White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of York.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/145544/

Version: Accepted Version

Proceedings Paper:
Dziurzanski, Piotr, Zhao, Shuai, Swan, Jerry et al. (2019) Solving the Multi-Objective Flexible Job-Shop Scheduling Problem with Alternative Recipes for a Chemical Production Process. In: Kaufmann, Paul and Castillo, Pedro A., (eds.) Applications of Evolutionary Computation - 22nd International Conference, EvoApplications 2019, Held as Part of EvoStar 2019, Proceedings. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer, pp. 33-48.

https://doi.org/10.1007/978-3-030-16692-2_3

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

Solving the Multi-Objective Flexible Job-Shop Scheduling Problem with Alternative Recipes for a Chemical Production Process

Piotr Dziurzanski[†], Shuai Zhao[†], Jerry Swan[†], Leandro Soares Indrusiak[†], Sebastian Scholze[‡], and Karl Krone[♯]

[†]Department of Computer Science, Univ. of York, Deramore Lane, Heslington, York, YO10 5GH, UK
[‡] Institut fur Angewandte Systemtechnik Bremen GmbH, Wiener Strasse 1, 28359 Bremen, Germany
[♯] OAS AG, Caroline-Herschel-Strasse 1, 28359 Bremen, Germany

Abstract. This paper considers a new variant of a multi-objective flexible job-shop scheduling problem, featuring multisubset selection of manufactured recipes. We propose a novel associated chromosome encoding and customise the classic MOEA/D multi-objective genetic algorithm with new genetic operators. The applicability of the proposed approach is evaluated experimentally and showed to outperform typical multi-objective genetic algorithms. The problem variant is motivated by real-world manufacturing in a chemical plant and is applicable to other plants that manufacture goods using alternative recipes.

1 Introduction

Manufacturing process scheduling is arguably one of the most widely studied optimisation problems [1]. In this problem, manufacturing jobs are assigned to machines at particular times in order to optimise certain key objectives, such as makespan or total workload of machines. Numerous versions of this problem have been proposed, starting from an original Job-shop Scheduling Problem (JSP) coined by R.L. Graham in 1996, and including flexible JSP (FJSP), where operations can be processed on any compatible resource [2]. However, the classic JSP and its popular extensions are limited to certain classes of rather artificial problems and do not scale well to the problem sizes found in industry [3]. Due to the NP-hard nature of the problem, exact solutions are not generally possible for real-world problems. The multi-objective nature of these problems further exacerbates the difficulty of obtaining good solutions. A variety of metaheuristics have been applied to the gamut of JSP variants. Amongst these, multi-objective genetic algorithms (GAs) have been applied particularly successfully, as surveyed in [4].

The real-world scenario motivating the research described in this paper is related to a manufacturing process for mixing/dispersion of powdery, liquid and paste components, following a stored recipe. The main optimisation objective

of this case study is to increase production line utilisation and, consequently, to decrease the makespan of batch production. The recipes can be executed on different compatible resources. Various recipes can be used to produce the same commodity. Consequently, the decision problem includes the selection of the multisubset (i.e. a combination with repetitions) of the recipes and their allocation to compatible resources, such that the appropriate amount of goods are produced with the minimal surplus in the shortest possible time. As such, the problem resembles, to a certain degree, FJSP with process plan flexibility (FJSP-PPF) [5] or the earlier-formulated "JSP with alternative process plans" [6]. However, none of the papers known to the authors considers process planning by recipe multisubset selection to satisfy both the criteria of the shortest makespan and the minimal surplus of the ordered commodities.

The main contribution of this paper is the formulation of a new variant of a multi-objective FJSP in which a number of commodities can be produced with a set of recipes. A single commodity can be manufactured with a few different recipes whose executions produce different amounts of the commodity and which have different resource compatibility and manufacturing time. The objectives are to minimise the makespan and produce the commodities in the amounts as close to the ordered ones as possible, i.e., to minimise the discrepancies between the ordered quantities and the manufactured ones for each resource. A chromosome encoding for the described problem has been proposed and the classic MOEA/D multi-objective genetic algorithm has been tuned with customised problem-specific genetic operators: mutation and elitism. The applicability of the proposed approach is evaluated experimentally and showed to outperform the classic multi-objective genetic algorithms.

The rest of this paper is organised as follows. After the brief survey of related works in Section 2, system model and problem formulation are presented in Section 3. The proposed approach for the targeted manufacturing scheduling problem is provided in Section 4. The motivating real-world use case is outlined in Section 5, followed by experimental results and conclusion in Sections 6 and 7, respectively.

2 Related work

One of the first applications of a multi-objective genetic algorithm to manufacturing scheduling was described in [7]. The authors of that paper aimed to find a set of nondominated solutions with respect to the minimal makespan, total flowtime and the maximum tardiness. The fitness value of an individual has been computed as a weighted sum of these three criteria, but the weight values were randomly specified whenever a pair of the parent solutions were selected. This led to the creation of the solution space where each point was generated using a different weight vector. These solutions were then improved by local search. The problem solved by this algorithm is a classic JSP, where the sizes of the assumed plant and taskset were limited. Hence, the settings can be viewed as rather abstract, and the manufacturing scheduling was used just for illustrating potential

applicability of the algorithm, rather for direct real-world applicability. In contrast, the problem considered in this paper describes a real-world scenario that is viewed as a challenge by a business partner. A more recent multi-objective genetic algorithm MOAE/D [8], used in this paper as the baseline, has employed some ideas from [7], such as generating various solutions from objective weighted sums.

Several real-world scheduling problems have been deeply researched, typically being solved by customised multi-objective GAs. For example, in [9], a real-world manufacturing problem originating from a steel tube production has been described by extending the classic FJSP and solved using a multi-objective GA with two objectives, namely reduction of the idle time on machines and waiting time of orders. The authors of that paper stressed that it was virtually impossible to apply the earlier research works on JSP in practice as they were based on overly simplified models and assumptions. In that paper, the production routes depend on the orders and a certain production stage could be processed on various homogeneous machines. The model proposed in that paper can be used in numerous job production problems, but is inappropriate in the case of batch manufacturing. In particular, it does not consider recipe selection or minimisation of the commodity surplus, which is addressed by the model proposed in this paper. Readers can refer to the survey presented in [10] to appreciate the complexity of the batch manufacturing in general. The factory model introduced in this paper is capable of describing the majority of the features from the general batch scheduling classification presented in [10], including the "sequence-depending setup", in which sequences of two manufacturing jobs scheduled to be processed subsequently by the same machine can require a time gap of a certain length between them (corresponding to e.g. cleaning the machine in a physical plant).

An interesting real-world problem related to textile batch dyeing scheduling has been described in [11]. Similarly to the problem described in this paper, both the temporal features and the weight of the products are considered. In the textile dying industry, cloths of the same colour can be batched together as long as their total weight does not surpass the capacity of the manufacturing resource. However, for the problem addressed in this paper, the resources are capable of producing only an exact weight of a given commodity, not lower or higher, and the total amount of a manufactured commodity is only influenced with the selection of the recipes multisubset to be executed. Instead of a batching heuristics, a method for recipe multisubset selection that optimises a set of criteria would be desirable.

A number of multi-objective GAs applied to manufacture scheduling problems has been surveyed in [4]. That survey covered assorted types of scheduling problems, including JSP, FJSP, dispatching in a flexible manufacturing system (FMS) and integrated process planning and scheduling (IPPS). The problem described in this paper follows certain realistic assumptions from those problems, such as the presence of alternative machines with different efficiency from FJSP or storage facilitation from FMS. The production planning and scheduling

are performed simultaneously as in IPPS. However, none of the reviewed papers allowed selection of a multisubset of recipes for producing the same type of a commodity. Similarly, none of those papers addressed the problem of minimising the surplus of the produced commodities. Both these features are essential to the problem analysed in this paper and they therefore feature in the proposed solution. This objective is also not mentioned in survey [12], which addressed the variability of the objective functions used for multi-objective FJPs. The objectives enumerated in that survey were related to various features of the production process, instead of the amount of the produced commodities.

From this literature survey, it may be concluded that to date there is no proposed FJSP variant that is compatible with the considered real-world scenario. Consequently, a new customisation of FJSP is needed, together with an algorithm capable of solving this problem on a practical scale. Both are presented in the following sections.

3 System model and problem formulation

The problem considered in this paper is an extended version of the classic FJSP, in which each taskset Γ includes a set of independent recipes γ_j, $j = 1, \ldots, n$. Recipe γ_j produces u_j units of certain commodity δ_l, $l = 1, \ldots, r$ and can be executed by one of resources defined by a set Λ_j, including at least one resource $\pi_i \in \Pi$, $i = 1, \ldots, m$. A recipe γ_j needs $t_{i,j}$ time units while executed on resource π_i.

The plant is supposed to satisfy order O, comprised of o_l units of commodities δ_l. The difference between the actually produced amount of commodity δ_l, θ_l and the ordered amount of commodity δ_l , o_l, is referred to as surplus and computed by:

$$\sigma_l = \theta_l - o_l. \tag{1}$$

Several instances of a single recipe γ_j can be scheduled to produce a sufficient amount of goods. These instances are later referred to as $\gamma_{j,k}$, $k = 1, \ldots, \mu_j$, where μ_j denotes the minimal number of recipe instances that satisfy the ordered amount of δ_l and is defined later in this paper.

Certain sequences of recipe instances γ_{j_1,k_1} and γ_{j_2,k_2}, which manufacture different commodities, can require a time gap of a certain length between them if scheduled to be processed subsequently by the same resources (it corresponds to e.g. cleaning the machine in a physical plant). Thus, the instance ordering can influence the makespan. This ordering is controlled with priority $p_{j,k} \in \mathbb{N}_0$ of a recipe instance $\gamma_{j,k}$. Priorities are ordered decreasingly, so from two recipe instances scheduled to a single resource, the one with a lower value of priority will be executed earlier.

Given a set of recipes Γ, a set of resources Π and an order O, the problem is to assign resources and priorities to a multisubset of recipes from Γ so that the total processing time (makespan) is minimised and the amount of each manufactured commodity is higher or equal to the order, $\theta_l \geq o_l$, but the surpluses of each commodity, σ_l, are minimised.

4 Proposed approach

As summarised by Michalewicz [13], to apply a genetic algorithm to solve a particular problem, each of solution representation, fitness function and evolutionary operators need to be specified. These aspects of the proposed algorithm are focused on in this section.

Let us consider recipe γ_j producing u_j units of a certain commodity δ_l. To determine the upper bound on the number of this recipe instances in the recipe multisubset to be allocated to resources, the lowest number of the recipe execution leading to producing sufficient units o_l of an ordered commodity θ_l needs to be determined. This value can be computed using equation

$$\mu_j = \left\lceil \frac{o_l}{u_j} \right\rceil . \tag{2}$$

Consequently, the cardinality of the multisubset is upperbounded with

$$\eta = \sum_{j=1}^{n} \mu_j. \tag{3}$$

The solution to the problem can be then described with a chromosome of length 2η (η for resource allocation and η for priorities), following the encoding proposed in the following subsection. Hence the genes can be addressed as $\tau_1, \ldots, \tau_\eta$, where τ_1 and τ_η correspond to recipe instances $\gamma_{1,1}$ and γ_{n,μ_n}, respectively.

As the considered real-world scenario includes several objectives aiming at minimising the makespan and the surplus of each commodity, the multi-objective genetic algorithm techniques briefly described later in this section needs to be applied.

4.1 Genetic representation of metrics

In genetic algorithms, candidate solutions are treated as individuals. During the optimisation process, these individuals are evolved using a set of bio-inspired operators, described briefly in Subsection 4.2. In this section, individuals' encoding that facilitates the manufacturing process optimisation and reconfiguration are proposed.

Since in the considered problem each metric assumes a value from a certain, predefined domain, so-called value encoding of chromosomes needs to be applied. This encoding, in contrast to e.g. the traditional binary encoding, allows each gene to directly correspond with a certain value of one variable of the optimisation problem and assume values from the domain of that variable only. For example, a gene representing a certain recipe instance allocation can assume only values corresponding to the compatible resources. To produce a required amount of the ordered commodities, a certain multisubset of recipe set Γ needs to be applied rather than all recipes from this set. The maximal number of recipes

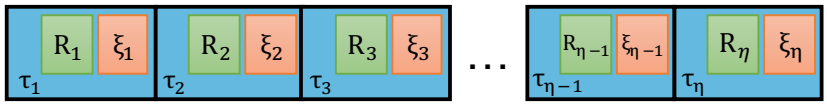

Fig. 1. Genes in a chromosome for manufacturing processes with alternative recipes

that needs to be considered is upperbounded to a certain value η, computed with equation (3).

The role of the GA is to allocate the recipe instances to resources and schedule them in time. The encoding has hence to embrace both the spatial and temporal scheduling. Consequently, in the proposed encoding a chromosome contains genes of two types, as shown in Figure 1. For η recipes (i.e., τ_1 to τ_η) that need to be scheduled, the number of genes is thus equal to 2η. The odd η genes (R_x in the figure) indicate the target resource for η recipe instances, $G_{2x+1} \in \{\emptyset, \pi_1, \ldots, \pi_m\}$, where symbol $\emptyset$ denotes the situation that certain recipe instance has not been scheduled for execution. The remaining η genes (ξ_x in the figure) specify the priorities of the recipe instances, $G_{2x} \in \mathbb{N}$, where $x = 1, \ldots, \eta$. The priorities are sorted in descending order, i.e. priority 0 is the highest. The aim of introducing priorities is to determine the processing orders of recipes allocated to the same resource and thus to determine the temporal scheduling. This ordering does not change the amount of produced commodities but can influence the makespan due to the sequence-dependent setups discussed earlier. The value of the solution represented by such chromosome is then evaluated using a plant model based on interval algebra described in [14]. The details of the applied plant modelling are out of the scope of this paper.

4.2 Evolution-inspired operators

In a typical GA, evolutionary operators (e.g., crossover and mutation) are applied to a set of individuals for advancing offspring with better solution quality. For the considered optimisation problem, a number of customised genetic operators needs to be proposed. The influence of the operators described below is experimentally evaluated in Section 6.

The mutation operator is customised for the studied optimisation problem in the following way. Instead of assigning random configurations (i.e., priorities and allocations) to recipe instances, the proposed operator mutates allocations of recipe instances via two approaches. Depending on the current allocation value of a given recipe instance, the first approach switches its allocation either to a randomly chosen resource that is compatible with the recipe (if this recipe instance is not allocated) or to *no allocation* (i.e., reject for production, in the case that the recipe instance has a valid allocation)[1]. The second approach mutates

[1] Note, due to the applied value encoding, a recipe instance can either be allocated to a resource among its feasible allocations or not be allocated at all. A recipe that is not allocated will be not scheduled for manufacturing.

the allocation of an allocated recipe instance to another compatible resource (if possible). If the recipe instance is not allocated, no action is performed under the second approach. A mutation factor $F \in [0.0, 1.0]$ is introduced to specify the approach to be applied for each allocation mutation. As for priorities, the proposed operator simply assigns random values (but within the predefined priority range) to recipe instances that are chosen to be mutated.

As shown in Section 6, compared to a traditional mutation operator, the proposed problem-specific mutation improves the quality of generated solutions and helps to expand the search range of the multi-objective genetic algorithm.

Elitism is often applied by GAs to guarantee the solution quality of each generated population, where a limited number of best solutions are passed to the next generation directly without any operation. In this work, the elitism mechanism is modified to serve the purpose of minimising the sum of discrepancy scores of all commodities. At the end of each generation, solutions that contain the least discrepancy score of each commodity are selected and are used to form an individual that contains the best manufacturing configuration (in terms of the discrepancy scores) being found for each commodity. This individual is then added to the population (if possible, by replacing a randomly chosen individual with a lower fitness) and will involve into future evolution.

Such a simple but effective mechanism can arguably improve the solution quality (in terms of the total discrepancy scores) produced by the proposed multi-objective GA. In particular, solutions with a minimised sum of discrepancy scores for all commodities can be obtained among all solutions generated by the GA. This feature is highly desirable as the cost of storing over-produced commodities can be effectively reduced.

4.3 Customisation of MOEA/D

The problem analysed in this paper is characterised with multi-objective criteria, since not only does the makespan need to be minimised, but also the amount of manufactured commodities should be as close to the ordered amounts as possible to minimise the storage costs. The diversity of these criteria makes it difficult to convert such multi-objective optimisation problem into a single-objective weighted sum of these objective values. Depending on the current situation, some solutions with a low weighted sum of objectives may not be acceptable due to, e.g., insufficient storage space for a certain commodity. An end-user should be then informed about a wide set of Pareto-optimal solutions to select the final solution based on his/her knowledge of the problem. The set of the alternative solutions presented to the end-user should be then diverse and, favourably, distributed over the entire Pareto front. This expectation is in line with the properties of the MOEA/D algorithm proposed by Zhang and Li in [8] with Tchebycheff Approach [15] adopted for multi-objective decomposition. With the Tchebycheff Approach, minimising a typical optimisation problem $F(x) = (f_1(x), \ldots, f_m(x))^T$ can be decomposed to the following:

$$\texttt{minimise}\ g^{te}(x|\lambda, z^*) = \max_{1 \le i \le m} \{\lambda_i (f_i(x) - z_i^*)\}, \tag{4}$$

Algorithm 1: Pseudo-code of MOEA/D-RS algorithm

inputs : Resource set Π; Chromosome size 2η;
Population size N;
Uniform spread of N weight vectors λ^1, λ^2,..., λ^N;
Neighbourhood size T;

outputs : EP (a set of recipe instance allocation and scheduling solutions);

1 Set $EP = \emptyset$
2 Generate N random individuals with recipe instance allocations (or recipe instance rejection) and priorities as the initial population;
3 Evaluate the key objective values of each individual in the initial population;
4 Compute the Euclidean distances between any two weight vectors and find T closest weight vectors to each weight vector. For each $i = 1, \ldots, N$, set $B(i) = \{i_1, ..., i_T\}$;
5 Initialise ideal points $z = (z_1, z_2, ..., z_m)$ based on the objective values obtained from all individuals of the initial population;
6 **while** *not termination condition* **do**
7 **for** *i=1,...,N* **do**
8 Randomly select two neighbours from $B(i)$, generate a new individual y via genetic operators proposed in Subsection 4.2 to the selected neighbours.
9 Evaluate the key objective values of y;
10 For each $j = 1, \ldots, m$, if $z_j > f_j(y)$, then set $z_j = f_j(y)$;
11 For each $j \in B(i)$, set $x^j = y$ if $g^{te}(y|\lambda^j, z) \leq g^{te}(x^j|\lambda^j, z)$;
12 Remove all individuals in EP that are dominated by y and add y to EP if no individuals dominate y.
  end
13 Generate an elite individual employing the operator described in Section 4.2, evaluate its objectives' values, and add it to the current population (if eligible), update z and EP.
 end
14 **return** EP;

where m is the number of objectives in the targeted optimisation problem F(x), $f_i(x)$ gives the value of objective i based on solution x, $x \in \Omega$ indicates a given solution in the decision space (i.e., Ω), λ is a set of uniformly distributed weight vectors, z^* indicates a set of reference points and $z_i^* = min\{f_i(x)|x \in \Omega\}$ gives the reference point of objective i.

The applied multi-objective optimisation algorithm is outlined in Algorithm 1, named MOEA/D-RS, i.e. MOEA/D for recipe scheduling problems. It follows the basic MOEA/D principles but is integrated with the proposed evolution-inspired operators for scheduling recipe in the context of manufacture.

The applied multi-objective optimisation algorithm treats differently the initial generation (lines 1-5) and the remaining generations (lines 6-14), as detailed below. For the initial population, MOEA/D calculates the neighbours (i.e., $B(i)$ for individual i) of each individual based on the Euclidean distance of their as-

sociated weight vectors (i.e., λ^i for individual i) (line 4). The set of ideal points (i.e., z) is initialised as the best objective values found in the initial population (line 5).

For the following populations, in each generation (line 6), MOEA/D iterates each individual in the current population (line 7) and selects individuals from the neighbours of the currently-examined individual (line 4) for creating new offspring. The rationale behind neighbourhood selection is the optimal solution of $g^{te}(x|\lambda^i, z^*)$ should be close to that of $g^{te}(x|\lambda^j, z^*)$ if the Euclidean distance between λ^i and λ^j is low, which indicates any g^{te}'s with a weight vector close to λ^i can help optimising $g^{te}(x|\lambda^i, z^*)$ [8]. With parents selected, new individuals are then generated by the proposed genetic operators described in Section 4.2.

Once a new individual (say individual y) is generated, values of its key objectives are calculated based on the fitness function (line 9). The ideal points' set z is updated if y contains a better value for any objective (line 10). This new individual will replace any given individual j in the neighbourhood of the currently-examined individual if $g^{te}(y|\lambda^j, z) \leq g^{te}(x^j|\lambda^j, z)$ (line 11). A set of recipe instance allocation and scheduling solutions, EP, is then updated (if necessary) based on the procedure given in line 12. At the end of each generation, we integrate the proposed elitism algorithm into the original MOEA/D and generate an elite individual as described in Section 4.2. Accordingly, z and EP are updated (if necessary) based on the objective values of the elite (line 13). Once the `termination condition` is met, the algorithm is finished with EP returned as the Pareto Front (line 14).

5 Real-world scenario

The considered real-world scenario is based on the process manufacturing of mixing/dispersion of powdery, liquid and paste recipe components, following a stored recipe. The main optimisation objective of this case study is to increase production line utilisation and, consequently, to decrease the makespan of batch production. The optimisation process can be viewed as scheduling operations, as described by recipes, both spatially (i.e. to a particular production line) and temporarily (i.e. to a particular time slot). Depending on the selected production line, the time to produce a product may vary significantly, which influences the percentage of manufacturing time that is truly productive, known as Overall Equipment Effectiveness (OEE). Further impact on the OEE is due to the size of batches to be produced.

In the considered scenario, the recipes for each batch produce a certain amount of commodity. Consequently, to satisfy a (daily) order for a certain commodity, one or more recipes for producing such commodity have to be selected and scheduled to resources. However, the sum of the commodity amount produced by any selection of recipes may be different from the daily order amount for that commodity. If a certain commodity cannot be produced in the required amount, some commodity surplus is expected. As the total amount of the produced commodity cannot be higher than the available storage space and the

surplus storage can be expensive, additional optimisation objectives can be defined: not only the makespan, but also the surpluses of each produced commodities have to be minimised. This observation leads to the conclusion that multi-objective optimisation techniques, as described earlier in this paper, can be applied.

The example plant consists of a set of mixers, Π. There are five identical 5 tonne mixers, named Mixer 1-5 (π_1, π_2, π_3, π_4, π_5, respectively), and two identical 10 tonne mixers, named Mixer 7 (π_6) and Mixer 8 (π_7). There are two special 10 tonne mixers: Mixer 9 (π_8) and Mixer 10 (π_9). Four types of white paints can be produced in the factory and each mixer can be used to produce any commodity. However, the amount of paint produced during one manufacturing process and processing time vary depending on the mixer type and paint type. For each combination of mixer type and paint type, there is a unique recipe, summarised in Table 1 (the paint type names are in German). The storage tanks, connected with the mixers via pipelines, limit the amount of the paints that can be produced as they have limited capacity and each tank can store only one type of paint. In case two recipes producing a different paint type are executed by the same mixer in sequence, a short sequence-dependent setup interval of the length provided by the business partner is enforced.

Table 1. Example recipe characteristics for a certain factory

Paint - commodity δ_l	Recipe γ_j	Compatible resources $\pi_i \in \Lambda_j$	Amount of produced commodity u_j	Time $t_{i,j}$ of executing recipe γ_j
Std Weiss δ_1	γ_1	$\{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5\}$	5 t	90 min.
	γ_2	$\{\pi_6, \pi_7\}$	10 t	60 min.
	γ_3	$\{\pi_8, \pi_9\}$	10 t	45 min.
	γ_4	$\{\pi_8, \pi_9\}$	10 t	45 min.
Weiss Matt δ_2	γ_5	$\{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5\}$	5 t	90 min.
	γ_6	$\{\pi_6, \pi_7\}$	10 t	60 min.
	γ_7	$\{\pi_8, \pi_9\}$	10 t	45 min.
	γ_8	$\{\pi_8, \pi_9\}$	10 t	45 min.
Super Weiss δ_3	γ_9	$\{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5\}$	4 t	120 min.
	γ_{10}	$\{\pi_6, \pi_7\}$	8 t	90 min.
	γ_{11}	$\{\pi_8, \pi_9\}$	8 t	60 min.
	γ_{12}	$\{\pi_8, \pi_9\}$	8 t	60 min.
Weiss Basis δ_4	γ_{13}	$\{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5\}$	6 t	60 min.
	γ_{14}	$\{\pi_6, \pi_7\}$	12 t	45 min.
	γ_{15}	$\{\pi_8, \pi_9\}$	12 t	30 min.
	γ_{16}	$\{\pi_8, \pi_9\}$	12 t	30 min.

6 Experimental Results

Based on the real-world case study described in Section 5, this section aims at the considered manufacture scheduling problem and presents experiments investigating (i) the optimisation results of NSGA-II [16], original MOEA/D [8] (referred to as MOEA/D hereafter) and the proposed problem-specific algorithm based on MOEA/D (i.e., MOEA/D-RS); (ii) the efficiency of the evolutionary operators proposed in Section 4.2; and (iii) the scalability of the proposed multi-objective genetic algorithm for optimisation problems in manufacture scheduling.

The factory setting is given in Table 1. Based on the amount to produce for each commodity, we first generate sufficient number of recipe instances for each recipe type. The number of instances for a given recipe γ_j can be determined using equation (2). For instance, a requirement of 45 tonnes of "Std Weiss" would lead to 5 instances of γ_1 and 4 instances of γ_2, γ_3 and γ_4, respectively. Each recipe instance is then assigned with random scheduling parameters (i.e., a random allocation from its compatible resources and a priority) and will be then added to the initial population. To provide fair comparison, general GA parameters applied in all tested algorithms include: `PopulationSize = 100`, `MaxGeneration = 100`, `CrossoverRate = 1.0`, `MutationRate = 0.8`. One-point crossover operator is applied. The factor that controls the mutation approach (i.e., F) in MOEA/D-RS is set to 0.3 across the evaluation.

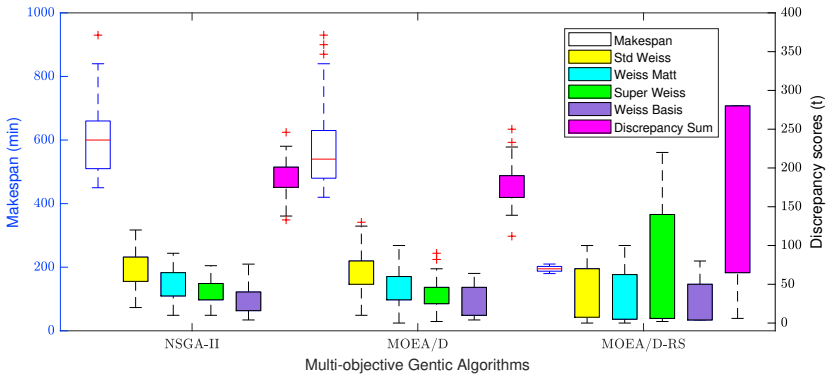

Fig. 2. Optimisation results NSGA-II, MOEA/D and MOEA/D-RS for $\delta_1 = 45$t, $\delta_2 = 40$t, $\delta_3 = 30$t, $\delta_4 = 20$t

Figure 2 presents the optimisation results of three multi-objective GAs for the considered optimisation problem with input values of $o_1 = 45$, $o_2 = 40$, $o_3 = 30$, $o_4 = 20$ (in tonnes). The makespan obtained by each GA is presented as a box with a blue frame and is associated with the blue Y-axis on the left-hand side of the figure while the discrepancy scores (boxes with black frames) are associated

with the Y-axis on the right-hand side of the figure. The navy box gives the sum of discrepancy scores of all commodities for each solution. As shown in the figure, the optimisation results given by NSGA-II and MOEA/D are similar, but are both outperformed by the proposed problem-specific GA in terms of the minimal value obtained for each objective. The Diversity Comparison Indicator (DCI) [17], a quality indicator commonly applied for assessing the diversity of Pareto front approximations in many-objective optimisation, is applied to the obtained Pareto Fronts and returns $(0, 0, 1)$[2], which indicates that at least one solution returned by MOEA/D-RS strictly dominates[3] any solution obtained by either NSGA-II or MOEA/D. Such observation shows that the generic multi-objective GAs may not be suitable for the studied problem, and certain problem specific multi-objective evolutionary algorithms (e.g., the one proposed in this paper) are desirable. The experiment has been repeated 5 times for 5 slightly different recipe characteristics and ordered amounts of commodities. In all the conducted experiments, the same value of DCI, i.e. $(0, 0, 1)$, has been obtained. These conclusions were confirmed by Sign Test with the probability exceeding 99.99%.

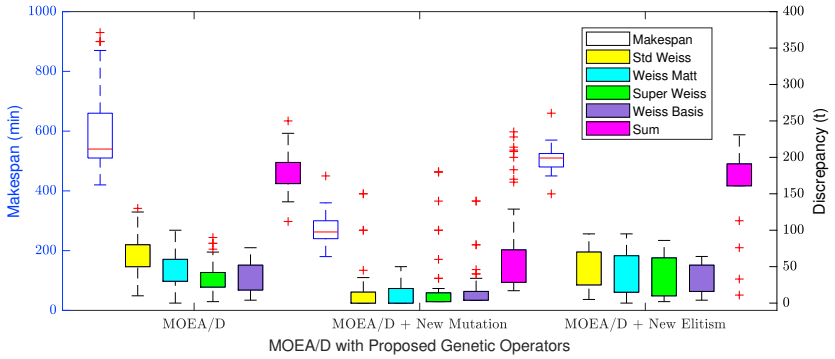

Fig. 3. Influence of the proposed evolutionary operators for an example order

Figure 3 investigates the efficiency of evolutionary operators proposed in Section 4.2, where each operator is integrated into MOEA/D alternatively and is compared with the original MOEA/D. As shown in the figure, MOEA/D with the proposed mutation operator applied demonstrates an overall better optimisation results for each objective compared to MOEA/D. A tuple returned by DCI for

[2] In general, each numerical value in a tuple obtained with DCI corresponds to a certain front quality in relation to the remaining fronts under comparison. These values are upperbonded with 1 and a higher value denotes a better relative front quality.

[3] For two solutions p and q, p strictly dominates q if p has a better value than q for any objective.

the two obtained fronts is equal to $(0, 1)$, which means that at least one solution generated by the customised algorithm strictly dominates any solution generated by the original MOEA/D. The efficiency of the elitism operator is demonstrated by the results of the sum of discrepancy scores for all commodities (i.e., the navy box). With this operator, the applied algorithm can have solutions that contain the least amount of over-produced commodities (i.e., the sum of discrepancy scores), and hence, can greatly reduce the storage cost required by factories. As reported by the DCI test, the values of the original MOEA/D and MOEA/D with elitism operator are 0.318 and 0.682, respectively.

As confirmed by the experiment given in Figure 2, by integrating all these operators into one algorithm, MOEA/D-RS demonstrates the best performance among all tested multi-objective algorithms under the studied problem scenario. The above experiments have been repeated for 5 slightly different recipe characteristics and ordered amounts of commodities. Similar results have been obtained with average DPI values $(0, 0, 1)$ for (NSGA-II, MOEA/D, MOEA/D-RS), $(0, 1)$ for (MOEA/D, MOEA/D + Customised Mutation), and $(0.389, 0.611)$ for (MOEA/D, MOEA/D + Customised Elitism) throughout the evaluation.

The scalability of the proposed problem-specific evolutionary algorithm is investigated with optimisation results of "makespan" and "sum of discrepancy scores" presented in Figure 4 and 5, respectively. This experiment is performed based on scaling both the size of the factory (i.e., number of mixers) and the amount of commodities required for production. A scale factor $i = 1, \ldots, 10$ is introduced to control the size of factory and production, where the number of resources $NoR = 10 \times i$ and $o_1 = 45 \times i$, $o_2 = 40 \times i$, $o_3 = 30 \times i$, $o_4 = 20 \times i$ (in tonnes). Accordingly, the set of compatible resources for each recipe type is modified to cope with the increased number of resources while scaling the factory size, where recipes $\{\gamma_1, \gamma_5, \gamma_9, \gamma_{13}\}$, $\{\gamma_2, \gamma_6, \gamma_{10}, \gamma_{14}\}$ and $\{\gamma_3, \gamma_4, \gamma_7, \gamma_8, \gamma_{11}, \gamma_{12}, \gamma_{15}, \gamma_{16}\}$ are compatible with resources $\{\pi_1, \ldots, \pi_{NoR\times 0.5}\}$, $\{\pi_{NoR\times 0.5+1}, \ldots, \pi_{NoR\times 0.7}\}$ and $\{\pi_{NoR\times 0.7+1}, \ldots, \pi_{NoR\times 0.9}\}$, respectively. The rest of resources that are not associated with any recipe (i.e., $\{\pi_{NoR\times 0.9+1}, \ldots, \pi_{NoR\times 1}\}$) are defined as not applicable for producing the given commodities, and hence, will not be considered in manufacturing.

As given in Figure 4, optimisation results of makespan obtained by MOEA/D demonstrate an observable increasing trend while incrementing scale factor i. In contrast, MOEA/D-RS has a lower increasing rate and outperforms MOEA/D with each i. Similar observations are obtained in Figure 5, where MOEA/D-RS again outperforms MOEA/D with each i. However, the increasing rate of results obtained by MOEA/D has a much higher increasing rate, where the minimal amount of over-produced commodities with $i = 10$ reaches 3651 tonnes. However, the results obtained by MOEA/D-RS remains better and effectively limit the amount of over-produced commodities to 289 tonnes with $i = 10$.

This experiment shows that under the considered manufacture optimisation problem, MOEA/D-RS yields a more diverse front (in terms of minimising objectives "makespan" and "sum of discrepancy scores") than that of MOEA/D, and confirms the efficiency of MOEA/D with enlarged size of the studied problem.

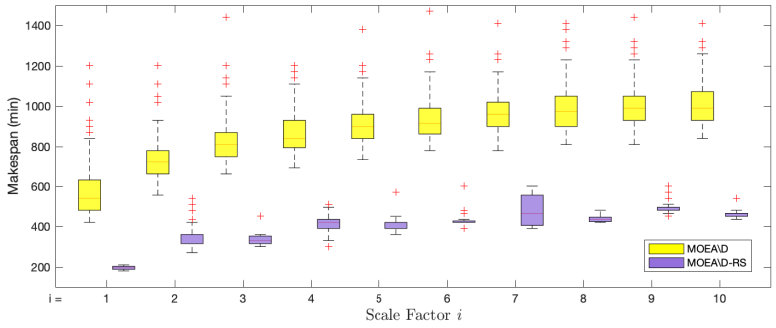

Fig. 4. Make span optimisation results of original MOEA/D and MOEA/D-RS by scaling an example scenario

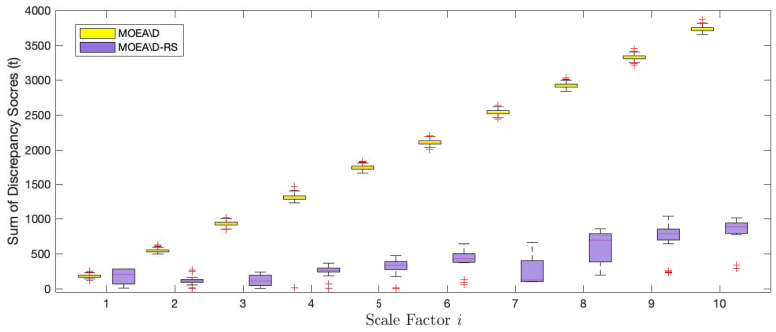

Fig. 5. Sum of discrepancy scores optimisation results of original MOEA/D and MOEA/D-RS by scaling an example scenario

In addition, the performance of the proposed multi-objective optimisation algorithm is not achieved via sacrificing its run-time efficiency, as shown in Figure 6. The execution times of both MOEA/D and the proposed MOEA/D-RS algorithm demonstrate an increasing trend with the increment of the factory size and production requirement. With $i \leq 8$, both algorithms demonstrate similar execution times yet MOEA/D-RS can provide results outperforming that of MOEA/D (see Figures 4 and 5). With $i > 8$, MOEA/D-RS has slightly higher execution time, yet the difference between the optimisation results by two algorithms is further enlarged, especially the sum of discrepancy scores given in Figure 5 for $i = \{9, 10\}$.

Summarising the experiments above, we conclude that compared to generic evolutionary operators, the proposed problem-specific operators have better efficiency for recipe scheduling problem in manufacturing. With the proposed operators, the modified MOEA/D algorithm (i.e., MOEA/D-RS) can outperform both the tested multi-objective genetic algorithms with the given problem sce-

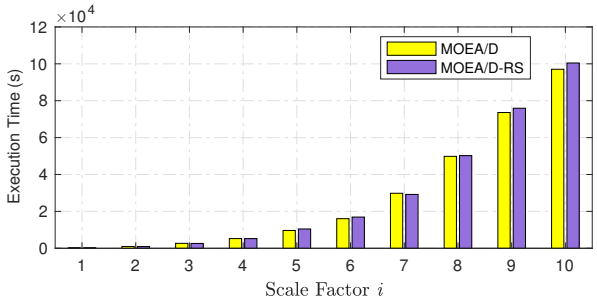

Fig. 6. Execution Times of original MOEA/D and MOEA/D-RS by scaling an example scenario

nario. This observation is further confirmed by an experiment that scales the problem size. From this experiment, we also observe that MOEA/D-RS requires similar execution time in comparison with MOEA/D, but can provide better optimisation results in terms of less time required for production and less amount of overproduced commodities (resulting in a reduced storage cost).

7 Conclusion

In this paper, a real-world factory scheduling problem has been described whose goal is not only to minimise the manufacturing makespan but also to minimise the production surplus via selecting recipes multisubset to be executed. As this problem was difficult to be solved by typical multi-objective genetic algorithms, a modification of MOEA/D has been proposed, which applies customised mutation and elitism operators developed specially for the studied problem. The experiments have demonstrated the superiority of the proposed algorithm (MOEA/D-RS) in comparison with state-of-the-art NSGA-II and original MOEA/D, where for the analysed cases, MOEA/D-RS produces solution that strictly dominates any solution obtained by both NSGA-II and MOEA/D (i.e., DCI values $(0, 0, 1)$ for NSGA-II, MOEA/D and MOEA/D-RS respectively). In particular, the proposed mutation operator improved the results with DCI values $(0, 1)$ of the original MOEA/D and MOEA/D with the customised mutation for all experiments. The proposed algorithm demonstrates similar scalability as the original MOEA/D does in general when being applied to relatively large factories and high quantity of production.

Acknowledgement

The authors acknowledge the support of the EU H2020 SAFIRE project (Ref. 723634).

Bibliography

[1] Ali, C.I., Ali, K.A.: A research survey: review of flexible job shop scheduling techniques. Int. Transactions in Operational Research 23(3), 551–591 (2015)

[2] Ham, A.: Flexible job shop scheduling problem for parallel batch processing machine with compatible job families. Applied Mathematical Modelling 45, 551–562 (2017)

[3] Wang, K., Choi, S.: A holonic approach to flexible flow shop scheduling under stochastic processing times. Computers & Operations Research 43, 157–168 (2014)

[4] Gen, M., Lin, L.: Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey. Journal of Intelligent Manufacturing 25(5), 849–866 (Oct 2014)

[5] Ozguven, C., Ozbakr, L., Yavuz, Y.: Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Applied Mathematical Modelling 34(6), 1539–1548 (2010)

[6] Thomalla, C.: Job shop scheduling with alternative process plans. International Journal of Production Economics 74(1-3), 125–134 (2001)

[7] Ishibuchi, H., Murata, T.: A multi-objective genetic local search algorithm and its application to flowshop scheduling. Trans. Sys. Man Cyber Part C 28(3), 392–403 (Aug 1998)

[8] Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Trans. on Evolutionary Computation 11(6), 712–731 (Dec 2007)

[9] Li, L., Huo, J.Z.: Multi-objective flexible job-shop scheduling problem in steel tubes production. Systems Engineering - Theory & Practice 29(8), 117–126 (2009)

[10] Méndez, C.A., et al.: State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers & Chemical Engineering 30(6-7), 913–946 (2006)

[11] Huynh, N.T., Chien, C.F.: A hybrid multi-subpopulation genetic algorithm for textile batch dyeing scheduling and an empirical study. Computers & Industrial Engineering 125, 615–627 (2018)

[12] Amjad, K.M., et al.: Recent research trends in genetic algorithm based flexible job shop scheduling problems. Mathematical Problems in Engineering pp. 1–32 (2018)

[13] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.). Springer-Verlag, Berlin, Heidelberg (1996)

[14] Indrusiak, L.S., Dziurzanski, P.: An interval algebra for multiprocessor resource allocation. In: Int. Conf. on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS). pp. 165–172 (July 2015)

[15] Miettinen, K.: Nonlinear multiobjective optimization, vol. 12. Springer Science & Business Media (2012)

[16] Deb, K., et al.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6(2), 182–197 (April 2002)

[17] Li, M., Yang, S., Liu, X.: Diversity comparison of pareto front approximations in many-objective optimization. IEEE Trans. on Cybernetics 44(12), 2568–2584 (Dec 2014)