimization problems is NP-hard, then the multi-threshold optimization version of the corresponding multi-criteria optimization problem is also NP-hard.

At the same time, finding a Pareto-optimal set of feasible solutions may be NP-hard even if none of the single-criterion problem is NP-hard, i.e., they are solvable in polynomial time. Can the multi-threshold optimization version of a multi-criteria optimization problem be solved in polynomial time, if all the corresponding single-criterion optimization problems are polynomial? In other words, suppose that the single-criterion problem of finding a feasible solution attaining the minimum value of the objective function f_i for $i = 1, \ldots, k$ can be solved in polynomial time. Then clearly, the decision version that seeks for a feasible solution $\sigma \in \mathcal{F}$ with $f_i(\sigma) \leq A$ is also polynomially solvable.

Unlike the Pareto-optimization problem, the multi-threshold optimization problem may be solvable in polynomial time even if all the corresponding single-criterion problems are NP-hard; whether it is solvable in polynomial time or not essentially depends on the particular threshold vector $\mathbf{A} = (A^1, \ldots, A^k)$. As we shall argue in the next sections, depending on the particular threshold values for each objective function, it might be possible to solve the multi-threshold optimization

problem in a low-degree polynomial time even if all the corresponding single-criterion problems are NP-hard. The given threshold values for each objective function may have a direct practical meaning. For practically useful values of the threshold vector **A**, the corresponding instance of the multi-threshold optimization problem might be solved in a low-degree polynomial time though it may be NP-hard, in general (for an arbitrary threshold vector, see Section 3).

3. Some basic single-criterion scheduling problems

In the rest of this chapter, we illustrate the Pareto-optimality and the multi-threshold optimization approaches for *scheduling problems*. For recent developments in multi-criteria optimization for scheduling problems, the reader is referred to a recent survey by Nagar et al. [3] and Parveen and Ullah [4] and for some earlier works approximately until the year 2005 to the earlier cited work by T'kindt and Billaut [1].

The scheduling problems arise in various practical circumstances. Examples of such problems are job shop problems in industry, scheduling of information and computational processes, and traffic scheduling and servicing of cargo trains, ships, and airplanes. There are scheduling problems of diverse types and different complexities. Saying generally, one deals with two primary notions: *job* (or *task*) and *machine* (or *processor*). A job is a part of the whole work to be done; a machine is the means for the performance of a job. A common restriction in scheduling problems is that a machine cannot handle more than one job at a time. Each job j is characterized by its *processing time* p_j, i.e., it needs this prescribed time on a machine. A job may have other parameters as well, which may yield additional restrictions and/or can be employed by an objective function. For instance, the *release time* r_j of job j is the time moment when job j becomes available (it cannot be scheduled before that time). The *due date* d_j of job j is the desirable completion time for job j (there may exist a penalty for the late or for the early completion of that job). A job *preemption* might be allowed, i.e., it might be split into portions, each portion being assigned at a different time interval to the machine(s). A *(feasible) schedule* assigns each job j to the machine(s) at the specified time moment(s) no less than r_j with the total duration of p_j so that no two jobs are assigned to the machine at any time moment (i.e., the job execution intervals cannot overlap in time). A job is *late* (*on time*, respectively) if it is completed after (at or before, respectively) its due date.

In the single-machine scheduling problems, there is a single machine on which all the jobs are to be scheduled. The majority of single-machine single-criterion scheduling problems are NP-hard, although there are polynomially solvable cases as well. For instance, if the objective function is the maximum job completion time called the *makespan* and denoted by $C_{\max}$, then the problem of minimizing $C_{\max}$, commonly abbreviated by $1||C_{\max}$ according to the standard Graham's notation for scheduling problems, is straightforwardly solvable if each job j has a single parameter p_j (the processing time): schedule the jobs in any order without creating machine idle time before the first scheduled job and between any pair of jobs. It is very easy to see that this list scheduling algorithm gives an optimal solution. If each job j has also a release time r_j (the problem $1|r_j|C_{\max}$), then scheduling the jobs in any order may not be good, but still there is a very simple greedy way to arrange them optimally: just order the jobs with non-decreasing release times and iteratively assign the next job from the list to the machine at the completion time of the previously assigned job or at the release time of the former job, whichever magnitude is larger.

Minimizing the makespan becomes more complicated with even two machines or if each job j has an additional job parameter called the *delivery time* q_j, which is an extra amount of time needed for job j for its full completion *once* it is already completed on the machine (the delivery of each job is accomplished independently of the machine immediately after its completion on the machine). Thus, job j will take p_j time on the machine and then an additional time q_j for its full completion (during which another job might be assigned to the machine). Then the maximum job completion time in the schedule σ (the makespan) is:

$$C_{\max}(\sigma) = \max_{j \in \sigma} \left\{ s_j(\sigma) + p_j + q_j \right\}. \tag{1}$$

The objective is to find a feasible schedule in which the maximum job completion time is the minimum possible one.

If there are no job release times, i.e., all jobs are released simultaneously (the problem $1|q_j|C_{\max}$), then the makespan can be minimized by the well-known Jackson heuristic [5]: first arranging the jobs in a non-increasing order of their delivery times and then scheduling them without leaving machine idle times, similarly as we did for the above versions. With job release times, however, the problem $1|r_j, q_j|C_{\max}$ becomes strongly NP-hard. Besides the $C_{\max}$ criterion, there are a number of other commonly used objective functions for scheduling problems. For instance, if for every job j its due date d_j is given, then several objective criteria can be used to measure the solution quality.

The *lateness* of a job j in a schedule σ:

$$L_j(\sigma) = d_j - \left(s_j(\sigma) + p_j \right) \tag{2}$$

(note that $s_j(\sigma) + p_j$ is the completion time of job j in the schedule σ). One of the most commonly used due date oriented objective functions is the maximum job lateness

$$L_{\max}(\sigma) = \max_{j \in \sigma} L_j. \tag{3}$$

The objective is to find a feasible schedule σ in which the maximum job lateness $L_{\max}$ is the minimum possible one. This problem $1|r_j|L_{\max}$ is, in fact, equivalent to the abovementioned one $1|r_j, q_j|C_{max}$ with job delivery times, and hence, it is also strongly NP-hard [7].

Another common due date-oriented objective function is the number of *late* jobs (the ones completed after their due date)

$$\sum_{j \in \sigma} U_j(\sigma), \tag{4}$$

where $U_j(\sigma)$ is a 0–1 function taking the value 1 if job j is late in the schedule σ and the value 0 otherwise. The objective here is to find a feasible schedule with the minimum possible value $\sum_{j \in \sigma} U_j(\sigma)$, equivalently, one maximizing the throughput, i.e., the number of jobs completed by their due dates (this model is motivated by applications in real-time overloaded systems, where the job due dates are crucial in a way that if a job is late, then it might rather be postponed for an undefined period of time in favor of other jobs which might be completed on time). Similarly to the above problems, if all jobs are simultaneously released, then the problem

$1||\sum U_j$ is polynomially solvable (by the algorithm of Moore and Hodgson); however, with job release times, the problem $1|r_j|\sum U_j$ is again strongly NP-hard.

Hoogeveen [6] has considered the no machine idle time version in a bi-criteria setting. Instead of minimizing the lateness, he has introduced the so-called target start time s_j of a job j: s_j is the desirable starting time for job j, similarly as the due date d_j is the desirable completion time for the job j. Together with the minimization of the maximum job lateness, the minimization of the maximum job promptness (the difference between the target and real start times of that job) can be considered. The above reference gives an algorithm that finds a Pareto-optimal set of feasible solutions for this bi-criteria scheduling problem.

4. Basic multi-criteria scheduling problems

We can combine the objective functions described in the previous section and obtain the corresponding multi-criteria scheduling problems. We consider these multi-criteria problems from the point of view of multi-threshold optimization and Pareto-optimization approaches.

We start by considering a bi-criteria problem with two objective functions, $C_{\max}$ and $L_{\max}$ obtained from the single-criterion problems $1|r_j|C_{\max}$ and $1|r_j|L_{\max}$, respectively (note that in the first problem, no job delivery times are given).

With the Pareto-optimization approach, we need to solve two relevant problems: (1) among all feasible schedules with a given maximum job lateness, find one with the minimum makespan, and (2) vice versa, among all feasible schedules with a given makespan, find one with the minimum maximum job lateness. Both of these problems are strongly NP-hard [7].

With the multi-threshold (bi-threshold) optimization approach, we are given two threshold values A^1 and A^2 on the functions $C_{\max}$ and $L_{\max}$, respectively. We would like to know if there exists a feasible schedule σ such that

$$C_{\max}(\sigma) \le A^1 \tag{5}$$

$$L_{\max}(\sigma) \le A^2. \tag{6}$$

As to condition (5), let us first construct a feasible schedule σ' in which the jobs are arranged in a non-decreasing order of their release times and are scheduled in this order without leaving unavoidable machine idle time. Recall that the schedule σ' (obtained in this way in $O(n \log n)$ time) is optimal for the problem $1|r_j|C_{\max}$. Hence, if $C_{\max}(\sigma') > A^1$, then there exists no (bi-threshold optimal) schedule σ with $C_{\max}(\sigma) \le A^1$ and we return a "no" answer. Otherwise, we know that there exists a feasible schedule σ' with $C_{\max}(\sigma) \le A^1$. In fact, if $C_{\max}(\sigma') = A^1$, then there are many such feasible schedules (we may introduce idle time intervals of a required total length in the schedule σ arbitrarily between neighboring jobs in different ways obtaining different feasible schedules satisfying inequality (5)). Let us denote the set of these feasible schedules by S_{A^1}.

Now it remains to verify condition (6), i.e., we wish to know if, among all schedules from the set S_{A^1}, there is one satisfying condition (6). In general, it may take an exponential time to answer this question for an arbitrary value A^2 since the corresponding decision problem is NP-complete. At the same time, it also might be possible to obtain an answer in polynomial time, depending on the value of A^2. The easiest way is to construct a greedy solution σ'' to the problem obtained, for instance, by the earlier mentioned Jackson heuristic. It is well-known that the

schedule σ'' minimizes the function $C_{\max}$. Hence, if $L_{\max}(\sigma'') \le A^2$, then we return the schedule σ'' with a "yes" answer. Otherwise, the answer may be "yes" or may also be "no." In this case, we need more costly calculations to seek for a feasible schedule σ from the set S_{A^1} with $L_{\max}(\sigma) \le A^2$. This may take an exponential time (as the second single-criterion problem $1|r_j|L_{\max}$ is NP-hard).

Combining the objective function $C_{\max}$ with $\sum_j U_j$, we obtain another bi-criteria problem from the single-criterion problems $1|r_j|C_{\max}$ and $1|r_j|\sum_j U_j$, respectively.

With the Pareto-optimization approach, we need to solve two relevant problems: (1) among all feasible schedules with a given maximum job lateness, find one with the minimum makespan, and (2) vice versa, among all feasible schedules with a given makespan, find one with the minimum number of late jobs. Both of these problems remain strongly NP-hard.

With the bi-threshold optimization approach, we are given two threshold values A^1 and A^3 on the functions $C_{\max}$ and $\sum_j U_j$, respectively. We would like to know if there exists a feasible schedule σ satisfying inequality (1) and the following inequality:

$$\sum_j U_j(\sigma) \le A^3. \tag{7}$$

Condition (5) can be treated as above. As to condition (7), we need to verify if, among all schedules from the set S_{A^1}, there is one satisfying this condition. As for condition (6), in general, it may take an exponential time to verify condition (7) for an arbitrary value A^3, since the corresponding decision problem with a single objective function $\sum_j U_j$ is NP-complete [8]. But it again might be possible to obtain an answer in polynomial time. Instead of Jackson's heuristic that we used for condition (6), now we use an extended version of the algorithm of Moore and Hodgson for the problem $1\|\sum U_j$. Recall that the latter algorithm is designed for simultaneously released jobs. It sorts all jobs in a non-decreasing order of their due dates and includes them in this order whenever the last included job completes by its due date. Otherwise, from the last block of the continuously scheduled jobs (there will be only one such block for simultaneously released jobs), it discards a longest job and repeats the same step until all jobs are considered in this way. Note that all the included jobs are completed on time. Finally, it adds the discarded jobs at the end of the resultant partial schedule in any order without leaving machine idle times (these jobs are late).

We modify the above algorithm by considering the jobs in the order as they are released, but order each group of currently released jobs similarly by non-decreasing due dates and accomplish the same steps for each such group of the already released jobs. Although the modified algorithm, in general, does not guarantee optimality, it may typically deliver a near-optimal solution to the version $1|r_j|\sum U_j$ with job release times. Let us denote the schedule delivered by the extended Moore and Hodgson algorithm by σ'''. It can be readily verified that the schedule σ''' minimizes the function $C_{\max}$. Hence, if $\sum_j U_j(\sigma''') \le A^2$, then we return the schedule σ''' with a "yes" answer. Otherwise, the answer may be "yes" or may also be "no." In this case, we need more costly calculations to seek for a feasible schedule σ from the set S_{A^1} with $\sum_j U_j(\sigma) \le A^2$, which, similarly as for the earlier bi-criteria problem, may take an exponential time.

Finally, combining all the three objective functions $C_{\max}$, $L_{\max}$, and $\sum_j U_j$, we obtain a more complicated three-criteria scheduling problem. Finding the Pareto-optimal set of feasible solutions obviously remains NP-hard. The

by-threshold problem gets also less accessible but still more flexible than the Pareto-optimality version, again essentially depending on the threshold values. We again consider the three conditions (5), (6), and (7) that come from the corresponding single-criterion problems and the set of feasible schedules S_{A^1} yielded by inequality (1). Using the fact that both schedules σ'' and σ''' are from the set S_{A^1}, it will suffice to verify whether

$$\sum_j U_j(\sigma'') \leq A^3 \tag{8}$$

or

$$L_{\max}(\sigma''') \leq A^2. \tag{9}$$

Intuitively, it is clear that the closer is A^3 to n (the total number of jobs) and the larger is A^2, the more probable it is that these inequalities will hold. Hence, the by-threshold problem will be solved in $O(n \log n)$ time (remind that the time complexity of all the three heuristics that we use for the creation of the schedules σ', σ'', and σ''' is $O(n \log n)$). If any of the conditions (6), (7), (8), or (9) is not satisfied, then an implicit enumeration algorithm that generates feasible schedules respecting the thresholds A^2 and A^3 can be applied.

5. Conclusions

We have seen that a multi-threshold optimization problem may solve practical multi-criteria problems in polynomial time while delivering a solution with an acceptable quality for a given threshold vector, which reflects real needs of a particular real-life application. We have compared the multi-threshold optimization problem with the Pareto-optimization problem for three basic multi-criteria scheduling problems on a single machine. It is clear that, in many multi-criteria applications, a practitioner may not be interested in a Pareto-optimal set of feasible solutions: an analysis of the set of Pareto-optimal solutions containing all non-dominated feasible solutions might be beyond the interest and capacity of the practitioner. In practice, a feasible solution that attains some threshold value for each objective function is required. For instance, take an automobile manufacturing and the three objective functions $C_{\max}$, $L_{\max}$, and $\sum_j U_j$ considered in the previous section. Clearly, the manufacturer is interested in minimizing the total production time $C_{\max}$, whereas he imposes a maximum possible lateness in the production of each car (which might be far above the minimum possible lateness), and there is a maximum admissible number of cars whose production might be late and be delayed for an infinitive amount of time (according to the current demand on the product). Two heuristic algorithms that we have considered in the previous section, in practice, may well deliver such solutions while minimizing the total production time. It is well-known that Jackson's heuristic, in practice, delivers near-optimal solutions with a value of the objective function close to the optimum [9]. At the same time, if the threshold for the criterion $\sum_j U_j$ is not too small, the solution delivered by the heuristic may also satisfy the threshold condition for that criterion. In fact, it might be possible to combine Jackson's heuristic with Moore and Hodgson's one in such a way that the resultant heuristic would provide a solution with the desired thresholds for both objective functions with some high probability. The construction of such heuristics that deliver a solution respecting the threshold vector for two or more objective criteria is an interesting line for further research.

We have illustrated the multi-threshold optimization approach on a few single-machine scheduling problems, though the approach can obviously be applied, in general, for different kinds of multi-objective optimization problems.

Author details

Nodari Vakhania[1*] and Frank Werner[2]

1 Centro de Investigación en Ciencias, UAEM, Mexico

2 Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, Germany

*Address all correspondence to: nodari@uaem.mx

References

[1] T'kindt V, Billaut J-C. Multicriteria Scheduling: Theory, Models and Algorithms. 2nd ed. Springer; 2006

[2] T'kindt V, Billaut J-C. Multicriteria scheduling: A survey, RAIRO. Operations Research. 2001;**35**:143-163

[3] Nagar A, Haddock J, Heragu S. Multiple and bicriteria scheduling: A literature review. European Journal of Operational Research. 1995;**81**(1): 88-104

[4] Parveen S, Ullah H. Review on job-shop and flow-shop scheduling using multi criteria decision making. Journal of Mechanical Engineering. 2010;**41**(2):130-146

[5] Jackson JR. Scheduling a Production Line to Minimize the Maximum Tardiness. Management Science Research Project. Los Angeles, CA: University of California; 1955

[6] Hoogeveen H. Single-machine bicriteria scheduling [Ph. D. Thesis]. Amsterdam: CWI; 1992

[7] Vakhania N. Scheduling a single machine with primary and secondary objectives. Algorithms. 2018;**11**:80. DOI: 10.3390/a11060080

[8] Garey MR, Johnson DS. Computers and Intractability: A Guide to the Theory of NP–Completeness. San Francisco: Freeman; 1979

[9] Vakhania N, Perez D, Carballo L. Theoretical expectation versus practical performance of Jackson's heuristic. Mathematical Problems in Engineering. 2015;**2015**:484671. DOI: 10.1155/2015/484671