An Extended Mutation-Based Priority-Rule Integration Concept for Multi-Objective Machine Scheduling

Jakob Bossek and Christian Grimme
University of Münster, Department of Information Systems
48149 Münster, Germany
Email: {bossek, christian.grimme}@uni-muenster.de

***Abstract*—There exist many optimal or heuristic priority rules for machine scheduling problems, which can easily be integrated into single-objective evolutionary algorithms via mutation operators. However, in the multi-objective case, simultaneously applying different priorities for different objectives may cause severe disruptions in the genome and may lead to inferior solutions. In this paper, we combine an existing mutation operator concept with new insights from detailed observation of the structure of solutions for multi-objective machine scheduling problems. This allows the comprehensive integration of priority rules to produce better Pareto-front approximations. We evaluate the extended operator concept compared to standard swap mutation and the stand-alone components of our hybrid scheme, which performs best in all evaluated cases.**

I. Introduction

Scheduling—the assignment of activities or jobs to limited resources over time—is an important and widely discussed topic in operations research. However, there is still a gap between problem-solving capabilities in single-objective and multi-objective scheduling. That gap is mainly rooted in the difficulties of transferring the theoretically well founded scheduling approaches from the single-objective domain to the multi-objective domain. Although theoreticians and practitioners agree on the importance of considering multiple objectives in real-world scheduling problems, theoretically founded approaches for multi-objective scheduling are scarce.

Unlike multi-objective scheduling, single-objective scheduling problems have been extensively investigated over the last 50 years. Research provides a huge amount of complexity results, optimal algorithms, and approximative heuristics (see, e. g., Pinedo [1] or Dutot et al. [2] for an introduction). What is most annoying: up to now, it seems almost impossible to make use of these results in an universal manner for the multi-objective domain [3].

The more general notion of optimality for multiple objectives is certainly one of the main reasons: For minimization problems with m objectives, an optimal compromise is reached, if for an objective γ_i, the value f_{γ_i} can only be decreased by increasing another objective value f_{γ_j} with $\gamma_i \neq \gamma_j$ and $i, j \in \{1, \ldots, m\}$. All objective vectors $F = (f_{\gamma_1}, \ldots, f_{\gamma_m})$ that fulfill this property are called members of the Pareto-front [4].

Often applied priority rules in the single-objective scheduling domain produce good solutions for a single objective. For multiple objectives, priority rules are usually as contradicting as the objectives themselves. This problem does not vanish, when applying meta heuristics like evolutionary algorithms for solving the multi-objective problems. In fact, the problem appears again, when trying to hybridize the respective evolutionary algorithm, i. e., when we include the single-objective expertise into some search mechanism of the general heuristic framework (e. g., inclusion into the variation operators).

In this paper, we address these challenges and propose a general mutation operator concept for effectively integrating single-objective priority rules for scheduling into multi-objective evolutionary search. This concept is based on a first proposal of Grimme et al. [3] and extended towards a general concept. Therefore, we include insights, which are gained by an analysis of optimal solutions for a special scheduling problem. After an introduction on multi-objective scheduling and common priority rules in section II, we analyze the structure of optimal multi-objective schedules in section III and derive the proposed extension of the current concept from these insights in section IV. In section V we analyze the applicability of our concept before concluding the paper in section VI.

II. Priority rules and Multi-Objective Scheduling

A multi-objective scheduling problem comprising m objectives can be formulated using the three-field-notation

$$\alpha|\beta|\gamma,$$

introduced by Graham et al. [5], where the α field contains the machine environment, the β field holds a list of constraints, and the γ field contains a list of m unprioritized objectives $\gamma_1, \ldots, \gamma_m$. Here we consider $\alpha = 1$ for single machine and $\alpha = P_m$ for parallel identical machine environments. Additionally, we consider release dates $r_j \in \mathbb{N}_0$ for a job $j \in I = \{1, \ldots, n\} \subset \mathbb{N}_{>0}$ as the only constraint. For each job j, we further denote the processing time as $p_j \in \mathbb{N}_{>0}$ and the due date as $d_j \in \mathbb{N}_{>0}$. For our study and operator design approach, we consider a subset of the common scheduling objectives, namely: (1) $\sum C_j = \sum_I C_j$ as sum of completion times C_j; (2) $L_{\max} = \max\{L_1, \ldots, L_n\}$ as maximum lateness

978-1-5386-2726-6/17/$31.00 ©2017 IEEE

with lateness defined as $L_j = C_j - d_j$; and (3) $\sum U_j = \sum_I U_j$ as number of tardy jobs with $U_j = 1$, if job j is late and $U_j = 0$ otherwise.

A. Single-Objective Priority Rules

In a nutshell, the general scheduling task is to find a sequence of jobs according to some priority rule before dispatching them on machines. Depending on the machine environment and objective under consideration, priority rules are often rather straight forward. The problem $1||\sum C_j$ is solved to optimality with the *shortest processing time first* (SPT) rule [6] while the $1||L_{\max}$ problem is optimally solved using the *earliest due date first* (EDD) ordering [7]. The latter is also the foundation for Moore's algorithm [8] for $1||\sum U_j$. The one machine problem $1|r_j|L_{\max}$ is $\mathcal{NP}$-complete. Here, EDD ordering serves as a simple heuristic without any performance guaranties. In the parallel machine environment SPT also solves the problem $P_m||\sum C_j$ to optimality [1]. The problems $P_m||L_{\max}$ and $P_m||U_j$ are $\mathcal{NP}$-hard. For the latter Süer et al. [9] proposed a good heuristic to determine job priority based on Moore's approach for the one machine case. A comprehensive collection of priority rules can be found at [10].

B. Multi-Objective Scheduling

Multi-objective scheduling problems can easily be defined by adding some objectives to the γ field of the above notation. However, the theoretical and heuristic results for solving these problems are rather scarce. In fact, only very few problems are known to be optimally solvable in polynomial time.

One such polynomially solvable problem is $1||\sum C_j, L_{\max}$, considering total completion time of all jobs and maximum lateness on a single machine simultaneously. Van Wassenhoven and Gelder [11] proposed an efficient algorithm for this problem, which considers the single-objective results for $1||\sum C_j$ and $1||L_{\max}$ as integral building blocks of the procedure.

In a first step, the so-called SPT/EDD-solution (i. e., the SPT rule is applied, breaking ties with the EDD rule) is computed as a stopping criterion for the algorithm. Then the algorithm mainly follows the EDD-rule with step-wise relaxation:

1) Determine $L = L_{\max}(\text{EDD})$, which is obviously smaller or equal to $L_{\max}(\text{SPT/EDD})$.
2) Redefine the current scheduling problem by adding L to the current due dates. Under the new problem, no job violates the due dates anymore. Optimize the schedule according to $\sum C_j$ as secondary criterion without violating a due date. This yields the Pareto-optimal point.
3) Determine the minimum increase L' of the current due dates to achieve a better value of $\sum C_j$.
4) Set $L = L'$. Go to step (2) unless the algorithm reaches $L_{\max}(\text{SPT/EDD})$, the stopping criterion.

This procedure generates all Pareto-optimal solutions for the problem with overall time complexity of $\mathcal{O}(n^3 \log n)$. Pinedo [1] provides a detailed description of the algorithm and an example.

The most interesting aspect of this algorithm is, that it allows insight into the generation of Pareto-optimal solutions for scheduling problems. We can observe, how the solution for a single objective is transformed to intermediate trade-off solutions and how the job sequence is changed during this process. Thus, we use this algorithm as a starting point for our analysis in section III.

C. Heuristic Approaches and Computational Intelligence

Apart from the few deterministic approaches, scheduling literature mainly provides discussion on the complexity of multi-objective problems as well as simple proofs of $\mathcal{NP}$-hardness (see, e. g., Chen and Bulfin [12] or Hoogeveen [13]). Thus, in practice, most problem solving approaches rely on general but heuristic schemes like multi-objective algorithms.

The basic idea of these algorithms stems from an abstract model of the Darwinian evolution theory, where a population of individuals is exposed to (environmental) selection pressure. Under this selection only the best adapted individuals survive and, thus, are able to reproduce. During the reproduction process, variation, and recombination of parental individuals (i. e., solutions) lead to slight deviations of the genetic structure and often to slightly different solutions. If an offspring's quality exceeds the parental quality, the offspring will more likely survive the selection process [14] and becomes able to reproduce.

Many successful approaches for evolutionary multi-objective optimization use the Pareto-dominance relation as primary selection mechanism supported by a secondary, diversity preserving mechanism (e. g., NSGA-II [15], SPEA2 [16], SMS-EMOA [17]). Apart from these mechanisms, most evolutionary multi-objective algorithms follow the standard evolutionary loop. Considering the case of integer or permutation encoded problems, solutions are varied (as in a single-objective evolutionary algorithm) with standard variation operators [18]. Thus, as in the single-objective case, variation operators provide the direct interface for generating problem-specific solutions in search space. On the other hand, Grimme et al. [3] argue, that integrating single-objective scheduling priority rules into these components is not trivial. As a consequence, the authors propose a decoupled predation-inspired approach to include multiple single-objective variation influences. Additionally, they propose a mutation operator for including scheduling preference rules, by applying then to small blocks in the schedule permutation encoding. Recently, Pereira et al. [19] adopted this approach for the unrelated machine model in the scheduling domain. In contrast, Lang and Grimme [20] propose a general framework for expert knowledge integration into preference-based evolutionary multi-objective algorithms like R-NSGA-II [21] and MOEA/D [22] and show the flexibility of this approach, also in the context of scheduling.

In this work, we address priority-rule integration into any general standard algorithm. As we are only interested in the variation operator as vehicle for this integration, the surrounding evolutionary meta-heuristic is of no importance. Thus, we rely on the probably most commonly used multi-objective

evolutionary algorithm, NSGA-II. Therein, during selection, the population is evaluated regarding Pareto-dominance first: All non-dominated individuals of the entire population are assigned rank 1 and removed from the population. From the remaining individuals the non-dominated ones are classified in rank 2. This procedure is repeated until all individuals are ranked. A rank-proportional fitness assignment guarantees, that individuals in rank 1 reproduce more often than individuals in rank 2 and so on. NSGA-II also takes the density of solutions around a particular solution into account. This secondary selection criterion is called *crowding distance* and applied as comparison operator during dominance selection in order to break ties.

III. Structures of Optimal Solutions

As stated in section II-B we strive for the integration of priority rules into variation operators of an evolutionary multi-objective algorithm. Therefore, we analyze the structure of Pareto-optimal solutions in search space, consisting of permutation strings of jobs. We specifically aim to analyze and verify a property implicitly claimed by the construction principle of the so-called σ-block mutation operator proposed by Grimme et al. [3]. This mutation operator works as depicted in figure 1. A position t in the index range of the permutation string is selected randomly. Then, a sub-string (block) of size $2\sigma + 1$ is selected symmetrically around t and sorted according to a given priority rule (e. g., SPT or EDD). The parameter σ, which determines the block size is selected normally distributed. Note, that the block is truncated, if it exceeds the boundaries of the permutation string.

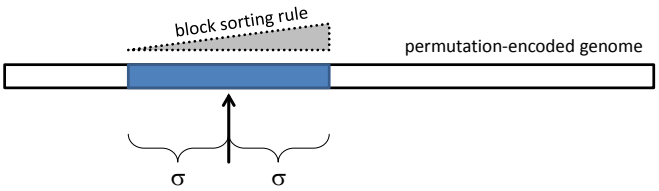

Fig. 1. Working principle of the σ-block mutation operator. A block of size $2\sigma + 1$ is randomly selected and sorted according to a priority rule.

Grimme et al. [3] show, that this operator improves the solution quality of an evolutionary multi-objective algorithm (specifically the used predator prey approach), when applied to a population using different objective-related priority rules. The operator implicitly postulates that Pareto-optimal solutions can be constructed from building blocks, which are internally sorted according to one of the applied priority rules.

In order to analyze and evaluate this claim and to additionally gain insight into trade-off construction, we solved 20 instances[1] of different size (ranging from 25 to 100 jobs) regarding $1||\sum C_j, L_{\max}$ to optimality using the before described algorithm of van Wassenhoven and Gelder (see section II-B). Then we compared the gained Pareto-optimal solutions to the extremal EDD and SPT(EDD) solutions, respectively. Specifically, we evaluate

- the *longest common sub-string* (i. e., consecutive block) of each solution compared with the extremal solutions and
- the *longest common sub-sequence* (i. e., longest not necessarily blocked sequence of same order) of each solution compared with the extremal solutions.

Both measures are normalized to $[0, 1]$ by the maximal possible common sequence length n. Note, that these measures are lower bounds of similarity: only the longest common sub-{sequence,string} is considered. However, there may be multiple intermingled sequences.

For each instance the results were plotted as box plots. Four instances are exemplarily shown in in figure 2. Obviously, the maximum length of common sub-strings of the EDD solution and all remaining Pareto-optimal solutions (leftmost box plot) is rather low (ranging between 0.11 and 0.14 for the depicted 25 job instance and between 0.01 and 0.10 for the 100 job instance). A similar behavior can be observed for the comparison of trade-off solutions with the SPT(EDD) solution (second from left box plot). Due to the working principle of van Wassenhoven and Gelder's algorithm, longest common blocks and sequences sometimes are equal to the SPT(EDD) solution.

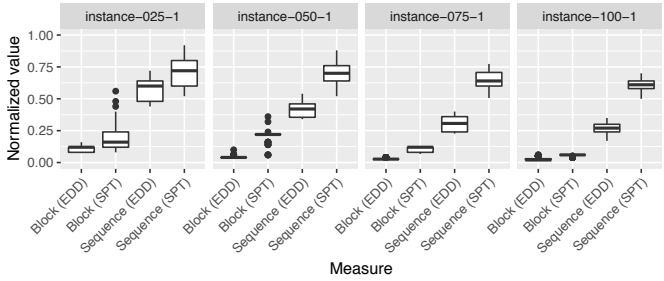

Fig. 2. Analysis of solution structure of the Pareto-optimal solutions for a scheduling instance with $n = 25, 50, 75, 100$ jobs respectively (from left to right) considering problem $1||\sum C_j, L_{\max}$. Depicted are the distributions of normalized longest sub-blocks (Block (EDD), Block (SPT)) and normalized longest sub-sequences (Sequence (EDD), Sequence (SPT)) compared to the pure EDD and SPT solutions respectively.

These findings suggest, that the Pareto-optimal solutions usually consist of very small blocks, which adhere to a single priority rule. This again suggests, that the concept of the σ-block mutation operator may not be sufficient to effectively integrate priority-rules.

Analyzing the longest common sub-sequences (two rightmost box plots) indicates that an operator constructing non-block-consecutive sequences throughout the string may also contribute to an effective construction of trade-off solutions. Both, EDD and SPT sub-sequence lengths exceed the maximum length of block sub-strings by far. This behavior was consistently observed for all 20 instances.

[1] A detailed description of the generation process and the properties of these instances is provided in section V-A.

IV. A Priority Mixture Operator

The findings from section III directly lead to a different operator concept for integrating priority-rule-based expertise into mutation. We redesign the σ-block mutation such that a sequence of not necessarily adjacent positions is sorted according to the respective priority rule (see figure 3):

1) Select λ different positions from the permutation string.
2) Apply the priority rule only among elements of the selected positions. Not selected elements are not considered.

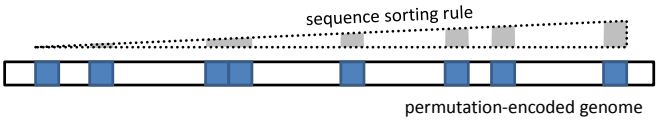

Fig. 3. Working principle of the priority mixture mutation operator. Random positions all over the permutation string are selected and reordered according to the given priority rule. Not selected positions are not considered during reordering.

This results in a permutation string, for which a subsequence of positions adheres to the selected priority rule. Clearly, that arbitrary priority rules (like SPT, EDD, or even more complex priorities based on multiple properties of the considered elements) can be integrated into this generic operator concept.

In that sense, multiple instances of this operator (with different priority rules attached) can be applied to the population of an evolutionary multi-objective algorithm. Following the approach of Lang [20], we can construct mutation operators in favor of the considered objectives. For the problem $1||\sum C_j, L_{\max}$ we may construct two operators with the SPT and EDD priority rules attached, respectively. They produce (at least partly) contradicting sub-sequences in the same permutation string and lead to a (hopefully) mixture of priorities, see figure 4.

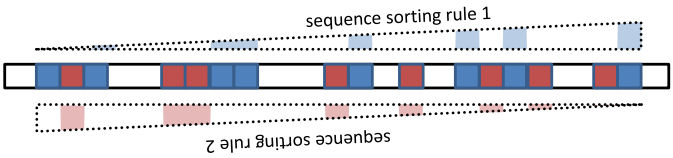

Fig. 4. Schema of solution generation by applying the proposed priority mixture mutation operator for two priority rules on a single permutation.

V. Experiments and Results

In order to evaluate the properties and benefit of the proposed priority mixture operator, we consider single and parallel machine scheduling problems. Therefore, we generate instances using a well defined process which is described in section V-A. In section V-B we detail our setup and finally present and discuss our results in section V-C.

A. Instances

Following [3], we create a benchmark set of 20 problem instances, 5 instances for each instance size $n \in \{25, 50, 75, 100\}$ with release and due dates based on a configurable three-step instance generation process. Figure 6 illustrates some of the generated instances.

1) First, for each job, a uniformly randomly distributed processing time $p_j \in \{p_{\min} = 1, \ldots, p_{\max} = 50\}$ is assigned.
2) In the second step, release dates are created: 50 % of the instances get a zero release date, i. e., $\lfloor \frac{n}{2} \rfloor$ jobs are available to the machine(s) from the beginning. The remaining $n - \lfloor \frac{n}{2} \rfloor$ jobs get a release date r_j, which is a realization of a truncated exponential distributed random variable $R_j \sim \text{TExp}\left(\frac{1}{\lambda}, t = C^1_{max} - p_j\right)$, where $C^1_{max} = \sum p_j$ is the non-delay makespan on a single machine, t is the truncation parameter, and λ is the rate parameter of the truncated exponential distribution. The choice of this parameter ensures that $P(R_j \leq C^1_{max}) = 1$. Thus, with probability 1 the release date is sampled before C^1_{max}. Additionally, due to the nature of the (truncated) exponential distribution) earlier release dates are more likely than late ones.
3) Finally, due dates are assigned: First a factor f_j is sampled uniformly at random from a $\mathcal{U}(3, 10)$ distribution. Next, due dates d_j are sampled from a truncated normal distribution $\mathcal{TN}(\mu_j, \sigma_j^2, a_j)$. Here, $\mu_j = r_j + f_j \cdot p_j$ is the mean, $\sigma_j = (f_j - 1)p_j/3$ is the standard deviation and $a_j = r_j + p_j$ is the left-hand truncation for the Normal distribution (see Figure 5). The choice of σ_j ensures, that with probability $\approx 99.7\%$ the due date deviates from its mean by at most $3\sigma_j$, since

$$P(|D_j - \mu_j| \leq 3\sigma_j) \approx 0.99.$$

The sampled factor f_j decides on the variance of the distribution, i. e., higher values lead to higher variance and thus, a higher probability for later due dates. The left-truncation of the due dates finally assures, that $d_j \geq r_j + p_j$ holds, i. e., that each job can be finished.

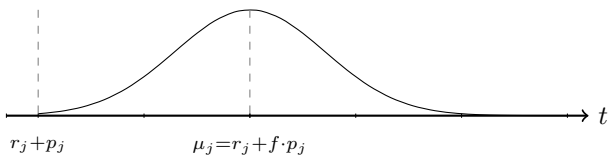

Fig. 5. Visualization of the density function of the truncated Normal distribution used to sample due dates in the instance generation process.

B. Experimental Setup

We apply the evolutionary multi-objective algorithm NSGA-II 10 times and for 2000 generations for each considered problem[2]. As problems we consider the simple single objec-

[2]Additionally, we examined the SMS-EMOA algorithm. Since no significant difference was was observed we limit our experimental evaluation to NSGA-II

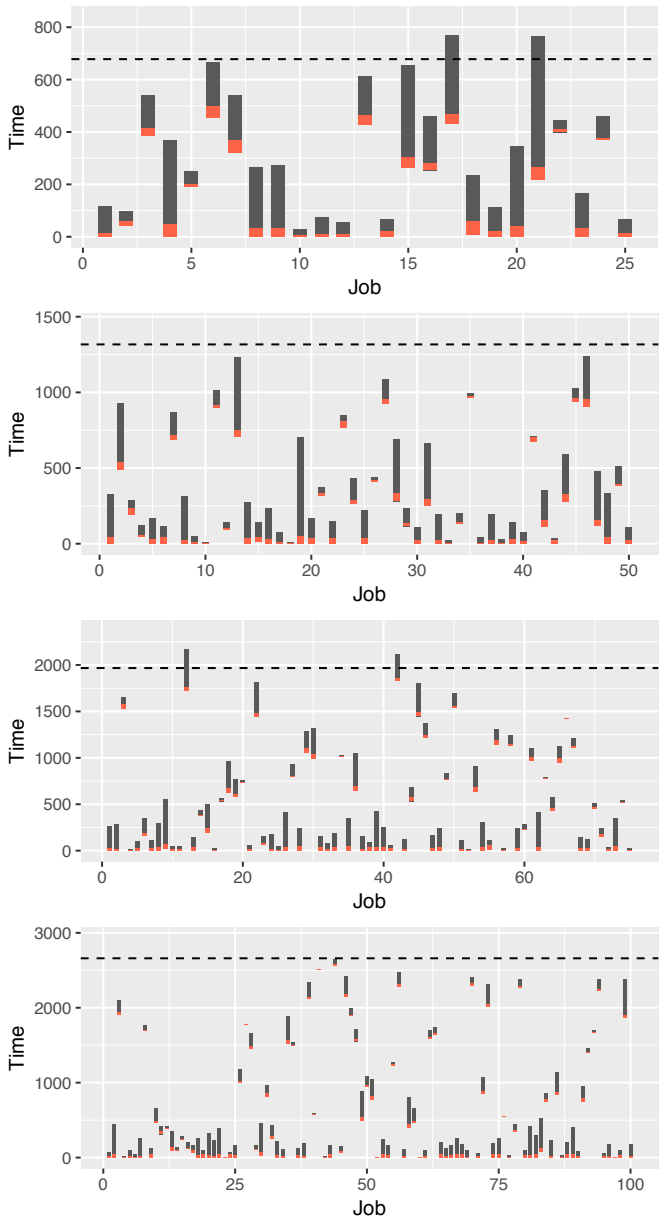

Fig. 6. Examplary instances with $n = 25, 50, 75, 100$ jobs (from top to bottom). The dark boxes show the time between release and due date, while the red boxes indicate the processing time. The dashed line highlights the makespan of the instance.

tive problem $1||\sum C_j, L_{\max}$, the NP-hard parallel machine problem $P3||\sum C_j, L_{\max}$, and the three-objective problem $1|r_j|\sum C_j, L_{\max}, \sum U_j$. For each problem we generate 5 instances for sizes $n \in \{25, 50, 75, 100\}$ (see Section V-C).

The parental and offspring population sizes are set to the number of jobs of the problem instance at hand. Defaults were used for all remaining algorithm parameters. As we especially investigate the effects of mutation operators, crossover operators are generally deactivated during our experiments. For the applied σ-block mutation, σ is set to $0.1 \cdot n$ with n being the instance size. The subsequence length for the mixture operator is bounded by $2 \cdot 0.1 \cdot n$.

All experiments were conducted using the statistical programming language R on a parallel linux computer cluster, which consists of 3528 processor cores in total. The utilized compute nodes are 2,6 GHz machines with 2 hexacore Intel Westmere processors, 12 cores per node, and 2 GB main memory per core. The used NSGA-II implementation can be found in the R package `ecr` [23].

C. Results

In a first step, we investigate the isolated effects of priority-rule integration via a single operator. This is done by either applying the mixture mutation operator (denoted as `randomPosition`) or the σ-block mutation (denoted as `sigmaInterval`) with only SPT priority or EDD priority included.

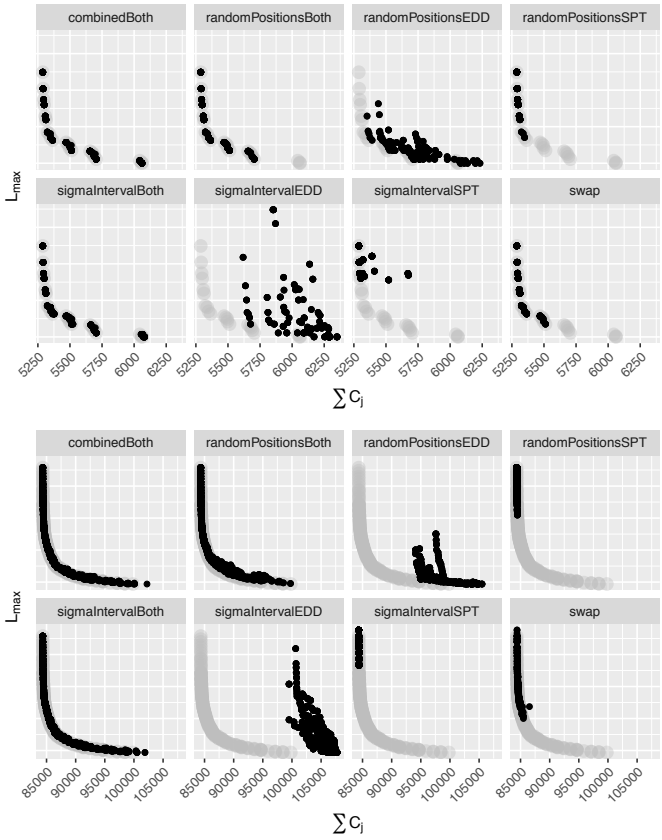

Fig. 7. Objective space for all considered mutation operators for an instance with $n = 25$ jobs (top) and $n = 100$ jobs (bottom) respectively for $1||\sum C_j, L_{\max}$. The true Pareto-front (big gray points) points is visualized besides the union of approximations from all 10 runs on the instance (small black points).

Figures 7 and 8 show the results denoted as `randomPositionEDD`, `randomPositionSPT`, `sigmaIntervalEDD`, and `sigmaIntervalSPT` for $1||\sum C_j, L_{\max}$ as well as $P3||\sum C_j, L_{\max}$ and for instances comprising 25 or 100 jobs. Here we find the expected behavior: positions of solutions on the Pareto-front reflect the integrated priority rules. For an integrated EDD rule, both

operators force solutions to small $L_{\max}$ values (i. e., the right tail of the Pareto-front), while SPT-biased mutation forces the solutions to small $\sum C_j$ solutions (i. e., the left tail of the Pareto-front).

Result 1: The mutation operators (σ-block and mixture mutation) allow the integration of priority rules and are able to force solutions towards the desired area of the Pareto-front.

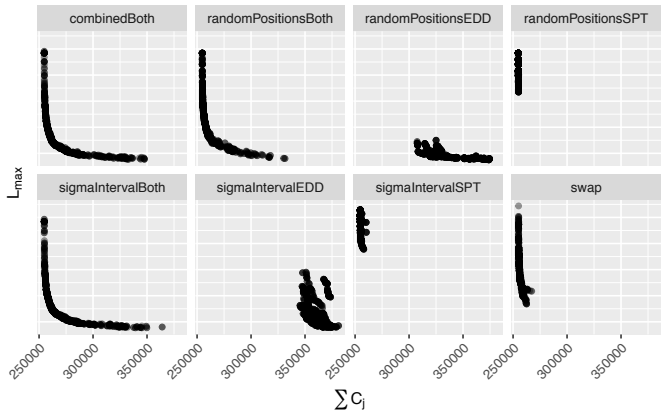

Fig. 8. Pareto-approximations for all considered mutation operators and combinations of operators for an instance with $n = 100$ jobs and $P3||\sum C_j, L_{\max}$.

A closer look to the results from applying mutation operators separately shows, that the σ-block mutation tends to produce especially good solutions around the most extremal positions of the Pareto-front. The mixture operator, however, also allows solutions which cover larger areas of the favored Pareto-front area. This is especially true for the EDD-biased operator. Certainly, the property of generating consecutive blocks of priority-ordered jobs in the σ-block mutation leads—under selection pressure—to the emergence of almost completely EDD- or SPT-ordered genomes. In contrast, the mixture operator, allows largely priority-compliant sequences which are nevertheless interrupted by single jobs or small blocks.

In a second step, we apply each type of operator for SPT and EDD at the same time for all three problems. In each mutation step of NSGA-II, simply one of the mutation operators is selected with equal probability (`sigmaIntervalBoth` and `randomPositionsSPT` respectively). Figures 7 and 8 also show the aggregated results of all runs for the one-machine and three-machines scenarios. All experiments yield a good coverage of the whole Pareto-front. Additionally, we combine both types of mutation operators and denote the plots with `combinedBoth` in our figures. Interestingly, a detailed analysis of the results via the Hypervolume enclosed by the solution fronts shows, that this combination of σ-block mutation and mixture mutation leads in most cases to best results. However, suprisingly the results for `sigmaIntervalBoth` are more effective than the achieved results for `randomPositionsSPT`. We would have expected the reverse case according to the observations of the similarity analysis (see e. g., Fig. 2). Our guess is that the repeated application of σ-block mutation with different sampled σ values also leads to a disjoint sequence ordering as performed by the mixture operator.

Figure 9 shows the Hypervolume results for all 100-job-instances for all three problems and for all 10 runs comparing the performance of pure swap mutation, the application of both mutation operator types separately as well as the combined application of both types. We find, that the mixture operator alone performs not as stable as the σ-block operator. When investigating the single solution fronts, we find that solutions under the mixture operator tend to the middle of the efficient front and miss the extremal areas of the Pareto-front.

TABLE I
AVERAGE HYPERVOLUME RANKS. THE DOMINATED HYPERVOLUME WAS RANKED IN EACH REPLICATION AND AVERAGED OVER REPLICATIONS AND PROBLEM INSTANCES.

Problem	Mutator	Avg. HV rank
$1\|\|\sum C_j, L_{\max}$	combinedBoth	1.43 **(1)**
	randomPositionsBoth	2.88 **(3)**
	sigmaIntervalBoth	1.77 **(2)**
	swap	3.92 **(4)**
$1\|r\_j\|\sum C_j, L_{\max}, \sum U_j$	combinedBoth	1.83 **(1)**
	randomPositionsBoth	2.98 **(3)**
	sigmaIntervalBoth	3.17 **(4)**
	swap	2.02 **(2)**
$P_3\|\|\sum C_j, L_{\max}$	combinedBoth	1.60 **(1)**
	randomPositionsBoth	2.75 **(3)**
	sigmaIntervalBoth	2.12 **(2)**
	swap	3.52 **(4)**

However, table I reveals that the combined application of the mixture operator and the σ-block operator yields the best average dominated hypervolume in all considered scenarios. For the problems $1||\sum C_j, L_{\max}$ and $P_3||\sum C_j, L_{\max}$ the exclusive usage of the mixture operator or the σ-block operator are ranked second and third, respectively. The simple swap

TABLE II
BEST MUTATION OPERATORS BASED ON MEDIAN DOMINATED HYPERVOLUME FOR EACH COMBINATION OF INSTANCE AND SCHEDULING PROBLEM.

	$1\|\|\sum C_j, L_{\max}$	$1\|r_j\|\sum C_j, L_{\max}, \sum U_j$	$P_3\|\|\sum C_j, L_{\max}$
$n = 2$	sigmaIntervalBoth	swap	combinedBoth
	combinedBoth	swap	combinedBoth
	combinedBoth	swap	combinedBoth
	combinedBoth	swap	combinedBoth
	combinedBoth	swap	combinedBoth
$n = 50$	combinedBoth	swap	combinedBoth
	combinedBoth	swap	combinedBoth
	combinedBoth	swap	combinedBoth
	combinedBoth	swap	combinedBoth
	combinedBoth	combinedBoth	combinedBoth
$n = 75$	combinedBoth	combinedBoth	combinedBoth
	sigmaIntervalBoth	combinedBoth	combinedBoth
	sigmaIntervalBoth	combinedBoth	combinedBoth
	sigmaIntervalBoth	combinedBoth	combinedBoth
	combinedBoth	swap	sigmaIntervalBoth
$n = 100$	combinedBoth	combinedBoth	combinedBoth
	combinedBoth	combinedBoth	sigmaIntervalBoth
	combinedBoth	combinedBoth	sigmaIntervalBoth
	sigmaIntervalBoth	combinedBoth	sigmaIntervalBoth
	sigmaIntervalBoth	combinedBoth	combinedBoth

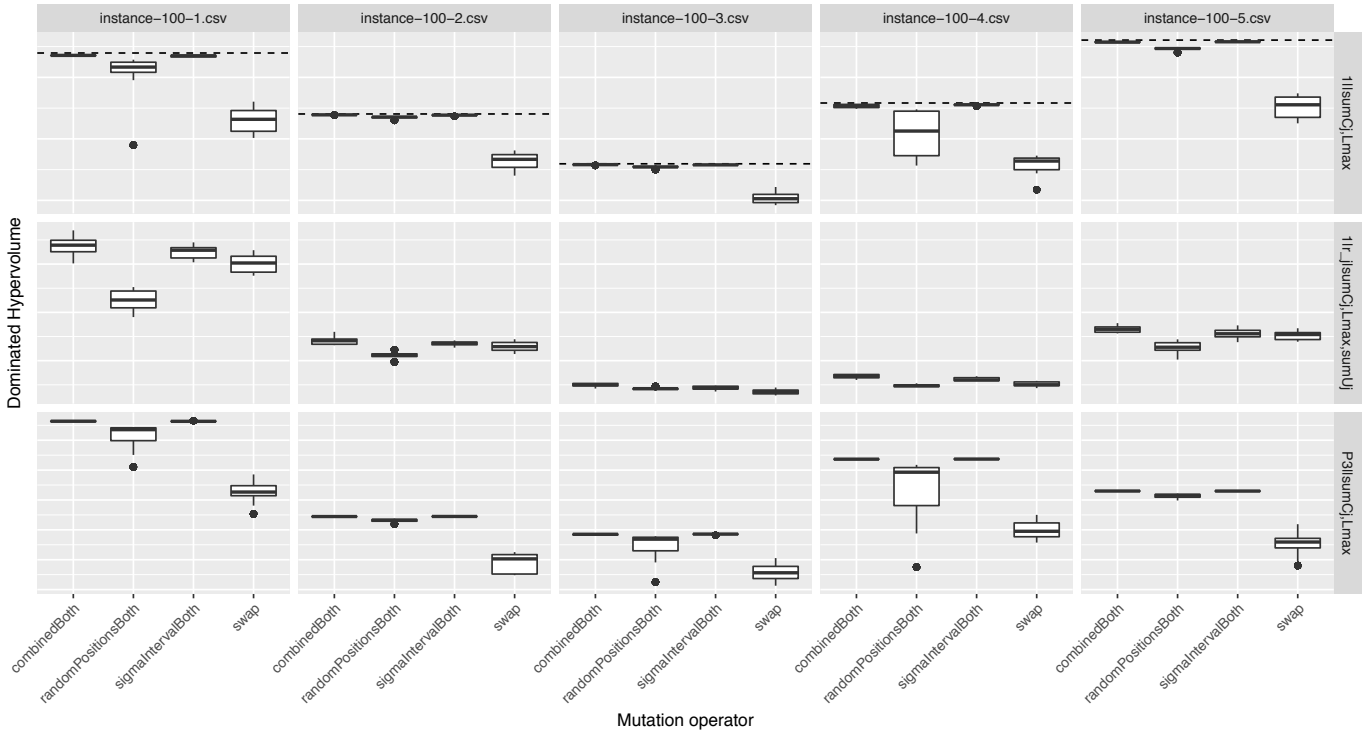

Fig. 9. Distribution of dominated Hypervolume over all 10 runs of the EA for all instances with $n = 100$ jobs split up by the considered scheduling problem and mutation operator. The dashed line in the results for $1||\sum C_j, L_{\max}$ indicates the Hypervolume of the true Pareto-front.

mutation is ranked worse for both problems. However, for the problem $1|r_j|\sum C_j, L_{\max}, \sum U_j$ swap mutation is second best on average. Here, only the combination of the priority-rule operators is slightly better. Table II gives a less aggregated summary of the results (median hypervolume measure). We see that the mixture operator and the σ-block operator leads to best performance with respect to the median Hypervolume measure in $2/3$ of all cases. Additionally, we find that the swap operator yields most beneficial results for small and medium size instances of the three-objective problem. This can be attributed to a probably missing priority rule which respects the third objective $\sum U_j$ and the release date constraint. For larger instances, we speculate that this effect is partly compensated by superior solution properties for the $\sum C_j$ and $L_{\max}$ objectives.

Result 2: The mixture operator alone is not superior to the σ-block operator. However, the combined application of mixture mutation and σ-block mutation leads to superior solution quality in most cases.

VI. Conclusion

In this paper, we discussed and highlighted the problem of expert knowledge integration in the scheduling domain and specifically focused on the integration of common and usually well founded priority rules from single-objective optimization. We exemplarily analyzed the emergence and properties of Pareto-optimal solutions utilizing a polynomial time solver for a single machine problem. The results lead us to proposing a mixture mutation operator, that allows the construction of priority-ordered (not necessarily consecutive) sub-sequences in a permutation encoded genome. We experimentally evaluated this operator inside NSGA-II together with standard swap mutation and the so-called σ-block mutation from literature and were able to show, that the operator effectively supports the solution generation also for more complex problems. Further, we (1) show, that the inclusion of expert knowledge via mutation operators is possible and (2) extended the set of useful operators in this domain.

Future research should consider further scheduling problems and a more comprehensive set of priority rules. Also, the analysis of optimal trade-off solutions may be extended to specifically analyze the parametrization of the operators (e.g., how to set the σ for the block operator).

References

[1] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. Springer, 2009.

[2] P.-F. Dutot, K. Rzadca, E. Saule, and D. Trystram, *Introduction to Scheduling*, 1st ed. CRC Press, 2010, ch. Multi-Objective Scheduling, pp. 219–251.

[3] C. Grimme, J. Lepping, and U. Schwiegelshohn, "Multi-criteria scheduling: an agent-based approach for expert knowledge integration," *Journal of Scheduling*, vol. 16, no. 4, pp. 369–383, 2013.

[4] K. Miettinen, *Nonlinear Multiobjective Optimization*, ser. Kluwer's International Series in Operations Research & Management Science. Boston: Kluwer Academic Publishers, 1999.

[5] R. L. Graham, E. L. Lawer, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[6] W. E. Smith, "Various Optimizers for Single-stage Production," *Naval Research Logistics Quarterly*, vol. 3, pp. 59–66, 1956.
[7] J. R. Jackson, "Scheduling a Production Line to Minimze Maximum Tardiness," University of California, Los Angeles, Management Science Research Project, Research Report 43, 1955.
[8] J. M. Moore, "An n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs," *Management Science*, vol. 15, pp. 102–109, 1968.
[9] G. A. Süer, E. Báez, and Z. Czajkiewicz, "Minimizing the Number of Tardy Jobs in Identical Machine Scheduling," *Computers and Industrial Engineering*, vol. 25, no. 1–4, pp. 243–246, 1993.
[10] R. Haupt, "A Survey of Priority Rule-Based Scheduling," *OR Spectrum*, vol. 11, no. 1, pp. 3–16, March 1989.
[11] L. N. van Wassenhove and F. Gelders, "Solving a Bicriterion Scheduling Problem," *European Journal of Operational Research*, vol. 2, no. 4, pp. 281–290, 1980.
[12] C.-L. Chen and R. L. Bulfin, "Complexity of single machine multi-criteria scheduling problems," *European Journal of Operational Research*, vol. 70, pp. 115–125, 1993.
[13] H. Hoogeveen, "Multicriteria scheduling," *European Journal of Operational Research*, vol. 167, no. 3, pp. 592–623, December 2005.
[14] H.-P. Schwefel, *Evolution and Optimum Seeking*, 1st ed. Wiley, 1995.
[15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
[16] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," in *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, K. Giannakoglou *et al.*, Eds. International Center for Numerical Methods in Engineering (CIMNE), 2002, pp. 95–100.
[17] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653 – 1669, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221706005443
[18] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Berlin: Springer, 1999.
[19] A. A. Pereira, H. Barbosa, and H. Bernardino, "Predator-prey techniques for solving multiobjective scheduling problems for unrelated parallel machines," in *Proceedings of the 9th International Conference on Evolutionary Multi-Criterion Optimization*, ser. LNCS, T. et al., Ed., vol. 10173, 2017, pp. 471–485.
[20] M. Lang and C. Grimme, "Towards standardized and seamless integration of expert knowledge into multi-objective evolutionary optimization algorithms," in *Proceedings of the 9th International Conference on Evolutionary Multi-Criterion Optimization*, ser. LNCS, T. et al., Ed., vol. 10173, 2017, pp. 366–380.
[21] K. Deb, J. Sundar, N. Udaya Bhaskara Rao, and S. Chaudhuri, "Reference point based multi-objective optimization using evolutionary algorithms," *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 273–286, 2006.
[22] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.
[23] J. Bossek, "ecr 2.0: A Modular Framework for Evolutionary Computation in R," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '17, 2017, pp. 1187–1193.