applied sciences

MDPI

Article

The Problem of Machine Part Operations Optimal Scheduling in the Production Industry Based on a Customer's Order

Predrag Mitić [1], Suzana Petrović Savić [1], Aleksandar Djordjevic [1,*], Milan Erić [1], Enes Sukić [2], Dejan Vidojević [3] and Miladin Stefanovic [1]

[1] Faculty of Engineering, University of Kragujevac, Sestre Janjic 6, 34000 Kragujevac, Serbia; predrag2904@gmail.com (P.M.); petrovic.suzana@gmail.com (S.P.S.); ericm@kg.ac.rs (M.E.); miladin@kg.ac.rs (M.S.)
[2] Faculty of Information Technology and Engineering, University Union-Nikola Tesla, Gagarina 149a, 11070 Belgrade, Serbia; sukic.enes@gmail.com
[3] Department of Informatics and Computing, University of Criminal Investigation and Police Studies, Cara Dušana 196, 11080 Belgrade, Serbia; dejan.vidojevic@kpu.edu.rs
* Correspondence: adjordjevic@kg.ac.rs

Abstract: This research focuses on small- and medium-sized businesses that provide machining or other process services but do not produce their own products. Their daily manufacturing schedule varies according to client needs. Small- and medium-sized businesses strive to operate in these circumstances by extending their customer base and creating adequate production planning targets. Their resources are limited, including the technical and technological components of their equipment, tools, people resources, time, and capacities. As a result, planning operations with the present resources of small- and medium-sized businesses in the midst of the global economic crisis is a widespread issue that must be addressed. This study seeks to offer a novel mathematical optimization model based on a genetic algorithm to address job shop scheduling and capacity planning difficulties in small- and medium-sized businesses, therefore improving performance management and production planning procedures. On the basis of the created optimization model, an appropriate software solution, and quantitative data concerning the job shop scheduling and capacity planning challenges of manufacturing operations in small- and medium-sized businesses, the study findings will be obtained. The practical implications include the establishment and development of a decision support system based on the genetic algorithm optimization method, which may improve the effectiveness of the flexible job shop scheduling problem and capacity planning in the production planning process. The given model and the application of the differential precedence preservative crossover operator within genetic algorithms are what constitute the novelty of this study.

Keywords: genetic algorithm; optimal scheduling; small and medium enterprises; metalworking industry

Citation: Mitić, P.; Petrović Savić, S.; Djordjevic, A.; Erić, M.; Sukić, E.; Vidojević, D.; Stefanovic, M. The Problem of Machine Part Operations Optimal Scheduling in the Production Industry Based on a Customer's Order. *Appl. Sci.* **2023**, *13*, 11049. https://doi.org/10.3390/app131911049

Academic Editor: Alexandre Carvalho

Received: 11 September 2023
Revised: 28 September 2023
Accepted: 30 September 2023
Published: 7 October 2023

1. Introduction

Outsourced production has emerged as a result of globalization and rapid technological development; it positively impacts several aspects of sustainability by facilitating cooperation and resource sharing among facilities to improve responsiveness, quality, efficiency, and waste management [1]. However, the global crisis has not bypassed the production industry either. Consequently, large manufacturing organizations engaged in producing goods in various industry branches focused on market research activities, product design and testing, assembly, delivery to the customer, and after-sales activities are looking to have as few unnecessary costs as possible. What is more devastating, the crisis has left its most significant impact on vulnerable small and medium enterprises (SMEs) with limited resources [2,3], for which every savings in time, money, and other resources represents a crucial point for survival. In such circumstances, SMEs may increase their

chances of survival if they apply optimization methods for allocating their limited technical and technological, human, time, and capacity resources.

An optimal schedule of operations and optimal resource allocation could be the most crucial instrument for production capacity planning in SMEs, i.e., deciding if it is possible to take new orders and deliver the products in the time intervals requested by customers. The necessity of the optimal scheduling of operations and resource allocation is derived from the fact that operation orders are completed in a given time and that the company's earnings on those operation orders in that period are optimized. A schedule of operations and optimal resource allocation serve as the nerve center of management decision-making in manufacturing systems and have the potential to facilitate autonomous manufacturing within decentralized supply chain systems. This is because shop scheduling helps integrate the physical and decisional aspects of production planning [4]. The issue presents a subclass of problems in the literature known as optimal scheduling of operations/jobs, or the problem of an optimal work schedule or activities. According to Gomes et al. [5], there are many kinds of research on this problem and methodology for its solution in engineering sciences, computer sciences, and process management, which result in a wide range of diverse literature. Methodologies for solving this problem can be classified as accurate or approximate, depending on whether their application requires an optimal or a near-optimal solution.

The schedule of operations and optimal resource allocation seeks to maximize production profit and efficiency under the premise of ensuring product quality and functionality. Single machine scheduling, parallel machine scheduling, flow shop scheduling, and the job shop scheduling problem (JSSP) are the most common types of scheduling [6]. The JSSP has attracted the attention of a significant number of scholars due to the significance of real-time scenarios. Due to the fact that the workshop scheduling problem has remained unresolved for more than two decades, it has continued to be a topic of interest up to the present day. The goal is to develop a timetable that takes into account many factors, such as makespan, flow-time, and tardiness, among others. A good schedule is an adequate operation permutation for all jobs which can lower the total makespan or idle time of machines, as stated by Dao et al. [7]. Many different factors affect the formation of the optimal schedule of operations and resource allocation, i.e., jobs. In production, the desire of decision-makers is to complete all tasks effectively and efficiently. It always boils down to reducing production time (makespan) and taking into account the limited resources that SMEs' have at their disposal. It is a complex analysis for which the SMEs selling the service do not have the appropriate resources. These SMEs comes to the market with the working hours of their resources, usually machines, with a price per hour that must be competitive. The goal that is in front of the decision-maker is to resolve capacity planning issues in SMEs based on the job shop scheduling in order to minimize the production time in a certain period and at the same time to have the maximum profits. These objectives are crucial for surviving on the market in the modern age and very often are the opposite in real systems and could create many problems during production planning. The decision-maker must have practical production planning tools that are useable in everyday business.

The main objective of this research is to present a novel optimization model to resolve capacity planning issues in SMEs based on the job shop scheduling for SMEs which sell a service to minimize the production time (makespan) in a certain period and at the same time to have the maximum profits for the SME. This objective is crucial for surviving on the market in the modern age. These two goals are often in opposition in real systems and could create many problems during production planning.

Considering that the cost of working hours of machines can be different for different machines, minimizing the makespan does not necessarily mean that costs are at a minimum i.e., it does not guarantee maximum profits. In this paper, the authors emphasize the term relative makespan, which is a solution to the optimization problem of minimum production time that guarantees maximum earnings. This model of optimal job scheduling for partial flexible job shop problems, especially suitable for the SME that sells services in

the metalworking industry, has not been considered in the available literature. With an innovative approach and a complete software solution applicable without modifications, the authors make available to SMEs a useful tool that they can use in everyday business and decision-making. The proposed model does not take into account the existence of interoperation warehouses, production disruptions, and urgent purchase orders, which in the case of large companies have a greater impact on the results.

The research results presented in this paper are theoretical and practical. The expected theoretical results are reflected in the defined model of relative makespan that connects two opposing goals in real systems, considering several factors that affect production planning and completely excluding the influence of the decision-maker on the production planning process. The model is based on data dictated by the SME environment. In addition to the previous one, the developed algorithm and software solution enable the practical application of the presented model to almost every SME of the metal processing industry, regardless of the machine park's type of production and technical and technological specifics.

The remaining sections are organized as follows. The second section is a survey of the relevant literature on this topic. In the third section, the modeling and mathematical formulation of the optimum job shop scheduling problem are stated. Also, in the third section, a method based on machine operations and optimal scheduling with an enhanced genetic operator is proposed. The fourth section presents an independent software solution with experimental findings. Conclusions are presented in the final section.

2. Literature Review

Current computer technologies and artificial intelligence (AI) approaches have become essential for small- and medium-sized enterprises (SMEs) that wish to thrive in the modern business market in order to accomplish the lowest costs and maximize resource utilization and activity scheduling [8,9]. Consequently, the method of resource use and schedule optimization will be explored.

The first problem recognized is setting up a model that fits service production with a daily varying volume in the metalworking industry. The second is to determine and apply an appropriate methodology for solving the problem. The optimization criteria are represented by an objective function functionally related to a set of control variables. The choice of the objective function is the most crucial step in solving an optimization problem. According to Živković [10] and Rao [11], there is no developed efficient or accurate algorithm for the solution of the NP-complete optimization problem, at least so far, so various authors who have dealt with this problem in recent years have used modern methods or metaheuristic algorithms to solve it. Unlike classical methods, all metaheuristic methods give results that are not accurate but converge to an optimum. Classical methods give results which with certainty represent a local extreme within the given precision. However, classical methods are applicable to solve NP-completeness only for smaller-scale problems.

The JSSP has been segmented into different categories according to the number of machines and jobs. The smaller issues can range from nine to twenty jobs and can use anywhere from four to ten machines. Medium-sized issues might range anywhere from 25 to 40 jobs and use anywhere from 15 to 20 machines. When dealing with large-scale issues, the number of jobs can range anywhere from 45 to 60, and the number of machines can range anywhere from 25 to 30 [12]. Over the course of the years, academics have suggested both accurate and heuristic approaches for solving the JSSP. These methods have many aims, such as lowering makespan, completion time variance (CTV), and the number of tardy jobs [13]. Because of the complexity of the computations involved, it is difficult to find ideal solutions for large-scale JSSPs. Instead, it is preferable to obtain near-optimal solutions through the use of heuristic methods [13,14]. Meng et al. [15] offer a unique proactive-reactive strategy to adapt to dynamic changes in working environments and to tackle the joint scheduling problem for machine tools and transportation resources. To adapt to dynamic events and construct the reschedule plan in a timely manner, the same authors developed a unique particle swarm optimization technique with an integrated ge-

netic algorithm (GA). A multiagent manufacturing system that takes into account dynamic events such as stochastic job insertions and unpredictable machine failures was presented by Zhang et al. [16]. This system was based on deep reinforcement learning and integrated the self-organization mechanism and self-learning strategy with the multiagent manufacturing system testbed. Wang and Wang [17] investigated numerous aspects and suggested a knowledge-based cooperative memetic algorithm for energy-aware distributed flow shop scheduling.

Because it has the potential to improve resource utilization and production efficiency while simultaneously lowering production costs, research on the flexible job shop scheduling problem (FJSP) has emerged as a topic of interest over the course of the past decade. To increase the scope of the JSSP, it should take into account the fact that each operation can be carried out on one of the machines that are a part of a subset of the machines in the production workshop [3,18]. Therefore, the FJSP is more difficult to solve than traditional scheduling problems. According to the results of the survey regarding the FJSP [19], it was found that 88 publications (44.67%) utilized makespan as the only objective function, while another 78 papers (39.59%) used makespan in conjunction with another objective function. The same survey shows that hybrid techniques were used to solve problems in 35% of the case studies, evolutionary algorithms in 23% of the case studies, heuristic techniques in 9% of the case studies, and other techniques in the remaining 33% of case studies. Since then, several metaheuristics have been proposed to solve the problem. By considering the characteristics of the FJSP, Jiang and Zhang [20] proposed a kind of adaptive discrete cat swarm optimization algorithm with three objectives to be optimized simultaneously, i.e., makespan, maximal machine workload, and total workload, and they employed a heuristic-based strategy to generate the initial population. Yang et al. [21] performed an evaluation with a combined improved version of nondominated sorting GA II and a surrogate measure to address the FJSP objectives, namely, makespan and robustness. Li et al. [22] proposed an efficient optimization algorithm that is a hybrid of the iterated greedy and simulated annealing algorithms to solve the FJSP with two objectives, which are simultaneously considered, namely, the minimization of the maximum completion time and the energy consumption during machine processing and material transportation. Türkyılmaz et al. [23] introduced a biobjective hybrid GA-hypervolume contribution measure that integrates GA with a multisearch algorithm and uses a hypervolume contribution measure (Δs) in its two-level selection strategy.

A suitable method for solving the FJSP that belongs to evolutionary algorithms is the GA, since, in most cases, it can find the global optimum with a very high probability [11]. One of the GA's main characteristics is to directly operate on the problem structure without derivation and function continuity limitations. GAs also have inherent implicit parallelism and global searching ability and can adjust search directions automatically and self-adaptively [24]. In recent years, the usage of GAs to solve optimal scheduling problems has increased. A GA employs a set or population of individual solutions as its searching space and is extremely resistant to a hasty approach to the local extreme [11]. Stanković et al. [25] compared meta-heuristic algorithms GA, Tabu search, and Ant colony optimization (ACO) to solve the FJSP in a flexible manner and to choose dominant solutions. Based on the literature sources supplied, a job or working order may be characterized as a sequence of distributed machine actions. Each job or distributed operation in the searching space represents a viable solution to the optimization problem. The basis of genetic engineering is the alteration of individual genes. As the gene sequence represents the individual, it is evident that if an adequate method of representing the job sequence as the individual is utilized, GAs can be very suitable for solving scheduling issues. Given the foregoing, a GA was selected as a solution to solve the problem described in this research. It should be noted that the efficiency of the GA is dependent on the genetic operators of crossover and mutation, which differ depending on the issue type [11].

In the literature, optimization of a single criterion, namely, optimizing the schedule of manufacturing processes to minimize the time necessary for their execution, predominates.

Nonetheless, it is vital to meet the makespan criterion, particularly for service-selling businesses. Even if makespan is minimized in a given time, the company's earnings must be sufficient to ensure its market viability. In this study, this sort of problem is considered.

3. Materials and Methods

3.1. Definition of the Proposed Model for Machine Part Operations Optimal Scheduling in the Production Industry Processes

The subject of inquiry is an SME in the metal processing industry. Assuming that the company has M machines, and if each machine or group of machines is a cost center determined by the price per unit time, and with capacity C, expressed in time units, and that N work orders with K operations on each work order must be completed, then the following model can be used to determine the optimal number of machines. One work order corresponds to one purchase order from a customer. Each work order must be complete by a certain time.

To that end, the authors made an effort to define the machine parts operation schedule in the manufacturing sector. Suppose that n jobs J_i (i = 1, . . ., n) should be distributed on m machine Mj (j = 1, . . ., m). A job's schedule is the distribution of one or more time intervals across one or more machines. In general, a machine is any workplace where a certain task is to be performed. Typically, the job schedule may be shown using Gantt charts that can be directed towards machine type or job type.

Different models of this problem are given in the literature depending on the goal set within the organization or process. Vujčić et al. [26] formulate one subclass of this problem as a problem of the optimal distribution of working labor, i.e., they stated that if $M_1, \ldots, M_n$ are machines to which n workers $R_1, \ldots, R_n$ should be assigned, and if it is known that the worker R_i on the machine M_j daily brings profits from c_{ij} monetary units, workers should be deployed so that the company has the maximum daily earnings. By further analysis, the authors show that the problem cannot be solved by examining all the possibilities because there are n! possibilities. However, it is essential to note that the possibilities of worker scheduling (in this subclass of problems) or activities or production operations can be viewed as permutations. It is also important to note that the authors reduce this problem to linear integer programming.

3.2. Mathematical Formulation of the Proposed Model

The considered case of optimization belongs to the FJSP, for which the formulation of the variables of the mathematical model can be conducted as follows [27]:

(i) Set of machines $M = \{M_1, M_2, M_3, \ldots, M_m\}$, where m is the total number of machines, with index h denoting each specific machine, $h \in \{1 \ldots m\}$;
(ii) Set of machine costs per time unit $K = \{K_1, K_2, K_3, \ldots, K_m\}$;
(iii) Set of working orders $N = \{N_1, N_2, N_3, \ldots, N_n\}$, where n is the total number of work orders, index i denotes each working order, $i \in \{1 \ldots n\}$;
(iv) Each work order consists of a sequence of operation $O = \{O_1, O_2, O_3, \ldots, O_k\}$, where k is the total number of operations within certain work order and index j denotes each working order operation, $j \in \{1 \ldots k\}$;
(v) Ordered set of operations $\forall N_i\ i \in \{1 \ldots n\}\ \exists$: $O_{ij} = \{O_{11}, O_{12}, O_{13}, \ldots, O_{nk^*}\}$, i.e., set of operations of the ith work order, where each work order i may have different number of k^* operations;
(vi) Ordered set of machines for the execution of the ith work order $\forall O_{ij}\ i \in \{1 \ldots n\}$, $j \in \{1 \ldots k^*\}\ \exists$: $M_{ij} \subseteq \{M_1, M_1, \ldots, M_m\}$. The execution of each operation j of a work order i (noted O_{ij}) requires one machine out of a set of given machines called $M_{ij} \subset M_m$;
(vii) Sequence l_{ij} of operation O_{ij}, $\forall N_i\ i \in \{1 \ldots n\}$ is the execution sequence of the jth operation of ith work order;
(viii) Execution time p_{ij} of operations $O_{ij}\ \forall N_i\ i \in \{1 \ldots n\}$ is the execution time of the jth operation of ith work order on hth machine;

(ix) B, a large enough positive number.

The model's control variables are the following:

In addition to the observed model's input variables, it is necessary to define a vector of control variables, that is, variables that describe the optimization objectives stated. In the observed model, these are:

$$t_{ijh}\text{—start time of operation } O_{ij} \text{ on machine h} \tag{1}$$

$$T_{hl}\text{—start time of the lth operation performed on the machine h} \tag{2}$$

$$ps_{ij}\text{—processing time of operation } O_{ij} \text{ on machine h} \tag{3}$$

$$X_{ijh} = \begin{cases} 1 & \text{if operation } O_{ij} \text{ is assigned to machine h} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$Y_{ijhl} = \begin{cases} 1 & \text{if operation } O_{ij} \text{ is assigned to machine h on lth place} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$Z_{ijhl} = \begin{cases} 1 & \text{if operation } O_{ij} \text{ can be performed on machine h} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$l_h\text{—number of assigned operation on hth machine} \tag{7}$$

$$e_{ijh}\text{—completion time of operation } O_{ij} \text{ on hth machine} \tag{8}$$

Constraints of the mathematical model:

The process time can be defined only after assigning the operation O_{ij} to the appropriate machine from the subset M_{ij}.

$$\sum_{h=1}^{m} X_{ijh} \cdot p_{ijh} = ps_{ij} \ \forall i, j \tag{9}$$

Two or more operations of the same work order cannot be executed simultaneously.

$$t_{ijh} + ps_{ij} \leq t_{i(j+1)} + \left(1 - X_{ijh}\right) \cdot B \ \forall i, h, j = 1, \ldots, k_i - 1 \tag{10}$$

The start time of each operation on the machine must follow after the end of the execution of the previous operation on that machine.

$$T_{hl} + ps_{ij} \leq T_{h(l+1)} + \left(1 - Y_{ijhl}\right) \cdot B \ \forall i, j, h, l = 1, \ldots, l_h - 1 \tag{11}$$

The beginning of each operation on the time axis must be simultaneous with the beginning of the operation on the machine on which the operation is performed.

$$e_{ijh} - t_{ijh} = \sum_{k:O_{ij} \in M_{ij}} \left(p_{ijh} \cdot X_{ijh}\right) \ \forall i, j \tag{12}$$

The difference between the end time and the start time of the operation O_{ij} must be equal to the execution time of the operation O_{ij} (process time) on the machine M_h from the subset M_{ij}.

$$T_{hl} \leq t_{ij} + \left(1 - Y_{ijhl}\right) \cdot B \ \Lambda \ T_{hl} + \left(1 - Y_{ijhl}\right) \cdot B \geq t_{ij} \ \forall i, j, h, l \tag{13}$$

At maximum, one operation O_{ij} can be performed on the lth place on the machine M_h from the subset M_{ij}.

$$\sum_{i=1}^{n} \sum_{j=1}^{k_i} Y_{ijhl} \leq 1 \; \forall h, l \tag{14}$$

One operation O_{ij} can be assigned for execution-only on one machine M_h from the subset M_{ij}.

$$\sum_{h=1}^{|M_{ij}|} X_{ijh} = 1 \; \forall i, j \tag{15}$$

One operation O_{ij} can be performed only once on an assigned machine M_h from the subset M_{ij}.

$$\sum_{l=1}^{l_h} Y_{ijhl} = X_{ijh} \; \forall i, j, h \tag{16}$$

Determining the subset of machines M_{ij} on which operation O_{ij} can be performed

$$Y_{ijhl} < Z_{ijhl} \quad \forall i, j, h \tag{17}$$

For each variable parameter, the following constraints are applied:

$$t_{ijh}, ps_{ijh}, T_{hl}, l_h, e_{ijh} \geq 0 \tag{18}$$

The objective function of the mathematical model embodies the optimization criteria. In the instance discussed in this study, the objective is to simultaneously minimize makespan and maximize revenues. In the current literature, such cases are not discussed enough. This goal consists of two opposite goals for the observed company model. One company in the production shop has, as mentioned, m machines. The operational costs of the machines are not the same. From costing analysis or analytical bookkeeping, the exact operational costs of every machine can be calculated. For example, if the execution time of one operation is 30 min, and suppose that on every machine that time is the same, the cost of execution on a machine M_1 will be $30 \cdot K_1 = 0.5 \text{ hour} \cdot 20 \frac{€}{\text{hour}} = 10 €$, and on machine M_3 it will be $30 \cdot K_3 = 0.5 \text{ hour} \cdot 15 \frac{€}{\text{hour}} = 7.5 €$. In real systems, the execution time for the same operation on a different machine is not the same, because even if the machine belongs to the same subset of machines M_{ij}, their technical and technological characteristics are different. With this in mind, it must be considered that one operation can be performed on several appropriate machines that belong to a subset of machines M_{ij} and the time for execution on every machine from the subset must be calculated as the input value of the model, as mentioned previously. K_h constants (used in the above example), representing the operational cost of machine/per time unit, are defined.

This problem could be considered as multi-criteria optimization, with two criteria: first to minimize the makespan and then to maximize earnings. Bagchi [28] and Rao [11] presented the methods for solving multi-criteria optimization problems; accordingly, in this case, the method of objective weighting should be adopted. Nevertheless, this creates new issues. The weight vector controls the optimal solution, posing a decision problem itself. Mathematically, a solution obtained with equal weights to all objectives may offer the least objective conflict. However, since the real working conditions often demand a more satisfactory solution, priority information must be induced from the decision-maker to form the objective [28].

In this paper, the authors presented a solution for this optimization problem that introduces the term relative makespan. It is clear that the time of completion of the last operation of the ith work order is, in fact, the time of the end of the ith work order, so using the already accepted notation C in the literature for the makespan of the ith work order can be written:

$$C_i = t_{i1h} + \sum_{j=1}^{k_i} \left(V_{ijh} + ps_{ij} \right) \tag{19}$$

where except V_{ijh}, which is the waiting time of the jth operation of the ith work order at the position l, other members of (19) are previously defined. In the observed model, the time of waiting is not considered, so Equation (19) can be written as

$$C_i = t_{i1h} + \sum_{j=1}^{k_i} ps_{ij} \quad (20)$$

The objective to minimize makespan can be written as

$$\min\left[\max\left(C_{1p}, C_{2p}, ..., C_{np}\right)\right] \quad (21)$$

where p = 1, . . ., and b and b are the total numbers of acceptable operations schedules.

Earnings are at a maximum when costs are at a minimum. So, to have maximum earning, costs should be at a minimum or below some value that guarantees survival on the market, and that value is the average cost production time unit. The average cost of one hour (or other time units) in the company which sell a service can be calculated as:

$$K_a = \frac{R_1 \cdot K_1 + R_2 \cdot K_{2+...+} R_m \cdot K_m}{R_1 + R_2 + \ldots + R_n}$$

where R_h h = 1,. . ., m is the total number of working hours for the machine M_h in some period (usually in real systems, that period is from 3 to 6 months), K_h h = 1,. . ., and m is the machine M_h cost per time unit. K_a means that if every work order has an average cost per time unit below or equal to the value of K_a, the earnings are at the maximum possible because the costs are paid, and the selling price is determined by management. Now, for every work order, the average cost can be calculated as:

$$K_i = t_{i1h} \cdot K_h + \sum_{j=1}^{k_i} ps_{ij} \cdot K_h \quad (22)$$

and (18) can be written as

$$C_{Ri} = t_{i1h} + \sum_{j=1}^{k_i} ps_{ij} \; \Lambda \; K_i \leq K_a \quad (23)$$

The objective to minimize relative makespan can be written as

$$\min\left[\max\left(C_{R1p}, C_{R2p}, ..., C_{Rnp}\right)\right] \quad (24)$$

Relative makespan represents makespan when the cost of production is at an average value. It also means that an acceptable schedule of operations must satisfy the condition $K_i \leq K_a$, which will be implemented by generating the initial population and through genetic operators.

3.3. Definition of GA-Related Terms in the Context of the Observed Model

A gene in the observed model is the primary carrier of information, i.e., operations. The operation contains the following information:

(i) Affiliation to work order (index i in O_{ij});
(ii) Sequence of operation O_{ij} in ith work order (index j in O_{ij});
(iii) The machine M_h on which operation O_{ij} is performed;
(iv) t_{ijh} the start time of operation O_{ij} on machine h;
(v) ps_{ij} processing time of operation O_{ij} on machine h.

A chromosome is represented by a combination of genes, which constitutes an acceptable solution. In the model observed, this would be an acceptable schedule of operations.

3.4. Flow Chart of the Proposed GA

A GA is applied to solve the scheduling problem, as shown in Figure 1, and the pseudo-code is given below (Algorithm 1).

Algorithm 1 Pseudo-code of GA applied to solve the scheduling problem

```
Start
Enter the input parameters
Create Initial population
    Calculate the fitness of individuals of the first generation
        Generation = 1
      Repeat
            Selection of parents
            Children = 0
                Repeat
                    Pick of two parents for a crossover (Parent1, Parent2)
                    Crossover DPPX (Parent1, Parent2)
                    Mutation
                    Children = Children + 1
                Until Children = population size-total number of parents
            Generation = Generation + 1
            Calculate the fitness of individuals for the current generation
             Show Gantt chart of the best schedule of the current generation
    Until generation = total number of generations
End.
```

The first step is to load the input data, which is read from the matrix record, and the rest are entered by the user (population size, the total number of parents, mutation probability and the total number of generations). After loading, the input execution of the GA begins.

3.5. Initial Population

The initial population is formed by defining the operations' schedule as an integer array over the numbers of work orders. The members of the integer array are the numbers of work orders from number 1 to number n. The array members are selected randomly, taking into account the number of operations in each work order. The integer array has as many members with the same number (the work order number) as that work order has operations. The machine is also selected randomly for each operation from a subset of machines where that operation can be performed.

3.6. Fitness Calculation

The arrays MachineTime, OperationTime, and EndTimeOij form the basis of the implemented algorithm for fitness calculation and the Gantt chart's formation, i.e., introducing the time axis. Also, within this procedure, chromosome decoding is performed. The MachineTime array represents the execution time of operations on machines, the OperationTime array represents the execution time of operations within one work order, and EndTimeOij the time of completion of the operation Oij on the machine h. The number of members of these arrays is defined by the number of operations in the schedule. In the fitness calculation array, Kij is also formed, which represents the average costs of an acceptable schedule of operations. The algorithm for forming the time axis is as follows:

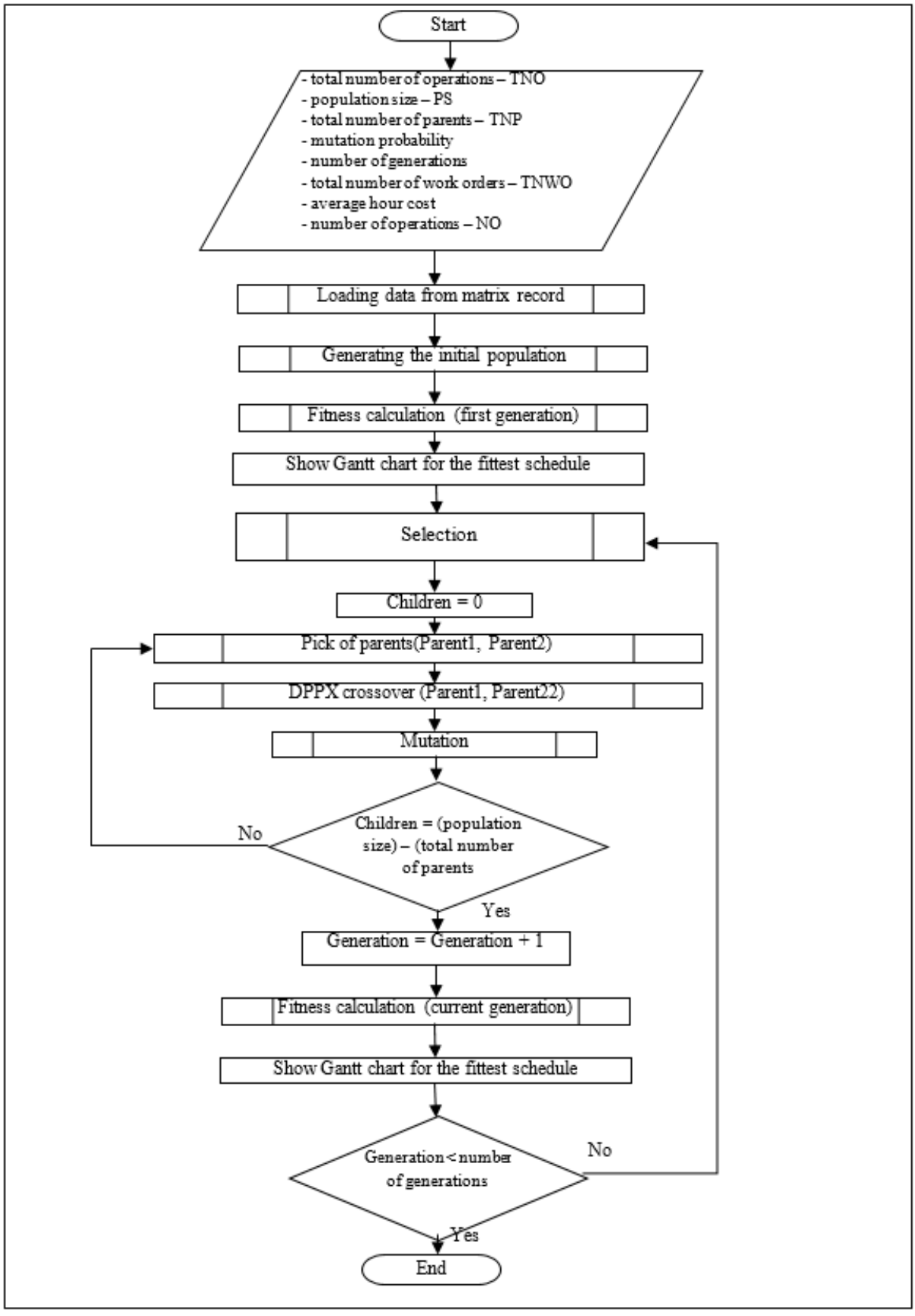

Figure 1. Flow chart of proposed GA.

The remaining variables and the limitations defined by the mathematical model are introduced in the fitness process. Most importantly, by finding the maximum of the array EndTimeOij, the observed schedule's fitness is obtained. It is now necessary to determine the best schedule of the current generation of individuals, which is determined by finding the schedule with minimum fitness from the current population, and its ordinal number is placed in the variable TopSchedule. This variable determines the best individual from the current population and thus implements the goal function defined by Equation (24) of the mathematical model. However, it should be noted that Equation (24) of the mathematical model is not implemented only in the first generation. Instead, the Equation (21) is implemented because after the application of crossover operator Differential Precedence Preservative Crossover—(DPPX), the only schedule which satisfies the condition $K_i \leq K_a$ will be transferred to the next generation, thus implementing the goal function of relative makespan defined by Equation (24) for every other generation except the first generation.

3.7. Selection of Parents

By comparing their fitness, this section of the GA decides which individuals or schedules are eligible to become parents. These individuals, i.e., those with lower execution times, should be parents. The comparison is repeated until the user-specified number of iterations indicated by the size of the parent set is reached. All individuals are initialized prior to comparing the fitness of the present population members. They are neither parents nor children, having acquired these characteristics from the preceding generation.

3.8. Crossover Operator DPPX

The goal of reproduction is to create new, preferably better individuals or create a new generation of individuals. For this purpose, the GA uses a genetic operator of crossing individuals or crossover. The first step is to choose two parents from a determined set of parents during the parents' selection. In this algorithm, two parents are chosen randomly. The only condition is that the parents cannot be the same individuals.

Xie and Chen [29] gave an overview of crossover operators used in flexible job shop problems, particularly the problems necessary to preserve the sequence of operations or previous relations because the schedule is predefined for each work order.

In this paper, the uniform precedence preservative crossover (PPX) operator was chosen as a crossover operator. Bierwirth [30] defined the PPX operator as a sequencing genetic crossover operator. Sequencing operators only change the sequence of the operations in the parent chromosomes, i.e., the assignment of operations to machines is preserved in the offspring. In this paper, a new modified PPX operator called DPPX usage will be presented, which introduces a fitness comparison between parents and the newly created individual in a crossover process as a first modification. As the problem in this paper is set so that one operation can be performed on multiple machines, another modification of the PPX operator was introduced in order to check for each operation if the machine is a member of a machine subset intended to execute the given operation.

The third modification is a condition that schedules can be transferred in the next generation only if their average production cost satisfies the condition given in Equation (22) of the mathematical model. An overview of the PPX operator will be given in the following example (Table 1).

The first step in the PPX operator algorithm is to form an array S (or a vector, as some authors call it) consisting of the digits 1 (first parent) and 2 (second parent) by random selection. This sequence defines the order in which genes are drawn from both parents. The number of members of the sequence is equal to the number of genes in the chromosome. If the member of the sequence S is number 1, then, going from left to right, the first gene of parent one is taken into the new individual, and it is deleted from both the first and second parent (going from left to right). If the member array S is number 2, the same procedure is repeated for the second parent. The procedure is repeated as long as there are genes in parents one and two. Table 2 shows the PPX operator's operation after the fourth iteration

and Table 3 at the end of the crossover process. The PPX operator is applied only to the part of the chromosome in the second row or to the operation sequence. As defined, it cannot be applied to the part of the chromosome that refers to machines, since it can lose the predetermined machine subset where an operation is performed. PPX is based on the exchange of genes between two parents while keeping the operations schedule within one work order, but not the machines on which this operation is performed.

Table 1. Example of two individuals for crossover.

Parent	**Order of Operations**								
Parent 1	O_{11}	O_{12}	O_{21}	O_{32}	O_{22}	O_{13}	O_{14}	O_{23}	O_{24}
Operation	1	1	2	3	2	1	1	2	2
Machine	1	3	1	1	5	6	8	7	9
Parent 2	O_{21}	O_{11}	O_{31}	O_{32}	O_{22}	O_{12}	O_{33}	O_{23}	O_{24}
Operation	2	1	3	3	2	1	3	2	2
Machine	1	1	1	4	4	2	7	8	3

Table 2. Overview of PPX crossover operator 4th iteration.

Crossover Process after 4th Iteration													
S array	2	1	1	2	2	2	1	1	2	1	2	1	1
Parent 1	O_{11}	O_{12}	O_{21}	O_{32}	O_{22}	O_{13}	O_{14}	O_{23}	O_{24}	O_{32}	O_{33}	O_{25}	O_{34}
Operation					2	1	1	2	2	3	3	2	3
Machine					5	6	8	7	9	4	6	8	4
Parent 2	O_{21}	O_{11}	O_{31}	O_{32}	O_{22}	O_{12}	O_{33}	O_{23}	O_{24}	O_{34}	O_{13}	O_{14}	O_{25}
Operation				3	2		3	2	2	3	1	1	2
Machine				4	4		7	8	3	9	9	6	8
Child	O_{21}	O_{11}	O_{12}	O_{31}									
Operation	2	1	1	3									
Machine	1	1	3	1									

Table 3. Overview of PPX crossover operator final phase.

Crossover Process Finished													
Child	O_{21}	O_{11}	O_{12}	O_{31}	O_{32}	O_{22}	O_{33}	O_{13}	O_{14}	O_{23}	O_{24}	O_{25}	O_{34}
Operation	2	1	1	3	3	2	3	1	1	2	2	2	3
Machine	1	1	3	1	4	4	7	6	8	7	9	8	4

Tables 2 and 3 clearly show that after applying the PPX operator, a new individual was created in which the final order of operations within the work orders was preserved. However, as the PPX operator keeps the machine on when the operation is performed, as it was selected during the formation of the initial population, it may happen that there is a better individual, that is, a schedule of operations with a better makespan, where certain operations are performed on other machines. The PPX operator in this form would be appropriate if any operation can be performed on any machine. However, this is a rare case in practice and represents only a particular case of this paper's observed model. The DPPX operator, after every iteration, checks whether the operation or withdrawn gene can be performed on the other machines from the defined subset of machines with a lower

execution time, having in mind its place in the child's chromosome. In the event of a positive result, it randomly selects a new machine from the appropriate subset of machines with a lower execution time and then reads the new execution time, which is entered with the machine in the chromosome of the new operation schedule.

The reason for the first modification lies in the fact that the number of possible operation schedules is equal to the number of permutations with repetitions and that all these schedules are acceptable from the point of view of the order of operations within work orders. As the machines on which the operations are performed are selected randomly, then the number of acceptable solutions is much higher, taking into account the arrangement of machines and that the number of GA iterations leading to the convergence of solutions is extensive as well the time of execution of the GA.

Instead of passing on every newly born individual to the next generation, a new crossover is performed regardless of whether it fits better than its parents do. An individual is passed on to the next generation only under the condition that its fitness is greater than or equal to one of the parents' fitness, i.e., only if the newly formed individual is better than one of both parents in its characteristics. If the probability is less than 50%, individuals' transfer to the new generation is carried out without comparison, and there is no rejection. If the probability is greater than 50%, the individual worse than both parents is rejected. In this way, the next generation's diversity is maintained (because not only the best individuals are transferred to the next generation, and at the same time, the number of iterations and execution time of the GA is significantly reduced). Also, it may happen that the new individual with the new machines is worse than the parents, but as previously described, it will only be discarded if it is worse than both parents. Finally, before transferring the newly formed individual to the next generation, the average cost is checked. If condition $K_i \leq K_a$ is not satisfied, a newly formed individual is rejected. The DPPX crossover operator defined in this way turns the GA presented in this paper into a semi-elitist one, the effectiveness of which will be presented in detail in the part of the paper related to the experimental results. The pseudo-code of the DPPX is shown below (Algorithm 2).

Algorithm 2 Pseudo-code of the DPPX

```
Start (Parent1-first parent, Parent2-second parent)
Set I = 0
While Parent (Schedules (I)) = true or Child (Schedules (I)) = true then I = I + 1
Parent = Schedules (I)
CParent1 = Schedule (Parent1)
CParent2 = Schedule (Parent2)
SParent = array of integer
Used1 = array of integer
Used2 = array of integer
Repeat

        for J = 1 to TNO steps of 1
        SParent(J) = Random number (1,2)
Used1(I) = 1
Used2(I) = 1
        Next J

for J = 1 to the Total number of operations in steps of 1
          If Parent (J) = 1 Then
          Counter = 1
          While Used1 (Counter) = 0 do increase Counter by 1
          IOp1 = 0
          Parent (Schedule (J)) = CParent1 (Schedule (Counter))
```

Algorithm 2 *Cont.*

```
                for InOp = 1 to j in steps of 1
                    If Parent (Schedule (InOp)) = CParent1 (Schedule (Counter))
                Then IOP1 = IOP1 + 1
                Next InOp
If the Check Machine (CParent1 (Schedule (Counter)), IOP1, CParent1 (Machine (Counter)) true
Then
SpecifyMachine (CParent1 (Schedule (Pos.)), IOP1)
Otherwise Parent (Machine (J)) = CParent1 (Machine (Counter))
Parent (TimeEx (J)) = ReadTimeEx (CParent1 (Schedule (Counter)), IOP1, CParent1 (Machine (J))
Delete withdrawn gene from parent 1 Used1 (Counter)) = 0
Position = 0

Repeat
Position = Position + 1
Until CParent2 (Schedule (Position)) = Parent (Schedule (J)) and Used2 (Position) <> 0
Used2 (Position) = 0

    If Parent (J) = 2 Then
    Counter = 1
    While Used2 (Counter) = 0 do increase Counter by 1
    IOp2 = 0
    Parent (Schedule (J)) = CParent2 (Schedule (Counter))

            for InOp = 1 to j in steps of 1
                If Parent (Schedule (InOp)) = CParent2 (Schedule (Counter))
            Then IOP2 = IOP2 + 1
            Next InOp

If the Check Machine (CParent2(Schedule (Counter)), IOP2, CParent2 (Machine (Counter)) true
Then
SpecifyMachine (CParent2(Schedule (Counter)), IOP1)
Otherwise PARENT (Machine (J)) = CParent2 (Machine (Counter))

Parent (TimeEx (J)) = ReadTimeEx (CParent2 (Schedule (Counter)), IOP2, CParent2 (Machine (J))
Delete withdrawn gene from parent 2 Used2 (Counter)) = 0
Position = 0

Repeat
Position = Position + 1
Until CParent1 (Schedule (Position)) = Parent (Schedule (J)) and Used1 (Position) <> 0
Used1(Position) = 0

Next J

Until ((the fitness of the new individual < from the fitness of the first parent) or (the fitness of the
new individual < from the fitness of the second parent) and the probability is greater than 50%)
and Ki ≤ Ka

End DPPX
```

3.9. Mutation Operator

A swap mutation operator, slightly modified due to the necessity to check the machine, is chosen in this model. The position of genes is randomly chosen. If these genes are the same, then the mutation does not make sense (the same individual was obtained) because the operation schedule must be preserved—operation O_{34}, which was in the 10th position when moved to the third position, becomes O_{31}. On the other hand, if the genes are not

the same or do not belong to the same work order, the mutation makes sense because then operation O_{23} becomes O_{33} and operation O_{32} becomes O_{22}.

It is necessary to check the machine for each gene in a new individual and assign a new one from the appropriate subset if necessary. Since the sequence of operations must be preserved within the work order, the genes to be included in the mutation are colored blue in Table 4, because although the same gene distribution remains 2-1-3 between the fourth and eighth positions of the mutated genes, they are not the same operations because the order in the work order has changed.

Table 4. Swap mutation operator.

	Individual before the Mutation												
Individual	O_{21}	O_{11}	O_{31}	O_{32}	O_{22}	O_{12}	O_{33}	O_{23}	O_{24}	O_{34}	O_{13}	O_{14}	O_{25}
Operation	2	1	3	3	2	1	3	2	2	3	1	1	2
Machine	1	1	1	4	4	2	7	8	3	9	9	6	8
	Individual after the Mutation												
Individual	O_{21}	O_{11}	O_{31}	O_{22}	O_{23}	O_{12}	O_{32}	O_{33}	O_{24}	O_{34}	O_{13}	O_{14}	O_{25}
Operation	2	1	3	2	2	1	3	2	2	3	1	1	2
Machine	1	1	1	4	6	2	4	7	3	9	9	6	8

4. Presentation of the Independent Software Solution and Experimental Results

4.1. Application of the Developed Solution

The application in which the previously presented solution is implemented is written in the object-oriented programming language Delphi 10.4. A database, i.e., a table with an example used to execute the application and display experimental results, was created in Microsoft Access 2016, and it consists of one specific table. The user can change the table, entering new work orders, operations, and times so that it is possible to test the displayed solutions for different data. Also, it is possible to increase or decrease the number of machines in the database. The table is arranged to correspond to the matrix record of operations shown in part 2 of this paper. Table 5 shows an example of a top relational database table containing data for application execution.

Table 5. Input data for application execution.

RnIndeks	**OIndeks**	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**	**M10**	**M11**
1	1	10	0	0	0	0	0	0	0	0	0	
1	2	0	11	14	0	0	0	0	0	0	0	
1	3	0	5	12	17	0	7	0	0	0	0	
1	4	0	0	0	36	27	40	36	12	22	0	
2	1	11	0	0	0	0	0	0	0	0	0	
2	2	0	0	0	8	8	0	0	0	0	0	
2	3	0	8	7	3	0	8	9	10	0	0	
2	4	0	3	3	9	0	0	0	11	17	0	
2	5	0	8	6	0	0	0	0	0	0	0	
3	1	5	0	0	0	0	0	0	0	0	0	
3	2	0	3	5	11	0	0	0	0	0	0	
3	3	0	9	3	0	5	3	14	0	0	0	
3	4	0	5	6	8	10	11	15	0	9	0	
3	5	0	3	5	12	7	8	20	0	0	0	

Table 5. *Cont.*

RnIndeks	OIndeks	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11
4	1	25	11	45	33	0	0	0	32	11	0	
4	2	16	0	0	25	22	33	28	0	0	0	
4	3	0	36	25	0	0	36	25	57	10	0	
4	4	0	0	25	30	35	40	0	0	0	0	
4	5	15	0	26	29	32	35	25	23	24	0	
0	0										0	

There are four work orders in the table; the first has four operations, and the others have five operations. The first column represents the work order number (RIndex), i.e., the work order index, and the second column represents the operation index (OIndex) within the work order. The following columns represent machines, and the execution time of that operation is in the intersection of the corresponding column of the machine and operation if it is predicted that the operation will be performed on that machine if the execution time is not 0. Execution times are given in minutes in this table. In this stage of the present research, execution times are intended to be entered in minutes, while the option of entering them in other time units will be provided in the future upgrade of the application. The table can be easily modified to expand with new machines and new operations. Each machine's operation cost per unit time is known and has been previously read from the appropriate database.

It should be noted that the total number of machines and the total number of operations that the user can modify must correspond to the number of machines and the number of operations located in the input data set.

4.2. Experimental Testing of Proposed GA

The purpose of this research was to create a helpful tool for SMEs to use in everyday business and decision-making focused on the FJSP. The basis of this tool is the proposed GA, and it was necessary to test it on existing problem instances from the available literature and compare its effectiveness with existing solutions. A large number of problem instances can be found in the available literature (http://www.idsia.ch/monaldo/fjsp.html, accessed on: 15 February 2023). For experimental testing, the data set defined by Brandimarte [31] was chosen, which consisted of 10 problems with the number of working orders ranging from 10 to 20, the number of machines between 4 and 15, and the number of operations for each working order ranging from 5 to 15. RMGA was tested on 2.20 GHz PC with 8 GB of memory.

For experimental testing, the application used the test data stored in txt files, as shown in Figure 2. In the upper right corner of the application, there is a list of.txt files, each of which represents an instance of a problem. The classic PPX crossover operator and the DPPX crossover operator were both used to test each of the problem situations.

In order to carry out our experiment, authors tested the algorithm with the following parameters:

- Population size: 400;
- Number of parents 30% of population size: 120;
- Mutation probability: 5%;
- Number of generations: 150.

The effectiveness of the proposed algorithm is compared with the existing algorithms of older dates from the available literature, like M&G Mastrolilli and Gambardella [32], GENACE Ho and Tay [33], Pezzela et al. [34], and Zhang et al. [35], and with the existing algorithms of newer dates such as Hybrid Genetic Tabu Search (HGTS) defined by Palacios et al. [36] and Driss et al. [37] and Teaching–Learning-Based Optimization (TLBO) proposed by Buddala et al. [38] and Kong et al. [24]. In order to obtain meaningful results, the authors

performed optimization five times for each instance. Table 6 shows the experimental results compared to the previously mentioned algorithms.

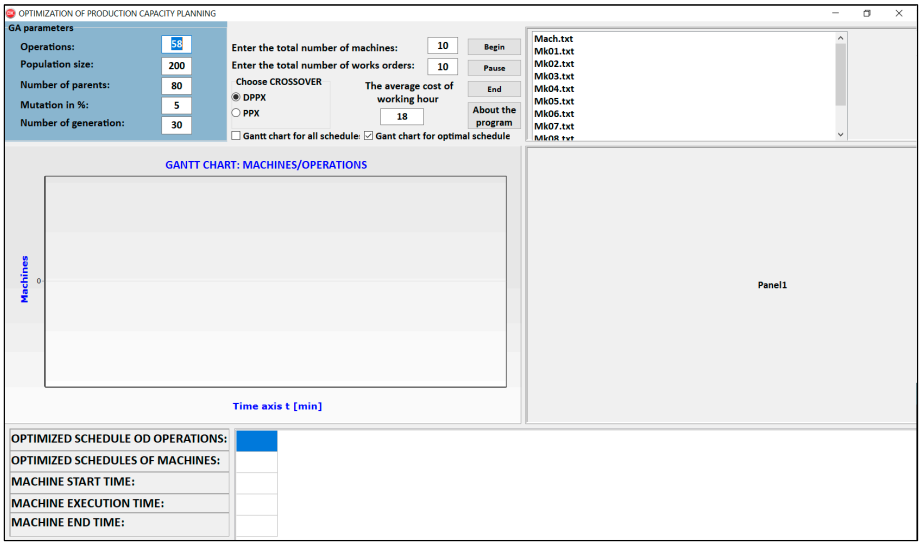

Figure 2. Initial form of application for experimental testing.

Table 6. Experimental results of GA execution and comparison with other GAs.

Problem Instance	n × m	BS	M & G	GENACE	Pezzella et al. [34]	Zhang et al. [35]	HGTS	Driss et al. [37]	TLBO	DIGWO	RMGA (PPX)	RMGA (DPPX)
		C_m	C_m	C_m	C_m	C_m	C_m	C_m	C_m	C_m	C_m	C_R
Mk01	10 × 6	37	40	40	40	40	40	37	40	40	38	38
Mk02	10 × 6	26	26	32	26	26	26	26	28	26	34	33
Mk03	15 × 8	204	204	N/A	204	204	204	204	204	204	199	191
Mk04	15 × 8	60	60	67	60	60	60	60	63	60	62	61
Mk05	15 × 4	172	173	176	173	173	172	173	172	173	173	172
Mk06	10 × 15	57	58	67	63	58	57	67	65	62	71	60
Mk07	20 × 5	139	144	147	139	144	139	148	144	140	168	158
Mk08	20 × 10	523	523	523	523	523	523	523	523	523	506	494
Mk09	20 × 10	307	320	320	311	307	307	307	311	307	356	344
Mk10	20 ×15	189	229	229	212	198	198	212	214	211	326	316

The first column reports the instance name, and the second column reports the number of machines and number of working orders for each instance, where m is the number of machines and n is the number of working orders. The third column reports the best-known lower bound, which was taken from Buddala et al. [38] and Driss et al. [37]. It should be noted that the best lower bound for the Mk01 problem instance is taken from Driss et al. [37]. From the fourth to the eleventh columns are the best results of the previously mentioned algorithms. The twelfth column shows the results of executing our algorithm named Relative Makespan GA (RMGA) with the classic PPX crossover, and the thirteenth shows the results of executing with the DPPX crossover.

Table 7 shows the deviation of RMGA results in relation to the best lower bound (the third column from Table 6). The deviation was calculated according to the formula (CM-CMRMGA)/CM in %, where CM is the best-achieved makespan of other GAs and CMRMGA is the best-achieved makespan with RMGA. The experimental results and deviation calculation show that RMGA achieves quite good (and in some cases excellent) results when the average number of machines per work order is less than or equal to two (instances Mk01, Mk03, Mk05, and Mk08). Its efficiency is significantly lower when the average number of machines is greater than two (instances Mk02, Mk06, Mk07, Mk09, and

Mk10) compared to other GAs (the average number of machines is the third number in first row of every txt file from Brandimarte's problem instances).

Table 7. Deviation of RMGA experimental results compared to best-known lower bound.

Problem Instance	n × m	BS	RMGA (PPX)	RMGA (DPPX)	RMGA (PPX)	RMGA (DPPX)
		C_m	C_m	C_R	Deviation	Deviation
Mk01	10 × 6	37	38	38	−2.70%	−2.70%
Mk02	10 × 6	26	34	33	−30.77%	−26.92%
Mk03	15 × 8	204	199	191	2.45%	6.37%
Mk04	15 × 8	60	62	61	−3.33%	−1.67%
Mk05	15 × 4	172	173	172	−0.58%	0.00%
Mk06	10 × 15	57	71	70	−24.56%	−22.81%
Mk07	20 × 5	139	168	158	−20.86%	−13.67%
Mk08	20 × 10	523	506	494	3.25%	5.54%
Mk09	20 × 10	307	356	344	−15.96%	−12.05%
Mk10	20 ×15	189	326	316	−72.49%	−67.20%

This is a consequence of using the DPPX sequential genetic operator, which, despite modifications compared to the classic PPX operator (where the deviations compared to the best lower limit are greater, proving the effectiveness of the DPPX operator), cannot ensure a sufficient degree of crossing between individuals. There is a need to change individuals "more" in order to achieve a better makespan, and this can be accomplished by increasing the mutation rate [39], because a mutation usually corresponds to the generation of a neighbor in some neighborhood if permutation coding is used. In this case, the mutation rate denotes the probability of applying a mutation to the current individual, and in these cases, the mutation rate is typically much larger than the five percent used in a first series of experiments.

The authors performed another test of the presented algorithm five times for every instance where the negative deviation was greater than three percent and tested different values of the algorithm parameters. The experimental results show that the following parameters are the most effective:

Small- and medium-scale problems (up to 100 operations on five machines):

- Population size: 250;
- Number of parents 40% of population size: 100;
- Mutation probability: 50%;
- Number of generations: 150.
- Large-scale problems (up to 240 operations on 15 machines):
- Population size: 500;
- Number of parents 50% of population size: 250;
- Mutation probability: 50%;
- Number of generations: 600.

The results of the second series of experiments are shown in Table 8.

The results of both series of experiments clearly show that RMGA is very effective on small- and medium-scale problems, and its effectiveness decreases slightly with large-scale problems. In seven of the ten instance problems considered, RMGA outperforms the best-known lower bound. For small- and medium-scale problems, the speed of RGMA execution was 10–20 s, but for large-scale problems, the speed was much higher, also because the application simultaneously creates Gantt charts and graphically displays the convergence of GA solutions in real time. In order to accurately determine the execution speed of RMGA, it is necessary to test only the algorithm itself without graphical representations of solution convergence or Gantt charts. Speed measurement and speed improvement of the RMGA will be part of future research.

Table 8. Results of the second series of experiments.

Problem Instance	n × m	BS	RMGA (DPPX)	AC	RMGA (DPPX)
		C_m	C_R	K_I	Deviation
Mk02	10 × 6	26	23	17.45	11.54%
Mk06	10 × 15	57	62	18.46	−8.77%
Mk07	20 × 5	139	125	17.43	10.07%
Mk09	20 × 10	307	308	18.37	−0.33%
Mk10	20 × 15	189	223	18.52	−17.99%

Figures 3 and 4 show Gantt charts and a graphical representation of the RMGA solution with the parameters used for problem instances Mk01 and Mk07. Taking into account the experimental results, it is clear that RMGA achieves very good results compared to other algorithms, primarily due to the application of the DPPX operator and a higher probability of mutation.

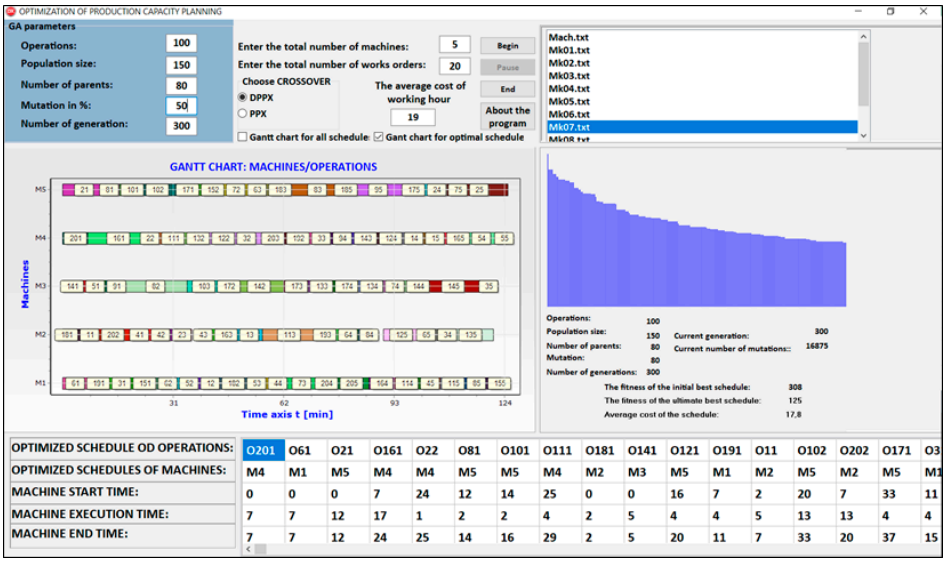

Figure 3. Gantt chart for problem instance Mk07.

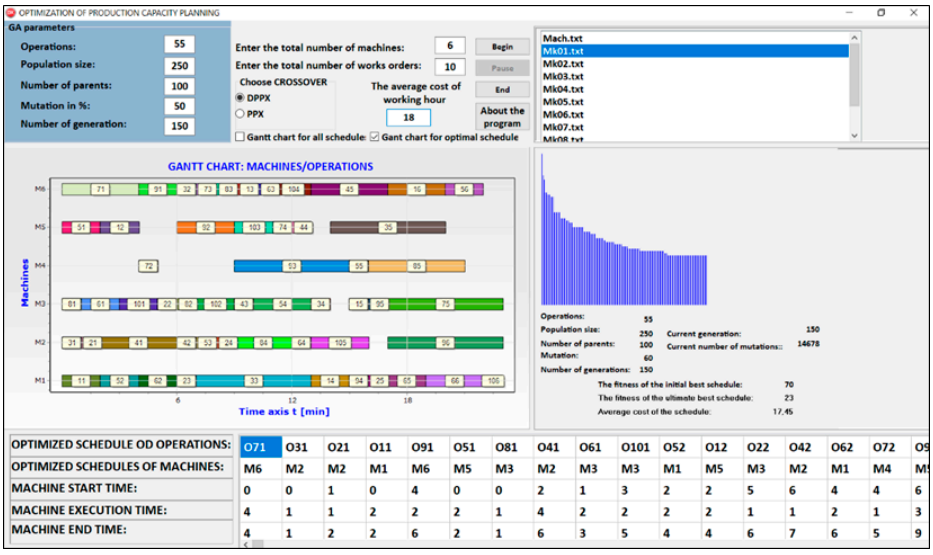

Figure 4. Gantt chart for problem instance Mk01.

By changing the numbers in Table 9, the lower limit of the average hourly cost of operations scheduling can be lowered or raised. This informs the decision-maker how much it will cost the SME to create that operation schedule.

Table 9. Example of average cost per hour for every machine.

Machine	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Average cost per hour	15	16	18	20	18	19	21	22	21	16	17	18	19	15	20

To show the influence of Ka on makespan, the authors conducted another series of experiments on instances Mk04, Mk08, and Mk09 that were randomly selected. In these experiments, the authors removed condition Ki $\leq$ Ka from DPPX crossover and ran the algorithm under the following parameters:

- Population size: 400;
- Number of parents 30% of population size: 120;
- Mutation probability: 50%;
- Number of generations: 100.

The results are shown in Table 10.

Table 10. Influence of average hour cost on behavior of RMGA.

Problem Instance	n × m	BS	RMGA with Ki $\leq$ Ka	RMGA without Ki $\leq$ Ka	AC	AC
		C_m	C_R	C_M	with Ki $\leq$ Ka	without Ki $\leq$ Ka
Mk04	15 × 8	60	61	49	18.50	18.71
Mk08	20 × 10	523	494	412	17.25	18.31
Mk09	20 × 10	307	308	304	18.37	18.52

From the experimental results, it is clearly seen that the makespan decreases compared to the best-known result, but the production costs increase and exceed the average price of 18 monetary units per hour that was entered as an input parameter. It is clear that the condition Ki $\leq$ Ka is necessary in the algorithm since it gives the decision-maker the opportunity to plan production in accordance with the best possible capacity utilization in the time frame but also with real production costs. It is clear that if the costs of each machine were identical, the relative makespan would not make sense, but in real production, this is not the case, and the model and software solution presented in this paper can be a very useful tool when planning production with cost control.

5. Discussion and Conclusions

In this paper, a novel optimization approach for resolving capacity planning difficulties in SMEs is proposed. The approach is based on job shop scheduling for SMEs to simultaneously minimize production time and maximize earnings, which is essential for SMEs' survival on the current market. The proposed approach, unlike most other models in the literature, takes into account the average cost price of production per unit of time and thus gives the possibility of determining the makespan when the production costs are average, i.e., so that there is always a profit. If the cost price of the working hour of the machine is not taken into account, it can happen that the makespan is minimal, but the earnings are minimal or even in extreme cases non-existent, because most of the operations in such a schedule are performed on machines whose working hour cost is high, and thus do not provide financial resources for survival on the market.

Through a mathematical model, the offered model defines relative makespan in a theoretical sense and demonstrates its practical applicability in the proposed software solution. Through the idea of relative makespan, the model combines two contradictory aims (minimize makespan and maximize profitability). It solves the P-FJSP issue, but it is

also conceivable to solve the T-FJSP (Total Flexible Job Shop Problem) when any operation may be performed on every machine by only modifying the matrix record input data. In addition, the model is not confined to the production plant, since any machine may be viewed as a workplace, and the model and software offered can address, without change, the classic problem of labor scheduling in other vocations. In addition to optimizing execution time, the proposed technique also optimizes the machine layout. It displays a machine-oriented Gantt chart that serves as the foundation for production planning. In contrast to the models in the published literature, the DPPX operator minimizes the number of iterations and retains the variety of generations, hence limiting the convergence of GA to the local optimum while simultaneously reducing GA execution time. The program may also be used as a production planning tool completely independently of other commercial software, making it an ideal alternative for small- and medium-sized enterprises.

However, the provided model has several drawbacks. The model excludes production disturbances such as cancellation of manufacturing orders, equipment malfunction, and urgent work orders. The inclusion of these limits might be one of the primary future research priorities. In addition, there are no intermediary warehouses between the machines in the provided model, thus waiting time and storage expenses are not accounted for. The most common disruptions in production, such as a machine breakdown or an urgent purchase order, do not greatly affect the practical applicability of the model. The proposed model is flexible and allows variations on a daily basis. A break point on the timeline in the operation schedule caused by, for example, a machine failure becomes the starting point of a new operation schedule in which the defective machine is removed from the schedule until it is repaired, so that the production process continues according to the newly formed operation schedule. The lack of inter-operational warehouses, however, limits the practical applicability of the proposed model in real conditions, since waiting time, transportation time, and storage costs are not taken into account.

As one of the primary subjects of future study, the establishment of restricted intermediate warehouses would enable the implementation of the model and software solution given to large organizations. The second most important goal of future research is testing the proposed model in real conditions and testing its efficiency and effectiveness, as well as further improving the algorithm based on the results obtained from the real-world case study.

Author Contributions: Conceptualization, P.M. and M.E.; methodology, P.M. and A.D.; software, P.M.; validation, S.P.S., E.S. and D.V.; formal analysis, M.S.; investigation, P.M.; resources, M.S.; data curation, S.P.S.; writing—original draft preparation, A.D.; writing—review and editing, P.M.; visualization, S.P.S.; supervision, M.E.; project administration, E.S.; funding acquisition, P.M., E.S., D.V., A.D. and M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The Delphi software solution data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Toptal, A.; Sabuncuoglu, I. Distributed scheduling: A review of concepts and applications. *Int. J. Prod. Res.* **2010**, *48*, 5235–5262. [CrossRef]
2. Caballero-Morales, S.O. Innovation as recovery strategy for SMEs in emerging economies during the COVID-19 pandemic. *Res. Int. Bus. Financ.* **2021**, *57*, 101396. [CrossRef]
3. Dai, M.; Tang, D.; Giret, A.; Salido, M.A. Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. *Robot. Comput.-Integr. Manuf.* **2019**, *59*, 143–157. [CrossRef]

4. Rauch, E.; Dallasega, P.; Matt, D.T. Sustainable production in emerging markets through Distributed Manufacturing Systems (DMS). *J. Clean. Prod.* **2016**, *135*, 127–138. [CrossRef]
5. Gomes, M.C.; Barbosa-Povoa, A.P.; Novais, A.Q. Optimal scheduling for flexible job shop operation. *Int. J. Prod. Res.* **2005**, *43*, 2323–2353. [CrossRef]
6. Alotaibi, A.; Lohse, N.; Vu, T.M. Dynamic agent-based bi-objective robustness for tardiness and energy in a dynamic flexible job shop. *Procedia Cirp* **2016**, *57*, 728–733. [CrossRef]
7. Dao, T.K.; Pan, T.S.; Pan, J.S. Parallel bat algorithm for optimizing makespan in job shop scheduling problems. *J. Intell. Manuf.* **2018**, *29*, 451–462. [CrossRef]
8. Mokhtari, H.; Hasani, A. An energy-efficient multi-objective optimization for flexible job-shop scheduling problem. *Comput. Chem. Eng.* **2017**, *104*, 339–352. [CrossRef]
9. Zhang, J.; Ding, G.; Zou, Y.; Qin, S.; Fu, J. Review of job shop scheduling research and its new perspectives under Industry 4.0. J. *Intell. Manuf.* **2019**, *30*, 1809–1830. [CrossRef]
10. Živković, M. *Algoritmi*; Mathematical Faculty: Belgrade, Serbia, 2000.
11. Rao, S.S. *Engineering Optimization: Theory and Practice*; John Wiley & Sons: Hoboken, NJ, USA, 2019.
12. Amjad, M.K.; Butt, S.I.; Kousar, R.; Ahmad, R.; Agha, M.H.; Faping, Z.; Asgher, U. Recent research trends in genetic algorithm based flexible job shop scheduling problems. *Math. Probl. Eng.* **2018**, *2018*, 9270802. [CrossRef]
13. Vital-Soto, A.; Baki, M.F.; Azab, A. A multi-objective mathematical model and evolutionary algorithm for the dual-resource flexible job-shop scheduling problem with sequencing flexibility. *Flex. Serv. Manuf. J.* **2022**, *35*, 626–668. [CrossRef]
14. Güçdemir, H.; Selim, H. Integrating simulation modelling and multi criteria decision making for customer focused scheduling in job shops. *Simul. Model. Pract. Theory* **2018**, *88*, 17–31. [CrossRef]
15. Meng, L.; Zhang, C.; Shao, X.; Ren, Y. MILP models for energy-aware flexible job shop scheduling problem. *J. Clean. Prod.* **2019**, *210*, 710–723. [CrossRef]
16. Zhang, Y.; Zhu, H.; Tang, D.; Zhou, T.; Gui, Y. Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems. *Robot. Comput.-Integr. Manuf.* **2022**, *78*, 102412. [CrossRef]
17. Wang, J.J.; Wang, L. A cooperative memetic algorithm with feedback for the energy-aware distributed flow-shops with flexible assembly scheduling. *Comput. Ind. Eng.* **2022**, *168*, 108126. [CrossRef]
18. Shen, L.; Dauzère-Pérès, S.; Neufeld, J.S. Solving the flexible job shop scheduling problem with sequence-dependent setup times. *Eur. J. Oper. Res.* **2018**, *265*, 503–516. [CrossRef]
19. Chaudhry, I.A.; Khan, A.A. A research survey: Review of flexible job shop scheduling techniques. *Int. Trans. Oper. Res.* **2016**, *23*, 551–591. [CrossRef]
20. Jiang, T.H.; Zhang, C. Adaptive discrete cat swarm optimisation algorithm for the flexible job shop problem. *Int. J. Bio-Inspired Comput.* **2019**, *13*, 199–208. [CrossRef]
21. Yang, Y.; Huang, M.; Wang, Z.Y.; Zhu, Q.B. Robust scheduling based on extreme learning machine for bi-objective flexible job-shop problems with machine breakdowns. *Expert Syst. Appl.* **2020**, *158*, 113545. [CrossRef]
22. Li, J.Q.; Du, Y.; Gao, K.Z.; Duan, P.Y.; Gong, D.W.; Pan, Q.K.; Suganthan, P.N. A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem. *IEEE Trans. Autom. Sci. Eng.* **2021**, *19*, 2153–2170. [CrossRef]
23. Türkyılmaz, A.; Senvar, O.; Ünal, İ.; Bulkan, S. A hybrid genetic algorithm based on a two-level hypervolume contribution measure selection strategy for bi-objective flexible job shop problem. *Comput. Oper. Res.* **2022**, *141*, 105694. [CrossRef]
24. Kong, X.; Yao, Y.; Yang, W.; Yang, Z.; Su, J. Solving the Flexible Job Shop Scheduling Problem Using a Discrete Improved Grey Wolf Optimization Algorithm. *Machines* **2022**, *10*, 1100. [CrossRef]
25. Stanković, A.; Petrović, G.; Ćojbašić, Ž.; Marković, D. An application of metaheuristic optimization algorithms for solving the flexible job-shop scheduling problem. *Oper. Res. Eng. Sci. Theory Appl.* **2020**, *3*, 13–28. [CrossRef]
26. Vujčić, V.; Ašić, M.; Miličić, N. *Matematičko Programiranje*; Mathematical Institute: Belgrade, Serbia, 1980.
27. Ortíz-Barrios, M.; Petrillo, A.; De Felice, F.; Jaramillo-Rueda, N.; Jiménez-Delgado, G.; Borrero-López, L. A dispatching-fuzzy AHP-TOPSIS model for scheduling flexible job-shop systems in industry 4.0 context. *Appl. Sci.* **2021**, *11*, 5107. [CrossRef]
28. Bagchi, T.P. *Multiobjective Scheduling by Genetic Algorithms*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1999.
29. Xie, N.; Chen, N. Flexible job shop scheduling problem with interval grey processing time. *Appl. Soft Comput.* **2018**, *70*, 513–524. [CrossRef]
30. Bierwirth, C.; Mattfeld, D.C.; Kopfer, H. On permutation representations for scheduling problems. In Proceedings of the PPSN IV: International Conference on Evolutionary Computation—The 4th International Conference on Parallel Problem Solving from Nature, Berlin, Germany, 22–26 September 1996; pp. 310–318.
31. Brandimarte, P. Routing and scheduling in a flexible job shop by tabu search. *Ann. Oper. Res.* **1993**, *41*, 157–183. [CrossRef]
32. Mastrolilli, M.; Gambardella, L.M. Effective neighbourhood functions for the flexible job shop problem. *J. Sched.* **2000**, *3*, 3–20. [CrossRef]
33. Ho, N.B.; Tay, J.C. GENACE: An efficient cultural algorithm for solving the flexible job-shop problem. In Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), Portland, OR, USA, 19–23 June 2004; Volume 2, pp. 1759–1766.
34. Pezzella, F.; Morganti, G.; Ciaschetti, G. A genetic algorithm for the flexible job-shop scheduling problem. *Comput. Oper. Res.* **2008**, *35*, 3202–3212. [CrossRef]

35. Zhang, G.; Gao, L.; Shi, Y. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Syst. Appl.* **2011**, *38*, 3563–3573. [CrossRef]
36. Palacios, J.J.; González, M.A.; Vela, C.R.; González-Rodríguez, I.; Puente, J. Genetic tabu search for the fuzzy flexible job shop problem. *Comput. Oper. Res.* **2015**, *54*, 74–89. [CrossRef]
37. Driss, I.; Mouss, K.N.; Laggoun, A. A new genetic algorithm for flexible job-shop scheduling problems. *J. Mech. Sci. Technol.* **2015**, *29*, 1273–1281. [CrossRef]
38. Buddala, R.; Mahapatra, S.S. An integrated approach for scheduling flexible job-shop using teaching–learning-based optimization method. *J. Ind. Eng. Int.* **2019**, *15*, 181–192. [CrossRef]
39. Werner, F. *Genetic Algorithms for Shop Scheduling Problems: A Survey*; Fakultät für Mathematik, Otto-von-Guericke-Universität: Magdeburg, Germany, 2011; Chapter 1; pp. 1–66.