J Sched (2013) 16:369–383
DOI 10.1007/s10951-011-0256-7

Multi-criteria scheduling: an agent-based approach for expert knowledge integration

Christian Grimme · Joachim Lepping · Uwe Schwiegelshohn

Published online: 5 October 2011

Abstract In this work, we present an agent-based approach to multi-criteria combinatorial optimization. It allows to flexibly combine elementary heuristics that may be optimal for corresponding single-criterion problems.

We optimize an instance of the scheduling problem $1|d_j|\sum C_j, L_{\max}$ and show that the modular building block architecture of our optimization model and the distribution of acting entities enables the easy integration of problem specific expert knowledge. We present a universal mutation operator for combinatorial problem encodings that allows to construct certain solution strategies, such as advantageous sorting or known optimal sequencing procedures. In this way, it becomes possible to derive more complex heuristics from atomic local heuristics that are known to solve fractions of the complete problem. We show that we can approximate both single-criterion problems such as $P_m|d_j|\sum U_j$ as well as more challenging multi-criteria scheduling problems, like $P_m||C_{\max}, \sum C_j$ and $P_m|d_j|C_{\max}, \sum C_j, \sum U_j$. The latter problems are evaluated with extensive simulations comparing the standard multi-criteria evolutionary algorithm NSGA-2 and the new agent-based model.

C. Grimme · U. Schwiegelshohn
Robotics Research Institute, TU Dortmund University,
Otto-Hahn-Strasse 8, 44227 Dortmund, Germany

C. Grimme
e-mail: christian.grimme@udo.edu

U. Schwiegelshohn
e-mail: uwe.schwiegelshohn@udo.edu

J. Lepping (✉)
ENSIMAG—antenne de Montbonnot, INRIA Rhône-Alpes,
Grenoble University, 51 avenue Jean Kuntzmann,
38330 Montbonnot Saint Martin, France
e-mail: joachim.lepping@imag.fr

Keywords Multi-criteria scheduling · Predator–prey model · Parallel machine scheduling · Evolutionary multi-criteria optimization

1 Introduction

Although theoreticians and practitioners agree on the fact that real-world scheduling optimization processes involve more than one criterion, the research area of algorithmic multi-criteria scheduling is a rather new topic posing many challenging, important, and unsolved problems. This might be because optimization in the multi-criteria domain follows a fundamentally different paradigm of optimality: while in the single-criterion case only one optimum has to be found, in the multi-criteria case optimal compromises or trade-offs have to be determined. Often, however, it is only possible to approximate the set of optimal trade-offs. For minimization problems with M criteria, an optimal compromise is reached, if for a criterion q, the value f_q can only be decreased by increasing the value f_p of some other criterion p with $q \neq p$ and $p, q \in \{1, \ldots, M\}$. All criteria vectors $\mathbf{F} = (f_1, \ldots, f_M)$ that fulfill this property are called members of the Pareto-front.

Since only a fistful of multi-criteria scheduling problems can be solved in polynomial time, many practitioners focus on the application of heuristics to approximate solution sets. Today, most problems can only be solved in two ways: On the one hand, problem specific approaches may be able to solve very specific problem instances. On the other hand, black-box optimizers like randomized search or evolutionary algorithms abstract from the specific problem and offer a universal methodology.

In contrast to multi-criteria scheduling, single-criterion scheduling problems have been extensively investigated over the last 50 years. Besides a vast amount of complexity results, researchers provided numerous optimal algorithms and approximation heuristics for many problems of that domain, see e.g., Pinedo (2009) or Dutot et al. (2010) for introduction and review. However, the most annoying fact is that it is so far almost impossible to make use of the comprehensive single-criterion knowledge base in a more generalized manner for the multi-criteria domain. It is easy to see that on the one hand a naive aggregation of single-criterion heuristics does not necessarily yield feasible non-dominated trade-off solutions. On the other hand, the monolithic architecture of popular multi-criteria evolutionary algorithms hardly allows to bring in any problem specific knowledge let alone to combine it in a flexible way. This drawback mainly results from the dominant selection procedure applied in most multi-criteria evolutionary algorithms which leaves almost no room to plug in special adaptation mechanisms. For this purpose, more advanced variation operators along with an alternative and more dynamic selection scheme is required which replaces the classic monolithic algorithmic architecture.

In this work, we propose an agent-based optimization model that offers a new way to flexibly integrate partial knowledge on specific aspects of the full problem. This model is basically inspired from the well-known predation paradigm from biology but reduced to a unidirectional interaction relation between predators and preys: a population of prey, which here represents the candidate solutions, is arbitrarily distributed on a spatial structure, normally an undirected graph. Predators pursue only *one criterion* each and move randomly along the edges to chase prey which are weak (i.e., the corresponding problem solution has a bad criterion value) regarding that specific criterion. The presence of several predators—each representing only a single criterion—is expected to force the prey to likewise adapt to the threats from all predators and thus result in suitable trade-off solutions for multi-criteria optimization problems.

The rest of the paper is organized as follows: in Sect. 2, we describe the problem context of multi-criteria scheduling problems, introduce the specific problems solved in this paper, and point out common solution strategies and complexity results. Then, in Sect. 3, we detail our alternative agent-based approach to multi-criteria scheduling. Next, in Sect. 4, we evaluate the performance of the afore defined algorithmic framework on several problems. In Sect. 5, we conclude this work and point to future directions.

2 Multi-criteria scheduling problems

In most practical scheduling scenarios and production planning tasks, the consideration of only a single criterion is insufficient to provide both quality assurance and customer satisfaction. A production scheme is usually judged with respect to several criteria considering, for instance, the overall production time (Makespan), average job flow time, job delay, or total number of late jobs.

However, for almost 30 years, scheduling theory and algorithm design were dedicated to solve single-criterion problems. Hoogeveen (2005) identifies the 1980s as a starting point for detailed investigation of multi-criteria scheduling problems and the emergence of a research area in its own rights. Since then, a lot of energy has been put to the determination of problem complexity, demonstrating the hardness of almost all problems. That took away the hope of solving those problems optimally in polynomial time. As always, this situation is the starting point for the development of heuristics which try to approximate optimal solutions in reasonable time.

In this section, we will introduce the concepts of scheduling in general, multi-criteria problems in specific, and a basic classification of solution strategies for these kinds of problem. Then, we briefly review the theoretical and heuristic research in single and parallel machine environments. Eventually, we consider solution approaches from computational intelligence and introduce their algorithmic concepts.

2.1 Basic notation and concepts

A multi-criteria scheduling problem with M criteria is also formulated using the $\alpha|\beta|\gamma$ three-field notation (Graham et al. 1979), where the γ field contains the list of criteria. We exclusively consider the enumeration problem of all Pareto-optima which is usually noted as $\#(\gamma_1, \ldots, \gamma_M)$, see T'kindt and Billaut (2006). Thus, we associate to this problem an *a posteriori* algorithm that does not use any aggregation methods but offers a set of solutions from which a decision maker can choose. For the matter of simplicity and if no confusions with other problem formulations is expected, we use the $\alpha|\beta|\gamma_1, \ldots, \gamma_M$ notation for the Pareto-enumeration problem throughout the paper. Further, we use the pure summation sign ($\sum$) as abbreviation for a sum over all jobs ($\sum_{j=1}^{n}$).

2.2 Single machine problems

The single machine scenario is obviously the simplest of all machine environments. As such, it can serve as a good starting point for providing problems that have known optimal solutions. Such are required to serve as a quality benchmark for both convergence and diversity of heuristic solutions.

Still, among the plethora of multi-criteria scheduling problems only very few are known to be solvable within polynomial time complexity. For a general discussion on complexity of multi-criteria single machine scheduling problems, see Chen and Bulfin (1993).

An example of such a polynomially solvable problem is $1|d_j|\sum C_j, L_{\max}$, which exposes Pareto-optimal solutions regarding total completion time of all jobs and maximum lateness. We further consider this problem due to two different reasons: On the one hand, the simple dispatching used for either criterion allows for the easy determination of extremal solutions in the multi-criteria search space. On the other hand, in 1980 van Wassenhoven and Gelder (1980) proposed an efficient algorithm for this problem, which is ideally suited for the matter of comparison. This algorithm bases on the feature that both extremal points can be determined separately: One can be computed by the SPT/EDD rule, i.e., minimizing $\sum C_j$ by a *shortest processing time first* (SPT) ordering (Smith 1956) with ties broken according to a downstream *earliest due dates first* (EDD) ordering (Jackson 1955). The resulting maximum lateness of this schedule is denoted by $L_{\max}$(SPT/EDD) and can be computed easily. The other extremal solution is determined using EDD as dominant approach, which yields $L_{\max}$(EDD) and is smaller or equal to $L_{\max}$(SPT/EDD) for the resulting schedules.

The algorithm now takes advantage of this by starting with an $L_{\max}$(EDD) schedule that generates the first Pareto-optimal point minimizing $\sum C_j$ as secondary criterion. From there, it determines the minimum increment in $L_{\max}$ needed to achieve a further reduction in $\sum C_j$. Then, the original and optimal $L_{\max}$(EDD) condition is relaxed by the determined value and the subroutine that further minimizes the schedule for $\sum C_j$ is called. This procedure is then repeated until the algorithm finally reaches $L_{\max}$(SPT/EDD) which marks the alternative extremal point in the front. This way, it generates all Pareto-optimal trade-off solutions in between. For a detailed description along with examples refer to T'kindt and Billaut (T'kindt and Billaut 2006). It has been shown that the maximum number of Pareto-optimal solutions is $n(n-1)/2$ which results in a complexity of $\mathcal{O}(n^2)$. As the generation of one Pareto-optimal schedule can be performed within $\mathcal{O}(n \log n)$ the overall complexity is determined by $\mathcal{O}(n^3 \log n)$.

2.3 Parallel identical machine problems

Among the multi-criteria problems on parallel machines, we first consider $P_m||C_{\max}, \sum C_j$, often abbreviated as MAXANDSUM, see Dutot et al. (2010), where n independent jobs have to be scheduled on m identical machines while the Makespan ($C_{\max} = \max_{j=1,\ldots,n}\{C_j\}$) and the total completion time $\sum C_j$ have to be minimized simultaneously. Although $P_m||\sum C_j$ is easy, the problem $P_2||C_{\max}$ is already NP-hard in the ordinary sense as it is equivalent to the PARTITION problem. According to Garey and Johnson (1978), $P_m||C_{\max}$ is strongly NP-hard for $m \geq 3$. Consequently, the multi-criteria problem consisting of NP-hard sub-problems is also NP-hard: Applying the general proof concept from Chen and Bulfin (1993) for the single machine case, we can certainly argue that a polynomial algorithm for a multi-criteria problem can also solve each sub-problem efficiently. As long as we cannot find an efficient approach for all NP-hard problems, there will be no algorithm for the multi-criteria problem that comprises some of those sub-problems.

However, Stein and Wein (1997) propose a general algorithmic framework that allows the approximation of a single schedule that minimizes two criteria at the same time. Assuming approximated schedules for both sub-problems, truncation and composition steps are executed on the respective schedules. It is supposed that the combined new schedule is still satisfactory for both criteria. In this way, one can find a solution closest to the Utopia point[1] which could be interpreted as perfect compromise (Dutot et al. 2010).

At least to the authors' knowledge there is no algorithmic approach to the three-criteria parallel machine problem $P_m|d_j|C_{\max}, \sum C_j, \sum U_j$. Nevertheless, the problem is in that sense interesting as it combines three fundamentally conflicting criteria: while $C_{\max}$ strongly focuses on utilization and favors the load-balancing among the parallel machines, $\sum C_j$ favors a higher throughput for single jobs. Additionally, $\sum U_j$ as due date related criteria brings a new focus into the problem which is fundamentally different to the processing time related criteria. Thus, it can be assumed that this problem is rather challenging and that the expected Pareto-front may have a large diversity.

Following Stein and Wein's framework idea of truncation and composition of schedules for sub-problems, we may be able to combine knowledge and solutions from sub-problems to clever compositions for the multi-criteria case. For all three considered sub-problems we have approximation algorithms or even optimal solution strategies ready at hand: $P_m||\sum C_j$ can be solved optimally using SPT while *longest processing time first* (LPT) guarantees an approximation quality of at least $\frac{4}{3} - \frac{1}{3m}$. The first rule can be proved with a simple interchange argument

[1] The Utopia point (in literature often also called the Zenith) represents a usually infeasible solution obtained by minimizing all criteria separately, see Vincent and Grantham (Vincent and Grantham 1981). It may serve as reference point for measuring the quality of potential compromise solutions.

while the second can be estimated via the smallest possible counter example, see Pinedo (2009). Finally, Süer et al. (1993) proposed SBC3 as an approach for $P_m|d_j|\sum U_j$ which is based on Moore's (1968) optimal algorithm for $1|d_j|\sum U_j$.

The current selection of heuristic approaches points again to conflicts that arise when sub-problems must be combined and solved in the multi-criteria domain. The construction of a common heuristic from SPT and LPT is obviously not easy at all. In the remainder of the paper we will guide toward a way that seems to be a promising approach.

2.4 Heuristic approaches from computational intelligence

A general scheme for optimization problems with multiple criteria is provided from the area of computational intelligence. There, various heuristics have been proposed which apply evolutionary and natural principles to multi-criteria optimization problems, see Deb (2001), or Coello et al. (2007) for a detailed review. The basic idea of most approaches is based on a simplified view on Darwinian evolution theory, where a population of individuals is exposed to (environmental) selection pressure. Under this selection only the best adapted individuals survive and are thus able to reproduce. During reproduction, variation and recombination of parental genes lead to slight deviations of the gene structure and consequently to slightly different solutions. If offspring exceed the parental qualities, new individuals will survive the next selection step as better adapted solutions (Schwefel 1995) and become able to reproduce.

Most successful algorithmic approaches use the Pareto-dominance relation to select efficient solutions during evolution: They apply non-dominated ranking and sorting to determine the reproduction candidates. As in multi-criteria optimization the Pareto-front approximation is the primary goal, algorithms must have the simultaneous ability of diversity preservation and good convergence to the optimum. Advanced algorithms like SPEA2 (Zitzler et al. 2001) and PAES (Knowles and Corne 2000) use an archive as a central component for solution conservation and offspring generation. Both rely on crowding-based measures (Deb et al. 2000) as a global component for diversity preservation: the former applies this mechanism on the population itself, while the latter uses it for archive reorganization. In contrast, indicator-based methods aggregate solution quality in a single value to enable comparison. Simple approaches aggregate via weighting; however, choosing adequate weights is an intricate process and requires adequate problem knowledge and at least a basic idea of the solution characteristics. This often leads to non-satisfactory results. More sophisticated approaches use elaborate aggregation methods. The $\mathcal{S}$-Metric Selection Algorithm (SMS-EMOA), for example, evaluates the whole Pareto-front with respect to a single value. Based on the hypervolume measure of Zitzler (1999), the algorithm selects solutions which contribute most to the overall hypervolume. Although this approach outperforms the dominance-based approaches for more than four criteria, the indicator calculation is rather expensive (Emmerich et al. 2005).

Among all these algorithm the Non-dominated Sorting Genetic Algorithm (NSGA-2) of Deb (2000) can be regarded as the most commonly used multi-criteria evolutionary algorithm and is thus also used as reference algorithm in this work. Following the traditional process of evaluation, selection, and variation the main modifications affect the selection process itself. There, the whole population is assessed regarding Pareto-dominance first: All non-dominated individuals of the entire population are assigned to a category classified at rank 1 and removed from the population. From the remaining individuals the non-dominated ones are classified in another category with rank 2. This procedure is done until all individuals are categorized. Due to a rank proportional fitness assignment, it is guaranteed that individuals in a category of rank 1 are breeding more offspring than the rest of the population. Additionally, NSGA-2 also takes the density of solutions surrounding a particular solution into account, because the average distance of two neighboring solutions for each criterion is determined. This is called *crowding distance* and used as an additional comparison operator during the selection.

It is obvious from the description above that *selection* constitutes the primary mechanism in NSGA-2 and *variation* like mutation and recombination play only a minor role. This is especially important for combinatorial problems where NSGA-2 performs best if in any generation as diverse solutions as possible generated. Thus, variation mainly acts as perturbation mechanism that gives the powerful selection mechanism of NSGA-2 the best choices.

However, it is commonly agreed that mainly variation determines the evolutionary path and efficiently strengthens the search. In a selection-focused algorithm like NSGA-2, it is—due to above mentioned reasons—almost impossible to support the evolutionary search with special variation operators as they will not have the desired influence. Unfortunately, those operators are the most convenient way to express special problem knowledge. Thus, the search of NSGA-2 can only hardly be supported by special problem knowledge. In order to deal with those weaknesses, we propose a lightweight algorithmic framework where the evolutionary concept is based on simple variation instead of complex selection. In this way, it becomes possible to plug in arbitrary operators expressing special problem knowledge.

3 The agent-based approach for multi-criteria optimization

In contrast to the monolithic structure of Pareto-dominance-based algorithms, Schwefel et al. (1998) proposed an agent-based approach motivated from natural interaction of predators and prey to specifically solve multi-criteria optimization problems. We adapt this algorithmic idea and transfer its application with several modifications to scheduling problems. After describing the basic principles of our algorithmic scheme, we specify all components needed to address scheduling problems and integrate problem knowledge via variation operators.

3.1 The essential algorithm structure

The natural principle of predators and prey interaction is considered by Schwefel et al. (1998) as abstract model for solving multi-criteria optimization problems. In that scheme, each predator represent an environmental influence, which refers to a specific single criterion. In fact, the predator component is represented by an agent that brings its single-criterion influence to a spatially distributed population of prey individuals. Each individual, however, represents a solution to the overall multi-criteria problem. While the agents are allowed to randomly move about the population, preys are—different from nature—immobile.

More detailed, the interaction environment of the model is usually represented by a toroidal grid to ensure an unrestricted motion of agents. According to Schwefel et al. (1998), this also guarantees that each prey is visited equally often by a predator on the long run. The grid is populated by both species, predator and prey. To represent solutions the latter use a certain encoding scheme, which is described in Sect. 3.1.1. There are as many prey individuals as graph nodes, representing the population. Furthermore, all prey are of equal kind and can therefore be classified as single species. On the contrary, predator individuals represent a single criterion of the multi-criteria optimization problem and randomly roam throughout the graph structure, see Fig. 1(a). After each movement step, a predator spans a selection neighborhood around its position containing surrounding prey and evaluates those individuals regarding its respective criterion, see Fig. 1(b). The worst prey is marked as candidate for replacement. Then, the remaining prey in the selection neighborhood are considered to reproduce a substitute prey for the worst one. Although the reproduction mechanism is unrestricted in the general model, we will apply mutation only, see also Sect. 3.1.2. That is, the best local individual is randomly varied by a specific mutation rule. Finally, the worst prey is replaced with the generated offspring, if its criterion value is exceeded by the offspring's fitness, see Fig. 1(c). As such, the replacement—in our specific implementation—accepts only superior solutions in a strictly elitist way. While predators roam throughout the population, their interaction with prey is locally restricted to the selection and reproduction neighborhood. As such, it can be continuously repeated for every predator in parallel.

Further, we base our approach on the generalized predator–prey model defined by Grimme et al. (2007) where predator configurations comprise not only a specific criterion but additional a whole *set* of variation operators for reproduction. We show this basic principle of a building block structure in Fig. 2. Further, the modular architecture allows to define a neighborhood function for determining the individuals that are exposed to selection or reproduction, and a walking function for the movement pattern on the spatial structure, see also Fig. 1. This approach allows us to attach various influences in a building block fashion to predators.

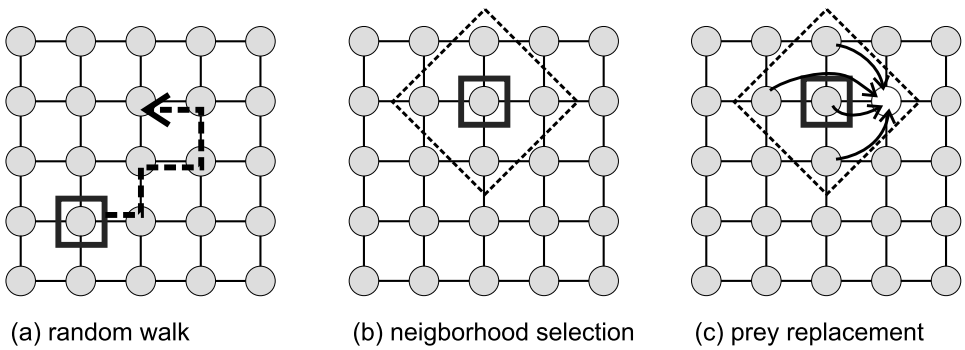


Fig. 1 Essential predator–prey algorithmic procedure: The instance exemplary shows a cut-out of the toroidal population structure that hosts prey individuals along with a single predator. The displayed movement involves two distinct walks (from positions 2, 3 to 4, 2 and from positions 4, 2 to 5, 3), each obeying the graph's vertices. Additionally, a neighborhood with a radius of 1 is shown, as well as one free vertex from which the prey has been removed. During the last step, the empty place is refilled with a new individual which is a variation of the neighboring prey

In this work, we focus on the variation operators as they are the critical building block for expert knowledge integration. In the following, we instantiate the model by discussing the problem encoding and operators.

3.1.1 Encoding of scheduling problems

Since we solve scheduling problems with n jobs, we use a standard permutation encoding of length n to represent the genotype of the problem. As we consider only offline scheduling problems, an algorithm has to find the set of optimal job permutations. The schedule construction from our chosen permutation representation is quite simple for single machine and identical parallel machine environments: the jobs are FCFS-dispatched in permutation order to the machines, see Fig. 3. Later, each criterion value can be calculated based on the resulting machine occupations.

3.1.2 Structure of the applied variation operators

Initial investigations by Grimme and Lepping (2007) show that special variation operators, triggered by autonomously acting predators, yield good approximation results especially for multi-criteria problems. For well examined scheduling sub-problems, different variation operators can be designed based on local search heuristics that assess a certain aspect of the considered multi-criteria problem.

The methodology of constructing predators from simple building blocks is founded on an analysis of variation operator influences. In former work (Grimme et al. 2007), understanding of building block behavior and the subsequent adroit combination realized by autonomous acting predators was beneficial for improving approximation results. For practical problems, the variation operators can be regarded as local search heuristics that asses a part of the considered multi-criteria problem to accelerate convergence to a fraction of the Pareto-front. They collectively and simultaneously effect the population due to their tight coupling to the parallel acting predators. Neither a crowding operator nor any other diversity preservation mechanism is used here. The diversity is only maintained by the interplay of multiple heuristics represented by predator influences.

We adapt this methodology by integrating problem specific expert knowledge into the operator design: The variation operators used for the reproduction of prey apply those strategies for single-criterion scheduling which are explained in Sects. 2.2 and 2.3. The following paragraphs describe these operators in detail.

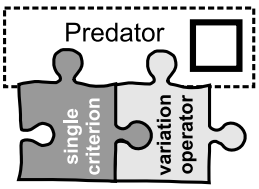


Fig. 2 The modular construction of a predator in the predator–prey model: A predator comprises a single-criterion selection building block and a variation operator building block. Only the implicit cooperation of predators with different configurations is expected to solve the multi-criteria problem

3.2 Integration of expert knowledge

For many single-criterion scheduling problems, expertise on solution strategies can be expressed by optimal sorting rules. For another set of single-criterion scheduling problems there exist either polynomial time algorithms or basic sorting rules that guarantee a certain solution quality which is expressed through an approximation factor. In the latter case, these orderings can serve as educated guess for a good solution to the problem and as a good starting point to bound the set of multi-criteria solutions.

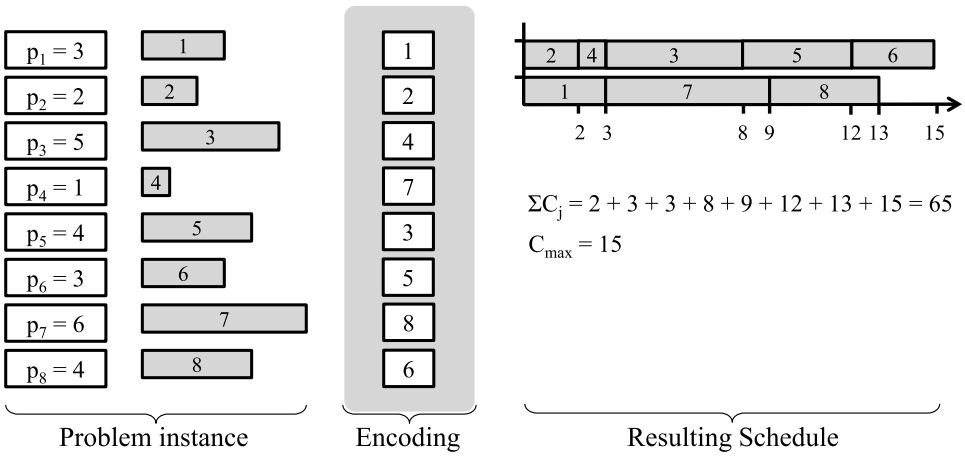


Fig. 3 Encoding based on permutation: A problem instance is encoded as sequence of jobs in an individual. Using FCFS dispatching the jobs are assigned to machines (here $m = 2$)

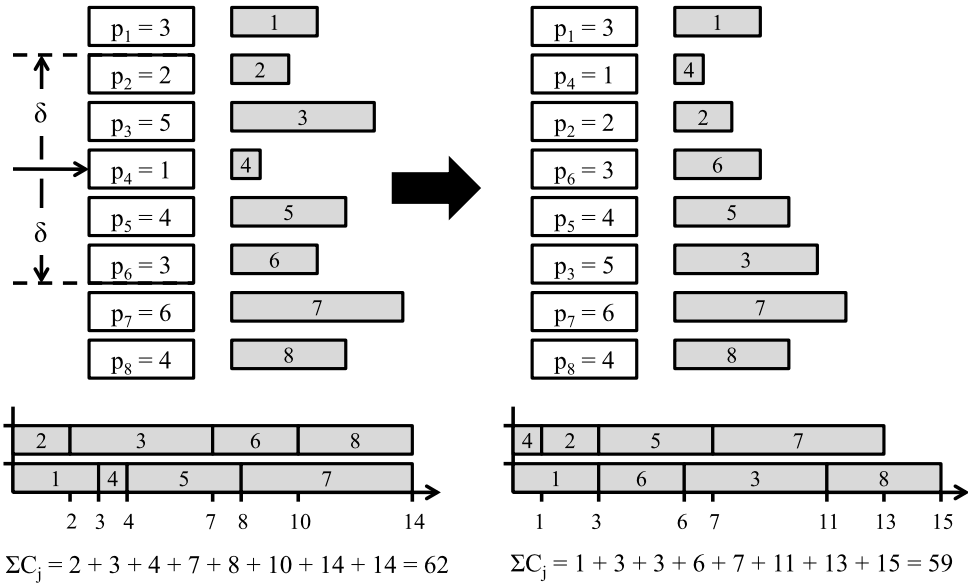


Fig. 4 Mutation operator concept with $\delta = 2$ and SPT mutation. This example does not consider any release dates ($r_j = 0$)

3.2.1 *Ordering as reasonable approximation for various single-criterion problems*

Consider the ascending order according to the known execution times of all jobs: The SPT ordering optimally solves the problems $1\|\sum C_j$ and $P_m\|\sum C_j$ both on a single and on parallel identical machines, see Pinedo (2009). Similarly, the single machine maximum lateness problem $1|d_j|L_{\max}$ is the best known special case of the more general cost function problem $1|\text{prec}|h_{\max}$. Here, the cost function is $h_j(C_j) = C_j - d_j$ and the general backward algorithm, see Pinedo (Pinedo 2009), yields a schedule that orders jobs in increasing order of their due dates for $1|d_j|L_{\max}$, i.e., *Earliest Due Date first* (EDD).

Another due date related criterion which we also consider is $\sum U_j$, the number of late jobs. In the single machine case, the problem $1|d_j|\sum U_j$ can be solved easily by applying Moore's algorithm (Moore 1968) that separates all jobs into a set of on-time and a set of late jobs. The first set has to be scheduled according to EDD to guarantee that $L_{\max}$ remains less or equal zero while the second set can be scheduled arbitrarily. The more detailed review of this approach by van den Akker and Hoogeven (2008) reveals that in fact Moore's algorithm is a dynamic algorithm with a special structure and that the EDD ordering is in any case crucial in the design of the algorithm.

For the parallel machine setting $P_m|d_j|\sum U_j$, Süer et al. (1993) developed the so called SBC3 heuristic that intuitively extends Moore's approach to the parallel machine environment. Also in their approach, jobs are sorted according to EDD first and then iteratively tested whether they become late when assigned to the machine with minimum load. If a late job occurs, the machine with minimum completion time is determined and the job is assigned to that machine. At the same time, the job with maximum processing time on that machine is removed from the schedule and marked late. This behavior is much more complex than other simple sorting rules, but EDD still serves as basic ordering. In Sect. 4.2.2 we therefore give a detailed analysis of this single-criterion sub-problem and explain how the solution strategy is modeled in the predator–prey model as specific mutation operator combination.

Finally, ordering jobs in increasing order of their processing time yields a reasonable schedule for $P_m||C_{\max}$. The *Longest Processing Time first* (LPT) rule leads to the well-known upper bound of $\frac{4}{3} - \frac{1}{3m}$ that specifies how well the heuristic is guaranteed to perform, see Graham (1969).

3.2.2 *Concept of a problem specific mutation operator*

Based on the discussion above, a general variation operator is designed which allows the integration of order-based expert knowledge. With this operator, the effect of SPT, LPT, EDD, or any other sorting schemes can be brought randomly and well-dosed into the population. Figure 4 exemplary depicts the application of this operator to a given sequence with processing times p_j: a position is selected randomly in the permutation representation of the genotype. Then, a subsequence of $2\delta + 1$ genes is sorted according to EDD, SPT, LPT, or any other rule. Here, we show the application of SPT sorting: A position is randomly selected while δ elements are involved left and right from this position. In this way, we have an easy mechanism for steering the mutation operator's strength varying the amount changed genes in the individual. However, it is reasonable to prefer

Table 1 Parameterization of the predators for the combined problem solving scenario

Predator	Criterion	Mutation	Parameter
P1	$L_{\max}$	EDD	$\sigma = 4$
P2	$\sum C_j$	EDD	$\sigma = 4$
P3	$L_{\max}$	SPT	$\sigma = 4$
P4	$\sum C_j$	SPT	$\sigma = 4$

slight changes over strong perturbations as the inner structure of individuals should be principally conserved over generations. Thus, we chose the δ-window from a normal distribution with an externally adjustable step size of σ. This value is chosen statically at the beginning of the optimization and is no further adjusted during the application. Obviously, $\delta = 0$ has no effect as only the initial gene at the current position is selected. On the other hand, a larger δ leads to a higher probability of a completely ordered genome which limits the Pareto-front regarding the respective criterion. In this example, the SPT sorting of a subsequence improves the total completion time criterion from $\sum C_j = 62$ to $\sum C_j = 59$.

3.2.3 Coupling of predators and variation operators

Due to the modular character of the considered agent-based optimization framework, predator agents and variation operators can be coupled in various ways. Two main classes of coupling can be identified: either a predator is coupled with its corresponding knowledge-based variation operator or coupled with an operator that supports another criterion. For the latter, one can consider a predator which selects regarding $\sum C_j$ either applying SPT or LPT mutation. For both cases, however, we expect a specific behavior which can be beneficially integrated into the algorithm setting.

In the first case, an operator that supports the predators criterion can be expected to favor convergence toward extremal solutions. That is, solutions rapidly converge to the criterion's optimal solution and fully neglect other criteria. A similar behavior was also observed by Grimme et al. (2007) and beneficially used to reach the outer perimeter of the desired Pareto-front.

In the second case, the selection/mutation coupling favors an implicit *lexicographical ordering* which conserves good solutions of the predator's criterion. Additionally, this configuration favors a subsequent optimization of the other criteria because the sorting is different from the actual selection. This approach can provide means to maintain good solutions while simultaneously exploring the search space regarding another criteria. For the evaluation, we consequently configure the predator–prey model with all possible mutation/selection combinations, see Table 1, to take advantage of both described effects.

A fine grained analysis of these effects and a proof of concept will be given during the following evaluation.

4 Evaluation

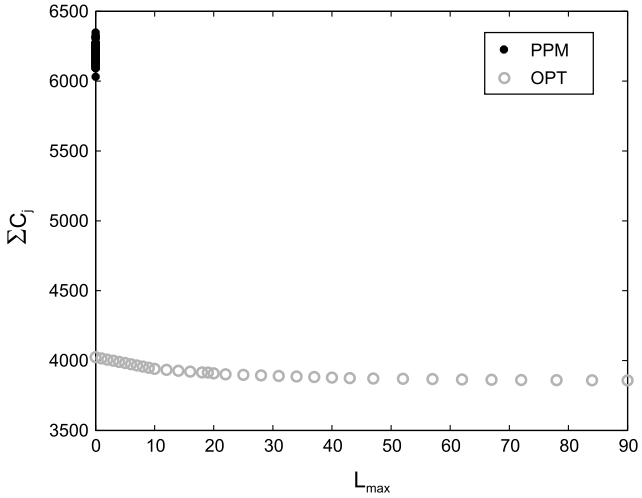
For our first test case with a single machine environment and two criteria, we generate a synthetic set of jobs $\mathcal{T}$, see Table 5. The processing times of this set was sampled using a uniform distribution as $p_j = \lfloor \mathcal{U}(1, 10) \rfloor$, $\forall j = 1, \ldots, n$. In order to guarantee that all due dates can be met, we determine correspondingly $d_j = p_j + \lfloor \mathcal{U}(1, 990) \rfloor$, $\forall j = 1, \ldots, n$. This ensures that many Pareto-optimal solutions exist as the widely distributed due dates allow for a larger variety of $L_{\max}$ values. That property was verified by the application of the polynomial algorithm, see Sect. 2.2, which results in 34 Pareto-optimal solutions that form a well distributed front; see Table 6 in the Appendix. Because we consider the single machine problem only as a first test case, we do not evaluate this setup with respect to statistical generality or robustness. If an optimal polynomial algorithm is available, the use of any evolutionary heuristic is pointless.

To analyze the performance and the behavior of our approach in detail, we generate a bunch of 100 synthetic job sets, half ($\mathcal{J}_1^{50} \ldots \mathcal{J}_{50}^{50}$) containing 50 jobs each and the other half ($\mathcal{J}_1^{100} \ldots \mathcal{J}_{50}^{100}$) comprising 100 jobs each. We sampled all sets with characteristics $p_j = \lfloor \mathcal{U}(1, 50) \rfloor$, $\forall j = 1, \ldots, n$ and $d_j = p_j + \lfloor \mathcal{U}(1, 100) \rfloor$, $\forall j = 1, \ldots, n$. For the parallel setup, we consider both 8 and 12 identical machines, i. e., in all P_m problems we set $m = 8$ and $m = 12$ leading to 200 configurations overall. The complete sets of job instances as well as the later discussed results are available via the authors' web page.[2]

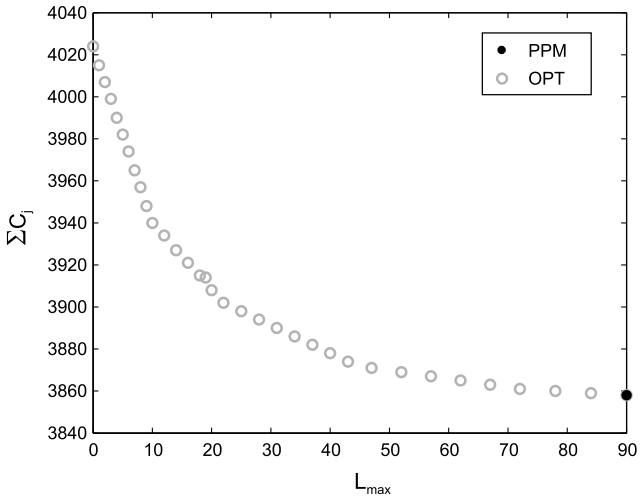
Initially, we instantiate the predator–prey model, in the rest of the paper abbreviated by "PPM", defining four fundamental building blocks constituting the runtime environment.

- The *spatial population structure* is represented by a two-dimensional toroidal grid with a size of 10×10 nodes initialized with 100 random individuals.
- The *movement* of a predator follows a uniformly distributed random walk pattern of stepping size 1, ensuring that each position is visited equally often.
- The *number of steps* for each model run is restricted to 6000 function evaluations. This value is independent of the actual number of involved predators.

[2] http://tinyurl.com/3ql3hpe.



(a) EDD mutation; selection regarding L_{max}



(b) SPT mutation; selection regarding $\sum C_j$

Fig. 5 Evaluation of the EDD and SPT mutation operators with corresponding selection. Additionally, the real Pareto-front is depicted as reference

– The *selection and reproduction neighborhood* of a predator is fixed to a radius of 1, resulting in a selection set of five prey individuals.

With this fixed setting at hand we can concentrate on the adjustments for scheduling specific problems into the general algorithmic scheme.

4.1 Evaluation of the Bi-criteria single machine problem

For the problem $1|d_j|\sum C_j, L_{\max}$, we first investigate the individual influence of our considered mutation operators and start with pure EDD mutation and the corresponding single-criterion selection followed by SPT mutation and selection regarding $\sum C_j$. As mutation width, see Sect. 3.2.2, we set $\sigma = 5$. Separated by criteria, the outcome is shown in Fig. 5.

As expected from the theoretical analysis, the EDD mutation solves the problem for the first criterion ($L_{\max}$) optimally. Therefore, when the predator selects according to $L_{\max}$, many solutions are found that reach the minimum possible criterion value of 0, see Fig. 5(a). As no subsequent SPT optimization (in a lexicographic fashion) is performed, solutions have bad $\sum C_j$ values.

The situation is different for pure SPT mutation, where we can find one optimal solution on the bottom right edge of the front ($\sum C_j = 3858$). The effect of SPT mutation seems to be stronger than the one of EDD mutation in terms of convergence, see Fig. 5(b) and it is also more robust to the selection criterion. The conclusion that can be drawn from this is ambivalent: While SPT mutation is able to favor convergence to the actual front, the population collapses into one optimal solution when the influence of the operator becomes too strong for the total completion time criterion. To realize an implicit lexicographic ordering of the criteria—which is especially advantageous to successfully solve this specific multi-criteria problem—we additionally apply both mutations together with the respective contrary criterion, see Sect. 3.2.3. The effects perceived during the isolated application of each of the operators indicate that their combination leads to a good approximation of the problem's Pareto-front. The final parametrization of each predator is shown in Table 1. We performed the tuning only for our exemplary test and did not spend much time to optimize the sigma values. It can be assumed that the sigma value has mainly an influence on the convergence speed and the achievable solution accuracy. However, the application of special parameter tuning strategies like *Sequential Parameter Optimization* (SPO), see Bartz-Beielstein et al. (2005), might be beneficial in this context. Here, we rely on rules of thumbs only.

The combined run results in the Paretofront approximation depicted in Fig. 6. It is remarkable that all identified characteristics are preserved in their combined application, resulting in a front that is almost covered as a whole compared to the optimal solution.

4.2 Evaluation of multi-criteria parallel machine problems

To evaluate our algorithmic framework on more challenging problems, we apply the agent-based approach to several parallel machine problems with two and three criteria. Additionally, we clarify how $P_m|d_j|\sum U_j$ can be approximated by the interplay of several agent. The mentioned SBC3 behavior can be modeled by the interaction of atomic sorting rules. This setup is then applied to approximate one criterion in a multi-criteria problem environment.

4.2.1 Solving the $P_m||C_{\max}, \sum C_j$ problem

As explained in Sect. 2.3, there is no specific solution algorithm to the bi-criteria problem $P_m||C_{\max}, \sum C_j$. We are

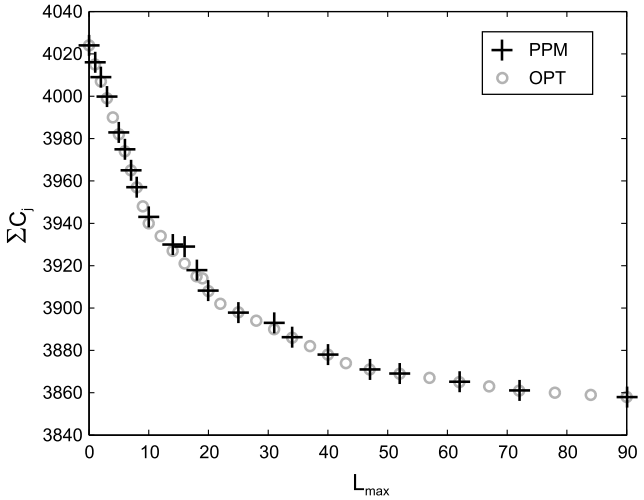


Fig. 6 Parallel application of all mutation operators in combination with two different selections ($L_{\max}, \sum C_j$) and the real Pareto-front (OPT)

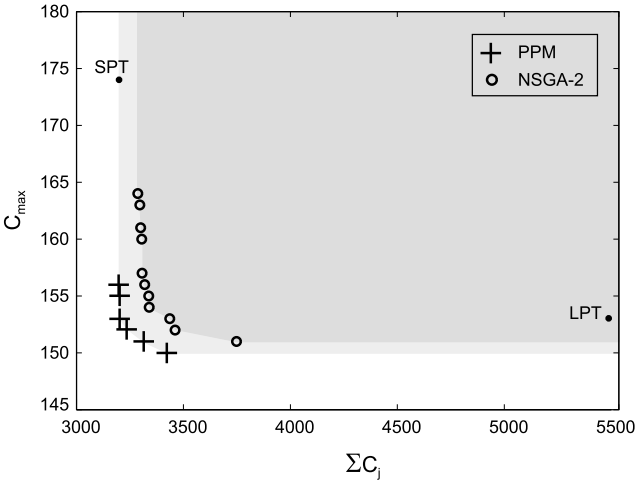


Fig. 7 Results for $\mathcal{J}_1^{50}$ with $m = 8$. *Circles*: NSGA-2; *crosses*: PPM

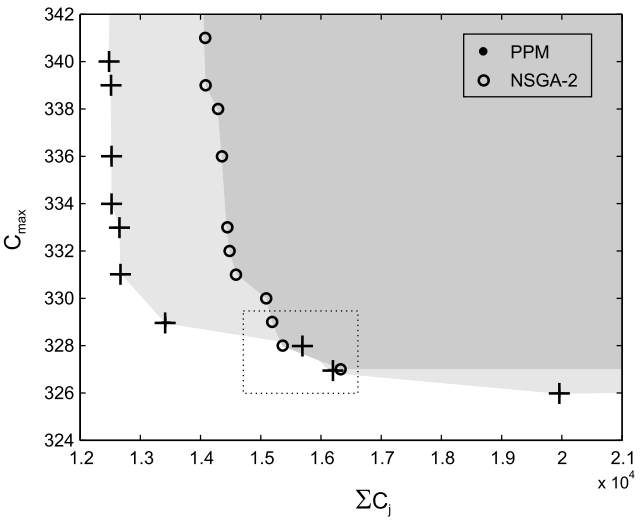


Fig. 8 Results for a specific case from $\mathcal{J}_{15}^{50}$ with $m = 8$, where PPM and NSGA results are incomparable regarding the ϵ-indicator. The area that hinders complete dominance of PPM is marked by the *dotted box*. *Circles*: NSGA-2; crosses: PPM

therefore forced to apply NSGA-2 to generate reference results. There, we are faced with the fundamental problem that NSGA-2 must be fine-tuned to achieve the best results for comparison. To this end, we performed several simulation studies and used the recommended NSGA-2 standard configuration, see Deb (2001), as well as the same special variation operators applied in the predator–prey model. However, as we expected, special operators applied in NSGA-2 did not lead to any better solutions than standard operators. Even all attempts with recombination schemes did not yield better results. Thus, we consider for comparison the best solutions achieved by NSGA-2 using a population size of 100 with exclusive swap mutation. This mutation randomly swaps eight jobs in the sequence and is applied to each individual (variation probability of 1.0). To be comparable with PPM we also restrict the number of steps per run to 6000 function evaluation.

The predator–prey model, however, uses two special tailored operators derived from the general mutation operator scheme, see Sect. 3.2.2: SPT mutation derived from the SPT rule, which solves $P_m||\sum C_j$ optimally as well as LPT mutation that approximates $P_m||C_{\max}$ and takes its cue from the earlier discussed LPT rule. We create a predator species for each criterion and connect them to both operators. Further, we discovered in preparatory experiments that LPT mutation should be emphasized over the quite efficient SPT mutation and choose therefore $\sigma = 10$ while SPT is applied with $\sigma = 5$. Regardless of the number of predators, we run the PPM for overall 6000 function evaluations per experiment and performed 50 runs of each setup; Figs. 7 and 8 show single solutions out of all runs for later explanation purposes only.

We analyzed the results of our algorithmic runs by applying two metrics: the hypervolume metric and the ε-dominance metric. The hypervolume metric computes the normalized volume in function space which is enclosed by the solution front and a reference point. The larger this value becomes, the more volume is enclosed by the determined solution set and the better is the approximation. This metric considers convergence behavior and diversity preservation at the same time. Still, some points on the approximated front may contribute more volume. The used reference points are listed in Table 2.

To provide an alternative viewpoint on our solutions, we also apply the ε-dominance metric $I_\varepsilon(A, B)$. It is a binary metric (Zitzler and Thiele 1999) which determines, whether a solution set A dominates another solution set B completely. In detail, the metric value denotes by how

Table 2 Reference points used for hypervolume computation

Problem	n	m	Reference point
$C_{\max}, \sum C_j$	50	8/12	[1000, 5000]
$C_{\max}, \sum C_j$	100	8/12	[1000, 20000]
$C_{\max}, \sum C_j, \sum U_j$	50	8/12	[1000, 7000, 50]
$C_{\max}, \sum C_j, \sum U_j$	100	8/12	[1000, 20000, 100]

much units in function space sets A or B have to be shifted in origin direction to completely dominate the respective other set. The metric has to be applied two-sided to be interpreted correctly. Only if $I_\varepsilon(A, B) > 0$ and the inverse comparison $I_\varepsilon(B, A) \leq 0$ the set A dominates set B completely. Otherwise, there are intersections of the determined solution fronts that do not allow a domination statement.

For hypervolume evaluation we find that all results of PPM enclose a significant larger volume than the results of NSGA-2. For significance analysis we apply a Wilcoxon rank-sum test ($p = 0.05$) on the determined hypervolume values for all 50 independent experiments of our 50 synthetical instances. Further, we test the variance of both algorithms to state on robustness (Fligner–Killeen test (Fligner and Killeen 1976) as most robust test for variance analysis according to Conover et al. (1981), $p = 0.05$). We find that the variance is significant smaller for the PPM than for NSGA-2 and conclude that the new approach behaves more robust on problems due to the integrated expertise knowledge. Exemplary, Fig. 7 shows both obtained Pareto-fronts (PPM and NSGA-2) and single-criterion SPT as well as LPT solutions in the criteria space. In that case, PPM clearly outperforms NSGA-2 in both convergence and diversity, although, on the first sight, one can get the impression that PPM generates a less divers front than NSGA-2. However, for the PPM the gray shaded "overall area of non-domination", i.e., the hypervolume, is larger. As PPM converges to $\sum C_j(OPT)$ value, only the minimum $C_{\max}$ value is at that point considered for the Pareto-front.

Looking at total domination with the ε-Indicator, we find the results shown in Table 3. Usually, the PPM completely dominates the NSGA-2 results, while NSGA-2 never dominates the PPM. However, there are some cases in which results are marked as incomparable, distinctively observable in the 8-machine scenario. Figure 8 shows this effect visually: a single solution of NSGA-2 dominates a solution of the PPM. Although the hypervolume is larger for the PPM solution, not all solutions of NSGA are dominated. Interestingly, this effect vanishes for larger setups with 12 machines. We hypothesize that this is due to the size of the solution set which may become smaller for multiple machine

Table 3 Results of the ε-Indicator evaluation, which judges on complete domination between two result sets. The columns show how often (in %) the results of PPM dominate the results of NSGA-2 and vice versa over all instances and for different machine configurations. The operator $A \rhd_\varepsilon B$ denotes the percentaged domination count of A over B with respect to ε-dominance

Job sets	m	PPM $\rhd_\varepsilon$ NSGA-2		NSGA-2 $\rhd_\varepsilon$ PPM
		% (mean)	std.	%
$\mathcal{J}_1^{50} \ldots \mathcal{J}_{50}^{50}$	8	72.76	16.31	0
$\mathcal{J}_1^{50} \ldots \mathcal{J}_{50}^{50}$	12	91.66	6.56	0
$\mathcal{J}_1^{100} \ldots \mathcal{J}_{50}^{100}$	8	56.54	12.32	0
$\mathcal{J}_1^{100} \ldots \mathcal{J}_{50}^{100}$	12	100	0.02	0

setups. Results from the real-valued problem domain have already proven the PPM's tendency to converge more in extremal regions of the Pareto-set (Grimme et al. 2007). Although diversity is still preserved due to the lexicographic combination of influences via selection and specific variation operators, see Sect. 3.2.3, the effect of diversity loss is also observed here and certainly has to be addressed in future work.

Before we present the PPM application to a three-criteria problem, we explain how the single-criterion total number of tardy jobs problem is approximated.

4.2.2 A preliminary study on $P_m|d_j|\sum U_j$

While the approximation of the total completion time $\sum C_j$ and Makespan $C_{\max}$ is comparatively easy, the total number of late jobs $\sum U_j$ is more challenging, as no atomic sorting heuristic is immediately applicable. As mentioned in Sect. 3.2.1, the SBC3 heuristic of Süer et al. transfers Moore's algorithm to the parallel machine environment. In general, we assume that EDD serves as basic principal while more detailed insights are actually required. Thus, we performed several experiments with different combinations of random swap mutation, EDD mutation, and SPT mutation. In Fig. 9, we depict the best results for an instance of $P_8|d_j|\sum U_j$ and several operator combinations. The *Random Swap Mutation* (RD) uniformly swaps two genes in the individual encoding and serves as adaptation of the standard mutation to a permutation encoding. The performance with pure RD mutation is relatively poor as no problem specific knowledge is provided and the algorithm has to rely on simple random search.

Recall that in SBC3, whenever a job misses its due date, the largest currently scheduled job is shifted toward the end of the sequence and marked late. Thus, the SBC3 behavior can be modeled by a combination of multiple atomic sorting rules, e.g., EDD and SPT while EDD is the general sorting rule and SPT heuristically produced subsequences where larger jobs are scheduled more to the end.

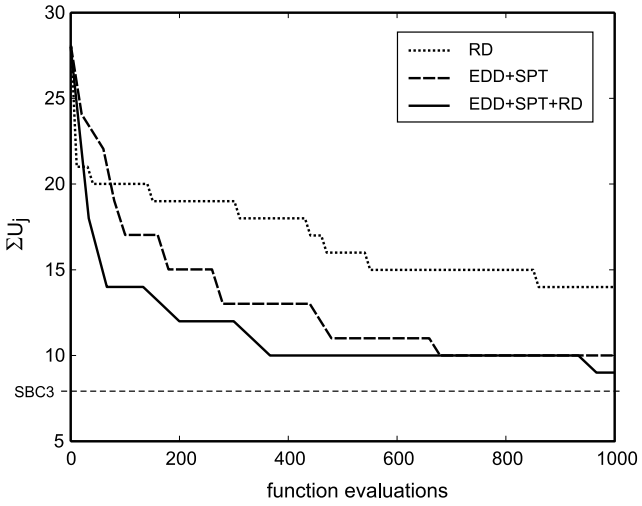


Fig. 9 Exemplary best results for $P_8|d_j|\sum U_j$ with several operator combinations and the result of the SBC3 heuristic ($\sum U_j = 8$). First, only a random mutation operator is applied (RD). Second, EDD and SPT favoring operators are applied (EDD + SPT). Finally, all operators are combined (EDD + SPT + RD)

Thus, we apply a combination of EDD + SPT and obtain much faster and better convergence than with exclusive RD mutation. Further, we discover that the performance of the PPM can be even more improved if some random influence is added. In this way, the PPM yields almost as good results as the SBC3 heuristic and we assume that the EDD + SPT combination is reasonable to approximate the $\sum U_j$ part of a corresponding multi-criteria problem.

4.2.3 *Solving the* $P_m|d_j|C_{\max}, \sum C_j, \sum U_j$ *problem*

With these findings at hand, we can now address the multi-criteria problem $P_m|d_j|C_{\max}, \sum C_j, \sum U_j$ with three contradictory criteria. We apply the same configuration concept as in the two-criteria case but use—as described above—additionally EDD mutations (with $\sigma = 5$) as they are supposed to lead solutions to the $\sum U_j$ criterion (in combination with the already available SPT and LPT mutations). Explicitly stated, we extend the predator configuration by adding a third species which selects regarding $\sum U_j$. Out of this species we generate three predators and attach EDD, SPT and LPT mutation to each. Finally, we instantiate two further predators, one from the species that selects regarding $\sum C_j$ and another one that selects regarding $C_{\max}$. Both agents are then equipped with an EDD mutation operator. This is a basic and straightforward scaling of the agents to the additional problem criterion.

As in the two-criteria case, we use our 200 problem configurations with 50 and 100 jobs as well as 8 and 12 machines for evaluating both the PPM and NSGA-2. Each configuration is evaluated 50 times to get statistically significant results. Further, we evaluate the performance of both approaches with the hypervolume metric and the ε-dominance indicator. Also in the three-criteria case PPM outperforms NSGA-2 significantly (Wilcoxon rank-sum test, $p = 0.05$) regarding hypervolume. In all 200 instances the hypervolume of PPM is significantly larger than that of the corresponding NSGA-2 solution sets. Thus, PPM is superior in convergence and diversity performance. The same holds for the evaluation of ϵ-dominance, see Table 4. In almost all configurations, the NSGA-2 results are completely dominated by PPM solutions. Only in very few cases no statement on full domination can be provided. However, none of the NSGA-2 results dominates PPM. Interestingly, for all instances with 100 jobs and 12 machines, PPM outperforms NSGA-2 by far leading to a domination count of 100%. An analysis regarding hypervolume shows that the difference in normalized hypervolume of PPM and NSGA-2 is always larger than 0.5, which indicates a loss in convergence and diversity for NSGA-2.

Table 4 Results of the ε-Indicator evaluation for the three-criteria case. The columns show how often (in %) the results of PPM dominate the results of NSGA-2 and vice versa over all instances and for different machine configurations

Job sets	m	PPM $\triangleright_\varepsilon$ NSGA-2		NSGA-2 $\triangleright_\varepsilon$ PPM
		% (mean)	std.	%
$\mathcal{J}_1^{50} \ldots \mathcal{J}_{50}^{50}$	8	99.36	1.79	0
$\mathcal{J}_1^{50} \ldots \mathcal{J}_{50}^{50}$	12	97.51	4.03	0
$\mathcal{J}_1^{100} \ldots \mathcal{J}_{50}^{100}$	8	99.89	0.38	0
$\mathcal{J}_1^{100} \ldots \mathcal{J}_{50}^{100}$	12	100	0.00	0

An even more detailed analysis of these findings shows a problem with convergence in all criteria, most severe for $\sum C_j$ and $\sum U_j$. For many configurations, PPM is able to generate solutions with no late jobs, while NSGA-2 stagnates at about half of all available jobs being late. Similarly, NSGA-2 is unable to approximate $\sum C_j$ as good as NSGA-2. We can explain this effect in a twofold way: First, NSGA-2 is known to expose convergence problems for multiple criteria (Knowles and Corne 2007). Due to its construction—which is based on non-dominated sorting and a diversity preserving mechanism—the algorithm tends to predominantly preserve diversity, if many solutions reside in the same front. Second, PPM enforces each criterion and as such favors convergence. Of course, the integrated heuristic methods mainly contribute to the rapid finding of better solutions, as shown in our preliminary experiments in Sect. 4.2.2. Here, we can most obviously show the advantage of integrating heuristic knowledge.

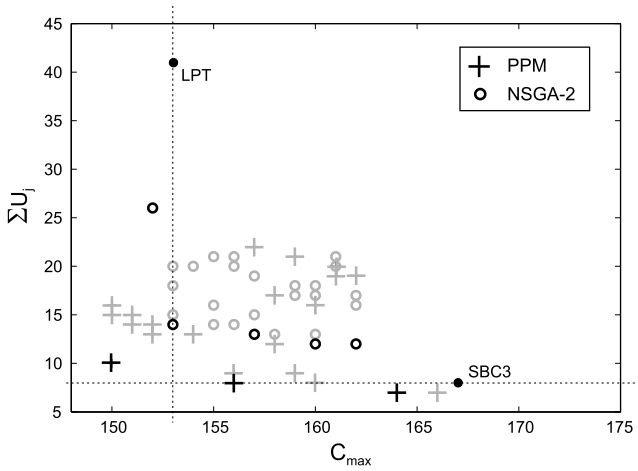
Finally, we exemplary show in Fig. 10 the different projections of a single Pareto-front approximation and corresponding heuristic results. In these projections it becomes obvious that PPM outperforms NSGA-2 for convergence and diversity. Figures 10(b) and (c) prove that even for this problem the optimal SPT solution is found by PPM while NSGA-2 fails. Further, PPM can find even better solutions than results of SBC3 and LPT heuristics; see Fig. 10(a). Viewing the results of hypervolume, ϵ-dominance together we certainly conclude that PPM is a powerful concept for solving scheduling problems with even more than two criteria.

5 Conclusion

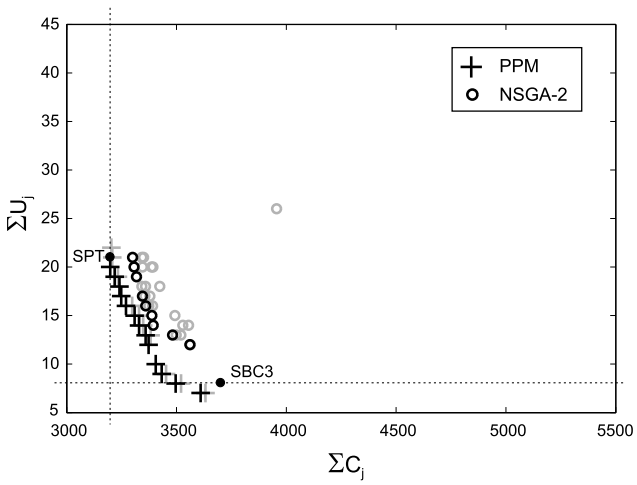
In this work, we presented an agent-based framework for solving multi-criteria scheduling problems. The lightweight design of the approach features the flexible combination of single-criterion solution heuristics to solve complex multi-criteria problems. It provides a decoupled and easy-to-realize randomized strategy which allows to seamlessly integrate theoretically founded results and rules from single-criterion scheduling. In this way, it becomes possible to use problem specific expert knowledge to cope with hard-to-solve problems. More specific, we can express problem specific single-criterion knowledge by corresponding variation operators. The isolated analysis of those operators shows that we can reliably cover certain regions of the Pareto-front. Thus, it is even possible to integrate both extremal convergence behavior and lexicographic optimization. The simultaneous and parallel realization of such effects on the population yields a good set of trade-off solutions.

We exemplary studied the behavior of our agent-based framework on three test problems. For the easy single machine problem $1|d_j|\sum C_j, L_{\max}$ the algorithm proves to be principally able to reach the optimal Pareto-front. Here, the successful combination of well-known solution strategies like SPT and EDD via specially designed operators is demonstrated. Further, the approach also proves to solve hard problems with multiple criteria. For both scheduling problems $P_m||C_{\max}, \sum C_j$ and $P_m|d_j|C_{\max}, \sum C_j, \sum U_j$ the combination of SPT, LPT, and EDD is advantageous in the randomized scheme, outperforming modern multi-criteria evolutionary search algorithms like NSGA-2.

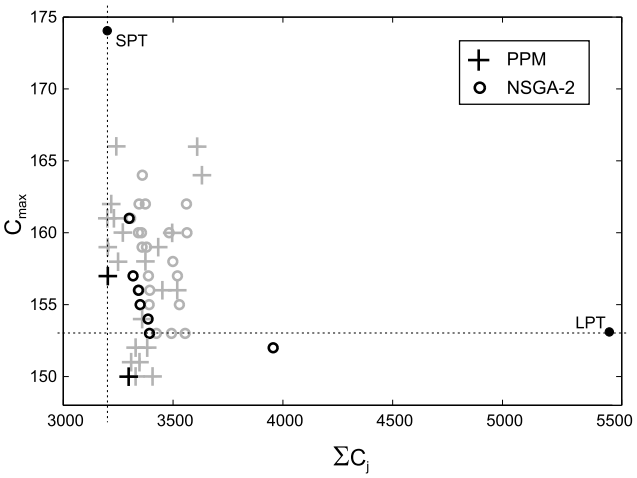
Acknowledgements The authors would like to thank M. Bender, J. Blazewicz, E. Pesch, D. Trystram, and G. Zhang for the organization of the 2010 "New Challenges in Scheduling Theory" workshop in Frejus, France. Further, the authors thank J. J. Durillo, A. J. Nebro, and E. Alba for providing their jMetal framework (Durillo et al. 2010) and A. Seshadri for his NSGA-2 implementation in MATLAB.



(a) Projection regarding $C_{max}, \sum U_j$



(b) Projection regarding $\sum C_j, \sum U_j$



(c) Projection regarding $\sum C_j, C_{max}$

Fig. 10 Results for $m = 8$ with three criteria. *Circles*: NSGA-2; *crosses*: PPM

Appendix 1: Job set $\mathcal{T}$ and solutions for $1|d_j|\sum Cj, L_{\max}$

Table 5 Properties of $\mathcal{T}$ as problem instance for $1|d_j|L_{\max}, \sum Cj$

j	p_j	d_j	j	p_j	d_j	j	p_j	d_j	j	p_j	d_j
1	10	793	14	5	170	27	3	526	40	7	100
2	1	827	15	1	158	28	7	859	41	10	645
3	1	50	16	4	522	29	2	571	42	10	776
4	1	484	17	9	533	30	8	393	43	5	190
5	3	513	18	3	829	31	2	610	44	7	10
6	6	69	19	1	167	32	2	156	45	9	530
7	3	824	20	3	928	33	3	632	46	10	362
8	1	951	21	5	292	34	1	357	47	7	487
9	7	537	22	8	391	35	2	849	48	1	800
10	1	90	23	2	151	36	3	785	49	8	466
11	1	968	24	9	302	37	6	192	50	5	749
12	6	702	25	5	279	38	2	678			
13	3	4	26	10	375	39	8	260			

Table 6 Maximum lateness ($L_{\max}$) and total completion time ($\sum C_j$) of all 34 Pareto-optimal solutions for job set $\mathcal{T}$

j	$L_{\max}$	$\sum C_j$	j	$L_{\max}$	$\sum C_j$	j	$L_{\max}$	$\sum C_j$	j	$L_{\max}$	$\sum C_j$
1	0	4024	10	9	3948	19	25	3898	28	57	3867
2	1	4015	11	10	3940	20	28	3894	29	62	3865
3	2	4007	12	12	3934	21	31	3890	30	67	3863
4	3	3999	13	14	3927	22	34	3886	31	72	3861
5	4	3990	14	16	3921	23	37	3882	32	78	3860
6	5	3982	15	18	3915	24	40	3878	33	84	3859
7	6	3974	16	19	3914	25	43	3874	34	90	3858
8	7	3965	17	20	3908	26	47	3871			
9	8	3957	18	22	3902	27	52	3869			

References

Bartz-Beielstein, T., Lasarczyk, C. W. G., & Preuss, M. (2005). Sequential parameter optimization. In *IEEE congress on evolutionary computation* (Vol. 1, pp. 773–780). New York: IEEE Press.

Chen, C. L., & Bulfin, R. L. (1993). Complexity of single machine multi-criteria scheduling problems. *European Journal of Operational Research*, *70*, 115–125.

Coello, C. A. C., Lamont, G. B., & Veldhuizen, D. A. V. (2007). *Evolutionary algorithms for solving multi-objective problems*. *Genetic and evolutionary computation* (2nd ed.). New York: Springer.

Conover, W. J., Johnson, M. E., & Johnson, M. M. (1981). A comparative study of tests for homogeneity of variances, with applications to the outer continental shelf bidding data. *Technometrics*, *4*(23), 351–361.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. *Wiley-interscience series in systems and optimization* (1st ed.). New York: Wiley.

Deb, K., Agrawal, S., Pratab, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In M. Schoenauer et al. (Eds.), *Lecture notes in computer science: Vol. 1917. Proceedings of the conference on parallel problem solving from nature* (pp. 849–858). Berlin: Springer.

Durillo, J., Nebro, A., & Alba, E. (2010). The jMetal framework for multi-objective optimization: design and architecture. In *IEEE congress on evolutionary computation* (Vol. 5467, pp. 4138–4325). Barcelona, Spain. Berlin: Springer.

Dutot, P. F., Rzadca, K., Saule, E., & Trystram, D. (2010). Multi-objective scheduling. In *Introduction to scheduling*, (1st ed.). (pp. 219–251). Boca Raton: CRC Press.

Emmerich, M., Beume, N., & Naujoks, B. (2005). An EMO algorithm using the hypervolume measure as selection criterion. In *Proceedings of the international conference on evolutionary multi-criterion optimization* (pp. 62–76).

Fligner, M. A., & Killeen, T. J. (1976). Distribution-free two-sample tests for scale. *Journal of the American Statistical Association*, *71*(353), 210–213.

Garey, M. R., & Johnson, D. S. (1978). "Strong" NP-completeness results: motivation, examples, and implications. *Journal of the ACM*, *25*(3), 499–508.

Graham, R. L. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, *17*, 416–429.

Graham, R. L., Lawer, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, *5*, 287–326.

Grimme, C., & Lepping, J. (2007). Designing multi-objective variation operators using a predator–prey approach. In *Lecture notes in computer science: Vol. 4403. Proceedings of the international conference on evolutionary multi-criterion optimization* (pp. 21–35). Berlin: Springer.

Grimme, C., Lepping, J., & Papaspyrou, A. (2007). Exploring the behavior of building blocks for multi-objective variation operator design using predator–prey dynamics. In D. Thierens et al. (Eds.), *Proceedings of the genetic and evolutionary computation conference* (pp. 805–812). New York: ACM.

Hoogeveen, H. (2005). Multicriteria scheduling. *European Journal of Operational Research*, *167*(3), 592–623.

Jackson, J. R. (1955). *Scheduling a production line to minimize maximum tardiness* (Management Science Research Project, Research Report 43), University of California, Los Angeles.

Knowles, J., & Corne, D. (2000). Approximating the nondominated front using the Pareto archived evolution strategies. *Evolutionary Computation*, *8*(2), 149–172.

Knowles, J., & Corne, D. (2007). Quantifying the effects of objective space dimension in evolutionary multiobjective optimization. In *Proceedings of the 4th international conference on evolutionary multi-criterion optimization* (pp. 757–771). Berlin: Springer.

Laumanns, M., Rudolph, G., & Schwefel, H. P. (1998). A spatial predator–prey approach to multi-objective optimization: a preliminary study. In T. Bäck et al. (Eds.), *Proceedings of the conference on parallel problem solving from nature* (pp. 241–249). Berlin: Springer.

Moore, J. M. (1968). An *n* job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science*, *15*, 102–109.

Pinedo, M. (2009). *Scheduling: theory, algorithms, and systems* (3rd ed.). Berlin: Springer.

Schwefel, H. P. (1995). *Evolution and optimum seeking* (1st ed.). New York: Wiley.

Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, *3*, 59–66.

Stein, C., & Wein, J. (1997). On the existence of schedules that are near-optimal for both makespan and total weighted completion time. *Operations Research Letters*, *21*, 115–122.

Süer, G. A., Báez, E., & Czajkiewicz, Z. (1993). Minimizing the number of tardy jobs in identical machine scheduling. *Computers & Industrial Engineering*, *25*(1–4), 243–246.

T'kindt, V., & Billaut, J. C. (2006). *Multicriteria scheduling. Theory, models and algorithms* (2nd ed.). Berlin: Springer.

van den Akker, M., & Hoogeveen, H. (2008). Minimizing the number of late jobs in a stochastic setting using a chance constraint. *Journal of Scheduling*, *11*(1), 59–69.

van Wassenhove, L. N., & Gelders, F. (1980). Solving a bicriterion scheduling problem. *European Journal of Operational Research*, *2*(4), 281–290.

Vincent, T. L., & Grantham, W. J. (1981). *Optimality in parametric systems* (1st ed.). New York: Wiley.

Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: methods and applications*. Ph.D. thesis, ETH Zürich.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, *3*(4), 257–271.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). *SPEA2: Improving the strength Pareto evolutionary algorithm* (Technical Report 103). Computer Engineering and Communication Networks Lab (TIK), ETH Zürich.