Evolutionary algorithms for multi-objective flexible job shop cell scheduling

Derya Deliktas[a,*], Ender Özcan[a], Ozden Ustun[b], Orhan Torkul[c]

[a] *Computational Optimisation and Learning (COL) Lab, School of Computer Science, University of Nottingham, NG8 1BB, Nottingham, UK*
[b] *Kütahya Dumlupınar Universiy, Faculty of Engineering, Department of Industrial Engineering, Kütahya, Turkey*
[c] *Sakarya University, Faculty of Engineering, Department of Industrial Engineering, Sakarya, Turkey*

Abstract

The multi-objective flexible job shop scheduling in a cellular manufacturing environment is a challenging real-world problem. This recently introduced scheduling problem variant considers exceptional parts, intercellular moves, intercellular transportation times, sequence-dependent family setup times, and recirculation requiring minimization of makespan and total tardiness, simultaneously. A previous study shows that the exact solver based on mixed-integer nonlinear programming model fails to find an optimal solution to each of the 'medium' to 'large' size instances considering even the simplified version of the problem. In this study, we present evolutionary algorithms for solving that bi-objective problem and apply genetic and memetic algorithms that use three different scalarization methods, including weighted sum, conic, and tchebycheff. The performance of all evolutionary algorithms with various configurations is investigated across forty-three benchmark instances from 'small' to 'large' size including a large real-world problem instance. The experimental results show that the transgenerational memetic algorithm using weighted sum outperforms the rest producing the best-known results for almost all bi-objective flexible job shop cell scheduling instances.

Keywords: Genetic algorithms; Memetic algorithms; Cell scheduling; Scalarization methods; Pareto frontier

1. Introduction

In a competitive world-wide market, the manufacturing companies should be more flexible and efficient to be able to respond rapidly to technological innovations and changing customer demands. The system flexibility along with the appropriate choice of planning and operational approaches becomes crucial providing means for effective allocation and use of required resources. Cellular manufacturing has emerged as a complex system that increases the productivity and flexibility of production by reducing the throughput times and enables the manufacturing of a variety of products in small batches (Salmasi, 2005).

In a cellular manufacturing system, the machines are grouped into cells, and parts to be produced are grouped into families. Each regular part is processed in one cell only. Hence, there is no need for the flow of parts between different cells, apart from the exceptional parts. As a consequence, intercell moves caused by the exceptional parts, intercellular transportation times, and material handling costs are reduced. In addition since parts with similar processing requirements are processed in a cell, the setup time, work-in-process inventory, flow time, and production lead time are reduced (Kamrani & Logendran, 1998). A major setup is also required for switching from one part family to another one while processing jobs, although a negligible or minor setup time is usually included within the processing time for each job. The time required for this part family setup may or may not depend on the sequence in which various families are processed. Hence, the cell scheduling problems involving setup times can be divided into two classes: *sequence-independent family setup times* and *sequence-dependent family setup times* (SDFSTs) (Li et al., 2015). In cell scheduling problems,

*Corresponding author: Address: Computational Optimisation and Learning (COL) Lab, School of Computer Science, University of Nottingham, NG8 1BB, Nottingham, UK

Email addresses: deryadeliktas@hotmail.com; derya.deliktas@nottingham.ac.uk (Derya Deliktas), ender.ozcan@nottingham.ac.uk (Ender Özcan), ozden.ustun@dpu.edu.tr (Ozden Ustun), torkul@sakarya.edu.tr (Orhan Torkul)

there are *exceptional parts* that require processing on machines in two or more cells and *re-entrant parts* that are allowed to visit a (group of) machine(s) more than once.

The flexible job shop cell scheduling problem with SDFSTs and intercellular transportation times (FJCS-SDFSTs-ITTs) is the focus of this study. FJCS-SDFSTs-ITTs require minimization of both *makespan*, i.e., the completion time of the last job leaving the system, and the total *tardiness*, i.e., a measure of the delay in completing all jobs. A Pareto front represents a set of the 'optimal' trade-off solutions, for which one of the objectives cannot be improved without sacrificing the other one. In an earlier study, Deliktas et al. (2019) formulated the bi-objective FJCS-SDFSTs-ITTs as a nonconvex optimization problem and applied an exact approach based on a mixed-integer nonlinear programming model. In their study, based on a sample instance, they have illustrated that the Pareto front is not trivial, containing unsupported solutions that are optimal but not on the convex hull of the Pareto front. This implies that the Pareto optimal solutions would have been missed if solving FJCS-SDFSTs-ITTs optimally using linear combinations of the makespan and total tardiness and applying the exact approach since any linear combination will not be able to access the unsupported regions of the Pareto front. Hence, the authors used a conic scalarization method (Kasimbeyli, 2013). However, the exact approach was still unable to solve the 'medium' to 'large' problem instances. The flexible job shop scheduling problem represents a simplified form of FJCS-SDFSTs-ITTs, which is a complex combinatorial problem proven to be NP-hard (Garey et al., 1976). Moreover, Deliktas et al. (2019) applied their exact approach to the 'small' to 'medium' size FJCS-SDFSTs-ITTs problem instances considering minimization of makespan only. They were able to produce optimal results for the 'small' instances while the majority of the 'medium' instances couldn't be solved to optimality even after 24 hours. Metaheuristics are commonly preferred approaches when the exact approaches fail to produce even feasible solutions to the given problem instances (Sörensen & Glover, 2013). *Population-based* metaheuristics, such as evolutionary algorithms using multiple interacting solutions during the search process, are known to be successful in solving real-world problems, particularly that are multi-objective, being able to produce multiple trade-off solutions.

In this study, we present a variety of evolutionary algorithms with novel components for solving the bi-objective FJCS-SDFSTs-ITTs combinatorial optimization problem. In particular, we have investigated Genetic Algorithms (GAs) and Memetic Algorithms (MAs) hybridizing GAs with hill-climbing varying their operators and considering three scalarization methods: weighted sum (WSM), conic (CSM) and Tchebycheff (TSM) to detect the best algorithmic choices. We have also created a benchmark of fourty three bi-objective FJCS-SDFSTs-ITTs problem instances [1] in various sizes ranging from small to large, each with a different characteristic. The forty-two benchmark instances are adopted from the existing problem domains, including the single objective FJCS-SDFSTs-ITTs cell formation and operations routing. The last instance is a real-world problem instance from a locomotive and wagon production factory. The empirical results on the benchmark are extremely encouraging since one of the proposed approaches, namely the memetic algorithm obtained the best-known solutions to all benchmark problem instances, even including the large scale ones.

The structure of this paper is as follows. Section 2 introduces the problem formally and provides an overview of the previous work on cell scheduling problems as well as multi-objective flexible job shop scheduling problems. The proposed EAs are explained in Section 3. Section 4 provides a case study from Turkey and benchmark instances from the literature with a comparison of the performance, and the experimental results and analysis in this section. Finally, Section 5 concludes this study.

2. A flexible job shop cell scheduling problem

2.1. Problem description

Although the multi-objective scheduling problems have been widely studied in recent years, there is far less research in the literature on the multi-objective FJCS problems (including SDFSTs and ITTs or not). Hence, this study fills a gap by investigating different approaches to bi-objective FJCS-SDFSTs-ITTs. In this section, we provide a description of the problem.

Let's assume a flexible job shop environment in cellular manufacturing where c is the set of cells, l is the set of part families, i is the set of jobs, and q is the set of machines. Each job i includes op_i operations and

[1] This benchmark will be made publically accessible via ResearchGate (`https://www.researchgate.net/profile/Derya_Deliktas/publications`)

each operation has its route to follow through the shop. It could be completely or partially different from the processing route of the other jobs. For each operation j of job i, there are identical parallel machines. A job is completed when all of its operations are handled. A cellular manufacturing system includes several manufacturing cells consisting of the different number of machines and allocated for the production of various parts. Each type of machine can process different parts and there are some particular parts referred to as exceptional parts that can move between cells. A work center indicates a production unit where a single or a group of machines can be run at the same time. Identical parallel machines are also allowed. For the reentrant parts, a job may visit a machine or work center more than once requiring the same or a different processing time (Pinedo, 2000).

Each job has the processing time and a due date representing the desired completion time. Processing time includes the intracellular move time. Those moves are considered as independent from the operation sequence while intercellular transfer times are taken into account because of exceptional parts caused the intercellular movements. Moreover, this study deals with the job shop cell scheduling problem with flexible machines and sequence-dependent family setup times which indicate that the family setup times are determined by the current part family as well as the direct previous part family processed on the same machine. Makespan and the total tardiness time are the objectives of the problem, which are to be minimized simultaneously.

A more formal definition of the FJCS-SDFSTs-ITTs problem is provided as follows (Deliktas et al., 2019). Firstly, we have the following assumptions:

(1) The cells, part families, and layout of cells are known.
(2) Machine failures and vehicle failures are ignored.
(3) Input and output buffers are unlimited.
(4) Each operation can be only assigned to one of the eligible parallel machines.
(5) The processing time for operations of each part type on a machine is known and fixed.
(6) The job release times and machine availability times are zero, which means all machines and jobs are available at the beginning of the planning horizon or time zero.
(7) Intracellular transfer times are ignored.
(8) Once an operation is treated on a machine, it can't be interrupted until it is finished (no preemption is allowed).
(9) Intracellular setup times are included in the operation times.
(10) Each machine can process at most one operation at the same time.
(11) All jobs have equal priorities.

We use the following notation for defining FJCS-SDFSTs-ITTs.

q Machine index, $q = 1, 2, \ldots, m$

i Job index, $i = 1, 2, \ldots, n$

c Cell index, $c = 1, 2, \ldots, C$

l Part family index, $l = 1, 2, \ldots, L$

j Operation index for the ith job, $j = 1, 2, \ldots, op_i$

op_i The total number of operations of job i

t_{op} The total number of all operations ($\sum_{\forall i} op_i$)

m The total number of machines

n The total number of jobs

L The total number of part families

C The total number of cells

O_{ij} The jth operation of job i

p_{ijq} The processing time of jth operation of job i if it is processed by machine q

$t_{cc'}$ Transportation time from cell c to cell $c^{'}$

$s_{ll'}$ Setup time for part family $l^{'}$ if processed immediately after part family l

x_{qc} 1, if machine q is located in cell c and 0 otherwise

y_{il} 1, if job i belongs to part family l and 0 otherwise

r_{ijq} 1, if operation j of job i is processed on machine q among eligible machines, and 0, otherwise

d_i The due date of job i

M A large positive number

C_{ijq} The completion time of jth operation of job i on machine q

u_{ijq} 1, if operation j of job i is processed on machine q in a flexible environment, and 0, otherwise

$z_{iji'j'q}$ 1, if jth operation of job i precedes $j^{'}$th operation of job $i^{'}$ on machine q, and 0, otherwise

C_{max} Makespan

T_i The tardiness of job i

The bi-objective problem of FJCS-SDFSTs-ITTs requires optimization of makespan (Equation (1)) and total tardiness (Equation (2)) simultaneously subject to the following constraints.

$$\min\{\ C_{max}\ \},\ \text{and} \tag{1}$$

$$\min\{\ \sum_{i=1}^{n} T_i\ \}, \tag{2}$$

subject to

$$\sum_{q=1}^{m} u_{ijq} = 1, \qquad \forall i,j \tag{3}$$

$$u_{ijq} \leq r_{ijq}, \qquad \forall i,j,q \tag{4}$$

$$\sum_{q=1}^{m} u_{ijq}.C_{ijq} \geqslant \sum_{q=1}^{m}\sum_{q'=1}^{m} u_{ijq}.u_{i(j-1)q'}.(C_{i(j-1)q'} + p_{ijq} + \sum_{c=1}^{C}\sum_{c'=1}^{C} x_{qc}.x_{q'c'}.t_{c'c}), \qquad \forall i, j \geq 2 \tag{5}$$

$$C_{ijq} \geqslant C_{i'j'q} + \sum_{l=1}^{L}\sum_{l'=1}^{L} y_{i'l'}.y_{il}.s_{l'l} - M.z_{iji'j'q} + p_{ijq} - M(2 - u_{ijq} - u_{i'j'q}),\ \forall i \geq 2, j, q, i^{'}, j^{'}, i \neq i^{'} \tag{6}$$

$$C_{i'j'q} \geqslant C_{ijq} + \sum_{l=1}^{L}\sum_{l'=1}^{L} y_{il}.y_{i'l'}.s_{ll'} - M.(1 - z_{iji'j'q}) + p_{i'j'q} - M(2 - u_{ijq} - u_{i'j'q}),\ \forall i \geq 2, j, q, i^{'}, j^{'}, i \neq i^{'} \tag{7}$$

$$C_{ijq} \geqslant p_{ijq}, \qquad \forall i,j,q \tag{8}$$

$$C_{max} \geqslant C_{ijq}, \qquad \forall i,j,q \tag{9}$$

$$T_i \geqslant C_{ijq} - d_i, \qquad \forall i,j,q \tag{10}$$

$$C_{ijq},\ C_{i'j'q},\ T_i \geqslant 0,\ z_{iji'j'q},\ u_{ijq} \in \{0,1\},\qquad \forall i,j,q,i^{'},j^{'} \tag{11}$$

Equation (3) forces exactly one machine alternative to be selected for each operation j of job i. Equation (4) determines the capable machines for each operation. r_{ijq} determines whether the operation j of job i is processed on machine q or not. For example, if the j^{th}operation of job i does not get processed on machine 4, the r_{ij4} parameter is set to 0 in Equation (4). The value of u_{ijq} depends on the value of r_{ij4} and in this case, u_{ij4} also gets assigned to 0. Equation (5) ensures that each part is processed depending on the defined precedence of its operations and intercellular transportation times. Equations (6) and (7) guarantee that at most, one part is processed by each machine at a time considering the setup times for the part family. Equation (8) imposes that the completion time of an operation cannot be less than its processing time. Equations (9) and (10) calculate the makespan and the tardiness of each job, respectively. Equation (11) defines the binary decision variables.

For illustration purposes, let's use the FJCS-SDFSTs-ITTs problem instance#5 (see Table 5) as an example, for further explanation of our problem. As characterized in Tables 1 and 2, there are 4 parts and 4 machines in instance#5. The parts are grouped into 2 part families, each containing 2 items. The machines are also grouped into 2 cells, both containing 2 machines.

Table 1. Part-machine matrix, processing times, and routing

		Cell 1		Cell 2		Routing	Due Dates
		M_1	$M_2(2)^*$	$M_3(2)$	M_4		
Part family 1	P_1	4**	5	6		1-**2**-3-**2**	98
	P_2	3	4			**2**-1-**2**	77
Part family 2	P_3		7	2	4	4-2-3	3
	P_4			6	2	**3**-4-**3**	75

* Value in parentheses shows that there are two identical parallel machines for machines 2 and 3 if the problem is in the flexible environment. There is no available identical machine in the job shop environment.

** M_1 is an eligible machine for the $1st$ operation of job 1, since this operation is processed on machine 1. Thus, the r_{111} parameter is set to 1 in Equation (4). P: Part, M: Machine

As an example, Table 1 shows that P_2 with the due date of 77 belongs to the first part family requiring a processing time of 4 on M_2. The machine M_2 is in the first cell and there are two identical parallel machines. The operations for manufacturing the part P_2 should be routed through the machines M_2, M_1, and then M_2 in that order. Although this is not relevant to P_2, Table 2 contains the required transportation times for the movement of operations between cells. For example, if an operation goes from the first cell to the second cell, that would require 3 units of time. The machine setup times are also provided in the table. So, for example, if a part P_2 operation is scheduled to a machine that just completes the processing of operation for a part belonging to the second part family, then a setup time of 4 units of time will be required by that machine.

Table 2. Family setup times for each part family and transportation times for each cell

Transportation times: from cell c to cell $c^{'}$ row c, column $c^{'}$			*Setup times*: from part family l to cell $l^{'}$ row l, column $l^{'}$		
$cc^{'}$	Cell 1	Cell 2	$ll^{'}$	Part family1	Part family2
Cell 1	—	3	Part family1	—	3
Cell 2	4	—	Part family2	4	—

Figure 1 provides a feasible solution to instance#5. In the figure, each operation for the jobs 1, 2, 3, and 4 are indicated as solid red, solid black, dashed red, and dashed black line, respectively. The arrows show the direction of the routing sequence. The processing time of an operation is displayed above each line. The transportation and setup times are also shown over the forklift and gear icons, respectively. Let's illustrate how we compute the objective values based on that solution starting with the partial fitness calculation for part 1 (P_1). Reading the data depicted in Figure 5 from left to right, it is observed that O_{11} is the second job of M_1 as shown in Fig. 2 (a). Since it is in the same part family with the first job (P_2) of M_1, part

family setup time between them is zero. Therefore, P_1 processes on M_1 with just its processing time. For the second process of this part, it is attained to M_2^2 which is one of the identical parallel machines. This part visits M_2^2 after P_2 completes. After that, it is transferred to M_3^1 which is one of the identical parallel machines in Cell 2 and the first part of M_3^1 which will be processed. In addition to its processing time, there is also transportation time between cells as shown in Fig. 2 (b). As the fourth process, P_1 visits M_2^1 in Cell 1 which P_3 was processed before. In addition to its processing time, both transportation and part family setup times are added to the objective value of makespan (see Eq. 1) because there is both intercellular movement between cells and setup process between part families. According to this, makespan is obtained as 34 as shown Fig. 2. The tardiness (see Eq. 2) of each job is computed considering the completion time of each job and its due date. The tardiness of the third job (part) is computed as 17 because it is completed at 20 (see Fig. 2 (b)) and its due date is 3 (see Table 1), while the other jobs do not have any tardiness. The total tardiness objective value is obtained with the tardiness of all jobs which yields 17 for this problem instance.

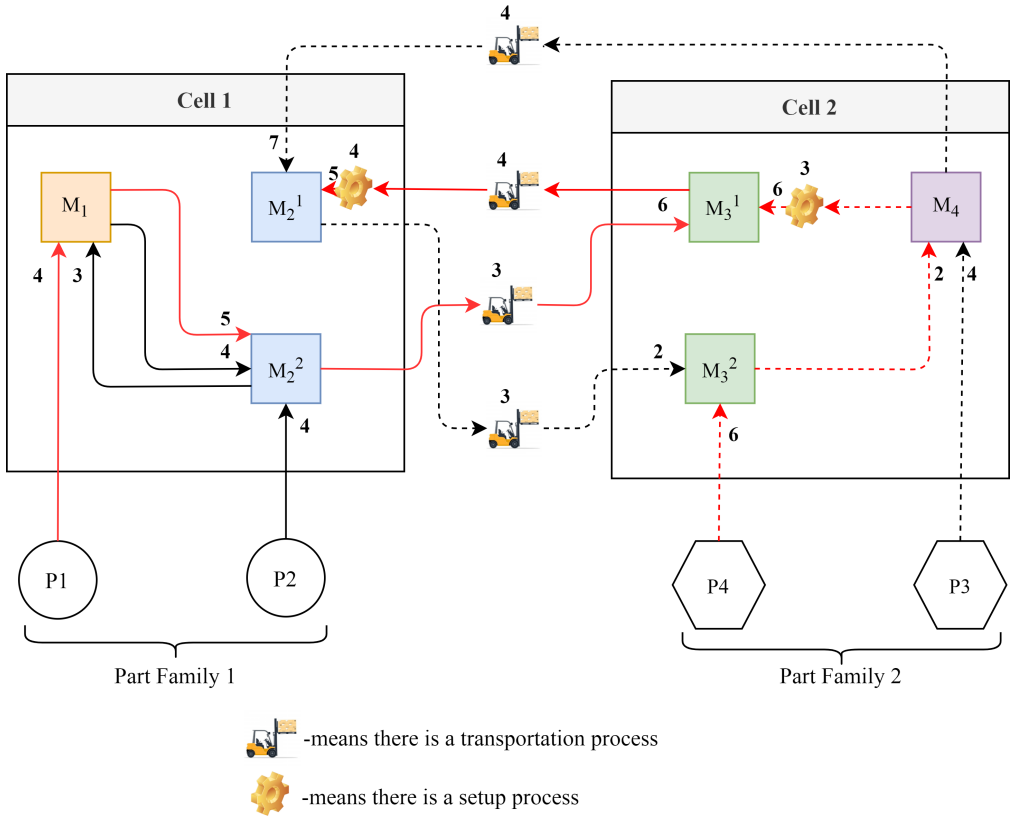
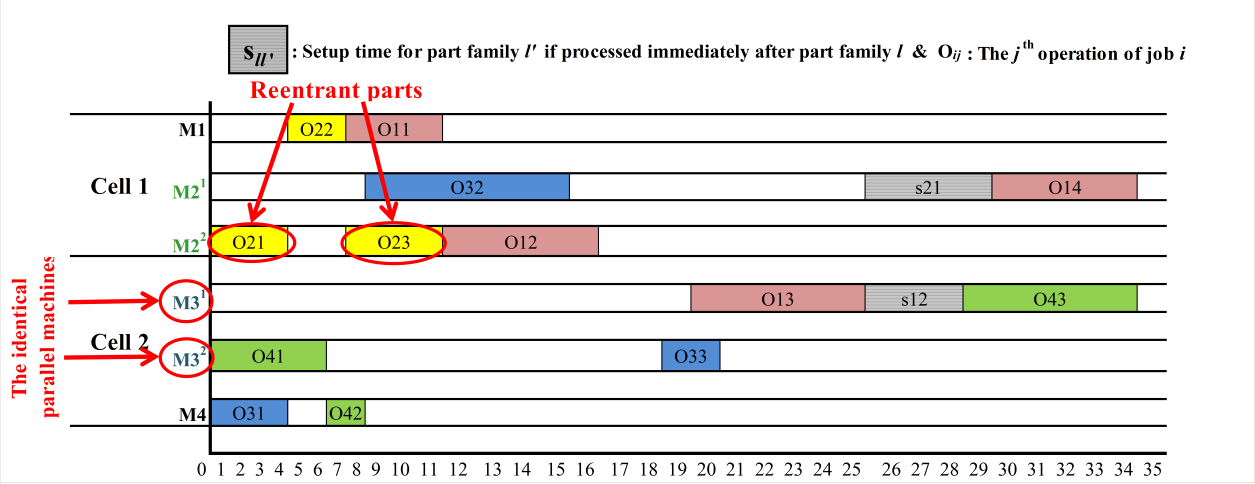


Figure 1: Feasible solution for the illustrative example

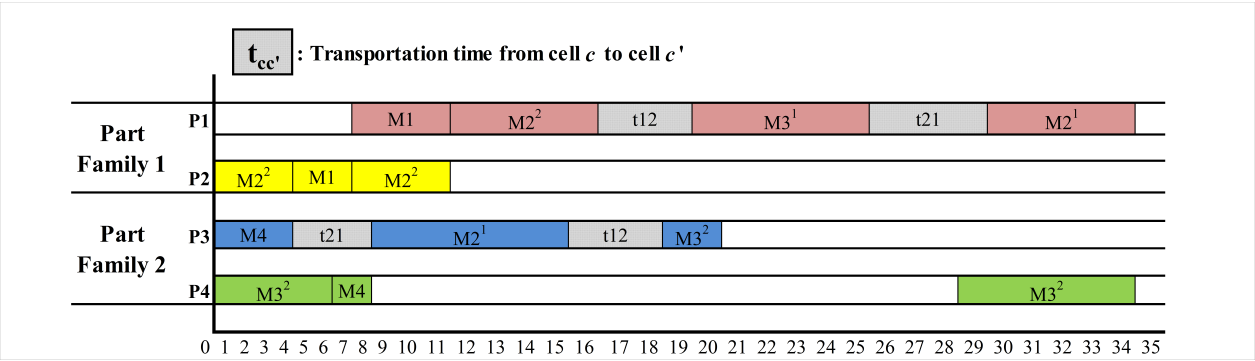
2.2. Related work

The flexible job shop cell scheduling (FJCS) problem requires a search for the best sequence of part families and individual parts within each cell for manufacturing. The bi-objective FJCS-SDFSTs-ITTs is a variant of the FJCS problem that introduces additional problem complexities, including setup-dependent/independent family times, intercellular transportation times, reentrant parts, and flexible routes. This multi-objective problem is not investigated in sufficient depth yet in the field. However, there are many studies on various solution methods considering multiple objectives for FJCS, as summarised in Table 4.

Table 3 shows that in almost all multi-objective flexible job shop scheduling problems makespan is the main objective taken into account, except (Li & Huo, 2009; Piroozfard et al., 2018). The majority of the studies in our sample survey present solution methods optimizing the objectives of makespan, the total workload of machines, and critical machine workload simultaneously. As for the solution methods, a variety of



(a)



(b)

Figure 2: The Gantt charts representing the solution (in Figure 1) for instance#5 based on (a) machine and (b) parts.

evolutionary algorithms has been used for tackling each particular multi-objective flexible job shop scheduling problem.

During the past two decades, there are only a few studies on the FJCS problem. As seen in Table 3, four out of six studies belonging to the FJCS problem have a single objective. De Giovanni & Pezzella (2010), Lu et al. (2018), Wu et al. (2017), and Chang & Liu (2017) proposed different variants of genetic algorithm with minimization of the makespan for solving distributed and flexible job-shop scheduling (DFJS) problems which involve three scheduling decisions such as job-to-cell assignment, operation sequencing, and operation-to-machine assignment. Ziaee (2014) developed a heuristic algorithm for solving the DFJS problem minimizing the makespan. Only one study considers two objectives with minimization the weighted makespan and the total traveling distance (Kesen et al., 2010). Table 4 provides a brief overview of the previous work on cell scheduling from the scientific literature. This overview shows that most of those previous studies focus on the flow shop cell scheduling problem and only a few of them consider the multi-objective variant of those problems. Metaheuristics, including genetic algorithms, simulated annealing, tabu search, and their hybrids are commonly applied to those multi-objective cell scheduling problems as the solution methods. However, none of the previous work performed an investigation as substantial as ours looking into the particular bi-objective FJCS-SDFSTs-ITTs problem and comparing the performance of genetic as well as memetic algorithms.

In this study, we have designed four evolutionary algorithms for the bi-objective FJCS-SDFSTs-ITTs problem to minimize both makespan and the total tardiness times. In multi-criteria decision making, the

Table 3. An overview of previous (meta)heuristic optimization algorithms applied to the multi-objective flexible job shop scheduling

Ref.	Objective Functions																		Solution methods	Descriptions
	C_{max}	TWM	CMW	MMW	S	E&T	RC	TPC	TCF	TLWC	EC	NE	TT	RT	STD&TD	ST	WT	PC		
Feng et al. (2018)	✓	✓																	Three-layer chromosome GA	Dynamic cellular manufacturing, inter-cell movement
Lu et al. (2017)	✓						✓												Multi-objective discrete virus optimization algorithm	Controllable processing times
Ahmadi et al. (2016)	✓				✓														Non-dominated Sorting Genetic Algorithm (NSGA-II) and non-dominated ranking genetic algorithm	Random machine breakdown
Anvari et al. (2016)	✓							✓											MOGA	Resource-constrained scheduling
Kaplanoğlu (2016)	✓	✓		✓															Simulated annealing (SA) algorithm	
Piroozfard et al. (2018)									✓	✓									MOE algorithm	Resource-constrained scheduling
Yin et al. (2017)	✓										✓	✓							MOGA	Simplex lattice design
Shen & Yao (2015)	✓			✓	✓								✓						ϵ-domination based steady-state MOE algorithm	Random events in dynamic flexible job shops
Gao et al. (2014)	✓					✓													Pareto-based grouping discrete harmony search (HS) algorithm	Local search methods
Jia & Hu (2014)	✓	✓		✓															Multi-objective tabu search (TS) algorithm	Back-jump tracking, path-relinking
Li et al. (2014a)	✓	✓	✓																Discrete artificial bee colony algorithm	Preventive maintenance constraints non-maintenance constraints, TS based local search
Shahsavari-Pour & Ghasemishabankareh (2013)	✓	✓		✓															Hybrid GA/SA algorithm	
Xiong et al. (2013)	✓													✓					MOE algorithm	Random machine breakdowns
Li et al. (2012)	✓	✓	✓																Shuffled frog-leaping algorithm	Local search method
Li & Pan (2012)	✓	✓	✓																Discrete chemical-reaction optimization algorithm	Preventive maintenance constraints non-maintenance constraints, TS based local search
Moslehi & Mahnam (2011)	✓	✓	✓																Multi-objective particle swarm optimization algorithm (PSO)	Local search method
Fattahi & Fallahi (2010)	✓														✓				GA	
Li et al. (2010)	✓	✓	✓																Hybrid TS algorithm	Variable neighborhood search
Li & Huo (2009)																✓	✓		Modified GA	The parallel machines with capacity, speed constraint, maintenance of machines
Xing et al. (2009)	✓	✓	✓																Simulation modelling	
Zhang et al. (2009)	✓	✓	✓																Integrated PSO/TS algorithm	
Saad et al. (2008)	✓	✓	✓			✓												✓	MOGA	The Choquet Integral aggregative methods, ordered weighted averaging operators
Xia & Wu (2005)	✓	✓	✓																Hybrid PSO/SA algorithm	
Kacem et al. (2002)	✓	✓		✓															Hybrid fuzzy logic and evolutionary algorithm	

C_{max}: Makespan, TWM: Total workload of machines, CMW: Critical machine workload, MMW: Maximal machine workload, S:Stability, E&T: The mean of earliness and tardiness time, RC: Resource consumption, TPC: Total project cost, TCF:Total carbon footprint, TLWC: Total late work criterion, EC: Energy consumption, NE: Noise emission, TT: Total tardiness, RT: Robustness time, STD&TD: Starting time deviation&total deviation, ST: The total setup time on machines, WT: Waiting time of jobs, PC: The production cost

Table 4. A brief survey of studies on cell scheduling problems

Ref.	Type of problem	Solution methods	Number of Objective Functions			Descriptions
			Single	Bi	Multi	
Delgoshaei et al. (2016)	Cell scheduling	MM & Hybrid GA and SA			✓	Taguchi method, inter and intracell moves, cost uncertainty
Li et al. (2013)	PS problem in CMS	MM & Pheromone based approach			✓	Intercell move times, flexible routes, multi-agents
Solimanpur & Elmi (2013)	JSCS problem	MM & Nested TS algorithm	✓			SIFSTs, intercell moves
Karthikeyan et al. (2012)	JSCS problem	SA & TS	✓			
Shen & Buscher (2012)	JSCS problem	TS	✓			SDFSTs
Elmi et al. (2011)	JSCS problem	MM & SA	✓			Intercell move times, reentrant parts, SDFSTs
Tang et al. (2010)	JSCS problem	MM & Scatter search approach	✓			Intercell move times
Tavakkoli-Moghaddam et al. (2010)	JSCS problem	PSO algorithm		✓		Intercell move times
Zeng et al. (2015)	JSCS problem	MM & GA	✓			Intercell move times
Tang et al. (2014)	JSCS problem	MM & Lagrangian relaxation decomposition method	✓			Intercell move times
Solimanpur et al. (2004)	JSCS problem	SVS algorithm	✓			SIFSTs, intercell moves
Lu et al. (2018)	FJCS problem	GA	✓			Intercell move times, flexible routes
Wu et al. (2017)	FJCS problem	GA	✓			Flexible routes
Chang & Liu (2017)	FJCS problem	Hybrid GA	✓			Taguchi method, flexible routes
Ziaee (2014)	FJCS problem	Heuristic algorithm	✓			Good initial solution, flexible routes
De Giovanni & Pezzella (2010)	FJCS problem	GA	✓			Flexible routes
Kesen et al. (2010)	FJCS problem	MM & GA		✓		Virtual cell
Elmi & Topaloglu (2016)	FSCS problem	ACO	✓			
Costa et al. (2017)	FSCS problem	GA & random sampling search methods	✓			SDFSTs
Li et al. (2015)	FSCS problem	MM & HS		✓		SDFSTs, local search method
Balaji & Porselvi (2014)	FSCS problem	AIS & SA algorithm	✓			SDFSTs
Ebrahimi et al. (2014)	FSCS problem	NSGAII & MOGA		✓		SDFSTs, uncertain due dates
Ibrahem et al. (2014)	FSCS problem	GA & PSO	✓			SDFSTs, DOE
Hendizadeh et al. (2008)	FSCS problem	TS	✓			SDFSTs
Logendran et al. (2005)	FSCS problem	Heuristic algorithm	✓			SDFSTs, DOE, flexible routes
França et al. (2005)	FSCS problem	GA & MA	✓			SDFSTs, local search method
Schaller et al. (2000)	FSCS problem	Heuristic algorithm	✓			SDFSTs
Shahvari & Logendran (2016)	Hybrid FSCS problem	MM & TS			✓	SDFSTs
Logendran et al. (2006)	Flexible FSCS problem	TS	✓			Initial solution finding mechanism, SDFSTs

MM: Mathematical model, ACO: Ant colony optimization, SA: Simulated Annealing, PSO: Particle Swarm Optimization, GA: Genetic Algorithm, TS: Tabu Search,
AIS: Artificial immune system, SIFSTs: Sequence-independent family setup times, FSCS: Flow shop cell scheduling, JSCS: Job shop cell scheduling, PS:Part scheduling,
DOE: Design of Experiments, WT: Waiting time of jobs, PC: The production cost, MA: Memetic Algorithm

weighted sum is a commonly used scalarization method mapping multiple objective values into a single objective value that guides the metaheuristic search process. Table 4 also shows that due to the lack of studies on multi-objective variants of the cell scheduling problem, there are only a few studies considering different scalarization methods in the area. None of those studies are on the flexible job shop scheduling problem in a cellular manufacturing environment with consideration of exceptional parts, intercellular moves, intercellular transportation times, sequence-dependent family setup times, and recirculation. As a disadvantage, the weighted sum scalarization method might not obtain all of the Pareto-optimal points for a non-convex optimization problem. A different scalarization method was considered reaching both supported and unsupported efficient solutions based on non-convex multi-objective models solving small and some medium-scale problem instances in Deliktas et al. (2019). However, to the best of our knowledge, none of the previously proposed algorithms are applied even to the large-scale FJCS-SDFSTs-ITTs instances as we do in this study.

3. Evolutionary algorithms for solving the bi-objective FJCS-SDFSTs-ITTs

FJCS-SDFSTs-ITTs is an NP-hard problem even in its simplified form (Garey et al., 1976). It has been observed that obtaining even a near-optimal solution could be extremely time-consuming and sometimes not even possible using an exact approach, such as a mathematical model, especially to the large-scale problem instances as shown in Deliktas et al. (2019) which considered minimization of makespan. Many researchers and practitioners resort to (meta)heuristic optimization methods in such situations to obtain a solution to a problem instance of reasonable quality in a reasonable amount of time. Evolutionary algorithms (EAs) are nature-inspired population-based metaheuristics (Sörensen & Glover, 2013), designed based on the principles of evolution and the concept of survival of the fittest. They have been successfully applied to a range of combinatorial optimization problems (Sörensen & Sevaux, 2006; Beasley & Chu, 1996). In this study, we investigate the performance of two classes of EAs, genetic and memetic algorithms for the bi-objective FJCS-SDFSTs-ITTs (Holland et al., 1992; Moscato et al., 1989). The description of those algorithms and their components are presented in the following sections.

3.1. Genetic and Memetic Algorithms

A generic genetic algorithm (GA) performs the search for the (near-)optimal solution(s) using a *population* (set) of candidate solutions which are referred to as *individuals* (or *chromosomes*) as illustrated in Algorithm 1. An initial randomly generated population of $nPop$ solutions (Pop_{init}) goes through a number of evolutionary cycles (generations) until the termination criteria are satisfied with the hope that the best solution in the final population is the (near-)optimal solution. The *fitness function* (objective function) evaluates the quality of a candidate solution, providing means for the comparison of solutions and guidance for the search process. At each cycle, a *mating pool* is formed using a *mate selection* method and then a pair of individuals (*parents*) from that pool undergo the *crossover* operation which exchanges genetic material between two solutions creating two new solutions, namely *offspring* (*children*). Then both offspring are perturbed using *mutation* and added to the *offspring pool*. This pool forms the next generation of solutions and a *replacement* method is used to choose which individuals from the offspring pool and current generation survive to the next generation. A memetic algorithm (MA) is an extension to GA, hybridizing GA with hill-climbing. MA applies hill-climbing to both children for exploitation after mutation (see line 11 of Algorithm 1). Those population-based algorithms have been shown to be successful in solving many real-world discrete optimization problems (Wu & Chow, 1994; Yang et al., 2008; Nalepa & Czech, 2013; Nalepa & Kawulok, 2014; Nalepa & Blocho, 2015; Nalepa et al., 2015). Hence, we tested both approaches for solving the discrete optimization problem of FJCS-SDFSTs-ITTs.

A variety of combinations of algorithmic components is studied to identify the best configuration for both genetic and memetic algorithms. The following sections provide the details for the domain-specific design of those components.

3.2. Representation

The first step in designing a metaheuristic is to represent a candidate solution for the problem. A chromosome in our representation is composed of two parts: (*i*) operation sequence vector denoted as v_1 indicating the sequence of operations for each job, and (*ii*) machine assignment vector denoted as v_2 indicating sequence of machine IDs selected to process the operations. Based on this information, we can generate the

Algorithm 1 Pseudo-code of the Evolutionary Algorithm

Randomly generate an initial population Pop_{init} of $nPop$ individuals
Calculate the fitness of each individual (see Eqs (12, 13, 14))
$Pop_{current} \leftarrow Pop_{init}$
while termination criteria not satisfied **do**
    while $OffspringPool$ is not full **do**
        Parent$_1$ $\leftarrow$ Select-Mate ($Pop_{current}$)
        Parent$_2$ $\leftarrow$ Select-Mate ($Pop_{current}$)
        Child$_1$, Child$_2$ $\leftarrow$ Apply-Crossover (Parent$_1$, Parent$_2$)
        Child$'_1$ $\leftarrow$ Apply-Mutation (Child$_1$)
        Child$'_2$ $\leftarrow$ Apply-Mutation (Child$_2$)
        Calculate the fitness of Child$_1$ and Child$_2$ (see Eqs (12, 13, 14))
        $OffspringPool \leftarrow$ Add(Child$'_1$, Child$'_2$)
    end while
    $Pop_{current} \leftarrow$ Apply-Replacement($Pop_{current}$, $OffspringPool$)
end while
Return the best solution from $Pop_{current}$

part family vector denoted as v_3 indicating part family belonging to the corresponding job, and cell assignment vector denoted as v_4 representing the cell belonging to the selected machine. From this point onward, when we use the term gene, that will refer to the locus from a given vector and allele will refer to the assigned value. As an example, for instance#5, let's assume that the manufacturing of each part i (P_i) is the job that is indicated as J_i and there are 13 operations to be scheduled. J_1 has four operations O_{11}, O_{12}, O_{13}, and O_{14}; J_2 has three operations O_{21}, O_{22}, and O_{23}; J_3 has three operations O_{31}, O_{32}, and O_{33}; J_4 has three operations O_{41}, O_{42}, and O_{43} (see *2.1 Problem description* section).

3.2.1. Operation sequence vector

In this study, operation-based representation is used where an ordered array of integers indicates the operation that will be processed next for the given job (Zhang et al., 2011). Hence, the length of the chromosome equals to the total number of operations. Thus, the chromosome includes 13 genes encoding the operation sequence. The index i of job J_i appears in the operation sequence vector (v_1) n_i times to represent its n_i ordered operations. For example, the first appearance of number 2 represents O_{21}, the second appearance of 2 means O_{22}, and so on. Moreover, because of the existence of precedence constraints, using such representation provides a means to avoid repair generating feasible solutions.

For the instance#5, one possible solution encoding the operation sequence vector is shown in Fig. 3. Reading the data from left to right and increasing the operation index for each job, the operation sequence depicted in Fig. 3 could be interpreted into a list of ordered operations as follows:

$O_{21} \succ O_{41} \succ O_{22} \succ O_{31} \succ O_{11} \succ O_{32} \succ O_{23} \succ O_{12} \succ O_{42} \succ O_{13} \succ O_{33} \succ O_{43} \succ O_{14}$

Positon: r	1	2	3	4	5	6	7	8	9	10	11	12	13
Job indicated	J_2	J_4	J_2	J_3	J_1	J_3	J_2	J_1	J_4	J_1	J_3	J_4	J_1
Operation sequence: $v_1(r)$	2	4	2	3	1	3	2	1	4	1	3	4	1
Operation indicated	O_{21}	O_{41}	O_{22}	O_{31}	O_{11}	O_{32}	O_{23}	O_{12}	O_{42}	O_{13}	O_{33}	O_{43}	O_{14}

Figure 3: An Illustration of a operation sequence vector as a part of the solution representation for the problem instance#5

3.2.2. Machine assignment vector

To generate a feasible schedule, all machines are assigned for a job following the machine assignment vector. Hence, the length of this vector also equals to the total number of operations. Each entry of the machine assignment vector v_2, $v_2(s)$ represents the selected machine for the corresponding operation indicated

at position s in the operation sequence vector. In Fig. 4, for instance, position 2 represents the O_{41}, and $v_2(s)$ indicates the machine assigned for O_{41}. For operation O_{41}, two machines are eligible to assign: machines M_3^1 and M_3^2. These are duplicate machines to manufacture a product in the flexible environment. For example, M_3^1 indicates the first identical parallel machine for M_3 while M_3^2 represents the second identical parallel machine for M_3. The IDs 1 and 2 indicated in each machine assignment vector v_2 show the array of the identical parallel machine set for each operation. If there is no identical parallel machine available for the corresponding machine, the allele for the corresponding machine should be 1 as given in Fig. 4. This assignment representation supports recirculation as well.

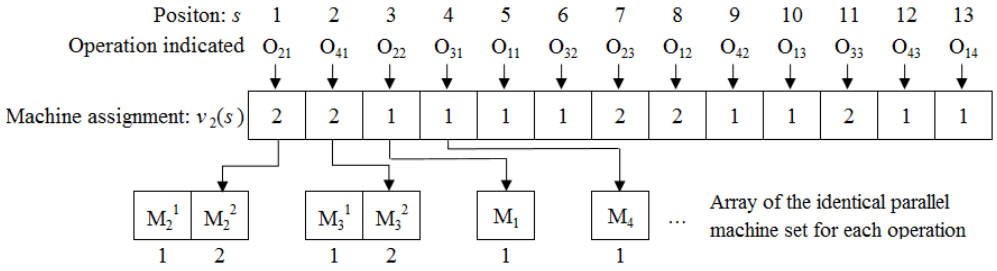


Figure 4: An illustration of a machine assignment vector as a part of the solution representation for the problem instance#5

3.2.3. Part family and cell assignment vectors

Two additional vectors v_3 and v_4 are used as a supporting data structure for the calculation of fitness. Those vectors are provided as input to our approach defining a problem instance and do not change during the search process. $v_3(t)$ contains the part family of each job at locus t in $v_1(t)$. $v_4(t)$ points to the cell that the selected machine at locus t belongs to. In Fig. 5, for instance, $v_1(6)$ represents the operation O_{32} belonging to the job/part 3 assigned to M_2^1 as indicated by $v_2(6)$. As for the supporting data structures since P_3 belongs to the part family 2, $v_3(6)$ which is set to 2 (Table 1). Also, M_2^1 belongs to cell 1, hence $v_4(6)$ is set to 1 (Table 1). After creating a new solution using any of the algorithmic components, if there is any change to a machine assignment at a locus, the supporting data structure v_4 gets updated at that locus, accordingly.

Positon: t	1	2	3	4	5	6	7	8	9	10	11	12	13
Operation sequence: $v_1(t)$	2	4	2	3	1	3	2	1	4	1	3	4	1
Machine assignment: $v_2(t)$	2	2	1	1	1	1	2	2	1	1	2	1	1
Machine selected	M_2^2	M_3^2	M_1	M_4	M_1	M_2^1	M_2^2	M_2^2	M_4	M_3^1	M_3^2	M_3^1	M_2^1
Part family assignment: $v_3(t)$	1	2	1	2	1	2	1	1	2	1	2	2	1
Cell assignment: $v_4(t)$	1	2	1	2	1	1	1	1	2	2	2	2	1

Figure 5: An illustration of the solution representation (v_1, v_2) and supporting data structures (v_3, v_4) for instance#5

3.3. Fitness function

After computing the objective values for makespan and the total tardiness time, the fitness values are calculated using a scalarization method based on those values. We experimented with three different fitness functions using weighted sum (WSM), conic (CSM), and Tchebycheff (TSM).

WSM (Miettinen, 2012; Ehrgott et al., 2014) is one of the well-known scalarization methods which can be used to obtain Pareto efficient solutions to a multi-objective optimization problem. Based on a given solution, multiple objective values are crashed into a single objective value by computing their weighted sum.

$$\min \{ w_1.C_{max} + w_2.\sum_{i=1}^{n} T_i \} \tag{12}$$

where w_1 and w_2 represent the importance weights for makespan (Equation 1) and tardiness (Equation 2), respectively, and $w_1 + w_2 = 1$ assuming $w_1, w_2 \geqslant 0$, and Equations (3)-(11) are all satisfied.

CSM introduced by Gasimov (2001) is based on supporting the image set of the problem by using cones instead of the hyperplanes used in weighted scalarization (Kasimbeyli, 2013). The scalarization of makespan and total tardiness is computed using the following equation:

$$\min \{ w_1.(C_{max} - Rf_1) + w_2.(\sum_{i=1}^{n} T_i - Rf_2) + \alpha.(|C_{max} - Rf_1| + |\sum_{i=1}^{n} T_i - Rf_2|)\} \quad (13)$$

subject to Equations (3)-(11), where $w_1, w_2 > 0$ are the importance degree of makespan and total tardiness, respectively, α is used to change the apical angle of a supporting cone, $0 \leqslant \alpha < min\{w_1, w_2\}$, and Rf_1 and Rf_2 are arbitrarily fixed reference points/values from a certain interval for makespan and total tardiness, respectively, in this study.

TSM was first presented in Steuer & Choo (1983) and Steuer & Steuer (1986), suggesting the use of a Tchebycheff metric based on reference points for the scalarization of multiple objective values. The previous work shows that if the reference points are feasible, the minimization of *achievement function* produces a solution that maximizes the distance to the Pareto optimal set. Otherwise, the minimization of achievement function produces a solution that minimizes the distance to the Pareto optimal set. The advantage of achievement function is that any arbitrary weakly Pareto optimal or Pareto optimal solution can be obtained by moving the reference points only. Wierzbicki (1986) discovered that the solution to the achievement function depends on Lipschitz-continuity of the reference points. Although many achievement functions are satisfying the imposed conditions (Miettinen, 2012), the one used in this study is as follows:

$$\min \{ \max \{ w_1.(C_{max} - Rf_1), w_2.(\sum_{i=1}^{n} T_i - Rf_2) \} + \rho.((C_{max} - Rf_1) + (\sum_{i=1}^{n} T_i - Rf_2)) \} \quad (14)$$

subject to Equations (3)-(11), where $w_1, w_2 > 0$ are the importance degree of makespan and total tardiness, respectively, $\rho > 0$ is a sufficiently small positive scalar, and Rf_1 and Rf_2 are arbitrarily fixed reference points/values from a certain interval for makespan and total tardiness, respectively, in this study.

3.4. Operators

All algorithmic components/operators specific to the FJCS-SDFSTs-ITTs problem are briefly explained in this section, including the parent selection, crossover, mutation, hill climbing, and replacement (Azadeh et al., 2017).

3.4.1. Crossover

As a mate selection method, the generic tournament method with a tour size of two is used in all evolutionary algorithms, and crossover is applied using the selected solutions, i.e., the crossover probability is 1.0. The possibility of crossover creating an infeasible offspring is eliminated by using tailored crossover operators. We implemented three different crossover operators: precedence preserving order-based crossover operator, order-based uniform crossover operator, and order-based one-point crossover operator, which are described below.

Precedence preserving order-based crossover (POX): We have used POX as one of the crossover operators which was proposed by Lee et al. (1998). This operator has been used in many previous studies as well (Zhang et al., 2011; Deng et al., 2017; Rahmati et al., 2013; Wang et al., 2012). Fig. 6 illustrates how POX works. A subset of jobs is randomly selected for an offspring. The operations belonging to that subset of jobs from the first parent (p1) are copied into the first offspring (o1) without changing their locations along with the machine assignments. For example, in Fig. 6, for the first offspring J_4 and J_3 are selected. Then the operations belonging to the selected jobs are deleted from the second parent (p2). Finally, all the remaining operation and machine assignment pairs from the second parent (p2) are copied into the empty locations into the first offspring in the same order. This process is repeated for the second offspring in the same manner.

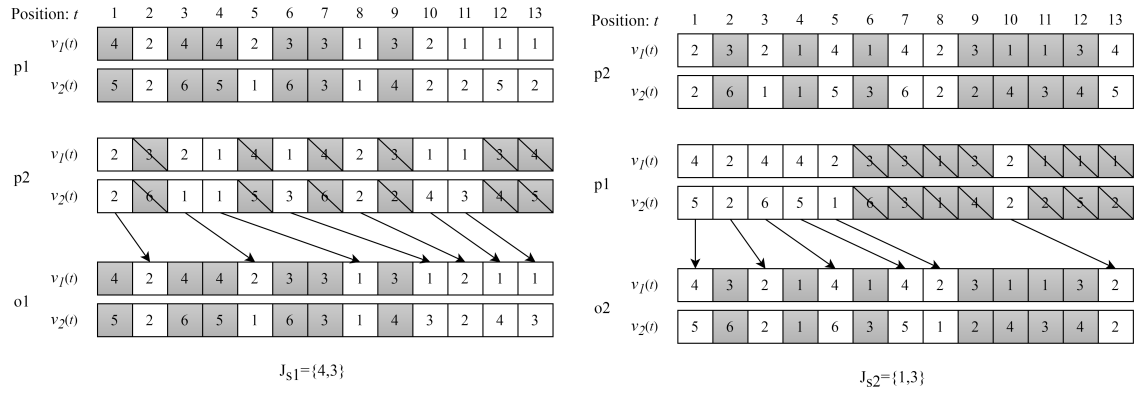


Fig. 6: An example application of POX

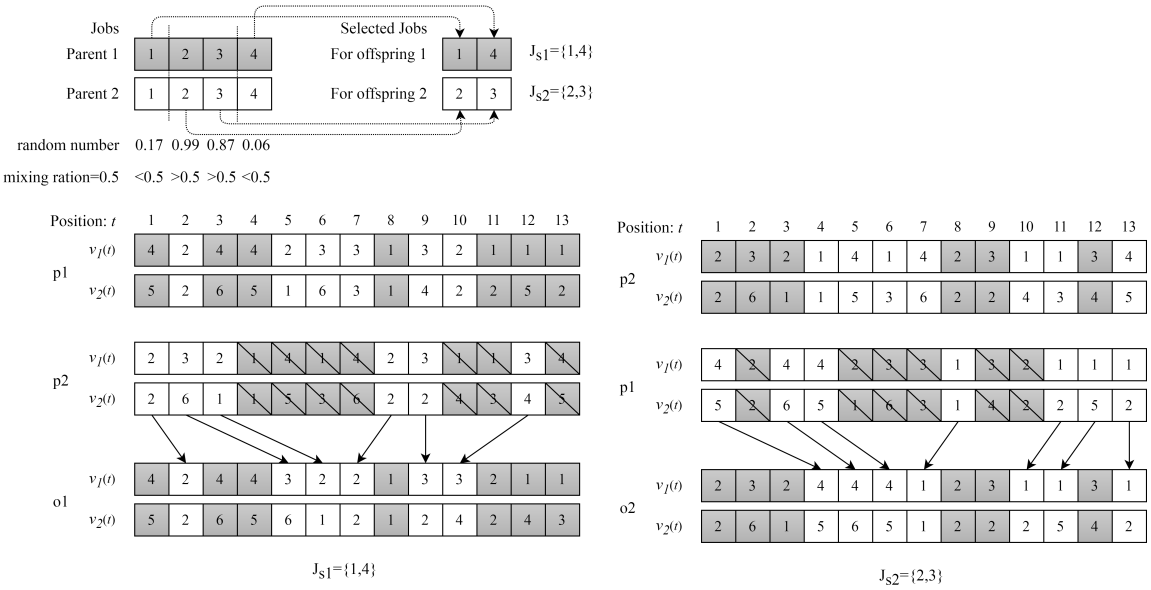


Fig. 7: An example application of OUX

Order-based uniform crossover (OUX): OUX has a parameter referred to as the mixing ratio which defines the percentage of inherited genes from the first and second parents (Li et al., 2014b). In this study, the mixing ratio is fixed as 0.5. Fig. 7 illustrates how OUX works. OUX randomly creates two partitions of jobs based on the mixing ratio, where each partition is used to create a separate offspring. Then OUX uses the same approach as POX for carrying out an exchange of the genetic material. Instead of a randomly selected subset of jobs, the associated partition is used.

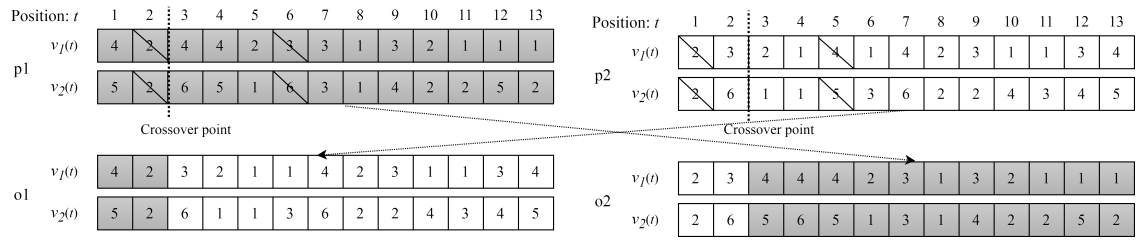


Fig. 8: An example application of OOX

Order-based one-point crossover (OOX): We have tailored a one-point crossover for FJCS-SDFSTs-ITTs. A random crossover point on the chromosome is selected first. Then all genetic material in parents to the crossover point is copied into two offspring. The operations already assigned are deleted from the other parents along with the associated machine assignments. Then the remaining genetic material from each parent is copied into the offspring without changing their order (Ruiz et al., 2006). The OOX is illustrated in Fig. 8.

3.4.2. Mutation

Mutation traverses all genes for each operation and perturbs the allele values at a locus with the standard probability of 1/(*chromosome length*). The mutation operator flips a coin and either applies a swap (SPM) operation (Piroozfard et al., 2018) or randomly changes the machine assignment for the gene in question without changing the operation. SPM exchanges two randomly selected operations and their machine assignments keeping the routing of the operations over the machines unchanged.

3.4.3. Hill-climbing

The genetic algorithms (GAs) do not employ hill climbing. In this study, we tested memetic algorithms (MAs) using two different hill-climbing methods, denoted as HC_1 and HC_2 for local search. The hill climbing is applied to the offspring right after the mutation. HC_1 is the same neighborhood structure as described in Karimi et al. (2012) as LN_1. We have built HC_2 by integrating LN_1 and LN_3 in Karimi et al. (2012) as illustrated in Algorithm 2. HC_1 operates similarly as HC_2, where the lines 5 and 7 of Algorithm 2 are discarded.

Algorithm 2 Pseudo-code of the hill-climbing method HC_2

```
1: procedure HillClimbing-2(Population, v1, v2, depth-of-search)
2:     for each chromosome in population do
3:         Detect operation set (S) that have two and more identical parallel machine from v1
4:         Determine a memory set (MS) for the selected operations, MS= {∅}
5:         Select two positions randomly as r1 and r2 between 1 and the length of chromosome (r1 ≠ r2)
6:         for k ← 1 to depth-of-search do
7:             Invert the values in substring located between r1 and r2
8:             Select an operation Oij randomly from set (S) (except operations in MS)
9:             Detect machine Mq from v2 that is currently selected to process Oij
10:            Detect a set of machines that can process Oij
11:            Select a random machine from this set of machines (except Mq) to process Oij
12:            Add Oij to MS
13:        end for
14:        Find the chromosome that has the lowest fitness function value out of the chromosome
15:    end for
16: end procedure
```

3.4.4. Replacement

In this study, we have designed two types of evolutionary algorithms with (MA) and without using hill climbing (GA), namely transgenerational (TMA and TGA) and steady-state (SSGA and SSMA) approaches.

The transgenerational evolutionary algorithms (TMA and TGA) create a 'large' pool of offspring, then a replacement scheme is applied to choose the individuals surviving to the next generation. We have tested two replacement schemes here. Assuming that the population size is N, the first replacement scheme (R_1) creates a pool of $N-1$ offspring. Then the best solution in the current population survives to the next generation along with the whole offspring pool. R_1 is a generic replacement scheme that maintains the best solution found so far through the generations. The second replacement scheme (R_2) uses strong elitism merging the current population with the offspring pool, sorting their fitness values ($2N$), and choosing the best half to survive to the next generation.

The steady-state evolutionary algorithms (SSMA and SSGA), two selected parents generate two offspring. Then, the replacement operator with strong elitism chooses the best two solutions from the offspring and

the worst two individuals of the current population to survive to the next generation (R_3). Those best two solutions replace the worst two individuals in the current population.

4. Experimental results

We performed two main sets of experiments. In the first set of experiments, we tested all combinations of algorithmic components and scalarization methods to detect the best configuration for each of the evolutionary algorithms with different replacement strategies using arbitrarily selected small, medium, and large size problem instances. Afterward, we have extended our experiments to compare the performances of those tuned evolutionary algorithms on forty-three problem instances including the real-world case study. The hypervolume (HV) (Brockhoff et al., 2008) is used as the performance indicator in this study. HV measures the quality of the trade-off solutions achieved by the algorithms at approximating the Pareto front (Zitzler & Thiele, 1998a,b). The higher the HV value, the better the performance of an algorithm is.

In the following section, we discuss the details of the benchmark problem instances adopted in the experiments. Then we cover the results obtained from the algorithm tuning experiments on selected instances as well as the experimental results comparing the performance of the tuned evolutionary algorithms for the bi-objective FJCS-SDFSTs-ITTs. All approaches are implemented using C# 2013 and the experiments are run on a laptop with 2.1 GHz Core i7 CPU and 8.00 GB RAM. The population size is set to half of the total number of operations (chromosome length) to be scheduled and the initial population is generated randomly. An algorithm terminates when the maximum number of fitness evaluations (which is a factor of the population size) is exceeded.

4.1. Benchmark problem instances

We have converted a set of forty-three problem instances from various relevant problem domains published in the scientific literature (Won & Kim, 1997; Tang et al., 2010; Deliktas et al., 2019; Zeng et al., 2015; Halat & Bashirzadeh, 2015; Tang et al., 2010; Harhalakis et al., 1990; Arkat et al., 2007) into FJCS-SDFSTs-ITTs problem instances. We discuss the adaptation process briefly below for forming the new FJCS-SDFSTs-ITTs benchmark. Table 5 provides the characteristics of each benchmark problem instance ranging from small to large sizes. The cellular information of each problem instance is used as reported in the corresponding papers as indicated in Table 5. Some of the instances already have cell formation, the processing time, routes, and setup and transportation times, whereas we randomly generated the missing relevant information for the others. For example, where missing, the processing times for each operation are randomly drawn as a discrete value from a uniform distribution within the interval of [1, 10], while setup and transportation times are obtained from the interval [1, 8]. The discrete due date for each job is generated randomly within the interval of [1, 110]. The size of a problem is classified considering the total number of parts, operations, machines, cells, and part families as summarized in Table 5.

The largest problem is our real-world case study, which has 58 parts, 41 machines, and 4 cells and part families while the smallest one has 4 parts, 6 machines, and 2 cells and part families adopted from Won & Kim (1997). This case study is based on a real industrial problem occurring at a large locomotive and wagon production factory in Turkey. The actual production data from this factory was gathered to show the performance and characteristics of the proposed EA algorithms in the paper. Also, parts or products produced in this factory, which have a complex production process and long routing, and which are critical for the factory, have been identified. It is seen that the locomotive and wagon factory is suitable for cellular production because the plant has a functional plant layout, medium degree of product variety, and production volume. The production is based on customer's demand.

4.2. Tuning experiments

In this initial set of experiments, we have tuned two GA variants, and two MA variants embedding different algorithmic components to detect their best configurations for FJCS-SDFSTs-ITTs. The combinations of three different crossover operators $\{CO_1, CO_2, CO_3\}$, two different local search heuristics used with the memetic algorithms $\{HC_1, HC2\}$ and no hill-climbing for the genetic algorithms, three replacement schemes $\{R_1, R_2, R_3\}$ under three different scalarization methods {WSM, CSM, and TSM} summing up to 81 different EA configurations are considered. The experiments to detect the best configuration for each EA variant

Table 5. The characteristics of the benchmark problem instances and the intervals for generating the reference points Rf_1 and Rf_2 used in the fitness functions CSM and TSM for each instance.

Size	Ref.	Inst.#	$n/L/m/C/t_{op}$	I_{Rf_1}	I_{Rf_2}	Size	Ref.	Inst.#	$n/L/m/C/t_{op}$	I_{Rf_1}	I_{Rf_2}
Small	Won & Kim (1997)	1	4/2/6/2/13	[28;33]	[33;34]	Large	Harhalakis et al. (1990)	22	20/4/26/4/81	[30;53]	[95;170]
		2	5/2/6/2/14	[28;33]	[33;50]			23	20/5/26/5/81	[34;52]	[102;203]
	Tang et al. (2010)	3	4/2/7/2/13	[16;27]	[11;11]		Tang et al. (2010)	24	22/4/27/4/71	[62;77]	[172;398]
		4	8/2/7/2/27	[31;71]	[23;54]		Zeng et al. (2015)	25	21/4/26/4/86	[63;93]	[200;440]
	Deliktas et al. (2019)	5	4/2/6/2/13	[27;50]	[11;24]			26	24/4/22/4/96	[51;71]	[260;494]
		6	6/2/6/2/21	[28;67]	[34;56]		Halat & Bashirzadeh (2015)	27	25/5/33/5/90	[67;77]	[178;466]
		7	6/2/8/2/18	[26;49]	[33;39]			28	25/5/35/5/96	[48;68]	[159;375]
		8	7/3/7/2/25	[37;81]	[21;56]			29	26/5/38/5/87	[57;66]	[146;352]
		9	7/2/9/3/24	[29;63]	[25;38]			30	27/5/30/5/97	[67;81]	[208;502]
		10	7/3/8/3/24	[29;62]	[36;56]			31	27/5/30/5/132	[91;105]	[347;831]
		11	8/4/11/4/25	[29;74]	[38;63]			32	27/5/40/5/101	[42;59]	[173;337]
		12	9/3/9/3/30	[34;71]	[74;135]			33	27/5/40/5/101	[48;65]	[158;375]
Medium	Deliktas et al. (2019)	13	10/4/11/4/33	[34;73]	[60;108]			34	27/5/29/5/104	[33;48]	[130;219]
		14	10/4/13/4/31	[31;67]	[60;91]		Zeng et al. (2015)	35	30/5/26/5/111	[48;74]	[261;519]
	Zeng et al. (2015)	15	11/3/11/3/37	[35;55]	[75;134]		Tang et al. (2010)	36	30/5/30/5/105	[47;75]	[152;384]
	Halat & Bashirzadeh (2015)	16	10/2/8/2/33	[47;55]	[85;153]			37	35/5/30/5/126	[34;68]	[149;330]
		17	12/2/8/2/38	[52;61]	[117;233]		Arkat et al. (2007)	38	30/4/25/4/113	[51;75]	[248;551]
		18	12/4/12/4/38	[35;48]	[75;125]		Zeng et al. (2015)	39	35/5/27/5/132	[51;71]	[215;500]
	Tang et al. (2010)	19	12/3/9/3/38	[42;58]	[141;239]		Tang et al. (2010)	40	40/6/25/6/138	[59;94]	[239;638]
	Deliktas et al. (2019)	20	14/4/13/4/45	[35;77]	[84;192]			41	45/6/28/6/157	[59;92]	[313;758]
		21	15/4/16/4/48	[31;86]	[63;129]			42	50/6/30/6/175	[59;93]	[340;820]
							Case study	43	58/4/41/4/269	[1671;2181]	[1449;16595]

are performed on the three sample instances. The FJCS-SDFSTs-ITTs instances #12, #21 and #43 are arbitrarily selected from the small, medium and large size instances, respectively.

Each EA is run for 1000 times using random weights generated within the interval of [0,1] and with the constraint $w_1 + w_2 = 1$ for each of the scalarization method for solving a problem instance. Additionally, 1000 different reference points for CSM and TSM are randomly created within a certain interval. The minimum value of makespan is determined as the lower bound based on the 31 runs by setting $w_1 = 1$ ($w_2 = 0$), while the maximum objective value of makespan is determined as the upper bound using 31 results obtained after setting $w_1 = 0$ ($w_2 = 1$). The lower and upper bounds of total tardiness are computed in the same way for the intervals. Table 5 provides the intervals from which the reference points for CSM and TSM are randomly drawn. α in Eq (13) is computed as $\alpha = [\min\{w_1, w_2\} - 0.01]$. ρ in Eq (14) is fixed as 0.01. The number of fitness evaluation is used as the stopping criterion, which is a factor of population size (for TMA with WSM approach, the number of maximum iteration is 100 and depth-of-search parameter is 4). The parameter settings of each operator used in the evolutionary algorithms are discussed in Section 3.4.

Each of the evolutionary algorithm variant with the scalarization method is run by considering different combinations of operators on the three selected sample instances for 1000 times. At each trial, fitness function (WSM, CSM, or TSM) is varied through a random setting of their relevant parameters, such as weights, reference points. Non-dominated solutions are obtained from those 1000 trials, for which hypervolume is calculated for each EA configuration per instance. Then the mean hypervolume for each EA configuration is computed averaging the normalized hypervolume values over the three sample instances.

Table 6 summarises the results based on mean hypervolume (*Avr.*). The trivial observation is that the choice of algorithmic components is influential on the performance of the evolutionary algorithms for FJCS-SDFSTs-ITTs. The best approach for FJCS-SDFSTs-ITTs from the tuning experiments turns out to be the transgenerational genetic algorithm with the best mean hypervolume value of 0.9856. This evolutionary algorithm employs the order-based uniform crossover (CO_1), generic replacement scheme (R_1), and TSM as the fitness function. One point crossover makes the performance of the genetic algorithm poorer regardless of the replacement scheme used. The steady-state genetic algorithm using the one point crossover performs the worst among all evolutionary algorithms. In almost all cases, the use of local search regardless of the hill-climbing method used improves the performance of the genetic algorithm version. The two interesting exceptions are the best performing approach and also the steady-state genetic algorithm using CO_3 and TSM as the fitness function, which yields the mean hypervolume value of 0.9643. The proposed hill-climbing method HC_2 seems to perform slightly better than HC_1 in the overall. Similarly, the transgenerational evolutionary algorithm performs better than the steady-state evolutionary algorithms.

Based on the results in Table 6, the best configuration for each of the evolutionary algorithms is as follows: (i) *TMA with CSM* performs the best using the order-based one-point crossover (CO_3), generic replacement scheme (R_1), and LN_1 (HC_1), (ii) *SSMA with TSM* performs the best using the order-based one-point

crossover (CO_3), steady-state replacement with strong elitism (R_3), and integrated LN_1 and LN_3 (HC_2), (iii) *TGA with TSM* performs the best when using the order-based uniform crossover (CO_1), and generic replacement scheme (R_1), and (iv) *SSGA with TSM* performs the best when using the order-based one-point crossover (CO_3), and steady-state replacement with strong elitism (R_3). Interestingly, if we consider the average performances of algorithms and choose the best algorithms focusing on the scalarization methods based on mean hypervolume turns out to be (i) for CSM, (iii) for TSM, however for WSM the winner becomes (v) *TMA with WSM* using the order-based one-point crossover (CO_3), generic replacement scheme (R_1), and LN_1 (HC_1). Hence, the hypervolume values belonging to the best configurations are (i) 0.9851, (ii) 0.9437, (iii) 0.9856, (iv) 0.9643, and (v) 0.9849 tuned evolutionary algorithms at their best are used in the next set of experiments.

4.3. Performance comparison of tuned evolutionary algorithms

To further evaluate the performance of the tuned EAs obtained from the previous set of experiments, namely TMA with WSM, SSMA with TSM, TGA with TSM, and SSGA with TSM, we have performed additional computational experiments on 43 different instances and analyzed the results. Each experiment is repeated 31 times under the same configuration of operators and parameters for all of the four evolutionary algorithms. Table 7 illustrates the performance comparison of all MOEAs across all problem instances based on hypervolume. For statistical analysis of paired comparisons of TMA with WSM and each of the other EAs, we have applied Wilcoxon signed rank test using 31 results from the algorithms based on hypervolume. The following notation is adopted in Table 7. For a given instance, $>$ ($<$) indicates that the TMA with WSM performs better (worse) than the algorithm indicated on the right side of the notation in the column label and this performance difference is statistically significant within a confidence interval of 95%. If there is no statistically significant performance variation between TMA with WSM and the other evolutionary algorithm then this is indicated with the notation '$\approx$'.

We also performed comparison of each pair of four evolutionary algorithms considering the best results for each of the 43 instances. The total number of different pair-wise comparisons of algorithms sums up to 172 (43×4). TMA with WSM, 89 out of 172 comparisons performs better than the other algorithms. This performance difference is statistically significant for the 15 of the cases while there is no significant performance variation for 74 of them. More specifically, Table 7 shows that TMA with WSM delivers a similar or better performance when compared to SSMA with TSM, TGA with TSM, and SSGA with TSM over 38, 42 and all of the 43 instances, respectively.

All evolutionary algorithms perform similarly on almost all the small instances and there is no winner. Only for instance#12 SSMA with TSM performs the best and this performance variation is statistically significant as indicated in Fig. 9(a). The second best performing EA is TMA with WSM on this problem instance. However, performance differences start to amplify on the medium instances. On 4 of the medium instances including #13, #14, #16 and #18 all four algorithms still perform similarly. As for the remaining medium instances, memetic algorithms, namely TMA with WSM and SSMA with TSM perform significantly better than the genetic algorithm variants. For the instances #19-#21, SSMA with TSM is the best, which is also confirmed for the medium instance#21 in Fig. 9(c). TMA with WSM is the best for instance#15. Both memetic algorithms deliver a similar performance on the five out of nine medium instances. On 17 out of 22 large instances including the real-world case study, TMA with WSM outperforms all other evolutionary algorithms considering the Wilcoxon signed-rank test. This performance difference is statistically significant. Fig. 9(c) to 9(f) confirms this observation on the instances #21, #35, #40, #43, respectively. SSGA with TSM is the worst performing algorithm among four EAs on the large instances. Hence, in the overall TMA with WSM using precedence preserving order-based crossover and LN_1 hill-climbing is the best evolutionary algorithm for solving the FJCS-SDFSTs-ITTs. The number of fitness evaluations for each instance in each trial (run) is kept the same for each algorithm for fairness. Due to the data structures used and additional complexities such as the computation of the scalarization method used, an algorithm could run slightly more or less than the others. The total running time of an algorithm varies from 1 to 90 seconds for a single trial depending on the size of an instance. In overall, TMA with WSM runs slightly faster than the all others on the majority of the instances.

As a sample, we plotted the Pareto fronts consisting non-dominated trade-off solutions achieved by each MOEA on one small, two medium, and three relatively large arbitrarily chosen instances, including the instance#12 (Fig. 10(a)), instance#15 (Fig. 10(b)), instance#21 (Fig. 10(c)), instance#35 (Fig. 10(d)),

Table 6. Normalized hypervolume values belonging to every evolutionary method with scalarization methods for every configuration. **Bold** entries highlight the best performing configurations for the tansgenerational/steady-state genetic and memetic algorithms based on the mean normalized hypervolume averaged over three problem instances.

SM	Inst.#	Transgenerational Genetic Algorithm						Steady State Genetic Algorithm		
		CO_1; R_1	CO_1; R_2	CO_2; R_1	CO_2; R_2	CO_3; R_1	CO_3; R_2	CO_1; R_3	CO_2; R_3	CO_3; R_3
WSM	12	0.7949	0.8462	0.8974	0.6410	1.0000	0.6410	0.8462	0.5641	1.0000
	21	0.9581	0.8878	0.2568	0.0324	0.6108	0.1243	0.9554	0.1487	0.6054
	43	0.9774	0.9635	0.8244	0.8298	0.9741	0.9512	0.9708	0.8364	0.9615
	Avr.	0.9101	0.8992	0.6595	0.5011	0.8616	0.5722	0.9241	0.5164	0.8557
CSM	12	1.0000	0.8205	0.5385	0.3846	0.9487	0.7180	0.7949	0.2051	0.8462
	21	0.9730	0.1635	0.3595	0.2824	0.9622	0.8730	0.5608	0.4460	0.9081
	43	0.9777	0.9586	0.8397	0.8214	0.9753	0.9455	0.9765	0.8070	0.9658
	Avr.	0.9836	0.6476	0.5792	0.4962	0.9621	0.8455	0.7774	0.4860	0.9067
TSM	12	1.0000	0.8462	0.7436	0.4872	1.0000	0.8462	0.7436	0.0000	1.0000
	21	0.9797	0.4122	0.3257	0.0000	0.9730	0.1689	0.8784	0.1703	0.9338
	43	0.9771	0.9536	0.8411	0.8222	0.9733	0.9530	0.9672	0.8025	0.9590
	Avr.	**0.9856**	0.7373	0.6368	0.4364	0.9821	0.6560	0.8631	0.3243	**0.9643**

SM	Inst.#	Transgenerational Memetic Algorithm								
		CO_1;R_1 HC_1	CO_1;R_2 HC_1	CO_1;R_1 HC_2	CO_1;R_2 HC_2	CO_2;R_1 HC_1	CO_2;R_2 HC_1	CO_2;R_1 HC_2	CO_2;R_2 HC_2	CO_3;R_1 HC_1
WSM	12	1.0000	1.0000	0.8462	0.9487	0.8205	0.7949	1.0000	1.0000	1.0000
	21	0.9973	0.9743	1.0000	0.9905	0.9824	0.6014	0.9851	0.9946	0.9973
	43	0.9546	0.9742	0.9506	0.9310	0.9310	0.9551	0.9271	0.9220	0.9574
	Avr.	0.9840	0.9828	0.9323	0.9567	0.9113	0.7838	0.9707	0.9722	***0.9849***
CSM	12	1.0000	1.0000	1.0000	0.8974	0.8462	0.9744	1.0000	0.9487	1.0000
	21	0.9973	0.9676	0.9946	0.9959	0.9770	0.9351	0.9892	0.9811	0.9973
	43	0.9476	0.9675	0.9543	0.9653	0.9432	0.9449	0.9259	0.9148	0.9579
	Avr.	0.9816	0.9784	0.9830	0.9529	0.9221	0.9515	0.9717	0.9482	**0.9851**
TSM	12	0.8462	0.9487	1.0000	0.8974	1.0000	0.7436	0.8974	0.8974	1.0000
	21	0.9919	0.9905	1.0000	0.9932	0.9838	0.9054	0.9973	0.9905	0.9973
	43	0.9529	0.9700	0.9454	0.9558	0.9421	0.9459	0.9244	0.9049	0.9554
	Avr.	0.9303	0.9697	0.9818	0.9488	0.9753	0.8650	0.9397	0.9309	0.9842

SM	Inst.#	Transgenerational Memetic Algorithm			Steady Stade Memetic Algorithm					
		CO_3;R_2 HC_1	CO_3;R_1 HC_2	CO_3;R_2 HC_2	CO_1;R_3 HC_1	CO_1;R_3 HC_2	CO_2;R_3 HC_1	CO_2;R_3 HC_2	CO_3;R_3 HC_1	CO_3;R_3 HC_2
WSM	12	0.6923	0.8462	0.8462	0.7179	1.0000	0.9744	0.8462	0.8974	0.8974
	21	0.9878	0.9973	0.9946	0.9973	1.0000	0.9473	1.0000	0.9946	1.0000
	43	0.9679	0.9533	0.9591	0.7993	0.8250	0.8251	0.8337	0.8206	0.8392
	Avr.	0.8827	0.9323	0.9333	0.8382	0.9417	0.9156	0.8933	0.9042	0.9122
CSM	12	0.7949	1.0000	0.8974	1.0000	1.0000	0.7692	1.0000	1.0000	0.8974
	21	0.9811	0.9973	0.9973	0.9932	0.9959	0.9297	0.9932	0.9905	0.9973
	43	0.9695	0.9556	0.9636	0.7837	0.8180	0.8124	0.8113	0.8008	0.8343
	Avr.	0.9152	0.9843	0.9528	0.9257	0.9380	0.8371	0.9348	0.9304	0.9097
TSM	12	0.7436	1.0000	1.0000	1.0000	1.0000	0.4872	1.0000	1.0000	1.0000
	21	0.9919	1.0000	0.9932	0.9919	0.9973	0.9149	0.9878	0.9932	0.9959
	43	0.9624	0.9518	0.9537	0.7891	0.8289	0.8107	0.8239	0.8217	0.8351
	Avr.	0.8993	0.9839	0.9823	0.9270	0.9421	0.7376	0.9373	0.9383	**0.9437**

CO_1: Order-based Uniform Crossover; CO_2: Order-based One-point Crossover; CO_3: Precedence Preserving Order-based Crossover
SM: Scalarization Method; HC_1: LN_1; HC_2: Integrated LN_1&LN_3

Table 7. Performance comparison of TMA with WSM, TMA with CSM, SSMA with TSM, TGA with TSM, and SSGA with TSM for the bi-objective flexible job shop cell scheduling problem based on hypervolume indicator across 43 benchmark problem instances

Inst.#	TMA with WSM			SSMA with TSM			TMA with CSM			TGA with TSM			SSGA with TSM	
	Mean	Std Dvd	vs.	Mean	Std Dvd	vs.	Mean	Std Dvd	vs.	Mean	Std Dvd	vs.	Mean	Std Dvd
1	**0.353365**	**0**	≈	**0.353365**	**0**	≈	**0.353365**	**0**	≈	**0.353365**	**0**	≈	**0.353365**	**0**
2	**0.549550**	**2.26E-16**	≈	**0.549550**	**2.26E-16**	≈	**0.549550**	**2.26E-16**	≈	**0.549550**	**2.26E-16**	≈	**0.549550**	**2.26E-16**
3	**0.592593**	**0**	≈	**0.592593**	**0**	≈	**0.592593**	**0**	≈	**0.592593**	**0**	≈	**0.592593**	**0**
4	**0.864756**	**2.26E-16**	≈	**0.864756**	**2.26E-16**	≈	**0.864756**	**2.26E-16**	≈	**0.864756**	**2.26E-16**	≈	**0.864756**	**2.26E-16**
5	**0.800000**	**3.39E-16**	≈	**0.800000**	**3.39E-16**	≈	**0.800000**	**3.39E-16**	≈	**0.800000**	**3.39E-16**	≈	**0.800000**	**3.39E-16**
6	**0.759259**	**3.39E-16**	≈	**0.759259**	**3.39E-16**	≈	**0.759259**	**3.39E-16**	≈	**0.759259**	**3.39E-16**	≈	**0.759259**	**3.39E-16**
7	**0.750000**	**0**	≈	**0.750000**	**0**	≈	**0.750000**	**0**	≈	**0.750000**	0	≈	**0.750000**	**0**
8	**0.887535**	**5.64E-16**	≈	**0.887535**	**5.64E-16**	≈	**0.887535**	**5.64E-16**	≈	**0.887535**	**5.64E-16**	≈	**0.887535**	**5.64E-16**
9	**0.804688**	**0**	≈	**0.804688**	**0**	≈	**0.804688**	**0**	≈	**0.804688**	**0**	≈	**0.804688**	**0**
10	**0.742857**	**4.51E-16**	≈	**0.742857**	**4.51E-16**	≈	**0.742857**	**4.51E-16**	≈	**0.742857**	**4.51E-16**	≈	**0.742857**	**4.51E-16**
11	**0.722276**	**3.39E-16**	≈	**0.722276**	**3.39E-16**	≈	**0.722276**	**3.39E-16**	≈	**0.722276**	**3.39E-16**	≈	**0.722276**	**3.39E-16**
12	0.758821	8.89E-05	<	**0.758865**	**3.02E-05**	⩾	0.758816	7.99E-05	>	0.758743	1.53E-04	>	0.758536	2.15E-04
13	**0.762499**	**0**	≈	**0.762499**	**0**	≈	**0.762499**	**0**	≈	**0.762499**	**0**	≈	**0.762499**	**0**
14	**0.726323**	**5.64E-16**	≈	**0.726323**	**5.64E-16**	≈	**0.726323**	**5.64E-16**	≈	**0.726323**	**5.64E-16**	≈	**0.726323**	**5.64E-16**
15	**0.775388**	**1.69E-05**	⩾	0.775381	2.15E-05	⩾	0.775368	6.14E-05	>	0.772845	1.40E-03	>	0.771958	2.26E-03
16	**0.732236**	**5.64E-16**	≈	**0.732236**	**5.64E-16**	≈	**0.732236**	**5.64E-16**	≈	**0.732236**	**5.64E-16**	≈	**0.732236**	**5.64E-16**
17	**0.805324**	**5.64E-16**	≈	**0.805324**	**5.64E-16**	≈	**0.805324**	**5.64E-16**	>	0.804197	1.74E-03	>	0.802855	2.94E-03
18	**0.777473**	**3.39E-16**	≈	**0.777473**	**3.39E-16**	≈	**0.777473**	**3.39E-16**	≈	**0.777473**	**3.39E-16**	≈	**0.777473**	**3.39E-16**
19	0.769646	1.20E-05	⩽	**0.769648**	**1.13E-016**	⩾	0.769641	3.28E-05	>	0.766577	3.53E-03	>	0.765310	4.16E-03
20	0.814591	1.34E-04	<	**0.814801**	**1.02E-04**	⩾	0.814512	2.06E-04	>	0.807380	4.35E-03	>	0.802825	5.27E-03
21	0.815199	5.13E-05	<	**0.815239**	**4.39E-05**	⩾	0.815196	5.91E-05	>	0.811686	3.93E-03	>	0.808018	6.07E-03
22	**0.764529**	**2.26E-16**	≈	**0.764529**	**2.26E-16**	≈	**0.76452**	**2.26E-16**	≈	**0.764529**	**2.26E-16**	⩾	0.764505	7.53E-03
23	0.790317	6.33E-06	⩽	**0.790318**	**3.39E-16**	⩽	**0.790318**	**3.39E-16**	⩽	**0.790318**	**3.39E-16**	>	0.790306	1.94E-05
24	**0.759888**	**1.13E-16**	≈	**0.759888**	**1.13E-16**	≈	**0.759888**	**1.13E-16**	>	0.759411	9.29E-04	>	0.758378	1.54E-03
25	**0.754495**	**1.01E-04**	>	0.754343	3.81E-04	>	0.754423	1.29E-04	>	0.754132	5.92E-04	>	0.752348	1.27E-03
26	**0.772663**	**1.08E-03**	>	0.768215	1.77E-03	>	0.771805	1.10E-03	>	0.764228	2.44E-03	>	0.760359	3.74E-03
27	0.733979	3.59E-05	⩾	0.733875	3.82E-04	⩽	**0.733983**	**4.30E-05**	>	0.733939	2.68E-04	>	0.732295	2.12E-03
28	**0.752377**	**3.39E-16**	≈	**0.752377**	**3.39E-16**	≈	**0.752377**	**3.39E-16**	≈	**0.752377**	**3.39E-16**	≈	**0.752377**	**3.39E-16**
29	**0.774487**	**4.51E-16**	≈	**0.774487**	**4.51E-16**	≈	**0.774487**	**4.51E-16**	>	0.773994	1.31E-03	>	0.772907	2.03E-03
30	**0.733672**	**2.18E-04**	>	0.732538	1.09E-03	⩾	0.733555	4.59E-04	>	0.732351	9.83E-04	>	0.728798	2.58E-03
31	**0.729524**	**8.14E-04**	>	0.724272	2.22E-03	⩾	0.728977	1.07E-03	>	0.723105	2.08E-03	>	0.715103	3.44E-03
32	**0.778097**	**1.58E-03**	>	0.776046	7.80E-04	⩾	0.778029	1.58E-03	>	0.772514	1.84E-03	>	0.768104	2.54E-03
33	**0.773665**	**4.51E-16**	>	0.773356	7.63E-04	⩾	0.773661	1.21E-05	>	0.772800	1.12E-03	>	0.769606	2.80E-03
34	**0.764252**	**0**	>	0.764044	5.51E-04	≈	**0.764252**	**0**	>	0.763725	7.66E-04	>	0.762700	9.91E-04
35	**0.765244**	1.56E-03	>	0.759397	2.38E-03	⩾	0.764728	1.58E-03	>	0.758826	2.50E-03	>	0.753363	3.48E-03
36	0.808274	3.38E-05	>	0.808004	6.08E-04	⩽	**0.808281**	**2.47E-05**	>	0.806118	1.61E-03	>	0.802159	2.19E-03
37	**0.836030**	**6.52E-04**	>	0.831053	1.82E-03	⩾	0.835691	8.36E-04	>	0.814596	4.79E-03	>	0.787353	6.06E-03
38	**0.775813**	**1.67E-03**	>	0.77102	1.98E-03	⩾	0.775647	1.76E-03	>	0.769625	1.88E-03	>	0.763267	2.88E-03
39	**0.772614**	**6.50E-04**	>	0.767662	2.22E-03	⩾	0.772466	8.93E-04	⩾	0.771814	1.56E-03	>	0.761611	3.93E-03
40	**0.853198**	**2.12E-03**	>	0.841404	3.47E-03	>	0.832820	2.23E-02	>	0.846085	3.45E-03	>	0.834003	4.07E-03
41	**0.789603**	**3.00E-03**	>	0.773132	4.44E-03	⩾	0.789275	3.51E-03	>	0.781478	3.44E-03	>	0.761292	3.90E-03
42	**0.825841**	**2.84E-03**	>	0.812883	2.68E-03	>	0.825446	2.64E-03	>	0.822401	2.77E-03	>	0.804421	4.00E-03
43	**0.977989**	**8.28E-04**	>	0.971360	2.17E-03	⩾	0.977899	9.79E-04	⩾	0.977466	1.03E-03	>	0.957841	4.06E-03

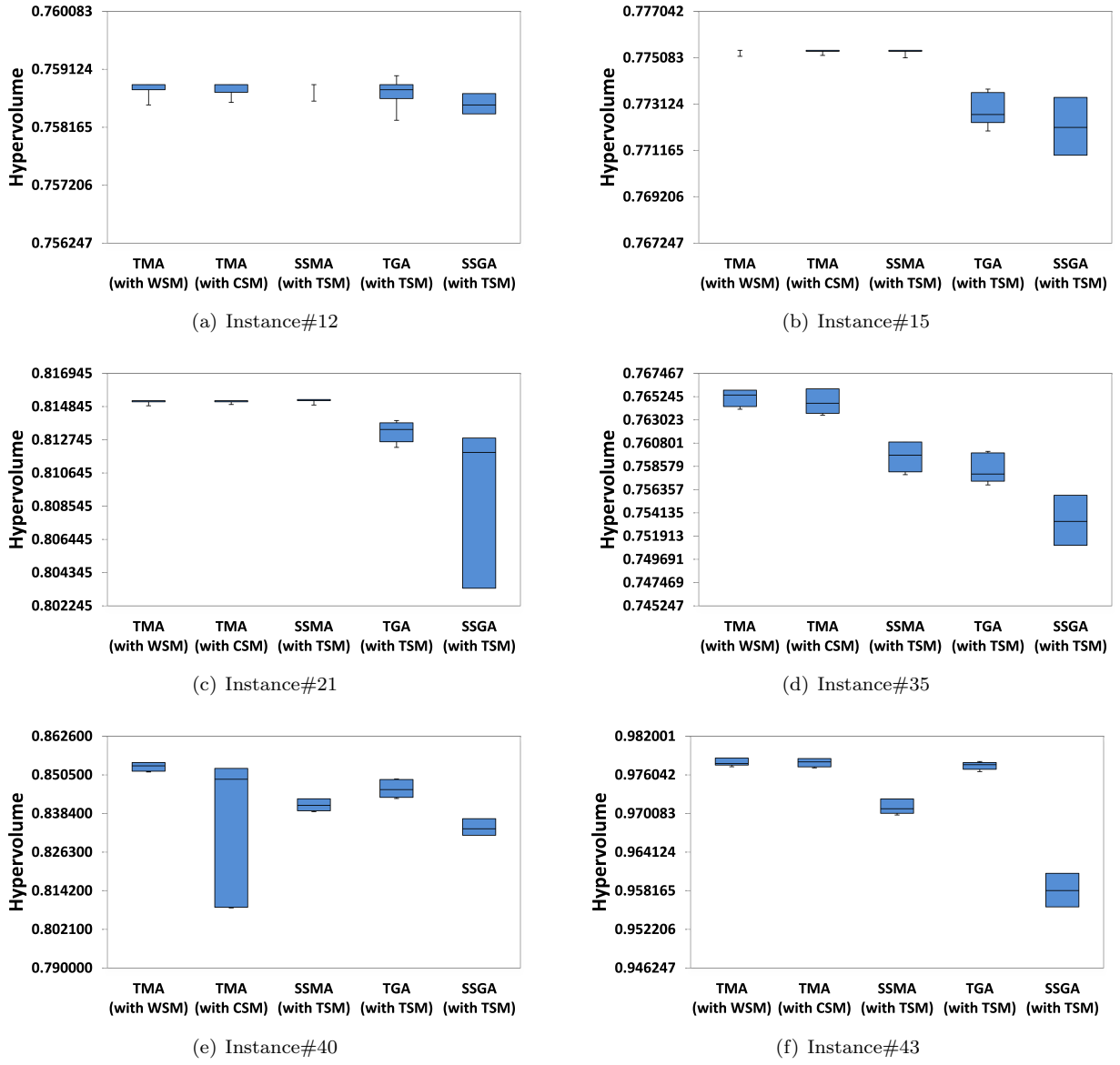
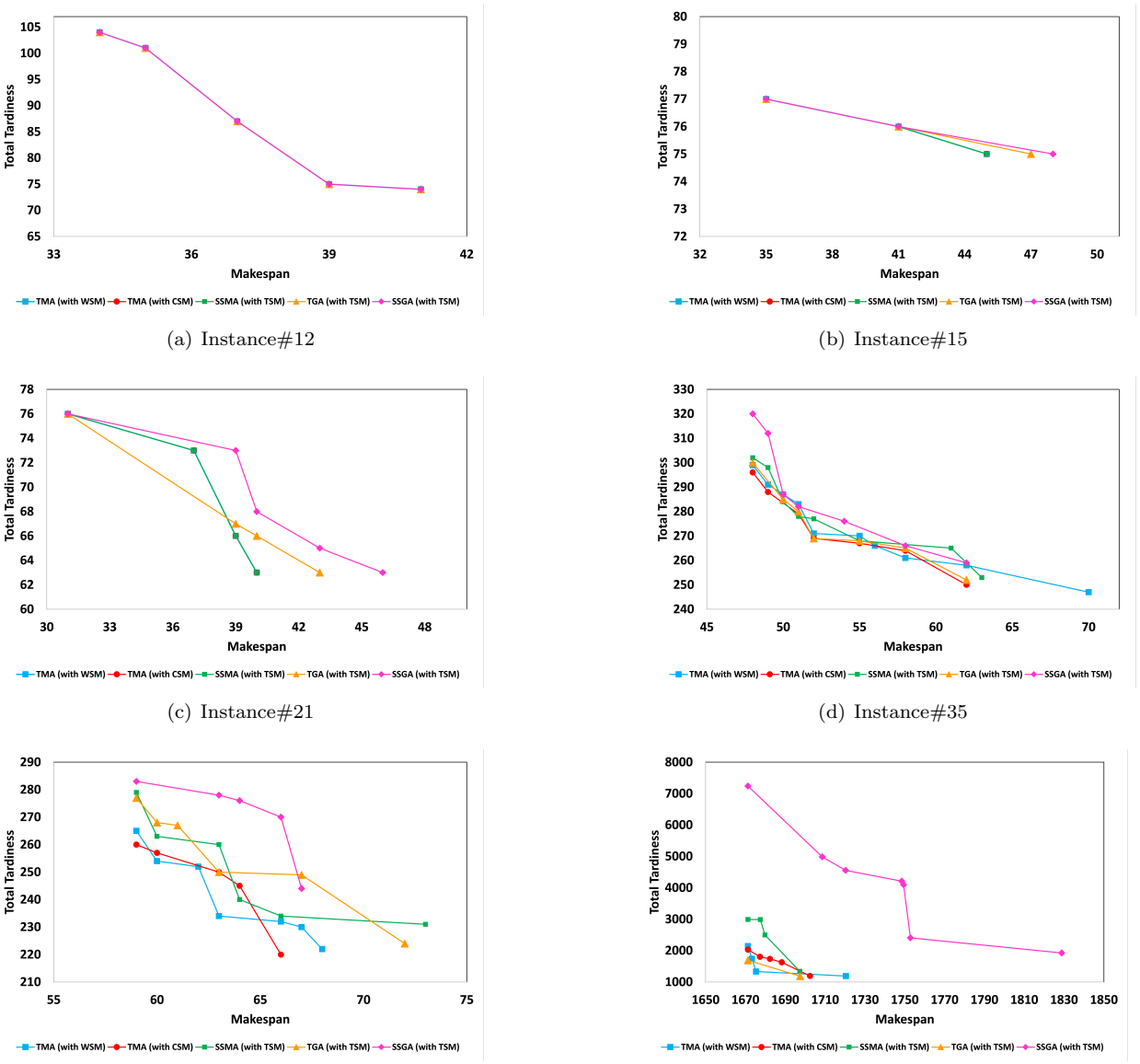


Fig. 9: Box-plots of the hypervolume values obtained from 31 runs of each evolutionary approach for six sample instances

instance#40 (Fig. 10(e)), and instance#43 (Fig. 10(f)). The Pareto fronts are obtained from each EA's best run yielding the best hypervolume. Fig. 10 illustrates that the best sets of trade-off solutions from all MOEAs for the small instance#12 have the same Pareto front plots while the TMA with WSM produces better results when compared to the other algorithms for the remaining cases. This approach also produces a wider spread of trade-off solutions on the Pareto front for the instance#35. Moreover, the majority of the plots of the best Pareto fronts are not convex and the results show that a Pareto front obtained for an FJCS-SDFSTs-ITTs problem instance might contain unsupported trade-off solutions, for example, as in Fig. 10(d) where TGA with TSM has an unsupported solution with the makespan value of 58 which is not on the convex hull of the whole Pareto front.



(a) Instance#12

(b) Instance#15

(c) Instance#21

(d) Instance#35

(e) Instance#40

(f) Instance#43

Fig. 10: The illustration of Pareto fronts consisting of non-dominated trade-off solutions obtained from the best of runs with each random weight considering 1000 weights in the WSM calculations for six sample instances

5. Conclusion and future work

In this study, we addressed the problem of multi-objective flexible job shop cell scheduling in the presence of exceptional and the reentrant parts, intercellular moves, intercellular transportation times, and sequence-dependent family setup times. We have tested various algorithmic components as a part of genetic and memetic algorithms for solving the bi-objective FJCS-SDFSTs-ITTs problem. We have also compared the performance of the tuned evolutionary algorithms using weighted sum, conic, and tchebycheff scalarization methods. The empirical results illustrate the success of a novel transgenerational memetic algorithm using a 'new' hill-climbing algorithm, guided by an objective function based on weighted sum during the search. The proposed TMA with WSM approach outperforms the remaining evolutionary algorithms producing a similar or better performance over at least 38 of the 43 instances and all solutions to the instances represent

the best-known solutions so far.

Moreover, the sample plots of the Pareto fronts obtained from the memetic algorithm concur with the same observations as in Deliktas et al. (2019). The Pareto fronts for the bi-objective FJCS-SDFSTs-ITTs problem instances are not trivial, and they contain unsupported trade-off solutions that are not on the convex hull of the Pareto front.

As a trivial future research direction, we can test single point based search methods for multi-objective optimization based on the selected operators that we have designed for the population-based evolutionary algorithms in this study. Moreover, the performance of those approaches using scalarization can be compared to the multi-objective evolutionary algorithms, such as NSGA-II, SPEA2. As future work, the proposed approach can be extended further and applied to the other variants of the problem, for example, that considers intracellular setup times and intracellular transfer times.

Acknowledgement

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under the BIDEB-2219 International Postdoctoral Research Fellowship Programme grant number 1059B191800037.

References

Ahmadi, E., Zandieh, M., Farrokh, M., & Emami, S. M. (2016). A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. *Computers & Operations Research*, *73*, 56–66.

Anvari, B., Angeloudis, P., & Ochieng, W. (2016). A multi-objective ga-based optimisation for holistic manufacturing, transportation and assembly of precast construction. *Automation in Construction*, *71*, 226–241.

Arkat, J., Saidi, M., & Abbasi, B. (2007). Applying simulated annealing to cellular manufacturing system design. *The International Journal of Advanced Manufacturing Technology*, *32*, 531–536.

Azadeh, A., Elahi, S., Farahani, M. H., & Nasirian, B. (2017). A genetic algorithm-taguchi based approach to inventory routing problem of a single perishable product with transshipment. *Computers & Industrial Engineering*, *104*, 124–133.

Balaji, A., & Porselvi, S. (2014). Artificial immune system algorithm and simulated annealing algorithm for scheduling batches of parts based on job availability model in a multi-cell flexible manufacturing system. *Procedia Engineering*, *97*, 1524–1533.

Beasley, J. E., & Chu, P. C. (1996). A genetic algorithm for the set covering problem. *European journal of operational research*, *94*, 392–404.

Brockhoff, D., Friedrich, T., & Neumann, F. (2008). Analyzing hypervolume indicator based algorithms. In *International Conference on Parallel Problem Solving from Nature* (pp. 651–660). Springer.

Chang, H.-C., & Liu, T.-K. (2017). Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms. *Journal of Intelligent Manufacturing*, *28*, 1973–1986.

Costa, A., Cappadonna, F. A., & Fichera, S. (2017). A hybrid genetic algorithm for minimizing makespan in a flow-shop sequence-dependent group scheduling problem. *Journal of Intelligent Manufacturing*, *28*, 1269–1283.

De Giovanni, L., & Pezzella, F. (2010). An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. *European journal of operational research*, *200*, 395–408.

Delgoshaei, A., Ali, A., Ariffin, M. K. A., & Gomes, C. (2016). A multi-period scheduling of dynamic cellular manufacturing systems in the presence of cost uncertainty. *Computers & Industrial Engineering*, *100*, 110–132.

Deliktas, D., Torkul, O., & Ustun, O. (2019). A flexible job shop cell scheduling with sequence-dependent family setup times and intercellular transportation times using conic scalarization method. *International Transactions in Operational Research*, *26*, 2410–2431.

Deng, Q., Gong, G., Gong, X., Zhang, L., Liu, W., & Ren, Q. (2017). A bee evolutionary guiding nondominated sorting genetic algorithm ii for multiobjective flexible job-shop scheduling. *Computational intelligence and neuroscience*, *2017*. doi:https://doi.org/10.1155/2017/5232518.

Ebrahimi, M., Ghomi, S. F., & Karimi, B. (2014). Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates. *Applied Mathematical Modelling*, *38*, 2490–2504.

Ehrgott, M., Ide, J., & Schöbel, A. (2014). Minmax robustness for multi-objective optimization problems. *European Journal of Operational Research*, *239*, 17–31.

Elmi, A., Solimanpur, M., Topaloglu, S., & Elmi, A. (2011). A simulated annealing algorithm for the job shop cell scheduling problem with intercellular moves and reentrant parts. *Computers & industrial engineering*, *61*, 171–178.

Elmi, A., & Topaloglu, S. (2016). Multi-degree cyclic flow shop robotic cell scheduling problem: Ant colony optimization. *Computers & Operations Research*, *73*, 67–83.

Fattahi, P., & Fallahi, A. (2010). Dynamic scheduling in flexible job shop systems by considering simultaneously efficiency and stability. *CIRP Journal of Manufacturing Science and Technology*, *2*, 114–123.

Feng, Y., Li, G., & Sethi, S. P. (2018). A three-layer chromosome genetic algorithm for multi-cell scheduling with flexible routes and machine sharing. *International Journal of Production Economics*, *196*, 269–283.

França, P. M., Gupta, J. N., Mendes, A. S., Moscato, P., & Veltink, K. J. (2005). Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups. *Computers & Industrial Engineering*, *48*, 491–506.

Gao, K.-Z., Suganthan, P. N., Pan, Q.-K., Chua, T. J., Cai, T. X., & Chong, C.-S. (2014). Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling. *Information Sciences*, *289*, 76–90.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, *1*, 117–129.

Gasimov, R. N. (2001). Characterization of the benson proper efficiency and scalarization in nonconvex vector optimization. In *Multiple criteria decision making in the new millennium* (pp. 189–198). Springer.

Halat, K., & Bashirzadeh, R. (2015). Concurrent scheduling of manufacturing cells considering sequence-dependent family setup times and intercellular transportation times. *The International Journal of Advanced Manufacturing Technology*, *77*, 1907–1915.

Harhalakis, G., Nagi, R., & Proth, J. (1990). An efficient heuristic in manufacturing cell formation for group technology applications. *The International Journal of Production Research*, *28*, 185–198.

Hendizadeh, S. H., Faramarzi, H., Mansouri, S. A., Gupta, J. N., & ElMekkawy, T. Y. (2008). Meta-heuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times. *International Journal of Production Economics*, *111*, 593–605.

Holland, J. H. et al. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Ibrahem, A.-m., Elmekkawy, T., & Peng, Q. (2014). Robust metaheuristics for scheduling cellular flowshop with family sequence-dependent setup times. *Procedia CIRP*, *17*, 428–433.

Jia, S., & Hu, Z.-H. (2014). Path-relinking tabu search for the multi-objective flexible job shop scheduling problem. *Computers & Operations Research*, *47*, 11–26.

Kacem, I., Hammadi, S., & Borne, P. (2002). Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and computers in simulation*, *60*, 245–276.

Kamrani, A. K., & Logendran, R. (1998). *Group technology and cellular manufacturing: Methodologies and applications* volume 1. Taylor & Francis.

Kaplanoğlu, V. (2016). An object-oriented approach for multi-objective flexible job-shop scheduling problem. *Expert Systems with Applications*, *45*, 71–84.

Karimi, H., Rahmati, S. H. A., & Zandieh, M. (2012). An efficient knowledge-based algorithm for the flexible job shop scheduling problem. *Knowledge-Based Systems*, *36*, 236–244.

Karthikeyan, S., Saravanan, M., & Ganesh, K. (2012). Gt machine cell formation problem in scheduling for cellular manufacturing system using meta-heuristic method. *Procedia engineering*, *38*, 2537–2547.

Kasimbeyli, R. (2013). A conic scalarization method in multi-objective optimization. *Journal of Global Optimization*, *56*, 279–297.

Kesen, S. E., Das, S. K., & Güngör, Z. (2010). A genetic algorithm based heuristic for scheduling of virtual manufacturing cells (vmcs). *Computers & Operations Research*, *37*, 1148–1156.

Lee, K.-M., Yamakawa, T., & Lee, K.-M. (1998). A genetic algorithm for general machine scheduling problems. In *1998 Second International Conference. Knowledge-Based Intelligent Electronic Systems. Proceedings KES'98 (Cat. No. 98EX111)* (pp. 60–66). IEEE volume 2.

Li, D., Wang, Y., Xiao, G., & Tang, J. (2013). Dynamic parts scheduling in multiple job shop cells considering intercell moves and flexible routes. *Computers & Operations Research*, *40*, 1207–1223.

Li, J., Pan, Q., & Xie, S. (2012). An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems. *Applied Mathematics and Computation*, *218*, 9353–9371.

Li, J.-Q., & Pan, Q.-K. (2012). Chemical-reaction optimization for flexible job-shop scheduling problems with maintenance activity. *Applied soft computing*, *12*, 2896–2912.

Li, J.-q., Pan, Q.-k., & Liang, Y.-C. (2010). An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, *59*, 647–662.

Li, J.-Q., Pan, Q.-K., & Tasgetiren, M. F. (2014a). A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. *Applied Mathematical Modelling*, *38*, 1111–1132.

Li, L., & Huo, J.-z. (2009). Multi-objective flexible job-shop scheduling problem in steel tubes production. *Systems Engineering-Theory & Practice*, *29*, 117–126.

Li, Y., Li, X., & Gupta, J. N. (2015). Solving the multi-objective flowline manufacturing cell scheduling problem by hybrid harmony search. *Expert Systems with Applications*, *42*, 1409–1417.

Li, Y., Yu, J., & Tao, D. (2014b). Genetic algorithm for spanning tree construction in p2p distributed interactive applications. *Neurocomputing*, *140*, 185–192.

Logendran, R., Carson, S., & Hanson, E. (2005). Group scheduling in flexible flow shops. *International Journal of Production Economics*, *96*, 143–155.

Logendran, R., deSzoeke, P., & Barnard, F. (2006). Sequence-dependent group scheduling problems in flexible flow shops. *International Journal of Production Economics*, *102*, 66–86.

Lu, C., Li, X., Gao, L., Liao, W., & Yi, J. (2017). An effective multi-objective discrete virus optimization algorithm for flexible job-shop scheduling problem with controllable processing times. *Computers & Industrial Engineering*, *104*, 156–174.

Lu, P.-H., Wu, M.-C., Tan, H., Peng, Y.-H., & Chen, C.-F. (2018). A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems. *Journal of Intelligent Manufacturing*, *29*, 19–34.

Miettinen, K. (2012). *Nonlinear multiobjective optimization* volume 12. Springer Science & Business Media.

Moscato, P. et al. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, *826*, 1989.

Moslehi, G., & Mahnam, M. (2011). A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, *129*, 14–22.

Nalepa, J., & Blocho, M. (2015). Co-operation in the parallel memetic algorithm. *International Journal of Parallel Programming*, *43*, 812–839.

Nalepa, J., Cwiek, M., & Kawulok, M. (2015). Adaptive memetic algorithm for the job shop scheduling problem. In *2015 international joint conference on neural networks (IJCNN)* (pp. 1–8). IEEE.

Nalepa, J., & Czech, Z. J. (2013). New selection schemes in a memetic algorithm for the vehicle routing problem with time windows. In *International Conference on Adaptive and Natural Computing Algorithms* (pp. 396–405). Springer.

Nalepa, J., & Kawulok, M. (2014). A memetic algorithm to select training data for support vector machines. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation* (pp. 573–580).

Pinedo, M. (2000). Scheduling: theory, algorithms, and systems. 1995.

Piroozfard, H., Wong, K. Y., & Wong, W. P. (2018). Minimizing total carbon footprint and total late work criterion in flexible job shop scheduling by using an improved multi-objective genetic algorithm. *Resources, Conservation and Recycling*, *128*, 267–283.

Rahmati, S. H. A., Zandieh, M., & Yazdani, M. (2013). Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, *64*, 915–932.

Ruiz, R., Maroto, C., & Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. *Omega*, *34*, 461–476.

Saad, I., Hammadi, S., Benrejeb, M., & Borne, P. (2008). Choquet integral for criteria aggregation in the flexible job-shop scheduling problems. *Mathematics and Computers in Simulation*, *76*, 447–462.

Salmasi, N. (2005). *Multi-stage group scheduling problems with sequence dependent setups*. PhD dissertation Oregon State University.

Schaller, J. E., Gupta, J. N., & Vakharia, A. J. (2000). Scheduling a flowline manufacturing cell with sequence dependent family setup times. *European Journal of Operational Research*, *125*, 324–339.

Shahsavari-Pour, N., & Ghasemishabankareh, B. (2013). A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling. *Journal of Manufacturing Systems*, *32*, 771–780.

Shahvari, O., & Logendran, R. (2016). Hybrid flow shop batching and scheduling with a bi-criteria objective. *International Journal of Production Economics*, *179*, 239–258.

Shen, L., & Buscher, U. (2012). Solving the serial batching problem in job shop manufacturing systems. *European Journal of Operational Research*, *221*, 14–26.

Shen, X.-N., & Yao, X. (2015). Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. *Information Sciences*, *298*, 198–224.

Solimanpur, M., & Elmi, A. (2013). A tabu search approach for cell scheduling problem with makespan criterion. *International Journal of Production Economics*, *141*, 639–645.

Solimanpur, M., Vrat, P., & Shankar, R. (2004). A heuristic to minimize makespan of cell scheduling problem. *International journal of production economics*, *88*, 231–241.

Sörensen, K., & Glover, F. (2013). Metaheuristics. In *Encyclopedia of Operations Research and Management Science* (pp. 960–970). Springer.

Sörensen, K., & Sevaux, M. (2006). Ma | pm: Memetic algorithms with population management. *Computers & Operations Research*, *33*, 1214–1225.

Steuer, R. E., & Choo, E.-U. (1983). An interactive weighted tchebycheff procedure for multiple objective programming. *Mathematical programming*, *26*, 326–344.

Steuer, R. E., & Steuer, R. (1986). *Multiple criteria optimization: theory, computation, and application* volume 233. Wiley New York.

Tang, J., Wang, X., Kaku, I., & Yung, K.-l. (2010). Optimization of parts scheduling in multiple cells considering intercell move using scatter search approach. *Journal of Intelligent Manufacturing*, *21*, 525–537.

Tang, J., Yan, C., Wang, X., & Zeng, C. (2014). Using lagrangian relaxation decomposition with heuristic to integrate the decisions of cell formation and parts scheduling considering intercell moves. *IEEE Transactions on Automation Science and Engineering*, *11*, 1110–1121.

Tavakkoli-Moghaddam, R., Jafari-Zarandini, Y., & Gholipour-Kanani, Y. (2010). Multi-objective particle swarm optimization for sequencing and scheduling a cellular manufacturing system. In *International Conference on Intelligent Computing* (pp. 69–75). Springer.

Wang, L., Zhou, G., Xu, Y., & Liu, M. (2012). An enhanced pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling. *The International Journal of Advanced Manufacturing Technology*, *60*, 1111–1123.

Wierzbicki, A. P. (1986). On the completeness and constructiveness of parametric characterizations to vector optimization problems. *Operations-Research-Spektrum*, *8*, 73–87.

Won, Y., & Kim, S. (1997). Multiple criteria clustering algorithm for solving the group technology problem with multiple process routings. *Computers & Industrial Engineering*, *32*, 207–220.

Wu, M.-C., Lin, C.-S., Lin, C.-H., & Chen, C.-F. (2017). Effects of different chromosome representations in developing genetic algorithms to solve dfjs scheduling problems. *Computers & Operations Research*, *80*, 101–112.

Wu, S.-J., & Chow, P.-T. (1994). Genetic algorithms for solving mixed-discrete optimization problems. *Journal of the Franklin Institute*, *331*, 381–401.

Xia, W., & Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, *48*, 409–425.

Xing, L.-N., Chen, Y.-W., & Yang, K.-W. (2009). Multi-objective flexible job shop schedule: Design and evaluation by simulation modeling. *Applied Soft Computing*, *9*, 362–376.

Xiong, J., Xing, L.-n., & Chen, Y.-w. (2013). Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns. *International Journal of Production Economics*, *141*, 112–126.

Yang, J.-h., Sun, L., Lee, H. P., Qian, Y., & Liang, Y.-c. (2008). Clonal selection based memetic algorithm for job shop scheduling problems. *Journal of Bionic Engineering*, *5*, 111–119.

Yin, L., Li, X., Gao, L., Lu, C., & Zhang, Z. (2017). A novel mathematical model and multi-objective method for the low-carbon flexible job shop scheduling problem. *Sustainable Computing: Informatics and Systems*, *13*, 15–30.

Zeng, C., Tang, J., & Yan, C. (2015). Job-shop cell-scheduling problem with inter-cell moves and automated guided vehicles. *Journal of Intelligent Manufacturing*, *26*, 845–859.

Zhang, G., Gao, L., & Shi, Y. (2011). An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, *38*, 3563–3573.

Zhang, G., Shao, X., Li, P., & Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, *56*, 1309–1318.

Ziaee, M. (2014). A heuristic algorithm for the distributed and flexible job-shop scheduling problem. *The Journal of Supercomputing*, *67*, 69–83.

Zitzler, E., & Thiele, L. (1998a). An evolutionary algorithm for multiobjective optimization: The strength pareto approach. *TIK-report*, *43*.

Zitzler, E., & Thiele, L. (1998b). Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature* (pp. 292–301). Springer.