Chapter 3

Evaluation of Non-Parametric Selection Mechanisms in Evolutionary Computation: A Case Study for the Machine Scheduling Problem

Maxim A. Dulebenets

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.75984

Abstract

Evolutionary Algorithms have been extensively used for solving stochastic, robust, and dynamic optimization problems of a high complexity. Selection mechanisms play a very important role in design of Evolutionary Algorithms, as they allow identifying the parent chromosomes, that will be used for producing the offspring, and the offspring chromosomes, that will survive in the given generation and move on to the next generation. Selection mechanisms, reported in the literature, can be classified in two groups: (1) parametric selection mechanisms, and (2) non-parametric selection mechanisms. Unlike parametric selection mechanisms, non-parametric selection mechanisms do not have any parameters that have to be set, which significantly facilitates the Evolutionary Algorithm parameter tuning analysis. This study presents a comprehensive analysis of the commonly used non-parametric selection mechanisms. Comparison of the selection mechanisms is performed for the machine scheduling problem. The objective of the presented mathematical model is to determine the assignment of the arriving jobs among the available machines, and the processing order of jobs on each machine, aiming to minimize the total job processing cost. Different categories of Evolutionary Algorithms, which deploy various non-parametric selection mechanisms, are evaluated in terms of the objective function value at termination, computational time, and changes in the population diversity. Findings indicate that the Roulette Wheel Selection and Uniform Sampling selection mechanisms generally yield higher population diversity, while the Stochastic Universal Sampling selection mechanism outperforms the other non-parametric selection mechanisms in terms of the solution quality.

Keywords: optimization, Evolutionary Algorithms, non-parametric selection mechanisms, machine scheduling problems, parameter tuning, computational time

1. Introduction

Evolutionary Algorithms (EAs) and other metaheuristic algorithms have been widely used for solving complex stochastic, robust, and dynamic optimization problems. These complex problems include but are not limited to the following: vertex cover problem, Boolean satisfiability problem, maximum clique size problem, Knapsack problem, traveling salesman problem, bin packing problem, machine scheduling problems, and others [1, 2]. Some of the aforementioned problems have a non-deterministic polynomial time complete (NP-complete) complexity, while the others are non-deterministic polynomial time hard (NP-hard). The exact solution algorithms cannot be used to solve NP-complete and NP-hard problems to the global optimality for the realistic size problem instances within an acceptable computational time. On the other hand, the approximation algorithms, including EAs and other metaheuristic algorithms, are able to provide good quality solutions within a reasonable computational time. Candidate solutions to the problem of interest are encoded in the chromosomes within EAs. Different types of chromosome representations have been reported in the EA literature. For example, canonical Genetic Algorithms, developed by Holland, rely on a binary chromosome representation; while canonical Evolutionary Strategies, proposed by Rechenberg, use a real-valued chromosome representation [3, 4]. On the other hand, Genetic Programming, developed by Koza, relies on a tree-based chromosome representation [3, 4].

Once the chromosome representation is selected, the initial population is generated, and fitness values of the initial population chromosomes are estimated. Then, the EA starts an iterative process, where the population chromosomes are continuously altered using selection and EA operators (e.g., crossover and mutation) from one generation to another, aiming to identify superior solutions. The EA is terminated, once a certain stopping criterion is met (in some EAs multiple stopping criteria can be imposed). Two types of selection mechanisms are applied throughout the EA evolution: (1) parent selection, which aims to identify a subset of individuals from the offspring chromosomes, survived in the previous generation, that will participate in the EA operations and generate the new offspring chromosomes; and (2) offspring selection, which aims to identify a subset of individuals from the generated offspring chromosomes that will survive in the given generation and will be moved to the next generation. A large number of different selection mechanisms have been reported in the EA literature, which can be categorized in two groups: (1) parametric selection mechanisms (e.g., Exponential Ranking Selection, Tournament Selection, Boltzmann Selection), and (2) non-parametric selection mechanisms (e.g., Roulette Wheel Selection, Stochastic Universal Sampling, Binary Tournament Selection, Ranking Selection, Uniform Sampling).

Each EA has several parameters (e.g., population size, crossover probability, mutation probability, and others), which are generally determined based on a parameter tuning [3, 4]. A "full factorial design" methodology has been widely used for the EA parameter tuning [5]. Based on the latter methodology, the algorithm has a number of parameters (or factors - f), which have a set of candidate values (or levels - l). In order to set the appropriate EA parameter values, a total of l^f algorithmic runs will be required throughout the parameter tuning analysis. Based on the analysis of a tradeoff between the objective function and computational time values, the most promising parameter combination will be chosen. Parametric selection mechanisms will increase the number of algorithmic runs to $l^{\left(f+N^{SEL}\right)}$, where N^{SEL} – is the number of parameters for a given selection mechanism. Such increase in the number of algorithmic runs can make the parameter tuning analysis computationally prohibitive due to significant computational time required. Moreover, the parameter values of the selection mechanisms, adopted for a given set of problem instances, may worsen the EA performance, when applied to a different set of problem instances.

In order to avoid the latter drawbacks and facilitate the EA parameter tuning analysis, this study solely applies non-parametric selection mechanisms throughout the EA design. Different EA categories, which rely on various non-parametric selection mechanisms, are evaluated based on the major algorithmic performance indicators, including the objective function value at termination, computational time, and changes in the population diversity throughout the algorithmic evolution. The computational experiments are conducted for the machine scheduling problem. The machine scheduling problem deals with allocation of the available handling resources (i.e., machines) for service of the tasks (i.e., jobs), which arrive at the given facility with a specific frequency [2]. The machine scheduling problem receives an increasing attention from the community, as it is considered as an important decision problem in manufacturing, service industries, and supply chain management [6–10]. Without efficient sequencing and scheduling, the supply chain players may not be able to meet specific deadlines, which are established for processing certain products. The latter may incur substantial monetary losses and, ultimately, can even result in the customer loss. In the meantime, poor utilization of the available handling resources may cause drastic monetary losses as well. Therefore, development of advanced decision support tools for the machine scheduling problems (including effective solution algorithms, which are the primary focus of this study) becomes critical in the current competitive environment.

Findings from this research are expected to provide important insights regarding non-parametric selection mechanisms, which can be further used in future for the design of EAs. Efficient non-parametric selection mechanisms will be critical for Hybrid EAs, which along with the standard EA parameters (e.g., population size, crossover probability, mutation probability) may require setting additional parameters for the local search heuristics. The remaining sections of this chapter are organized in the following order. The next section discusses the machine scheduling environment, where the developed EA will be applied. The third section presents a mixed integer mathematical model for the machine scheduling problem. The fourth section focuses on a detailed description of the main EA components. The fifth section discusses the computational experiments, which were conducted in this study for evaluation of non-parametric selection mechanisms. The last section summarizes findings and outlines potential directions for the future research.

2. Machine scheduling

The objective of the machine scheduling problems (MSPs) is to allocate the arriving jobs among the available machines and identify the processing order of jobs on each machine. A large number of various MSPs have been widely studied in the past, such as single machine, identical machines in parallel, machines in parallel with different speeds, unrelated machines in parallel, job shop, and others [2]. The aforementioned MSPs differ in terms of machine properties (e.g., machines at a given facility have identical properties vs. machines at a given facility have different properties), job type (e.g., the processing time of a given job may vary on two machines with the same speeds based on the job type), order of machines to be visited (e.g., a given job may have to be processed on several machines in a specific order), etc.

The unrelated MSP will be studied in this chapter. Let $I = \{1, \dots, m\}$ be a set of jobs, arriving at the facility, which should be processed on the available machines within a given planning horizon. Let $J = \{1, \dots, n\}$ be a set of machines available at the given facility within a given planning horizon. Let $K = \{1, \dots, p\}$ be a set of job processing orders. Each job should be assigned for processing on one of the available machines in one of the processing orders. The machines at the given facility are assumed to have different properties (e.g., different speeds); therefore, the processing time of a given job may vary depending on the machine assignment. Furthermore, the processing time on a given machine depends on the job type (i.e., the processing time for a given job on the machines with the same speed may be different due to the job type). The latter three aspects are common for the unrelated MSPs. The MSP environment, modeled in this study, is illustrated in **Figure 1**.

Once the job arrives at the facility, it will be directed to the assigned machine for processing. If the assigned machine is processing another job at the moment, the arriving job will be queued, while waiting to be processed (see **Figure 1**). It is assumed that the facility operator will incur the job waiting cost ($c_i^{WC}, i \in I$ in USD/hour), as increasing number of waiting jobs may cause congestion at the given facility. Furthermore, the facility operator will incur the cost of processing a given job on one of the available machines ($c_i^{HC}, i \in I$ in USD/hour). Each job, arriving at the facility, must be processed by specific time ($DP_i, i \in I$ in hours). If the job processing deadline is violated, the facility operator will incur the cost due to job processing delays ($c_i^{DC}, i \in I$ in USD/hour). The objective of the facility operator is to allocate the arriving jobs among the available machines and identify the processing order of jobs on each machine,

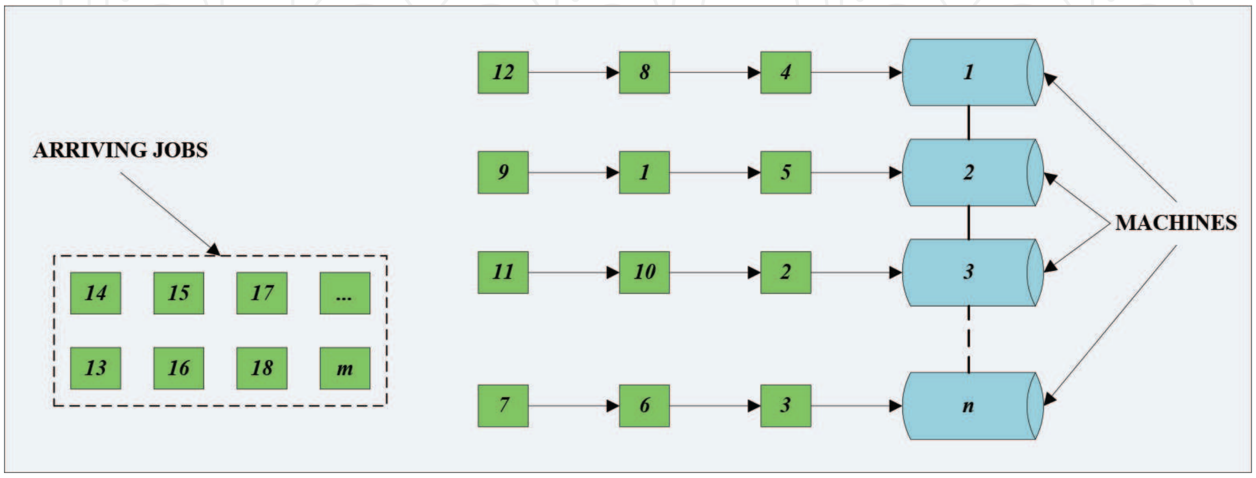


Figure 1. Machine scheduling environment.

aiming to minimize the total job processing cost, which includes: (1) the total job handling cost; (2) the total job waiting cost; and (3) the total cost due to job processing delays.

3. Mathematical model

This section of the chapter presents a mixed integer programming model for the machine scheduling problem (**MSP**), which is studied herein. A detailed description of notations used in the mathematical model and throughout this book chapter is provided at the end of the book chapter.

MSP: Machine Scheduling Problem

$$\min \left[\sum_{i \in I} \sum_{j \in J} \sum_{k \in K} \left(HT_{ij} \boldsymbol{x}_{ijk} c_i^{HC} \right) + \sum_{i \in I} \left(\boldsymbol{WT}_i c_i^{WC} \right) + \sum_{i \in I} \left(\boldsymbol{PD}_i c_i^{DC} \right) \right] \tag{1}$$

Subject to:

$$\sum_{j \in J} \sum_{k \in K} \boldsymbol{x}_{ijk} = 1 \forall i \in I \tag{2}$$

$$\sum_{i \in I} \boldsymbol{x}_{ijk} \leq 1 \forall j \in J, k \in K \tag{3}$$

$$\sum_{i^* \in I : i^* \neq i} \sum_{k^* \in K : k^* < k} \left(HT_{i^* j} \boldsymbol{x}_{i^* j k^*} + \boldsymbol{IT}_{i^* j k^*} \right) + \boldsymbol{IT}_{ijk} - AT_i \boldsymbol{x}_{ijk} \geq 0 \ \ \forall i \in I, j \in J, k \in K \tag{4}$$

$$\boldsymbol{SPT}_i \geq \sum_{i^* \in I : i^* \neq i} \sum_{k^* \in K : k^* < k} \left(HT_{i^* j} \boldsymbol{x}_{i^* j k^*} + \boldsymbol{IT}_{i^* j k^*} \right) + \boldsymbol{IT}_{ijk} - PN\left(1 - \boldsymbol{x}_{ijk}\right) \forall i \in I, j \in J, k \in K \tag{5}$$

$$\boldsymbol{WT}_i \geq \boldsymbol{SPT}_i - AT_i \forall i \in I \tag{6}$$

$$\boldsymbol{FPT}_i \geq \boldsymbol{SPT}_i + \sum_{j \in J} \sum_{k \in K} \left(HT_{ij} \boldsymbol{x}_{ijk} \right) \forall i \in I \tag{7}$$

$$\boldsymbol{PD}_i \geq \boldsymbol{FPT}_i - DP_i \forall i \in I \tag{8}$$

The objective function (1) of the **MSP** mathematical model minimizes the total job processing cost, which is composed of the following components: (1) the total job handling cost; (2) the total job waiting cost; and (3) the total cost due to job processing delays. Constraint set (2) guarantees that each job will be scheduled for processing on one of the available machines in one of the processing orders. Constraint set (3) ensures that no more than one job can be processed on each machine in a given processing order. Constraint set (4) ensures that the processing of a given job will not start before its arrival at the facility. Constraint set (5) calculates the start processing time for each job, arriving at the facility. Constraint set (6) computes the waiting time for each job, arriving at the facility. Constraint set (7) estimates the finish processing time for each job. Constraint set (8) calculates hours of delay in processing each job, arriving at the facility.

4. Evolutionary Algorithm description

MSPs belong to the class of NP-hard problems, which cannot be solved using the exact optimization algorithms to the global optimality for the realistic size problem instances within an acceptable computational time. Therefore, a set of EAs were developed in this study to solve the MSP mathematical model. EAs were differentiated based on the type of non-parametric selection mechanism adopted. This section provides an outline of the main EA steps and a detailed description of each step.

4.1. Main EA steps

The main EA steps are presented in **Algorithm 1**. The data structures for the EA variables are initialized in step 0. The initial population is generated in steps 1–2. After that, fitness of the initial population chromosomes is evaluated in step 3. Then, the EA algorithm starts an iterative process (steps 4–12), where the fittest individual is stored before applying the parent selection in step 6. The latter strategy is commonly referred to as "Elitist Strategy" in Evolutionary Computation. After that, the parent chromosomes are determined in step 7, while the offspring chromosomes are produced via the EA operations in step 8. Fitness of the offspring chromosomes is evaluated in step 9. After that, the offspring selection is executed to determine the offspring chromosomes that along with the fittest individual will be moved to the next generation (steps 10 and 11). The iterative process is continuously executed until a certain stopping criterion is met. At convergence, the proposed EA algorithm returns the best solution, which corresponds to the job to machine to processing order assignment with the least possible job processing cost. A detailed description of each EA component is presented in Sections 4.2–4.8.

Algorithm 1. Evolutionary Algorithm (EA).

```
$\boldsymbol{EA}(Data, \Omega, \sigma^C, \sigma^M, SC)$.
in: $Data$ - input data for the MSP mathematical model; $\Omega$ - population size; $\sigma^C$ - crossover probability; $\sigma^M$ - mutation probability; $SC$ - stopping criterion
out: $Solution$ - the best job to machine to processing order assignment
0: $|Population| \leftarrow \Omega$; $|Fitness| \leftarrow \Omega$; $|Parents| \leftarrow \Omega$; $|Offspring| \leftarrow \Omega$; $|Best| \leftarrow \oslash$
1: $gen \leftarrow 1$
2: $Population_{gen} \leftarrow \boldsymbol{InitPopulation}(Data, \Omega)$
3: $Fitness_{gen} \leftarrow \boldsymbol{FitnessEval}(Data, Population_{gen})$
4: while $SC \leftarrow FALSE$ do
5:     $gen \leftarrow gen + 1$
6:     $Best \leftarrow argmin(Fitness_{gen})$
7:     $Parents_{gen} \leftarrow \boldsymbol{ParentSel}(Population_{gen}, Fitness_{gen})$
```

http://dx.doi.org/10.5772/intechopen.75984

Algorithm 1. Evolutionary Algorithm (EA).

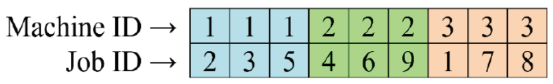
```
8:      $Offspring_{gen} \leftarrow \textbf{EAoperation}(Parents_{gen}, \sigma^C, \sigma^M)$
9:      $Fitness_{gen} \leftarrow \textbf{FitnessEval}\left(Data, Offspring_{gen}\right)$
10:     $Population_{gen+1} \leftarrow Population_{gen+1} \cup \{Best\}$
11:     $Population_{gen+1} \leftarrow \textbf{OffspringSel}\left(Offspring_{gen}, Fitness_{gen}\right)$
12: end while
13: $Solution \leftarrow argmin(Fitness_{gen} \cup Fitness_{Best})$
14: return $Solution$
```

4.2. Chromosome representation

Two-dimensional integer chromosomes will be used in this study to represent candidate solutions to the MSP mathematical model (i.e., job to machine to processing order assignments). Note that the term "chromosome" is used interchangeably with the term "individual" throughout this chapter, as both terms represent the same meaning [3]. An example of a chromosome is illustrated in **Figure 2**, where 9 jobs are scheduled for processing on 3 machines. Specifically, jobs "2", "3", and "5" are scheduled for processing on machine "1" (in that specific processing order); jobs "4", "6", and "9" are scheduled for processing on machine "2" (in that specific processing order); while jobs "1", "7", and "8" are scheduled for processing on machine "3" (in that specific processing order). The term "genes" will be used in this study to denote components of a chromosome (i.e., machine identifiers and job identifiers).

4.3. Initialization of the chromosomes and population

There are two major approaches for initializing the chromosomes and population within EAs. The first approach initializes the chromosomes and population randomly (i.e., the job to machine to processing order assignment is determined randomly). The second approach relies on application of the local search heuristics. A large number of the local search heuristics have been presented in the machine scheduling literature, such as [2]: First In First Out, First In Last Out, Shortest Processing Time First, Shortest Remaining Processing Time on the Fastest Machine, Shortest Setup Time First, and others. The local search heuristics may allow obtaining better quality solutions as compared to the random initialization mechanisms. However, the local search heuristics, which have been used for MSPs, are typically deterministic. Therefore, the population, initialized using deterministic local search heuristics, will have identical chromosomes, which will negatively affect the population diversity (i.e., only one domain of the



Machine ID →	1	1	1	2	2	2	3	3	3
Job ID →	2	3	5	4	6	9	1	7	8

Figure 2. Chromosome representation example.

search space will be explored at the population initialization stage). To avoid the latter drawback and ensure the population diversity, this study will use a random initialization mechanism to create the initial population. The number of individuals in the population is determined based on the population size parameter (Ω).

4.4. Fitness function

The fitness function of chromosomes is assumed to be equal to the objective function of the MSP mathematical model (i.e., total job processing cost). Application of various scaling mechanisms for the fitness function (e.g., linear scaling, sigma truncation, and power law scaling) to control the selection pressure throughout the algorithmic run will be one of the future research directions of this study.

4.5. Parent selection mechanism

The purpose of the parent selection mechanism is to determine a subset of individuals from the offspring chromosomes, survived in the previous generation, that will participate in the EA operations and generate the new offspring chromosomes. As discussed in the introduction section of this chapter, the main objective of this study is to evaluate various non-parametric selection mechanisms, commonly used in the literature, including the following [3, 4]:

a. **Roulette Wheel Selection** (also known as **Fitness Proportionate Selection**) – each individual of the population is assigned a portion of a roulette wheel, where a larger portion is allocated to the individual with a higher fitness value. Then, the roulette wheel is continuously rotated until the required amount of parent chromosomes has been selected.

b. **Stochastic Universal Sampling** – each individual of the population is assigned a portion of a straight line segment, where a larger portion is allocated to the individual with a higher fitness value (similar to the Roulette Wheel Selection mechanism). Then, the parent chromosomes are selected based on the evenly spaced fitness intervals (unlike Roulette Wheel Selection, which requires generating a random number each time in order to rotate the roulette wheel).

c. **Binary Tournament Selection** – multiple binary tournaments are executed, where two individuals are randomly sampled from the population during each tournament, and the individual with a higher fitness value is chosen to become a parent. The required number of tournaments is determined based on the population size.

d. **Ranking Selection** – the parent and offspring chromosomes from the previous generation are combined in a one data structure, sorted based on their fitness, and a subset of chromosomes with higher fitness values (out of the available parent and offspring chromosomes) will become parents. Such selection mechanism has been widely used in canonical Evolutionary Strategies [3] and is generally referred to as $(\mu + \lambda)$-selection, where parents (μ) are allowed to compete with offspring (λ).

e. **Uniform Sampling** – the parent chromosomes are selected from the population by uniform (or random) sampling. Unlike the aforementioned selection mechanisms, Uniform Sampling is not biased by fitness.

For a detailed description of the considered non-parametric selection mechanisms and illustrative examples of these mechanisms, this study refers to Eiben and Smith [3] and Sivanandam and Deepa [4]. Five categories of the EA algorithm, deploying different types of parent selection mechanisms, will be evaluated in this study, including the following: (1) EA with Roulette Wheel Selection (EA-RWS); (2) EA with Stochastic Universal Sampling (EA-SUS); (3) EA with Binary Tournament Selection (EA-BTS); (4) EA with Ranking Selection (EA-RS); and (5) EA with Uniform Sampling (EA-US).

4.6. EA operations

Once the parent chromosomes are selected, the developed EA algorithm applies the crossover and mutation operators in order to produce and mutate the offspring chromosomes. Both operators are described in sections 4.6.1–4.6.2 of the chapter.

4.6.1. Crossover

The order crossover is used to produce the offspring chromosomes. Selection of the latter crossover operator can be justified by the adopted chromosome representation. Specifically, certain crossover operators (e.g., N-point, whole arithmetic, uniform) may produce infeasible offspring for the integer chromosome representation [3]. On the other hand, the order crossover guarantees feasibility of the generated offspring chromosomes. An example of an order crossover operation is illustrated in **Figure 3**. Two chromosomes are randomly selected from the available parent chromosomes. The probability of parents to undergo a crossover operation is determined by the crossover probability parameter (σ^C). After that, a string of genes is copied from parent "1" to offspring "1". Note that the length of a string will be set randomly, and, therefore, may vary from one crossover operation to another. In the considered example, a string of genes with jobs "2", "6", "8", and "3" is copied from parent "1" to offspring "1". Then, the genes with missing jobs are copied from parent "2" to offspring "1". In the

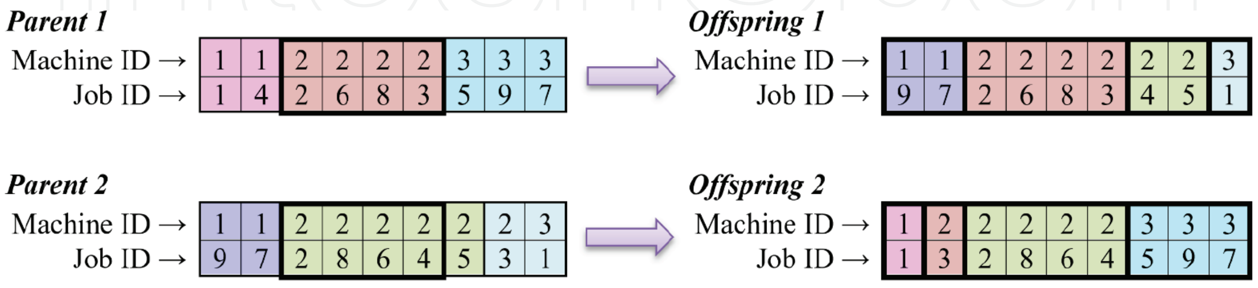

Parent 1
Machine ID → 1 1 2 2 2 2 3 3 3
Job ID → 1 4 2 6 8 3 5 9 7
Offspring 1
Machine ID → 1 1 2 2 2 2 2 2 3
Job ID → 9 7 2 6 8 3 4 5 1
Parent 2
Machine ID → 1 1 2 2 2 2 2 2 3
Job ID → 9 7 2 8 6 4 5 3 1
Offspring 2
Machine ID → 1 2 2 2 2 2 3 3 3
Job ID → 1 3 2 8 6 4 5 9 7

Figure 3. Order crossover operation example.

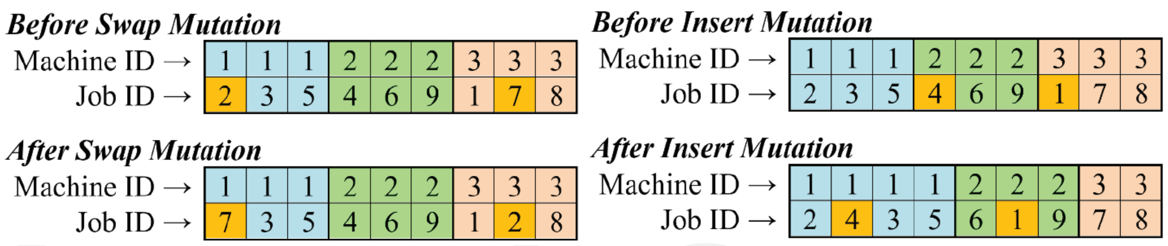

Before Swap Mutation
Machine ID → 1 1 1 2 2 2 3 3 3
Job ID → 2 3 5 4 6 9 1 7 8
After Swap Mutation
Machine ID → 1 1 1 2 2 2 3 3 3
Job ID → 7 3 5 4 6 9 1 2 8
Before Insert Mutation
Machine ID → 1 1 1 2 2 2 3 3 3
Job ID → 2 3 5 4 6 9 1 7 8
After Insert Mutation
Machine ID → 1 1 1 1 2 2 2 3 3
Job ID → 2 4 3 5 6 1 9 7 8

Figure 4. Mutation operation example.

considered example, jobs "9", "7", "4", "5", and "1" are copied from parent "2" to offspring "1". The offspring "2" is produced in a similar manner.

4.6.2. Mutation

The offspring chromosomes, produced via the order crossover, will be mutated. Two types of mutation operators will be applied in this study: (a) swap; and (b) insert. An example of a mutation operation is illustrated in **Figure 4**. In case of a swap mutation operation, job "2", initially scheduled for processing on machine "1" as the first job, is re-scheduled for processing on machine "3" as the second job. On the other hand, job "7", initially scheduled for processing on machine "3" as the second job, is re-scheduled for processing on machine "1" as the first job. In case of an insert mutation operation, job "4", initially scheduled for processing on machine "2" as the first job, is re-scheduled for processing on machine "1" as the second job. On the other hand, job "1", initially scheduled for processing on machine "3" as the first job, is re-scheduled for processing on machine "2" as the second job. Application of both swap and insert mutation operators allows altering job to machine and job to processing order assignments. The number of genes to be mutated throughout the mutation operation is determined by the mutation probability parameter (σ^M).

4.7. Offspring selection mechanism

The purpose of the offspring selection mechanism is to determine a subset of individuals from the generated offspring chromosomes that will survive in the given generation and will be moved to the next generation. This study relies on the generational offspring selection mechanism, where all offspring chromosomes will be moved to the next generation and become candidate parent chromosomes. Such offspring selection mechanism has been widely used in canonical Genetic Algorithms, proposed by Holland, and Genetic Programming, developed by Koza [3, 4].

4.8. Stopping criterion

The developed EA algorithm will be terminated, once a certain stopping criterion is met. The stopping criterion, adopted in this study, is the maximum number of generations (g^{MAX}).

5. Computational experiments

This section provides a detailed description of the computational experiments, which were conducted to evaluate the considered non-parametric selection mechanisms. Five EA categories, applying different non-parametric selection mechanisms (i.e., the EA-RWS, EA-SUS, EA-BTS, EA-RS, and EA-US algorithms, described in Section 4.5), were evaluated in terms of the objective function value at termination, computational time, and changes in the population diversity throughout the algorithmic run. All EA algorithms were coded in MATLAB 2016a. The computational experiments were executed on a CPU with Dell Intel(R) Core™ i7 Processor and 32 GB of RAM. Sections 5.1–5.3 elaborate on the input data selection for the MSP mathematical model, parameter tuning of the developed EA algorithms, and comprehensive comparative analysis of the considered non-parametric selection mechanisms.

5.1. Input data selection

The required input parameters for the MSP mathematical model were primarily generated based on the relevant literature [2, 6–36]. The adopted parameter values are presented in **Table 1**. A total of 40 problem instances were developed to conduct the computational experiments by changing the number of arriving jobs from 50 to 140 with an increment of 10 jobs, while the number of available machines was changed from 4 to 10 with an increment of 2 machines.

MSP parameter	Adopted value
Number of arriving jobs: m (jobs)	Varies based on the problem instance
Number of available machines: n (machines)	Varies based on the problem instance
Number of job processing orders: p (orders)	$p = m$ (considering the case, when all jobs are assigned for processing on one machine)
Arrival time of job i: $AT_i, i \in I$ (hours)	$Exponential(2)/60$
Handling time of job i on machine j: $HT_{ij}, i \in I, j \in J$ (hours)	$Uniform(20; 80)/60$
Deadline for processing job i: $DP_i, i \in I$ (hours)	$AT_i + Uniform(1.2; 1.5) \cdot \min_{j \in J}(HT_{ij})$
Unit handling cost for job i: $c_i^{HC}, i \in I$ (USD/hour)	$Uniform(200; 400)$
Unit waiting cost for job i: $c_i^{WC}, i \in I$ (USD/hour)	$Uniform(50; 100)$
Unit delayed processing cost of job i: $c_i^{DC}, i \in I$ (USD/hour)	$Uniform(300; 600)$
Large positive number: PN	10^6

$Exponential(a)$ – exponentially distributed pseudorandom numbers with a mean inter-arrival time of a; $Uniform(b; c)$ – uniformly distributed pseudorandom numbers, varying from b to c.

Table 1. MSP parameter values.

5.2. EA parameter tuning

A parameter selection analysis was performed for the EA-RWS, EA-SUS, EA-BTS, EA-RS, and EA-US algorithms to identify the appropriate parameter values. Each one of the developed EA algorithms has a total of 4 parameters, including the following: (1) population size – Ω; (2) crossover probability – σ^C; (3) mutation probability – σ^M; and (4) maximum number of generations – g^{MAX}. A "full factorial design" methodology [5], described in the introduction section of the chapter, was adopted for the EA parameter tuning. A total of 3 candidate values were considered for each parameter (i.e., 3^f factorial design). A total of 3 problem instances were chosen at random from the generated problem instances (see Section 5.1) in order to conduct the parameter tuning analysis.

A total of 10 replications were performed for each algorithm and each problem instance to obtain the average objective function and computational time values. The number of replications was found to be sufficient, as the objective function values did not vary substantially from one replication to another. Specifically, the coefficient of variation of the objective function values at termination did not exceed 1.00% over the performed replications for all the generated problem instances and the developed solution algorithms. Based on preliminary algorithmic runs, it was found that increasing number of replications would incur a significant increase in the computational time without a significant reduction of the objective function coefficient of variation for each EA. **Table 2** provides a summary of the parameter analysis for each EA, including the following data: (1) algorithm; 2) parameter; (3) considered candidate values for each parameter; and (4) the best parameter value, highlighted in bold font (determined based on the analysis of a tradeoff between the obtained objective function values and computational time required).

The parameter tuning analysis for the developed EA algorithms took more than 11 days (i.e., more than 51 hours for each EA). Application of parametric selection mechanisms would increase the computational time of the parameter tuning analysis even further. The latter highlights the importance of adopting non-parametric selection mechanisms.

5.3. Comparative analysis

This section focuses on a detailed comparative analysis of the considered EA algorithms, deploying different non-parametric selection mechanisms, in terms of the objective function

Algorithm\Parameter	Ω	σ^C	σ^M	g^{MAX}
EA-RWS	[40; 50; **60**]	[**0.25**; 0.50; 0.75]	[**0.01**; 0.02; 0.05]	[2000; 2500; **3000**]
EA-SUS	[40; 50; **60**]	[0.25; 0.50; **0.75**]	[**0.01**; 0.02; 0.05]	[2000; 2500; **3000**]
EA-BTS	[40; 50; **60**]	[0.25; 0.50; **0.75**]	[**0.01**; 0.02; 0.05]	[2000; 2500; **3000**]
EA-RS	[40; 50; **60**]	[0.25; 0.50; **0.75**]	[0.01; **0.02**; 0.05]	[2000; 2500; **3000**]
EA-US	[40; 50; **60**]	[0.25; 0.50; **0.75**]	[**0.01**; 0.02; 0.05]	[2000; 2500; **3000**]

Table 2. EA parameter tuning analysis summary.

values at termination and required computational time. Moreover, changes in the population diversity are analyzed throughout evolution of each EA.

5.3.1. Objective function and computational time

The developed EA-RWS, EA-SUS, EA-BTS, EA-RS, and EA-US algorithms were executed for all the generated problem instances, which were described in Section 5.1. A total of 10 replications were performed for each algorithm and each problem instance. Results of the conducted analysis are reported in **Table 3** for each algorithm and each problem instance, including the following data: (1) instance number; (2) number of arriving jobs (m); (3) number of available machines (n); (4) average objective function value at termination ($\boldsymbol{Z}$) for each EA algorithm; and (5) average computational time value (CPU) for each EA algorithm.

The average objective function values comprised 339.79 10^3 USD, 321.39 10^3 USD, 333.97 10^3 USD, and 324.14 10^3 USD, and 357.86 10^3 USD over the developed problem instances for the EA-RWS, EA-SUS, EA-BTS, EA-RS, and EA-US algorithms respectively. Therefore, EA-SUS that relies on Stochastic Universal Sampling outperformed the EAs with other non-parametric selection mechanisms in terms of the solution quality. Superiority of the EA-SUS algorithm can be explained by the fact that Stochastic Universal Sampling selects the parent chromosomes based on the evenly distributed fitness intervals and, therefore, ensures that high, medium, and low quality individuals will be given a chance to reproduce. The EA-RS algorithm, which deploys Ranking Selection, demonstrated a good performance; however, it was outperformed by the EA-SUS algorithm due to the fact that ranking is substantially biased by fitness. Ranking Selection allows only high and medium fitness chromosomes to become parents, while the individuals with low fitness values are not given any chance to reproduce.

The EA-RWS and EA-BTS algorithms were outperformed by both EA-SUS and EA-RS algorithms, as they do not guarantee that high and medium quality individuals will become parents. Although Roulette Wheel Selection and Binary Tournament Selection are biased by fitness, and the individuals with higher fitness have higher chances to reproduce, such selection mechanisms may allow a significant portion of low quality individuals to become parents, which negatively affects the objective function values and results in a premature convergence. The worst performance was recorded for the EA-US algorithm, which relies on Uniform Sampling. Uniform Sampling is not biased by fitness and gives all individuals equal chances to become parents, which may not be desirable in some cases (i.e., higher and medium quality individuals should have higher chances to reproduce, as compared to low quality individuals). Uniform Sampling can be advantageous when applied in combination with other selection mechanisms (e.g., Uniform Sampling is used at the parent selection stage, while Stochastic Universal Sampling is used at the offspring selection stage). Evaluation of the EA algorithms, which use a combination of various non-parametric selection mechanisms, will be one of the future research directions of this study.

An additional statistical analysis was conducted to investigate differences between the average objective function values at termination, suggested by the developed algorithms. The null hypothesis was assumed to be $H_0 : \mu_{EA_1} = \mu_{EA_2}$ (i.e., the average objective function value at

Instance	***m***	***n***	**EA-RWS**		**EA-SUS**		**EA-BTS**		**EA-RS**		**EA-US**	
			***Z*, 10^3 USD**	**CPU, sec**	***Z*, 10^3 USD**	**CPU, sec**	***Z*, 10^3 USD**	**CPU, sec**	***Z*, 10^3 USD**	**CPU, sec**	***Z*, 10^3 USD**	**CPU, sec**
1	50	4	141.61	52.72	137.06	56.26	138.30	56.61	138.86	55.11	143.75	61.19
2	50	6	84.43	51.76	82.89	53.53	82.94	58.46	87.32	54.45	89.29	62.14
3	50	8	59.35	52.92	56.85	54.81	57.81	59.75	57.84	55.42	61.45	63.01
4	50	10	44.52	54.54	42.98	56.07	45.19	60.40	45.14	56.32	48.43	63.94
5	60	4	198.97	58.39	190.43	60.07	191.37	65.79	192.01	60.47	195.31	68.20
6	60	6	121.49	59.69	111.83	61.33	114.81	67.33	113.47	62.06	123.11	69.39
7	60	8	82.89	61.17	80.00	62.16	80.71	68.59	80.60	63.14	84.01	70.83
8	60	10	62.31	62.12	60.37	63.35	62.98	70.17	60.61	64.00	66.13	72.39
9	70	4	278.59	65.71	259.23	67.40	269.27	74.93	267.85	68.11	285.21	77.10
10	70	6	164.86	67.19	159.63	69.17	161.25	76.34	159.87	69.38	176.77	78.77
11	70	8	119.25	68.20	111.44	70.07	114.07	77.85	113.52	71.61	122.13	80.20
12	70	10	87.04	69.71	84.72	70.99	88.47	79.31	85.13	72.72	92.40	82.75
13	80	4	358.82	73.98	341.87	75.49	347.55	84.01	342.51	77.36	368.74	87.29
14	80	6	214.52	75.25	204.72	76.47	212.32	84.90	206.70	78.60	222.90	88.00
15	80	8	148.00	76.39	142.60	79.78	148.80	85.86	143.65	79.50	155.65	89.31
16	80	10	112.84	77.59	106.88	79.47	110.21	87.52	107.23	80.47	124.04	91.83
17	90	4	460.90	81.58	444.03	83.76	446.23	92.98	446.81	85.02	484.48	96.91
18	90	6	277.98	82.84	269.76	84.89	271.57	93.84	271.19	86.96	297.47	92.80
19	90	8	191.22	83.95	180.81	86.22	195.22	95.37	180.97	88.43	203.99	94.31
20	90	10	151.94	85.54	135.86	87.94	142.50	97.32	136.17	89.75	159.42	95.93
21	100	4	600.06	89.07	564.90	92.29	580.63	101.97	568.24	94.03	601.43	101.61
22	100	6	355.99	89.84	343.57	93.10	348.84	101.66	346.33	94.63	384.70	102.99
23	100	8	249.85	91.11	228.49	94.42	243.06	103.32	229.37	95.94	260.30	103.12
24	100	10	190.11	92.81	171.88	95.95	184.94	104.81	174.24	97.51	196.91	104.42
25	110	4	720.16	96.39	678.10	99.21	706.05	109.70	678.90	101.52	745.49	108.94
26	110	6	440.85	98.03	419.34	101.31	429.49	111.74	421.62	102.51	461.96	110.18
27	110	8	300.11	99.59	280.76	102.57	292.38	112.76	281.59	104.07	317.98	111.86
28	110	10	223.92	100.81	208.87	103.90	220.49	114.61	210.76	105.66	245.74	113.24
29	120	4	858.23	104.78	802.81	108.03	848.19	120.34	816.61	110.30	900.09	120.61
30	120	6	539.24	105.52	488.90	109.46	514.46	120.73	498.86	112.01	549.63	122.26
31	120	8	356.24	107.50	343.26	111.21	363.44	122.90	345.24	113.77	389.83	123.98
32	120	10	273.65	109.06	249.24	112.04	267.37	125.11	250.87	114.76	284.92	124.56
33	130	4	1011.56	112.84	974.07	116.21	1001.36	123.02	979.15	119.14	1069.10	129.10
34	130	6	620.42	114.00	583.82	117.26	618.10	122.51	589.85	119.96	650.25	130.27

Instance	*m*	*n*	EA-RWS		EA-SUS		EA-BTS		EA-RS		EA-US	
			Z, 10^3 USD	CPU, sec	Z, 10^3 USD	CPU, sec	Z, 10^3 USD	CPU, sec	Z, 10^3 USD	CPU, sec	Z, 10^3 USD	CPU, sec
35	130	8	418.78	115.49	401.15	118.47	424.44	124.11	403.36	121.57	449.19	132.76
36	130	10	310.32	116.52	296.11	119.97	318.84	125.34	298.81	122.95	342.26	134.08
37	140	4	1184.91	120.74	1121.27	123.96	1168.76	129.81	1123.67	127.12	1267.22	138.33
38	140	6	712.51	121.49	680.00	125.12	696.63	130.66	690.01	129.63	757.52	139.52
39	140	8	499.65	123.61	465.71	126.37	483.23	131.65	469.30	129.50	529.86	141.40
40	140	10	363.36	124.70	349.41	127.91	366.61	133.38	351.45	130.79	405.48	142.81
Average:			339.79	87.38	321.39	89.95	333.97	97.69	324.14	91.66	357.86	100.56

Table 3. Objective function and computational time values for the considered EA algorithms.

termination of algorithm EA_1 $[\mu_{EA_1}]$ is equal to the average objective function value at termination of algorithm EA_2 $[\mu_{EA_2}]$), while the alternative hypothesis was assumed to be $H_a : \mu_{EA_1} < \mu_{EA_2}$ (algorithm EA_1 returns lower average objective function value at termination as compared to algorithm EA_2). The average objective function values were estimated over 40 problem instances for each EA algorithm. Based on the hypothesis testing results, no statistically significant difference has been identified among the average objective function values at termination, suggested by the EA-SUS algorithm and other developed EA algorithms, at significance level $\alpha = 0.05$. The latter finding can be justified by the fact that for some of the problem instances the developed algorithms did not demonstrate significant differences in terms of the objective function values (generally, the problem instances with lower number of arriving jobs and available machines – problem instances 1, 2, 5, 6, and others).

Furthermore, on average over all the generated problem instances the EA-SUS algorithm outperformed the EA-RWS, EA-BTS, EA-RS, and EA-US algorithms by 5.72, 3.91, 0.86, and 11.35%. However, for some of the problem instances the EA-SUS algorithm outperformed the EA-RWS, EA-BTS, EA-RS, and EA-US algorithms by up to 11.84, 7.97, 5.34, and 17.65%. Therefore, application of the EA-SUS algorithm is expected to become even more advantageous (in terms of objective function values at termination) with increasing problem size. The computational time of the developed EA algorithms did not exceed 142.81 sec over all 40 problem instances, which can be considered as acceptable.

5.3.2. Changes in the population diversity

The population diversity is critical in EAs especially at early stages of the search process. Without a diverse population, a given EA will not be able explore the available domains of the search space in an efficient manner. Lack of diversity in early generations of the EA algorithm may lead to negative consequences, including premature convergence. The population fitness values were recorded throughout evolution of the developed EA-RWS, EA-SUS, EA-BTS, EA-RS, and EA-US algorithms for each replication and each problem instance. The population fitness boxplots are illustrated in **Figures 5** and **6** for the first replication of each EA algorithm after the parent selection in generations 500, 1000, 1500, 2000, 2500, and 3000. Note that boxplots are presented only for the first replication of each EA algorithm and problem instances 37–40 (i.e., the problem instances with the largest number of arriving jobs), but

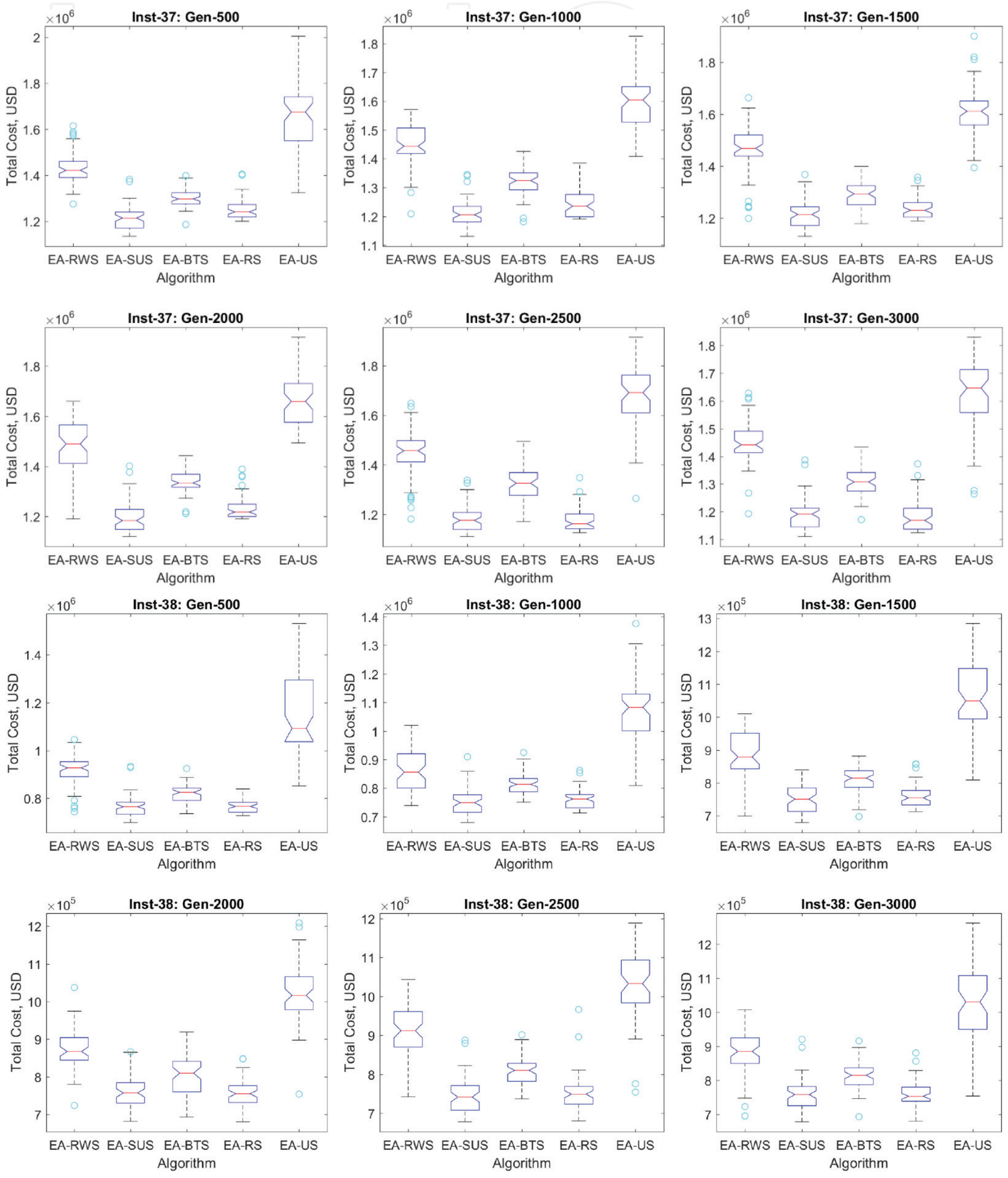


Figure 5. EA population fitness boxplots for problem instances 37 and 38.

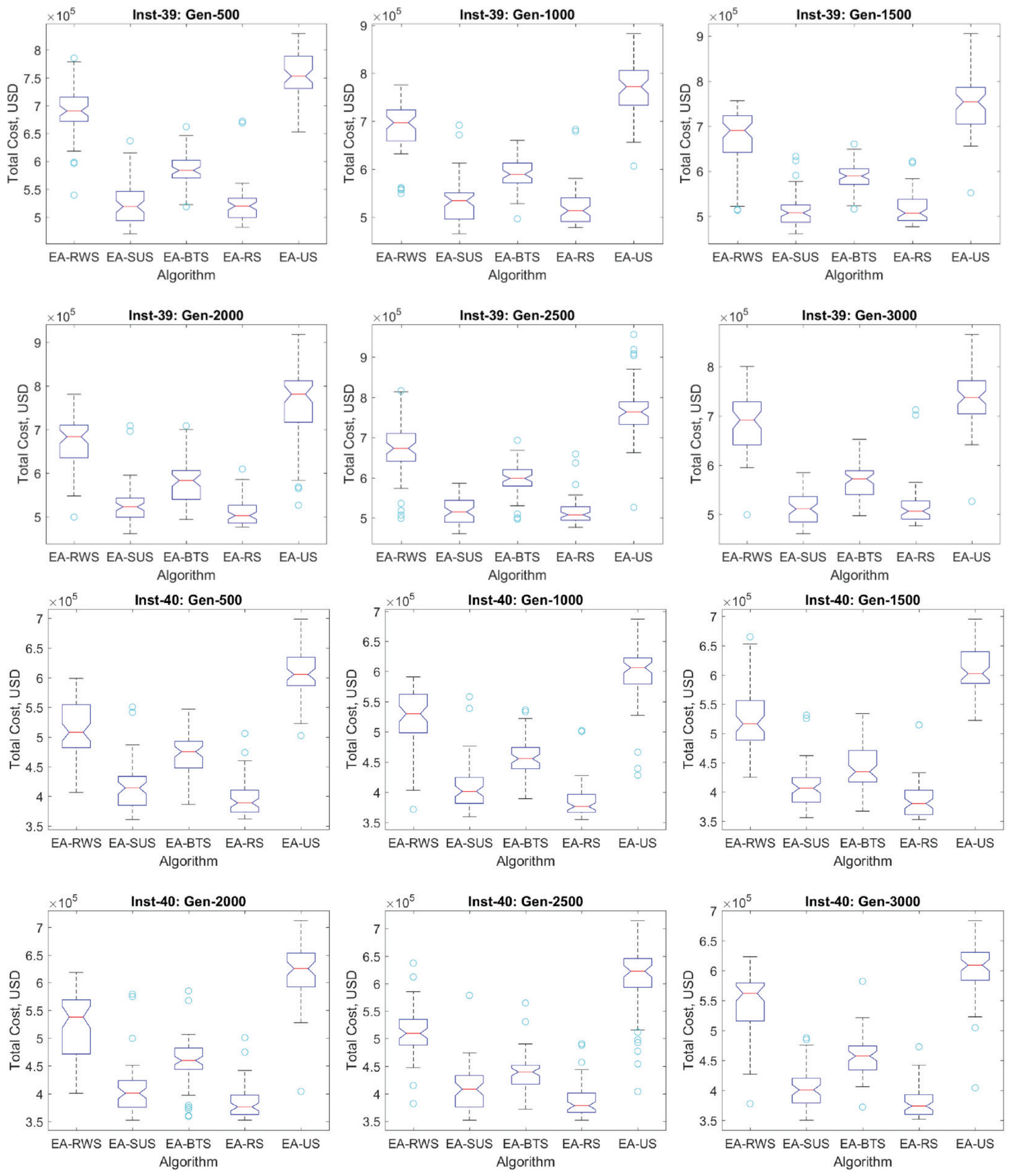


Figure 6. EA population fitness boxplots for problem instances 39 and 40.

similar patterns have been observed for the rest of algorithmic replications and problem instances. The population fitness boxplots have the following components: (1) rectangle, where the top and the bottom parts correspond to 75th and 25th population fitness value percentiles respectively; (2) median, which is shown using a red line; (3) whiskers, which are shown using dashed lines covering 99.30% of the population fitness value data points; and (4) extreme population fitness value points (falling outside of 99.30% of the population fitness value data points) or "outliers", which are shown using "°" symbol.

It can be observed that the population fitness boxplot whiskers of the EA-RWS and EA-US algorithms consistently cover a wider range of the population fitness values. The latter finding indicates that both EA-RWS and EA-US algorithms maintain a more diverse population, as compared to the EA-SUS, EA-BTS, and EA-RS algorithms. However, the quality of individuals within both EA-RWS and EA-US populations is significantly lower as compared to the EA-SUS, EA-BTS, and EA-RS populations. For example, the EA-RWS and EA-US algorithms cover the population fitness ranges of [1193.15; 1627.94] 10^3 USD and [1276.65; 1829.78] 10^3 USD respectively, while the EA-SUS algorithm covers the population fitness range of [1110.66; 1387.16] 10^3 USD for problem instance 37 at termination (i.e., in generation 3000). Therefore, as discussed in Section 5.3.1, the EA-RWS and EA-US algorithms were outperformed by the EA-SUS, EA-BTS, and EA-RS algorithms in terms of the objective function values at termination. The EA-SUS, EA-BTS, and EA-RS algorithms were able to maintain the adequate population diversity and return good quality job to machine to processing order assignments.

Throughout the computational experiments, it was found that the population diversity patterns did not change significantly from generation 500 up to generation 3000 (e.g., the range, covered by the population fitness boxplot whiskers, does not alter substantially throughout evolution of each EA after generation 500). The latter finding can be justified by the fact that the developed EAs relatively quickly identified the promising domains of the search space (i.e., within the first 400–500 generations), and then focused on exploiting the identified domains for the rest of generations, aiming to discover solutions with superior fitness values. Application of scaling mechanisms (such as linear scaling, sigma truncation, and power law scaling) will allow controlling the population diversity of the developed EA algorithms (e.g., reduce the population diversity towards the EA convergence and give higher reproduction chances to "super-individuals" – i.e. the individuals with the highest fitness values) and will be one of the future research directions of this study.

6. Concluding remarks and future research extensions

Evolutionary Algorithms and other metaheuristic algorithms have been extensively applied for solving complex stochastic, robust, and dynamic optimization problems. Two types of selection mechanisms are deployed within Evolutionary Algorithms, including the parent selection and the offspring selection. Evolutionary Algorithms have a lot of parameters, which are generally set based on the parameter tuning analysis. Parametric selection mechanisms (e.g., Exponential Ranking Selection, Tournament Selection, Boltzmann Selection) increase the number of parameters within a given Evolutionary Algorithm, which can make the parameter tuning analysis computationally prohibitive due to significant computational time required. To avoid the latter drawback and facilitate the parameter tuning analysis of Evolutionary Algorithms, this study focused on design of the Evolutionary Algorithm that solely relied on non-parametric selection mechanisms. Different categories of Evolutionary Algorithms, which applied various non-parametric selection mechanisms (Roulette Wheel Selection, Stochastic Universal Sampling, Binary Tournament Selection, Ranking Selection, Uniform Sampling), were evaluated based on the major algorithmic performance indicators.

A set of computational experiments were conducted for the unrelated machine scheduling problem, which is known to be NP-hard. The objective of the mathematical model, proposed for the problem, aimed to minimize the total job processing cost. Results indicate that the Evolutionary Algorithm with the Stochastic Universal Sampling selection mechanism outperforms the Evolutionary Algorithms with other selection mechanisms in terms of the objective function values. The worst performance was demonstrated by the Evolutionary Algorithm, which relied on the Uniform Sampling selection mechanism. Furthermore, the Evolutionary Algorithms with the Roulette Wheel Selection and Uniform Sampling selection mechanisms typically allowed maintaining higher population diversity; however, the quality of individuals within the population was lower as compared to the Evolutionary Algorithms with the Stochastic Universal Sampling, Binary Tournament Selection, and Ranking Selection mechanisms. The computational time of all the developed Evolutionary Algorithms did not exceed 142.81 sec over the considered problem instances, which can be considered as acceptable. Therefore, based on a comprehensive analysis of the commonly used non-parametric selection mechanisms, Stochastic Universal Sampling was found to be the most promising, as it was able to maintain the adequate population diversity throughout the algorithmic run and return good quality solutions at termination. Results from the conducted numerical experiments are expected to facilitate development of efficient Evolutionary Algorithms for the machine scheduling problems. Moreover, the developed problem instances and findings from this study can serve as benchmarks for the future machine scheduling studies.

The future research directions for this study include the following: (1) application of scaling mechanisms for the fitness function; (2) evaluation of the Evolutionary Algorithms, which use a combination of various non-parametric selection mechanisms (e.g., Uniform Sampling is used at the parent selection stage, while Stochastic Universal Sampling is used at the offspring selection stage); (3) consider alternative stopping criteria for the developed Evolutionary Algorithms; (4) compare various non-parametric selection mechanisms for the Hybrid Evolutionary Algorithms, which apply different local search heuristics along with the stochastic search operators; and (5) evaluate performance of the commonly used non-parametric selection mechanisms for other NP-hard problems (e.g., bin packing problem, Knapsack problem, traveling salesman problem).

Nomenclature

Sets

$I = \{1, \ldots, m\}$	set of arriving jobs
$J = \{1, \ldots, n\}$	set of available machines

$K = \{1, \ldots, p\}$	set of job processing orders

Decision variables

$\boldsymbol{x}_{ijk} \in \{0, 1\} \forall i \in I, j \in J, k \in K$	=1 if arriving job *i* is scheduled for processing on machine *j* in processing order *k* (=0 otherwise)

Auxiliary variables

$\boldsymbol{IT}_{ijk} \in R^{+} \forall i \in I, j \in J, \ k \in K$	idling time of machine *j* between processing job *i* and preceding job processed in order $(k-1)$ (hours)
$\boldsymbol{SPT}_i \in R^{+} \forall i \in I$	start processing time for job *i* (hours)
$\boldsymbol{FPT}_i \in R^{+} \forall i \in I$	finish processing time for job *i* (hours)
$\boldsymbol{WT}_i \in R^{+} \forall i \in I$	waiting time of job *i* (hours)
$\boldsymbol{PD}_i \in R^{+} \forall i \in I$	delay in processing job *i* (hours)

Parameters

$m \in N$	number of arriving jobs (jobs)
$n \in N$	number of available machines (machines)
$p \in N$	number of job processing orders (orders)
$AT_i \in R^{+} \forall i \in I$	arrival time of job *i* (hours)
$HT_{ij} \in R^{+} \forall i \in I, j \in J$	handling time of job *i* on machine *j* (hours)
$DP_i \in R^{+} \forall i \in I$	deadline for processing job *i* (hours)
$c_i^{HC} \in R^{+} \forall i \in I$	unit handling cost for job *i* (USD/hour)
$c_i^{WC} \in R^{+} \forall i \in I$	unit waiting cost for job *i* (USD/hour)
$c_i^{DC} \in R^{+} \forall i \in I$	unit delayed processing cost of job *i* (USD/hour)
$PN \in R^{+}$	large positive number

Author details

Maxim A. Dulebenets

Address all correspondence to: mdulebenets@eng.famu.fsu.edu

Department of Civil and Environmental Engineering, Florida A&M University-Florida State University, Tallahassee, FL, USA

References

[1] Hromkovič J. Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics. 2nd ed. Berlin, Germany: Springer International Publishing; 2002. p. 557. DOI: 10.1007/978-3-662-05269-3

[2] Pinedo M. Scheduling: Theory, Algorithms, and Systems. 5th ed. New York, USA: Springer International Publishing; 2016. p. 670. DOI: 10.1007/978-3-319-26580-3

[3] Eiben AE, Smith JE. Introduction to Evolutionary Computing. 2nd ed. Berlin, Germany: Springer International Publishing; 2015. p. 287. DOI: 10.1007/978-3-662-44874-8

[4] Sivanandam SN, Deepa SN. Introduction to Genetic Algorithms. 1st ed. Berlin, Germany: Springer International Publishing; 2008. p. 442. DOI: 10.1007/978-3-540-73190-0

[5] de Lima EB, Pappa GL, de Almeida JM, Gonçalves MA, Meira W. Tuning Genetic Programming parameters with factorial designs. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC); 18–23 July 2010; Barcelona, Spain. New York: IEEE; 2010. pp. 1-8

[6] Boysen N, Briskorn D, Meisel F. A generalized classification scheme for crane scheduling with interference. European Journal of Operational Research. 2017;**258**(1):343-357. DOI: 10.1016/j.ejor.2016.08.041

[7] Nagananda KG, Khargonekar P. An approximately optimal algorithm for scheduling phasor data transmissions in smart grid networks. IEEE Transactions on Smart Grid. 2017;**8**(4):1649-1657. DOI: 10.1109/TSG.2015.2497284

[8] Fernandez-Viagas V, Ruiz R, Framinan JM. A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. European Journal of Operational Research. 2017;**257**(3):707-721. DOI: 10.1016/j.ejor.2016.09.055

[9] Ozturk O, Chu C. Exact and metaheuristic algorithms to minimize the total tardiness of cutting tool sharpening operations. Expert Systems with Applications. 2018;**95**:224-235. DOI: 10.1016/j.eswa.2017.11.030

[10] Juarez F, Ejarque J, Badia RM. Dynamic energy-aware scheduling for parallel task-based application in cloud computing. Future Generation Computer Systems. 2018;**78**:257-271. DOI: 10.1016/j.future.2016.06.029

[11] Dulebenets MA. Application of evolutionary computation for berth scheduling at marine container terminals: Parameter tuning versus parameter control. IEEE Transactions on Intelligent Transportation Systems. 2018;**19**(1):25-37. DOI: 10.1109/TITS.2017.2688132

[12] Herrmann J, Proth JM, Sauer N. Heuristics for unrelated machine scheduling with precedence constraints. European Journal of Operational Research. 1997;**102**(3):528-537. DOI: 10.1016/S0377-2217(96)00247-0

[13] Weng MX, Lu J, Ren H. Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. International Journal of Production Economics. 2001;**70**(3):215-226. DOI: 10.1016/S0925-5273(00)00066-9

[14] Vallada E, Ruiz R. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. European Journal of Operational Research. 2011;**211**(3):612-622. DOI: 10.1016/j.ejor.2011.01.011

[15] Bank J, Werner F. Heuristic algorithms for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties. Mathematical and Computer Modelling. 2001;**33**(4):363-383. DOI: 10.1016/S0895-7177(00)00250-8

[16] Glass CA, Potts CN, Shade P. Unrelated parallel machine scheduling using local search. Mathematical and Computer Modelling. 1994;**20**(2):41-52. DOI: 10.1016/0895-7177(94)90205-4

[17] Pearn WL, Chung SH, Yang MH, Chen YH. Algorithms for the wafer probing scheduling problem with sequence-dependent set-up time and due date restrictions. Journal of the Operational Research Society. 2004;**55**(11):1194-1207. DOI: 10.1057/palgrave.jors.2601795

[18] Rabadi G, Moraga RJ, Al-Salem A. Heuristics for the unrelated parallel machine scheduling problem with setup times. Journal of Intelligent Manufacturing. 2006;**17**(1):85-97. DOI: 10.1007%2Fs10845-005-5514-0

[19] Kim DW, Na DG, Chen FF. Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. Robotics and Computer-Integrated Manufacturing. 2003;**19**(1):173-181. DOI: 10.1016/S0736-5845(02)00077-7

[20] Aspnes J, Azar Y, Fiat A, Plotkin S, Waarts O. On-line routing of virtual circuits with applications to load balancing and machine scheduling. Journal of the ACM (JACM). 1997;**44**(3):486-504. DOI: 10.1145/258128.258201

[21] Hsieh JC, Chang PC, Hsu LC. Scheduling of drilling operations in printed circuit board factory. Computers and Industrial Engineering. 2003;**44**(3):461-473. DOI: 10.1016/S0360-8352(02)00231-0

[22] Chen JF, Wu TH. Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints. Omega. 2006;**34**(1):81-89. DOI: 10.1016/j.omega.2004.07.023

[23] Jinsong B, Xiaofeng H, Ye J. A genetic algorithm for minimizing makespan of block erection in shipbuilding. Journal of Manufacturing Technology Management. 2009;**20**(4):500-512. DOI: 10.1108/17410380910953757

[24] Agnetis A, Flamini M, Nicosia G, Pacifici A. Scheduling three chains on two parallel machines. European Journal of Operational Research. 2010;**202**(3):669-674. DOI: 10.1016/j.ejor.2009.07.001

[25] Hu X, Bao JS, Jin Y. Minimising makespan on parallel machines with precedence constraints and machine eligibility restrictions. International Journal of Production Research. 2010;**48**(6):1639-1651. DOI: 10.1080/00207540802620779

[26] Driessel R, Mönch L. Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times. Computers and Industrial Engineering. 2011;**61**(2):336-345. DOI: 10.1016/j.cie.2010.07.001

[27] Agnetis A, Kellerer H, Nicosia G, Pacifici A. Parallel dedicated machines scheduling with chain precedence constraints. European Journal of Operational Research. 2012;**221**(2):296-305. DOI: 10.1016/j.ejor.2012.03.040

[28] Park C, Seo J. A GRASP approach to transporter scheduling and routing at a shipyard. Computers & Industrial Engineering. 2012;**63**(2):390-399. DOI: 10.1016/j.cie.2012.04.010

[29] Park C, Seo J. A GRASP approach to transporter scheduling for ship assembly block operations management. European Journal of Industrial Engineering. 2013;**7**(3):312-332. DOI: 10.1504/EJIE.2013.054133

[30] Rose CD, Coenen JM. Comparing four metaheuristics for solving a constraint satisfaction problem for ship outfitting scheduling. International Journal of Production Research. 2015;**53**(19):5782-5796. DOI: 10.1080/00207543.2014.998786

[31] Nicosia G, Pacifici A. Scheduling assembly tasks with caterpillar precedence constraints on dedicated machines. International Journal of Production Research. 2017;**55**(6):1680-1691. DOI: 10.1080/00207543.2016.1220686

[32] Dulebenets MA. The vessel scheduling problem in a liner shipping route with heterogeneous vessel fleet. International Journal of Civil Engineering. 2018;**16**(1):19-32. DOI: 10.1007/s40999-016-0060-z

[33] Dulebenets MA. The green vessel scheduling problem with transit time requirements in a liner shipping route with emission control areas. Alexandria Engineering Journal. 2018;**57**(1):331-342. DOI: 10.1016/j.aej.2016.11.008

[34] Dulebenets MA. A comprehensive multi-objective optimization model for the vessel scheduling problem in liner shipping. International Journal of Production Economics. 2018;**196**:293-318. DOI: 10.1016/j.ijpe.2017.10.027

[35] Kim DW, Kim KH, Jang W, Chen FF. Unrelated parallel machine scheduling with setup times using simulated annealing. Robotics and Computer-Integrated Manufacturing. 2002;**18**(3):223-231. DOI: 10.1016/S0736-5845(02)00013-3

[36] Caragiannis I. Efficient coordination mechanisms for unrelated machine scheduling. Algorithmica. 2013;**66**(3):512-540. DOI: 10.1007/s00453-012-9650-6