Multi-objective meta-heuristics applied to flexible job shop scheduling problems

Joana Cunha Cabral
joana.c.cabral@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

December 2020

Abstract

With a tremendous increase in computational power observed in the past two decades and the improvements in data collection and analysis, industries turned to digitalization. Within Industry 4.0, scheduling using multi-objective functions is now an opportunity to provide solutions that address several conflicting objectives and provide optimized decision-making. To address this problem a Multi-Objective Artificial Bee Colony algorithm was implemented. The proposed algorithm combines a coding method that only generates feasible solutions with different techniques to generate an initial population and to generate new solutions. To measure the quality of solutions and the overall performance of the algorithm, two metrics were used, the Hypervolume and the Mean Ideal Distance. To obtain the ideal set of parameters that maximizes the ABC algorithm performance, a Bayesian optimization algorithm was used. Moreover, this algorithm was validated and its performance evaluated and tested for the Flexible Job Shop Scheduling Problem using two different types of benchmark datasets, with a total of 15 instances evaluated. The proposed approach shows a good performance, similar to that of the state of the art.

Keywords: Flexible Job Shop Scheduling, Multi-objective, Artificial Bee Colony Algorithm, Parameter Tuning, Bayesian Optimization

1. Introduction

Throughout the last few decades, the world has experienced an exponential growth of data gathering, storage and treatment. This growth has been correlated with the just as well exponential increase, in computational power. A continuous exploration and effort to maximize efficiency in all sorts of processes and industries has taken humanity from the steam engine powered factories that define the first industrial revolution, in the late 19^{th} century, passing through incremental technological progress such as mass production and automated systems applied to industries, to the digital era of today, defined by a new stage of systems with new capabilities to integrate and interact with information, cyber-physical systems, the Internet of Things (IoT) and cloud computing [28, 18].

Firstly formalised in Germany, the term Industry 4.0 describes the fourth industrial revolution, marked by fast changing technologies like the ones already stated above. Industry 4.0 introduces a transformation from machine-based to evidence-based decision making manufacturing systems where product-machine interaction is possible without human involvement, making an impact on time and costs, by reducing planning time, achieving better production control, lower energy consumption, improving quality, among others [1]. Smart factories exchange large amounts of data and information in real time between the different systems within a factory, allowing the different systems to perform better-informed decision in order to improve production status, such as throughput rates, energy consumption, inventory management and more. [5, 17]

Scheduling for production management has always been an important problem in operations research, and within the scope of Industry 4.0 it is equally important, since it is a tool to improve the overall factory efficiency and productivity. From the collection of data from customers demands and factory state, scheduling can affect planning and optimize decisions, according to the objectives of the factory.

Scheduling focuses on finding the optimal or near-optimal schedule, which mainly is the job sequence and the operation-to-machine assignment, while subject to constraints such as time. This can be either a simple task or a very complicated one, depending on the process domain, the constraints involved in the process and its performance indicators [16].

Focusing on static environment scheduling the nature of the different problems found can be divided into two categories [14], deterministic, where processing times and other parameters are set and known *a priori*, or stochastic,where parameters and processing times, modelled as random variables that follow a probability distribution. Among the deterministic problems a further classification is made, distinguishing single from multiple machine problems, the difference between one another focuses on the increased complexity of the problem. When only one machine processes the whole set of finite jobs, the problem is to find the sequence in which to process the jobs. Multiple machines directly relate to more complex arrangements, these configurations, by ascending order of complexity are parallel machine, flow shop and job shop.

For job shop problems [3], the operations of each job can be assigned to a machine for processing, meaning that a job can befall on a machine more than once. The classical job shop problems has a set of identical machines, and each operation can only be performed in a specific machine. The flexible job shop problem is defined as a set of multi-purpose machines, where each operation can be performed on more than one machine.

Flexible Job Shop scheduling problems using multi-objective optimization can be solved by using three types of meta-heuristic approaches. The first is by aggregating the multiple objectives into an weighted sum. The second type of approaches are non-Pareto approaches, where each objective is handled separately. And finally Pareto optimality approaches. Out of these three the most commonly used are the weighted sum and the Pareto optimality approaches. Some examples of meta-heuristics approaches using the weighted sum to solve the multi-objective flexible job shop scheduling problem are given in [24, 12, 23, 26]. Some examples of meta-heuristics techniques using the Pareto optimality type of approach are given by [21, 10, 4, 19].

2. Background

The Flexible Job Shop Scheduling Problem (FJSSP) is the generalization of the classical Job Shop Scheduling Problem (JSSP). This means that the main objective of the classical static JSSP is to find the schedule that processes n jobs on m machines following a given objective function.

To generalize the JSSP, FJSSP has an additional condition that imposes job variability and also creates the need for an operation sequence solution, not only the machine assignment. FJSSP allows operations to be executed by a set of machines instead of a single machine, providing a more realistic approach to the scheduling problems.

The FJSSP is stated as follows: a set of jobs, independent from each other $J = \{J_1, ..., J_n\}$; a set of independent machines, $M = \{M_1, ..., M_m\}$; a sequence of operations from each job J_i, $O_{ik} = \{O_{i1}, O_{i2}, ..., O_{ih}\}$; each operation O_{ik}, must be processed by a subset of machines such that $M_{ik} \subset M$, if $M_{ik} = M \ \forall i, k$ then the problem becomes a flexible job shop problem; each machine can execute exactly one operation at a time; each job and each operation must be processed exactly one time; each job has a specific linear precedence structure with the sequence of machines it visits; ef the precedent operation is still being processed, the remaining operations wait until the processing is completed; the processing time PT_{ikm} is given by the time it takes for operation O_{ik} of job J_i to be performed in machine M_m, without interruption; machines must always be available at the start of the scheduling horizon; the start time of every operation O_{ik} on machine m is defined as ST_{ikm}; the finishing time of each operation O_{ik} on machine m is defined as FT_{ikm}.

The FJSSP can be divided into two subproblems, a routing problem that selects the machine to process each operation, and the sequencing problem on all the selected machines to obtain a feasible schedule. It has also some additional constraints, it requires the operation to only be processed after all precedent operations have been processed, another constraint requires that one machine handles one and only one operation at a time and, finally, the precedence constraint for operations of the same job.

2.1. Objective Functions

The FJSSP intends to obtain a feasible schedule while quantifying performance measures. Three objective functions were taken into consideration, the makespan, the total machine workload and the maximum machine workload. The makespan, given by equation 1 is the maximum completion time of all jobs. The total workload, given by equation 2 is the amount of work it takes to finish all jobs. Lastly, the maximum workload, given by equation 3 is the workload of the machine responsible for the maximum workload. These objectives were chosen not only by their significance as to measure the efficiency of a schedule, because they are also in conflicting with each other.

$$C_{max} = \max\left(C_1, ..., C_j, ..., C_n\right), \quad (j \in J, k \in M, i \in O_{ji}) \tag{1}$$

$$W_t = \sum_{k=1}^{m}\sum_{j=1}^{n}\sum_{i=1}^{n_l}\left(p_{jik}x_{jik}\right), \quad (j \in J, k \in M, i \in O_{ji}) \tag{2}$$

$$W_m = \max_{1\leq k\leq m}\left(\sum_{j=1}^{n}\sum_{i=1}^{n_l}\left(p_{jik}x_{jik}\right)\right), \quad (j \in J, k \in K, i \in O_{ji}) \tag{3}$$

The Multi-Objective Flexible Job Shop Scheduling Problem (MOFJSSP) determines the machine allocation for each operation and the sequence in which those operations will be performed in a way that optimizes multiple objectives. The mathematical formulation is presented below, based on [20]. Equation 4 is used to describe a multi-objective problem without losing generality, where x corresponds to the decision vector in space, meaning the solutions found, and y represents the objective vector with l objectives.

$$\min y = f(x) = \left(f_1(x), ..., f_l(x)\right), \quad (x \in \Omega, y \in R^l, l > 1) \tag{4}$$

2.2. Representation

The representation of the problem can be done by both a disjunctive graph and a Gantt chart.

The disjunctive graph is described by equation 5.

$$G = (V, Con \cup Dis) \tag{5}$$

In the graph, operations are denoted by nodes $(i/j) \in V$, with V being the set of nodes that represent the operations. The precedence constraints of each operation in each job is assured by the conjunctive arcs Con. The disjunctive arcs represent a sequence of jobs done on the same machine, represented in Dis.

A Gantt chart is a commonly used bar chart that gives a visual representation of a feasible solution. It represents the allocation of the operations of each job and its given sequence. It also gives a representation of this allocation, the starting time of each operation and its respective completion time and the idle time on each machine.

3. Implementation

This section describes the implementation of the Multi-Objective Artificial Bee Colony algorithm implemented, to solve the Flexible Job Shop Scheduling Problem. This implementation was inspired by works [19, 11, 15, 22, 25]. It also describes the Bayesian Optimization algorithm used to tune the parameters of the algorithm.

3.1. Artificial Bee Colony with Non-Dominated Sorting Algorithm

The ABC algorithm was initially introduced by [8], and later enhanced in [9]. Essentially inspired by the intelligent foraging behaviour of a honeybee swarm, it contains three types of foraging bees, the employed bees, the onlooker bees and the scout bee.

A first step is taken by generating random food sources and associating the employed bees to those food sources. The employed bees then perform exploration, at the next step, the onlooker bees perform exploitation. Finally, the scout bees explore the exhausting food source. This process will iterate until it meets a stopping criterion. To translate this process, each food source represents a feasible solution, its nectar amount corresponds to the quantitative values of each objective function.

3.2. Procedure to implement the proposed ABC algorithm

To properly define the algorithm implemented, first it is necessary to make some definitions.

Pareto dominance, straightforwardly, if a solution S_1 dominates another solution S_2 ($S_1 \prec S_2$, which also denotes that S_2 is dominated by S_1), then equations 6 and 7 must always be true.

$$\forall i \in \{1, 2, 3, ..., l\} : f_i(S_1) \leq f_i(S_2) \tag{6}$$

$$\exists i \in \{1, 2, 3, ..., l\} : f_i(S_1) < f_i(S_2) \tag{7}$$

A Pareto non-dominated solution is a solution that is not dominated by any solution within the set of solutions found. It is considered an optimal solution and will be included in the Pareto front. The Pareto front is the selection of the non-dominated solutions found as well as their respective image from the objective space.

As the number of solutions is rather diverse, and may vary from iteration to iteration, it is necessary to fractionate the solutions following a dominance level and to sort them by that same level and an additional method, the crowding distance [27]. The process of sorting and fractionating the solutions according to a dominance level d_i, will for now on be defined as Non-dominated sorting. The first level will accommodate the non-dominated solutions from the current set, the second level will uphold the solutions that are dominated by the ones from the first level, this process repeats until there are no more solutions to allocate.

Crowding distance for each solution is calculated after the non-dominated sorting is complete, so that for each level, the solutions are sorted in an ascent order. For the first solution S_1 and the last solution S_{last} the value of the crowding distance is not calculated, it will be set as infinity. As for the remaining solutions, the crowding distance cd_i will be calculated according to equation 8,

$$cd_i = \begin{cases} \infty \quad \text{if} \quad i = 1 \quad \text{or} \quad i = last, \\ \sum_{l=1}^{L} \frac{f_l(s_{i+1}) - f_l(s_{i-1})}{f_l^{max} - f_l^{min}} \quad \text{otherwise,} \end{cases} \tag{8}$$

where cd_i is the crowding distance of solution s_i, $f_l(s_i)$ represents the value of the l_{th} objective function for solution s_i, f_l^{max} and f_l^{min} represent, respectively, the maximum and minimum values of that l_{th} objective function.

3.3. ABC algorithm steps

A general procedure of the ABC steps can be given by the following statements:

- Initialization step;
- Repeat
  - Employed Bees Step;
  - Onlooker Bees Step;
  - Scout Bees Step;
  - Update Solutions Set;
- Until it reaches maximum generation.

In the initialization step, the parameters are set, a population is initialized, creating the initial food sources with their respective nectar amounts, and then the best ones are selected to continue.

The next step, the employed bees step, these bees will probe to new food sources with the soul objective of acquiring more nectar to their neighbourhood. When a new food source is found, an evaluation is made between the amount of nectar from the old and the new food sources, and the best one remains. Finally, these bees share their information with the next foraging bees, by dancing within their colony.

The next foraging bees start a new step in the algorithm, the onlooker bees step, these foragers will now select the food sources to explore through a tournament selecting one food source per bee. After this selection is done, the onlookers start exploring the neighbourhood of their selected food source, similarly to the employed bees, but with some added depth to their foraging.

The final step is the scout bee step, where a random food source is chosen, and enhanced by this bee exploring around the neighbourhood. Then if this bee finds a new food source with a similar or better fitness value. It will replace the food source with the worst fitness value from within the current set of food sources by the newly found one.

These moves will be performed until they satisfy the termination criterion, which in this case, is when it reaches the maximum number of generations.

3.4. Solution Representation and Encoding

Solutions will be represented by two vectors, a machine assignment vector and an operation sequence vector, both vectors will represent a feasible solution. The first one, the machine assignment vector, allocates each operation from each job to a machine within its machine set. The second one, operation sequence vector, conceals the sequence of operations for each job.

The machine assignment vector depicts that each integer from the vector corresponds to a machine assigned uninterruptedly for each operation. The operation sequence vector works a little differently, since the operations from each job correspond to their job number, and the *ith* happening of a job denotes the *ith* operation of that same job.

3.5. Solution Decoding

To transform the machine assignment vector and the operation sequence vector that constitute the encoded representation of a feasible solution, into a schedule is the action of decoding the solution. It is necessary to establish at what time each operation starts and ends, and also the idle time between operations in the same machine. The precedence constraints must allows uphold, this means that the second operation of a job, can only start after the first one has finished, and if that operation can not start immediately after, due to its machine being currently processing another operation, from another job, it will create idle time on that machine. Considering the makespan objective, a left-shift scheme may be applied in order to shift operations as much to the left as possible, thus compacting the schedule.

3.6. Population Initialization

An initial population needs to guarantee diversity and quality of its solutions. Another important point being the ability to avoid falling into local optima. Like so it was used an hybrid way of generating an initial population set with $P = PS \times m \times n$ solutions, where PS is a predefined parameter related to the size of the initial number of solutions. To generate initial machine assignment vectors, three rules were used: a random rule, a local minimum processing time rule, and a global minimum processing time rule. The operation sequence initialization process also presents three rules: a random rule, a most time remaining rule, and a most number of operations remaining rule.

3.7. Employed Bee Step

3.7.1 Exploitation Search for Machine Assignment

For each solution from the employed bee step the exploitation search for the machine assignment proceeds to generate an integer, from 1 to the total number of operations. Then, selects the positions randomly from the machine assignment vector and for each chosen position, it replaces the current allocated machine from another within the candidate machine set, in order to produce a new solution. After generating the new solution, it evaluates and compares with respect to the old solution.

3.7.2 Exploitation Search for Operation Sequence

For each solution from the employed bee step the exploitation search for the operation sequence proceeds to select two jobs, randomly, then fills the positions of the first job chosen within the positions of the second job, from left to right, and repeats the process respectively, to generate a new solution. After generating the new solution, it evaluates and compares it with the old solution.

3.8. Onlooker Bee Step

3.8.1 Tournament Selection for the Onlooker bee step

Due to the difficulty of properly calculating the fitness value that is associated with the probability of an onlooker bee choosing a food source, a problem inherent to the multi-objective nature of the algorithm, it was implemented a tournament selection with size three. This method is described as first selecting three solutions randomly from the employed bee solutions, then sorting the solutions according to the Pareto level and crowding distance and choosing the best solution to be the food source of the onlooker bee.

3.8.2 Exploitation search in Onlooker Bee Step

With all the onlooker bees having a selected food source from the tournament selection described above, every onlooker bee will perform the techniques described for the employed bee step to generate new neighbouring solutions and the better solutions found will update the population to establish a new population P_t.

3.8.3 Crossover Operators

The crossover operators were designed to enhance the exploring capabilities of the onlooker bees, or in other words, to share information in order to find more promising food sources (better solutions). Since the solution is composed by the machine assignment vector and the operation sequence vector, different crossover operators, one for each of these parts, were designed.

Starting by first describing the crossover operators for machine assignment, two operators were used, the two-point crossover operator and an uniform crossover operator. For the operation sequence, the crossover operator implemented is a modified precedence operation crossover operator (MPOX) according to the precedence preserving order-based crossover (POX).

3.9. Local Search Based on the Critical Path

A local search method was implemented in the onlooker bees step to promote the intensification of the exploration capacity of the algorithm.

The graph G has two dummy nodes, node 0 and node $E = N + 1$, where N is total number of operations. These nodes will have no processing time, the starting time of node 0 will be 0, and the completion time of node E is the makespan of the schedule. The earliest starting time of operation O_{ij}, expressed by equation 9, will be designated as ES_{ij} and will be defined as the maximum time value between the earliest completion time in the operation that precedes operation j in the job i, represented as $PO^J_{ij} = O_{i,j-1}$ and the earliest completion time of the previously executed operation on the same machine k, represented by PO^M_{ij}. The earliest completion time of operation O_{ij}, expressed in equation 10, is the sum between the earliest starting time of that same operation and its processing time on machine k.

$$ES_{ij} = \max\left(EC(PO^J_{ij}), EC(PO^M_{ij})\right) \tag{9}$$

$$EC_{ij} = ES_{ij} + PT_{ijk} \tag{10}$$

To compute the earliest starting and completion times, the process will perform iteratively from node 0 until node E, in other words, from left to right. To compute the latest starting and completion times, the process will be done backwards, starting from node E, ending in node 0.

The latest starting time of operation O_{ij} without delaying the makespan, given by equation 11, designated as LS_{ij} is given by the subtraction between the latest completion time of that same operation and its processing time machine k. The latest completion time of operation O_{ij} without delaying the makespan , given by equation 12, designated as LC_{ij} is given by the minimum time value between the latest starting time of the operation that follows j within job i, designated as $FO^J_{ij} = O_{i,j+1}$, and the latest starting time of the operation that follows O_ij on machine k, designated as FO^M_{ij}.

$$LS_{ij} = LC_{ij} - PT_{ijk} \tag{11}$$

$$LC_{ij} = \min\left(LC(FO^J_{ij}), LC(FO^M_{ij})\right) \tag{12}$$

It is also important to note that for the dummy nodes these values will remained unaltered. Also, the makespan will be given by the latest completion time of the ending node E. The total slack of each operation translates into the time amount as to which an operation can be delayed without delaying the makespan, intuitively, if an operation has zero slack, it means that if that operation delays, then the makespan will delay as well, thus making it a critical operation, and is expressed in equation 13.

$$TS_{ij} = LS_{ij} - ES_{ij} \tag{13}$$

Once a critical path is found, the critical operations will be moved towards minimizing the objective functions. Referencing the theorem in [25], in the schedule represented by graph G, if a new schedule represented by graph G' obtained by moving an operation $O_{ij} \notin \chi(G)$ in G, then $C_{max}(G) \leq C_{max}(G')$. This means that minimizing the makespan, and consequently total and maximum workload can only be achieved through moving the critical operations exclusively.

An operation can only be moved and inserted into another position if it satisfies the following inequality:

$$\begin{aligned} &\max\left(EC(PO^{M}_{(G_i^-,v)}), EC(PO^{J}_{(G_i^-,co_i)})\right) \\ &+PT_{co_i,k} \leq \min\left(LS(FO^{M}_{(G_i^-,v)}), EC(FO^{J}_{(G_i^-,co_i)})\right) \end{aligned} \tag{14}$$

Where v is the operation in G^- that will succeed operation co_i in the newly appointed position. This relates to another theorem from [25] which states that a schedule G' is obtained by inserting co_i into a position located before operation v on machine M_k in G_i^- under the circumstance that satisfies 14. The positions that will be available to insert co_i into, before v, are appointed by Γ, which is the set of positions before all the operations of δ_k excluding Ψ_k and after all the operations from Ψ_k excluding δ_k. $\psi_k(G)$ consists of the possible machines in which each critical operation can be performed sorted to minimize total and maximum workload. Where δ_t and δ_m are used as metrics to consider respectively the total and maximum workload.

3.10. Recombination and Selection

A final method in the onlooker bee step that will decide the population to proceed to the scout bee step is the recombination and selection approach. The recombination and selection method combines the populations within the Onlooker Bee Phase, then sorts the solutions with non-dominated sorting. Using the best solutions to update the set of solutions.

3.11. Scout Bee Step

The scout bee step is used to randomly generate a new food source that replaces the worst solution found. Global exploration will be applied to the solution to intensify the population diversity and improve results.

3.12. Parameter Tuning

The algorithm implemented to solve the MOFJSSP needs to initialize promptly a set of key parameters. This set of 8 parameters is composed by the parameter related to the maximum generation, Gen_{max}, the parameter related to the initial population size PS, the probability of the machine assignment to be executed by rule 1, and respectively the rule 3, the probability of the operation sequence to be executed by rule 1, and respectively the rule 3, the probability for solutions to participate in the tournament selection and finally the probability to perform the two-point crossover.

Such a number of parameters to be set can have different methodologies to tune these parameters. Also, since these parameters will affect the performance of the algorithm implemented, it was implemented a Bayesian optimization algorithm for tuning the parameter set.

3.12.1 Bayesian Optimization

The Bayesian Optimization (BO) algorithm was chosen for being a state-of-art solution in circumstances where a finite set of parameters will directly reflect on the performance of an algorithm. The choice of these parameters can not be done intuitively, also, it is a computationally expensive system and through other methods, such as a grid search, would consume a tremendous amount of time and could leave out of testing lots of parameter set combinations.

The main advantage of choosing Bayesian optimization is that the iterative process of retrieving the best combination of parameters is done through an unbiased utility function, that will maximize the performance of the proposed algorithm when choosing an new set of parameters to test, resulting in a lot of computational time saved and reducing the number of tests to be done.

3.12.2 Performance Metrics

To establish a proper evaluation of the performance of the proposed algorithm and how the parameters influence it, two metrics that evaluate both solutions diversity and convergence were used. The metrics used were the Hypervolume and the Mean Ideal Distance.

4. Results

In a first instance the results of the setting of parameters are presented. Then, the results from the implemented algorithm were tested using two different Benchmark datasets, the Brandimarte dataset

[2] and the Kacem dataset, from [7, 6]. These results are compared with several state-of-art algorithms, combining both weighted sum and Pareto optimality approaches to solve the multi-objective optimization problem.

4.1. Parameter Tuning
The parameter tuning was done using a Bayesian optimization algorithm. The algorithm was implemented with the aid of the *Matlab* function *bayesopt*.

It uses a specific parameter, the exploration versus exploitation ratio from the acquisition function, which was set to $\epsilon = 0.5$, to establish equal importance on exploration and exploitation. Also, the acquisition function used was the expected improvement, μ_{EI}, and the maximum number of iterations defined was 30. The search for the optimal set of parameters was done on two metrics, the Mean Ideal Distance and the Hypervolume, for each metric and for each benchmarked instance, the algorithm was executed 5 times to ensure consistency within the obtained sets of parameters.

Table 1 summarizes the final parameters obtained, and table 2 the final parameters that change according to each specific benchmarked instance.

Parameter	Value
P_{as}	0.634
P_c	0.624
$p_{m,rule1}$	0.1790
$p_{m,rule3}$	0.7940
$p_{o,rule1}$	0.2430
$p_{o,rule3}$	0.8120

Table 1: Parameters used in the proposed algorithm, P_{as}, P_c, $p_{m,rule1}$,$p_{m,rule3}$, $p_{o,rule1}$ and $p_{o,rule3}$

4.2. Results comparison
The algorithms chosen to compare the proposed implementation with involve algorithms that both multi-criteria approaches for the multi-objective optimization and Pareto optimality approaches, to verify the superiority of results from the latter one with respect to the first type of approaches. And also compare the proposed algorithm with other Pareto optimality approaches.

4.2.1 Kacem dataset results comparison

The proposed algorithm was tested with the five instances from the Kacem dataset, using the parameters described in the previous section. The results were compared with five already existing algorithms, such as Approach by Localization (AL+CGA) [7], Particle Swarm Optimization

Inst.	PS	Init.Pop.	Gen.	Max.Gen.
mk01	1.458	88	2.073	125
mk02	1.945	117	3.630	218
mk03	0.988	119	3.099	372
mk04	1.230	148	1.424	171
mk05	0.999	60	2.306	138
mk06	1.423	213	1.756	263
mk07	1.030	103	3.399	340
mk08	0.874	175	3.709	742
mk09	0.830	249	1.422	426
mk10	0.865	260	1.684	505
kacem1	1.900	38	2.445	49
kacem2	1.365	88	1.555	100
kacem3	1.365	96	2.381	167
kacem4	1.547	155	1.730	173
kacem5	1.131	170	2.565	285

Table 2: Parameter values for the initial population of solutions and maximum generation.

Combined with Simulated Annealing (PSO+SE) [23], Particle Swarm Optimization combined with Tabu Search (PSO+TS) [26], Discrete Artificial Bee Colony (P-DABC) [13] and an Enhanced Pareto Artificial Bee Colony (EPABC) [19]. Amongst these algorithms, PSO+SE and PSO+TS, both use a fitness function that aggregates the objective functions into a single fitness function to be minimized, turning the problem into multi-criteria evaluation. AL+CGA proposes an approach that simultaneously minimizes the makespan and balances the workload of each machine. It is only on P-DABC and EPABC that a Pareto optimality approach is used.

Algorithms		kacem5		
		C_{max}	W_T	W_M
AL+CGA	s_1	23	95	11
	s_2	24	91	11
PSO+SA	s_1	12	91	11
PSO+TS	s_1	11	93	11
P-DABC	s_1	12	91	11
	s_2	11	93	11
EPABC	s_1	11	91	11
	s_2	11	93	10
Proposed MO-ABC	s_1	11	91	11
	s_2	12	91	11

Table 3: Results comparison on the instance kacem5 (15×10).

Table 3 presents the results obtained by these five algorithms and the proposed algorithm, for the instance *kacem5*. The proposed algorithm can obtain either more non-dominated solutions or better solutions when compared with the first four algorithms presented in table 3. As for the last two, the P-DABC and the EPABC, the results are quite similar either by number of non-dominated solutions or by the quality of solutions. Since both these two algorithms already outperform the previous four mentioned, it is reasonable to say that the proposed algorithm has an overall good quality performance for these five Kacem instances.

4.2.2 Brandimarte dataset results comparison

The second benchmarks dataset used to test the implemented algorithm is the Brandimarte dataset. The instances of this dataset are generated through a uniform distribution between given limits. The proposed algorithm is using the parameters described in the previous section. The results from five other algorithms found in literature were used to compare the results of the implemented algorithm. These algorithms are a search algorithm developed by Xing [24], a Multi-Objective Genetic Algorithm (MOGA)[21], an Hybrid Tabu Search Algorithm (HTSA) [12], a Shuffle Frog-Leaping Algorithm (HSFLA) [10] and a Particle Swarm Optimization (MOPSO) [4].

Instance	Objective	Xing	MOGA	HTSA	HSFLA	MOPSO	Proposed MO-ABC
MK01	C_{max}	42	40	**40**	40	**40**	40
	W_T	162	169	**167**	165	**167**	170
	W_M	42	36	**36**	37	**36**	36
MK02	C_{max}	28	**26**	**26**	26	**26**	**26**
	W_T	155	**151**	**151**	152	**151**	**151**
	W_M	28	**26**	**26**	26	**26**	**26**
MK03	C_{max}	204	204	**204**	**204**	**204**	204
	W_T	852	855	**852**	**852**	**852**	993
	W_M	204	199	**204**	**204**	**204**	204
MK04	C_{max}	68	66	61	62	61	**60**
	W_T	352	345	366	364	382	**390**
	W_M	67	63	61	61	60	**60**
MK05	C_{max}	177	**173**	172	173	**173**	173
	W_T	702	**683**	687	685	**683**	686
	W_M	177	**173**	172	173	**173**	173
MK06	C_{max}	75	**62**	65	64	**62**	63
	W_T	431	**424**	398	403	**424**	425
	W_M	67	**55**	62	55	**55**	57
MK07	C_{max}	150	**139**	140	141	**139**	141
	W_T	717	**693**	695	696	**693**	697
	W_M	150	**139**	140	141	**139**	141
MK08	C_{max}	523	**523**	523	523	523	523
	W_T	2524	**2524**	2524	2524	2524	2524
	W_M	523	**515**	523	523	523	523
MK09	C_{max}	311	311	**310**	**311**	310	312
	W_T	2374	2290	**2294**	**2275**	2514	2424
	W_M	299	299	**301**	**299**	299	299
MK10	C_{max}	227	**214**	214	215	**214**	218
	W_T	1989	**2082**	2053	1957	**2082**	2038
	W_M	221	**204**	210	198	**204**	204

Table 4: Results comparison on the ten Brandimarte instances.

On the algorithm presented by Xing, and on HTSA, the multi-objective problem turns into a mono-objective problem by evaluating the objective functions with a weighted sum. MOGA, HSFLA and MOPSO, all use a Pareto optimality approach to evaluate non-dominated solutions.

Table 4 it shows the results for non-dominated solutions with the minimum makespan obtained. Observing the solutions for *mk01* and comparing them to the ones obtained by the proposed algorithm, it can be seen that the only algorithms that outperform are the MOPSO, HTSA and MOGA, in the Total Workload objective ,W_T. For *mk02* and *mk08*, the results are similar to the ones obtained by MOGA, HTSA and MOPSO, in terms of performance.

In *mk04*, the result obtained dominates all other results in comparison. In *mk05*, the solutions for MOPSO, HTSA and HSFLA also dominate the solution obtained by the proposed algorithm. In *mk03* it does not out perform the presented algorithms, as the total workload value is higher. In *mk09* it does not out perform any of the other algorithms. On *mk06* the solutions by MOGA and MOPSO dominate the solution obtained with the proposed algorithm. For *mk07* and *mk10* the only solution that the proposed algorithm dominates is the one proposed by Xing. By observing table 4 induces that the results from the proposed algorithm, although they are not best for all instances, do not disperse, thus it is noticeable that this algorithm provides reasonably good results.

5. Conclusions

The objective of this thesis was to develop tools able to create a schedule for a multi-objective optimization problem. To achieve this, an Artificial Bee Colony algorithm was implemented .

To accomplish the proposed objective, the implementation of the Artificial Bee Colony algorithm was done using *Matlab*. The presented algorithm makes use of an aggregation of techniques to generate a good quality initial population, a coding scheme that generates only feasible solutions and a non-dominated sorting algorithm to sort and maintain the fittest individuals. To enhance the search capabilities of the algorithm, a local search based on critical path was implemented.

To test the proposed algorithm, a Bayesian optimization algorithm was first implemented to find the set of parameters that maximizes the performance of the algorithm according to metrics that evaluate both convergence and diversity within the solutions found.

After testing the algorithm using two benchmark datasets and comparing it with several different algorithms, the implementation and quality of the solutions obtained were both verified. Although the solution found by the proposed algorithm showed not to be always the best solution for the instances from the Brandimarte dataset, one instance in particular outperformed other algorithms, the *mk03*. As for the other instances of this particular dataset when the solution presented by the proposed algorithm was not the best, the difference between the solution found and, the others presented by the algorithms in comparison, is relatively small, thus making them good quality solutions. Differently, in the instances from the Kacem dataset, the proposed algorithm obtained high quality solutions. For all five instances, the proposed algorithm found a good diversity of non-dominated solutions, when compared to the other algorithms from the state-of-art the solutions found were of considerably good quality.

References

[1] A. Azevedo and A. Almeida. Factory templates for digital factories framework. *Robotics and Computer-Integrated Manufacturing*, 27(4):755–771, 2011.

[2] P. Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3):157–183, 1993.

[3] K. Genova, L. Kirilov, and V. Guliashki. A survey of solving approaches for multiple objective flexible job shop scheduling problems. *Cybernetics and Information Technologies*, 15(2):3–22, 2015.

[4] S. Huang, N. Tian, Y. Wang, and Z. Ji. Multi-objective flexible job-shop scheduling problem using modified discrete particle swarm optimization. *SpringerPlus*, 5(1), 2016.

[5] F. Jovane and E. Westkämper. *The ManuFuture Road*. 2009.

[6] I. Kacem, S. Hammadi, and P. Borne. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 32(1):1–13, 2002.

[7] I. Kacem, S. Hammadi, and P. Borne. Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation*, 60(3-5):245–276, 2002.

[8] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University. Technical Report TR06, 2005.

[9] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3):459–471, 2007.

[10] J. Li, Q. Pan, and S. Xie. An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems. *Applied Mathematics and Computation*, 218(18):9353–9371, 2012.

[11] J. Q. Li, Q. K. Pan, and K. Z. Gao. Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. *International Journal of Advanced Manufacturing Technology*, 55(9-12):1159–1169, 2011.

[12] J. Q. Li, Q. K. Pan, and Y. C. Liang. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering*, 59(4):647–662, 2010.

[13] J. Q. Li, Q. K. Pan, and M. F. Tasgetiren. A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. *Applied Mathematical Modelling*, 38(3):1111–1132, 2014.

[14] A. Nagar, J. Haddock, and S. Heragu. Multiple and bicriteria scheduling: A literature survey. *European Journal of Operational Research*, 81(1):88–104, 1995.

[15] F. Pezzella, G. Morganti, and G. Ciaschetti. A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers and Operations Research*, 35(10):3202–3212, 2008.

[16] M. L. Pinedo. *Planning and Scheduling in Manufacturing and Services*. 2006.

[17] E. H. D. Ribeiro da Silva, A. C. Shinohara, E. Pinheiro de Lima, J. Angelis, and C. G. Machado. Reviewing digital manufacturing concept in the Industry 4.0 paradigm. *Procedia CIRP*, 81:240–245, 2019.

[18] M. Russmann, M. Lorenz, P. Gerbert, M. Waldner, J. Justus, P. Engel, and M. Harnisch. Industry 4.0: World Economic Forum. *The Boston Consulting Group*, pages 1–20, 2015.

[19] L. Wang, G. Zhou, Y. Xu, and M. Liu. An enhanced Pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling. *International Journal of*

Advanced Manufacturing Technology, 60(9-12):1111–1123, 2012.

[20] L. Wang, G. Zhou, Y. Xu, S. Wang, and M. Liu. An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 60(1-4):303–315, 2012.

[21] X. Wang, L. Gao, C. Zhang, and X. Shao. A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 51(5-8):757–767, 2010.

[22] M. Watanabe, K. Ida, and M. Gen. A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. In *Computers and Industrial Engineering*, volume 48, pages 743–752, 2005.

[23] W. Xia and Z. Wu. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering*, 48(2):409–425, 2005.

[24] L. N. Xing, Y. W. Chen, and K. W. Yang. An efficient search method for multi-objective flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, 20(3):283–293, 2009.

[25] Y. Yuan and H. Xu. Multiobjective flexible job shop scheduling using memetic algorithms. *IEEE Transactions on Automation Science and Engineering*, 12(1):336–353, 2015.

[26] G. Zhang, X. Shao, P. Li, and L. Gao. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers and Industrial Engineering*, 56(4):1309–1318, 2009.

[27] Y. Zhang, D. W. Gong, X. Y. Sun, and Y. N. Guo. A PSO-based multi-objective multi-label feature selection method in classification. *Scientific Reports*, 7(1):1–12, 2017.

[28] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman. Intelligent Manufacturing in the Context of Industry 4.0: A Review. *Engineering*, 3(5):616–630, 2017.