A memetic algorithm for minimizing the makespan in the Job Shop Scheduling problem

Un algoritmo memético para minimizar el makespan en el problema del Job Shop Scheduling

Um algoritmo memético para minimizar o makespan no problema do Job Shop Scheduling

Henry Lamos-Díaz*
Karin Aguilar-Imitola**
Yuleiny Tatiana Pérez-Díaz***
Silvia Galván-Núñez****

Fecha de recepción: 30 de abril de 2015
Fecha de aprobación: 26 de agosto de 2016

Abstract

The Job Shop Scheduling Problem (JSP) is a combinatorial optimization problem cataloged as type NP-Hard. To solve this problem, several heuristics and metaheuristics have been used. In order to minimize the makespan, we propose a Memetic Algorithm (MA), which combines the exploration of the search space by a Genetic Algorithm (GA), and the exploitation of the solutions using a local search based on the neighborhood structure of Nowicki and Smutnicki. The genetic strategy uses an operation-based representation that allows generating feasible schedules, and a selection probability of the best individuals that are crossed using the JOX operator. The results of the implementation show that the algorithm is competitive with other approaches proposed in the literature.

Keywords: Job Shop Schedule; local search; memetic algorithm; metaheuristics.

* Ph. D. Universidad Industrial de Santander (Bucaramanga-Santander, Colombia). hlamos@uis.edu.co.
** M. Sc. Universidad Industrial de Santander (Bucaramanga-Santander, Colombia). ka-rin.aguilar@correo.uis.edu.co.
*** Esp. Universidad Industrial de Santander (Bucaramanga-Santander, Colombia). yuleiny.perez@correo.uis.edu.co.
**** M. Sc. Universidad Industrial de Santander (Bucaramanga-Santander, Colombia). silgalnu@udel.edu.

Resumen

El *Job Shop Scheduling Problem* (JSP) es un problema de optimización combinatoria catalogado de tipo NP-Hard. Para dar solución a este problema han sido utilizados diversos métodos heurísticos y metaheurísticos. Con el objetivo de minimizar el makespan se propone un algoritmo memético (MA) que combina la exploración del espacio de búsqueda mediante un algoritmo genético (GA) y la explotación de las soluciones, usando una búsqueda local basada en la estructura de vecindario de Nowicki y Smutnicki. La estrategia genética usa una representación basada en operaciones que le permite generar programas factibles y una probabilidad de selección de los mejores individuos que son cruzados usando el operador JOX. Los resultados obtenidos en la ejecución demuestran que el algoritmo es competitivo frente a otros enfoques propuestos en la literatura.

Palabras clave: algoritmo memético; búsqueda local; Job Shop Schedule; metaheurísticas.

Resumo

O Job Shop Scheduling Problem (JSP) é um problema de otimização combinatória catalogado de tipo NP-Hard. Para dar solução a este problema têm sido utilizados diversos métodos heurísticos e metaheurísticos. Com o objetivo de minimizar o makespan propõe-se um algoritmo memético (MA) que combina a exploração do espaço de procura mediante um algoritmo genético (GA) e a exploração das soluções, usando uma busca local baseada na estrutura de vizinhança de Nowicki e Smutnicki. A estratégia genética usa uma representação baseada em operações que permite gerar programas factíveis e uma probabilidade de seleção dos melhores indivíduos que são cruzados usando o operador JOX. Os resultados obtidos na execução demonstram que o algoritmo é competitivo frente a outros enfoques propostos na literatura.

Palavras chave: algoritmo memético; busca local; Job Shop Schedule; metaheurísticas.

Cómo citar este artículo:
H. Lamos-Díaz, K. Aguilar-Imitola, Y. T. Pérez-Díaz, and S. Galván-Núñez, "A memetic algorithm for minimizing the makespan in the Job Shop Scheduling problem," *Rev. Fac. Ing.*, vol. 26 (44), pp. 113-123, Ene. 2017.

I. Introduction

The Job Shop Scheduling Problem (JSP) is one of the most studied scheduling problems in the literature [1]. Like other scheduling problems, it is classified as a problem of combinatorial nature since it requires to develop a configuration for programming a set of jobs in a set of machines, where each job has a series of operations that must be processed in a defined sequence, and in an established processing time. In most of the job shop scheduling cases, it is desirable to find the best configuration to minimize the makespan (total time where all jobs have been executed). Other goals directly related to timing are the minimization of tardiness, delay, and total flow. The JSP is considered an NP-Hard problem [2]; hence, it is computationally difficult to find an optimal solution in a reasonable time since the search space grows exponentially as the problem entries increase.

Since the early 1950's, numerous researches have focused on solving the JSP. In 1956, Jackson [3] proposed a new approach by generalizing the Johnson's Flow Shop algorithm [4]. Akers and Friedman [5] employed an algebraic approach to represent the processing sequences. Subsequently, Roy and Sussmann [6] proposed a representation with the disjunctive graph; and finally, Balas [7] applied an enumerative approach based on this graph. Among the different approaches for solving the JSP, it is common to find exact methods that aim to find optimal solutions at a high computational cost; however, they turn out to be efficient only for small applications. Applegate and Cook [8], and Brucker *et al*. [9] solved the Ft10 Benchmark problem, and tried with applications up to 30 jobs and 10 machines for the JSP, using the Branch and Bound method; other applications of this method are covered in [10, 11]. For some larger applications, however, some approximations such as heuristic algorithms are required. Some of them include priority dispatching rules [12, 13], and the mobile bottle neck algorithm [14, 15]. Recent researches have mostly focused on more advanced heuristic algorithms, better known as "metaheuristics", which propose several approaches like the tabu search [16-19], simulated annealing [20-22], ant colony optimization [23-26], particle swarm optimization [27-30], neuronal network [31, 32], and genetic algorithms (GA). In particular, the GA are based on Darwin's evolutionary theory, and they have been employed to provide successful solutions to various combinatorial problems (JSP included [33-36]) since they allow exploring in an efficient way the solution space; nevertheless, they may converge prematurely. That is why recent researches have aimed to combine the GA with other techniques that ameliorate its efficiency by developing hybrid methods as the Memetic Algorithm (MA).

The MA was first introduced by Moscato and Norman [37]. The basis of the MA lays on individual enhancements of the solutions of agents that interrelate one to another in a process that contains stages of cooperation and population competition. The MA has been successfully used in different areas and combinatorial problems, such as the knapsack problem [38-40], routing problems [41-43], quadratic assignment [44-45], and spanning tree [32, 46], among others. In order to give a solution to the JSP, some studies [1, 47-50] have proposed a MA, where the global search given by the GA is combined with a neighborhood structure based on Nowicki and Smutnicki [51], which allows the leading of the local search and the efficient exploitation of the solution space with the generation of three adjacent solutions for each initial solution; all of this with the final goal of minimizing the makespan. Here, we review the literature related to solve sequence problems with evolutionary "metaheuristics" algorithms, taking into account the JSP and the MA. In addition, we designed an algorithm to minimize the makespan, and studied the representation of the solution with chromosome, based on operations, and various ways of starting and building the solutions. Likewise, we fixed and established the genetic operators, as well as the searching algorithm, and designed an experiment to measure the effect of the algorithm parameters on the outputs. Finally, we evaluated the algorithm efficiency with reference problems from the OR-Library.

This paper is organized as follows: section 2 describes the JSP; section 3 presents the MA framework; section 4 analyzes the effect of the algorithm parameters on the makespan by experimentation, and evaluates the MA with benchmarking problems; and section 5 summarizes the conclusions.

II. Job shop scheduling problem definition

The JSP consists in a set of jobs that must be processed in a limited set of *M* machines. In the JSP, the following restrictions and assumptions are considered:

- Each machine is able to process one job at a time.
- Each job can be processed by only one machine at a time.
- The sequence of machines that a job visits is completely fixed, and has a linear precedence structure.
- All jobs must be processed for each machine only once, and there is a maximum of operations per job.
- Machines are always available and never interrupted.
- The processing time of all operations is known.

The JSP mathematical model is presented in equations (1) through (7). Equation (1) represents the makespan minimization function; meanwhile, equations (2-7) represent the problem constraints.

t_{ij}: Beginning time of each operation

J: Set of n jobs to be processed

M: Set of m machines

Oij: Job operation that must be processed by a machine in an interrupted time

$$CmaxMinimization = max(tij + \tau ij): \forall J\, i \in J, Mj \in M \quad (1)$$

Subject to:

- Starting time

$$t_{ij} \geq 0\ \{i, p\} \in J\ \{j, h\} \in M \quad (2)$$

- Precedence

$$t_{ij} - t_{ih} \geq \tau_{ih} \quad \text{If} \quad O_{ih}\ precedes\ O_{ij} \quad (3)$$

- Disjunctive

$$t_{pj} - t_{ij} + K(1 - y_{ipj}) \geq \tau_{ij}\ y_{ipj} = 1, if\ O_{ij}\ precedes\ O_{pj} \quad (4)$$

$$t_{ij} - t_{pj} + K(y_{ipj}) \geq \tau p_j \quad y_{ipj} = 0, In\ other\ case \quad (5)$$

$$\text{Where } K > \left(\sum_{i=1}^{n}\sum_{j=1}^{m}\tau_{ij} - \min(\tau_{ij})\right) \quad (6)$$

$$y_{ipj} = \{0,1\}, Binary\ variable \quad (7)$$

III. Memetic algorithm for the job scheduling problem

The general procedure for the development of the memetic algorithm (MA) is shown in the following pseudo code.

The memetic process begins with the generation of the initial population. For that purpose, the population size parameters are established as well as the number of generations, selection probability, and mutation. The initial population is a set of solutions presented as chromosomes that are decoded afterwards. A searching method is then applied to these solutions in order to generate adjacent (or neighbor) solutions. Subsequently, each individual from the initial population, as well as the chosen neighbors, are decoded and evaluated to obtain the makespan value. If the algorithm's termination criterion has been accomplished, the program stops and shows the best-found solution; if not, a new population is generated using the genetic operators' application (selection, crossing, and mutation) over the population. These new individuals are known as a generation, and the previous procedure is then repeated until the breakdown criterion is accomplished. The following sections describe each stage of the algorithm.

A. Chromosome coding and decoding

In the JSP solution using MAs, the coding is given by a chromosome for each individual that represents the programming or the schedule. The main purpose is to generate feasible schedules, avoid any reparation to individuals, and easily apply the genetic operators. In this study, we used the representation based on operations, which allows to code the schedule as an operation sequence.

For a given problem of jobs and machines, a chromosome is a permutation with job repetitions and genes. The operations are represented by the number of each job, and they appear several times in the chromosome. Each appearance of a same number indicates an operation within the programming

sequence for the given job. This type of representation always generates feasible programs. For example, a chromosome [2 3 2 1 1 3 2 3 1] is given, where 1, 2, and 3 correspond to jobs. Each job is repeated as many times as the number of machines the problem possesses, and they represent the operations for each job. In figure 1, by reading the chromosome from left to right, the first gene (2) represents the first operation of the second job to be processed first on the corresponding machine. The second gene (3) represents the first operation of the third job. Therefore, the third gene (2) represents the second operation of the second job because it is the second time this number appears in the chromosome. In this way, the chromosome [2 3 2 1 1 3 2 3 1] is a sequence (operation of job) that can be translated as an operation vector $O_{21}\ O_{31}\ O_{22}\ O_{11}\ O_{12}\ O_{32}\ O_{23}\ O_{33}\ O_{13}$.

The Gantt's diagram, which is used for schedule decoding, results from reading the genes in a chromosome from left to right, and programming operations considering the corresponding processing times and machine sequences. Each operation is assigned in the minimum possible time without violating constraints.

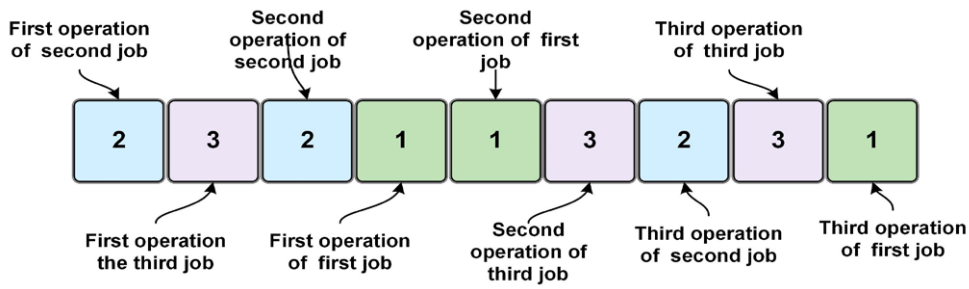

Fig. 1. Example of chromosome interpretation**.**

B. Initial population

Different methods have been proposed to generate the initial population: heuristics, as the mobile bottle neck [10], priority dispatching rules [9], or random approaches. This study uses the random generation of individuals because any job permutation will be a feasible schedule, due to the chosen representation type. An individual is a vector that contains positions, and is generated by randomly assigning the number of each job. This is repeated until achieving the population size.

C. Genetic operators

Genetic operators allow the population to evolve by generating new individuals with the main purpose of ameliorating the offspring, as well as exploring new searching space solutions. The used operators are selection, crossover and mutation.

The selection scheme combines both tourney and roulette selection. Basically, it consists on choosing the parents as a probability of the best population individuals, but at the same time allowing some not-so-good individuals to be part of the selected group. The chosen chromosomes form half of the previous population, and they are called parents.

In Crossover, first, a pair of parents (chromosomes) are randomly selected from the mating pool, and then new offspring is created by exchanging the parents' genetic information. Two parent strings are denoted as P1 and P2, and two children strings are denoted as H1 and H2. The procedure consists on randomly choosing one job, any gene in the parent P1, which is then retained in the same position in child H1; subsequently, the remaining empty positions in child H1 are filled with the genes of parent P2 that are different from the chosen job. The second child is generated in the same way, but exchanging the parent's role (Fig. 2). This operator is known as JOX (Job Order Crossover) [52].

Mutation introduces some extra variability into the population, and prevents its premature convergence. Each child generated with the crossing operator has a mutation probability associated (random number between 0 and 1), which is compared to a designated parameter at the beginning of the algorithm. If the mutation probability of a child is less or equal than the probability parameter, the mutation is then executed; otherwise, the individual is kept with no changes. The mutation procedure exchanges the position of two

genes within the chromosome. For instance, in figure 3, the chosen numbers are 2 and 6, which means, they are the positions to be exchanged. The first operation for job 3 is found in position 2, which will be moved to position 6, and the last operation for job 1 that is on position 6 will be moved to position 2. In this way, a child is altered, and the schedule is modified.

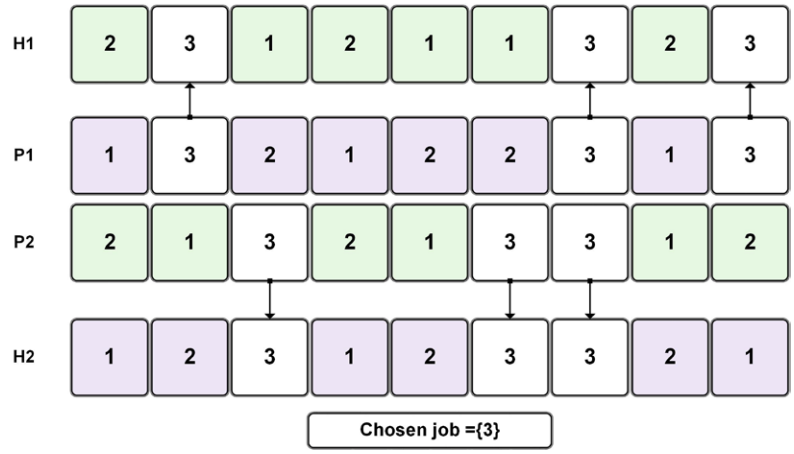

Fig. 2. Example of the JOX crossover procedure to generate children.

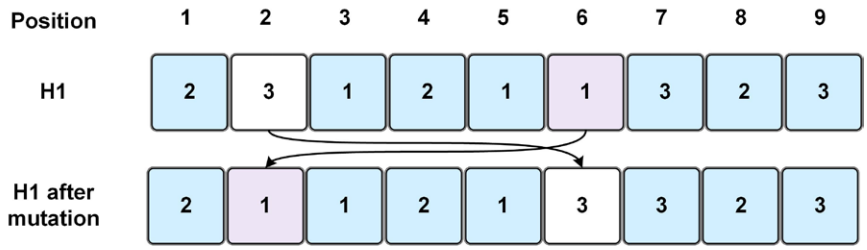

Fig. 3. Example of the mutation operator.

D. Local searching procedure

The memetic algorithm (MA) modifies the neighborhood strategy proposed by Nowicki and Smutnicki [51]. Nowicki and Smutnicki's proposal is based on the critical route, also known as N_5. However, in the MA, the decision on the schedule's critical route, in which local searching will be applied, is not random; on the contrary, the decision is made choosing the longest known route for that programming. In other words, the chosen route is the one with the largest number of critical operations. The main purpose of applying this neighborhood structure is to reduce the amount of movements by omitting some movements (proposed by Van Laarhoven *et al.* [19]) that will not immediately improve the makespan. The operation exchanges are only made in the block borders. The procedure consists on exchanging the last two operations of the first critical block, and the first two operations of the last block in the selected critical route. To perform this operation, some of the chosen blocks must contain at least two operations.

Using this local searching method, three neighbors (, and) are obtained from the chosen route: one neighbor per each executed exchange, and the third one from the two simultaneous exchanges. If the chosen critical route contains only one block, it is possible to generate only one neighbor. Once the three neighbors are found, the operations programming order is modified in the schedule. Afterwards, a decoding is carried out on Gantt's diagram, where the neighbor with the best fitness function value (makespan) will continue into the procedure performed by the genetic operators. If only one neighbor is generated, it will immediately advance to the selection process.

The local searching pseudo code is presented as follows:

Local searching pseudo code
Input: *Longest critical route* $S_0 \; \forall \; n \; \in \; P_i$
Begin
For $i{:}1{:}n$ ***do***
Matrix B = First select (B_1) *and lastly* (B_k) *critical block in* S_0
Generate several neighbors
If *B has two blocks*
$N'_i = exchange\ first\ two\ operations\ in\ B_k$ *(First neighbor)*
$N''_i = exchange\ last\ two\ operations\ in\ B_1$ *(Second neighbor)*
$N^*_i = N'_i + N''_i$ *(Third neighbor)*
Else
$N'_i = exchange\ last\ two\ operations\ in\ B_1$
End if
Decode and evaluate the neighbors
N_i *is the best neighbor solution*
End For
$Q''_i = N_i + \cdots + N_{i+n}$
Output: *local solution matrix* Q''_i

IV. Computational Results

A fractionated 2^{k-1} experiment was designed to select the best parameters to execute the numerical experiments. Population size, number of generations, selection percentage, and mutation probability were considered factors that may affect the computational performance. The experiments were performed using benchmark problems from Fisher *et al.*, better known as FT, and Lawrence's, better known as LA, all of them taken from the OR-Library [52-53]. For the first experiment, low and medium complexity problems (Ft06, La01, La06, La09, La11, La14, and La17) were used; meanwhile, for the second experiment, problems of high complexity (Ft10, Ft20, La16, La21, La 24, La25, and La 38) were used. For low and medium complexity instances, 8 replicates were made per each combination or treatment, while 5 were made for high complexity instances. The low and high values of each factor used in the low and medium complexity instances were population size {10, 30}, number of generations {20, 30}, selection probability {0.7, 0.9}, and mutation probability {0.05, 0.1}. The values for high complexity instances were population size {80, 150}, number of generations {100, 170}, selection probability {0.8, 0.9}, and mutation probability {0.05, 0.1}. The numerical experiments were developed in the Matlab® programming environment, in a PC with an Intel Core i7 processor, 3.40GHz, and 8GB memory. According to the analysis of variance (ANOVA), population size, number of generations, and selection percentage affect the solution; therefore, the ANOVA results allowed to adjust these factors, achieving a positive impact in most of the outputs. We found that mutation probability has a low significance in the evolutionary process, confirming the literature findings regarding this operator [54].

Table 1 summarizes the results of the experiments. The columns include the name of each test benchmark problem (Problem), the size of the problem (Size), the population size (POP), the number of generations (GEN), the selection probability (SP), the mutation probability (MP), the value of the best-known solution for each problem (BKS), the value of the best solution found by using the proposed MA (MA), and the solution obtained from other evolutionary approaches made by Hasan *et al.* [55], Gao *et al.* [52], and Wang *et al.* [36]. The solutions marked with an asterisk (*) are optimal. The proposed MA found the best-known solution in 19 instances (63.3 %) among the 30 evaluated problems. Also, the propose MA found 10 solutions worse than other approaches; however, the gap respect to BKS is less than 2 % in 87 % of the evaluated problems (Table 1).

Table 1
Comparisons of the results between proposed MA, BKS and other memetic approaches

Problem	**Size**	**PS**	**GEN**	**PS**	**PM**	**BKS**	**MA**	**MA(GR-RS)** [34]	**MA** [54]	**AMGA** [35]
Ft06	6x6	10	10	0.7	0.1	55	55*	55	55	-
Ft10	10x10	500	100	0.9	0.1	930	937	930	930	-
Ft20	20x5	500	80	0.9	0.1	1165	1182	1165	1165	-
La01	10x5	20	20	0.9	0.1	666	666*	666	666	666
La02	10x5	50	50	0.9	0.1	655	655*	655	655	655
La03	10x5	200	100	0.9	0.1	597	597*	597	597	597
La04	10x5	300	150	0.9	0.1	590	590*	590	590	590
La05	10x5	5	1	0.7	0.05	593	593*	593	593	593
La06	15x5	10	1	0.9	0.1	926	926*	926	926	926
La07	15x5	20	20	0.9	0.1	890	890*	890	890	890
La08	15x5	30	20	0.9	0.1	863	863*	863	863	863
La09	15x5	10	10	0.9	0.05	951	951*	951	951	951
La10	15x5	10	2	0.7	0.1	958	958*	958	958	958
La11	20x5	10	5	0.7	0.05	1222	1222*	1222	1222	1222
La12	20x5	10	5	0.9	0.1	1039	1039*	1039	1039	1039
La13	20x5	12	6	0.9	0.1	1150	1150*	1150	1150	1150
La14	20x5	10	5	0.7	0.05	1292	1292*	1292	1292	1292
La15	20x5	50	50	0.9	0.1	1207	1207*	1207	1207	1207
La16	10x10	PS0	170	0.8	0.1	945	946	945	945	945
La17	10x10	40	50	0.9	0.1	784	784*	784	784	784
La18	10x10	150	170	0.9	0.1	848	858	848	848	848
La21	15x10	150	170	0.9	0.1	1046	1081	1079	1055	1046
La22	15x10	300	80	0.9	0.1	927	954	960	927	-
La23	15x10	350	100	0.9	0.1	1032	1032*	1032	1032	-
La24	15x10	150	100	0.8	0.1	935	976	959	940	-
La25	15x10	300	100	0.9	0.1	977	999	991	984	-
La31	30x10	250	60	0.9	0.1	1784	1784*	1784	1784	1784
La32	30x10	250	100	0.9	0.1	1850	1868	1850	1850	-
La35	30x10	300	70	0.9	0.1	1888	1901	1888	1888	-
La38	15x10	250	100	0.8	0.1	1196	1258	1266	1216	-

V. Conclusions

This study developed a memetic algorithm (MA) to solve the Job Shop Scheduling Problem by using the JOX method in the crossing operator, and a new neighborhood structure for the local searching procedure. This new neighborhood structure generated three new solutions by exchanging operations in the first and last block of the critical path. The MA takes advantage of the genetic strategy that explores the solution space, and the local searching method, which intensifies the searching by exploiting every found solution to avoid being trapped in a local optimum. The random generation of initial individuals was successful to diversify the population, and the decoding was the process with the longest computational time. From the ANOVA, we can conclude that the population size is associated with the exploration capacity of the memetic algorithm. The MA was evaluated in 30 benchmark problems, and compared with the solutions obtained in the literature; the computational results obtained from the experiments demonstrated the efficiency of the proposed memetic algorithm.

Author's contributions

All authors contributed extensively to the work presented in this paper.

H. Lamos Díaz conceived, designed, and supervised the research, and critically reviewed the paper. K. Aguilar Imitola and Y. Pérez Díaz designed and programmed the algorithm, conducted the computational experiments, analysis and data interpretation, and drafted the manuscript. S. Galvan Núñez supervised the algorithm design and the drafting of the manuscript, and participated in the critical revision of paper.

References

[1] M. Frutos and F. Tohmé, "A Multi-objective Memetic Algorithm for the Job-Shop Scheduling Problem," *Oper. Res.*, vol. 13 (2), pp. 233–250, Jul. 2013. DOI: http://doi.org/10.1007/s12351-012-0125-y.

[2] M. R. Garey and D. S. Johnson, *Computer and Intractability: A Guide to the Theory of NP-Completeness*. 1979.

[3] J. R. Jackson, "Scheduling a production to minimize maximum tardiness," *Res. Report. Manag. Sci. Res. Proj.*, vol. 43, 1955.

[4] S. M. Johnson, "Optimal two and three stage production schedules with setup times included," *Naval. Res. Logist. Quart.*, vol. 1 (1), pp. 61–68, Mar. 1954. DOI: http://doi.org/10.1002/nav.3800010110.

[5] S. B. Akers and J. Friedman, "A Non-Numerical Approach to Production Scheduling Problems," *Oper. Res.*, vol. 3 (4), pp. 429–442, 1955. DOI: http://doi.org/10.1287/opre.3.4.429.

[6] B. Roy and B. Sussmann, "Les problems d'ordonnancement avec constraintes disjonctives," *Note D.S. 9, SEMA*, 1964.

[7] E. Balas, "Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm," *Oper. Res.*, vol. 17 (6), pp. 941–957, Dec. 1969. DOI: http://doi.org/10.1287/opre.17.6.941.

[8] D. Applegate and W. Cook, "A Computational Study of the Job Shop Scheduling Problem," *ORSA J. Comput.*, vol. 3 (2), pp. 149–156, May. 1991. DOI: http://doi.org/10.1287/ijoc.3.2.149.

[9] P. Brucker, B. Jurisch, and B. Sievers, "A branch and bound algorithm for the job-shop scheduling problem." *Discr. Appl. Math.*, vol. 49, pp. 107–127, 1994. DOI: http://doi.org/10.1016/0166-218X(94)90204-6.

[10] C. Mencía, M. R. Sierra, and R. Varela, "Depth-first heuristic search for the job shop scheduling problem," *Ann. Oper. Res.*, vol. 206 (1), pp. 265–296, Jul. 2013. DOI: http://doi.org/10.1007/s10479-012-1296-x.

[11] Y. Tan and Z. Jiang, "A branch and bound algorithm and iterative reordering strategies for inserting additional trains in real time: A case study in Germany." *Math. Probl. Eng.*, vol. 2015, pp. 1–12, 2015. DOI: http://doi.org/10.1155/2015/289072.

[12] K. Hadavi, Y.-W. Hou, W.-L. Hsu, D. Levy, and M. Pinedo, "Dispatching Issues in Job Shop Scheduling," in *New Directions for Operations Research in Manufacturing - Chapter 14*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 234–245. DOI: http://doi.org/10.1007/978-3-642-77537-6_14.

[13] S. Yokoyama, H. Iizuka, and M. Yamamoto, "Priority rule-based reconstruction for total weighted tardiness minimization of job-shop scheduling problem," *J. Adv. Mech. Des. Syst. Manuf.*, vol. 8 (5), pp. 1–6, 2014. DOI: http://doi.org/10.1299/jamdsm.2014jamdsm0073.

[14] J. Adams, E. Balas, and D. Zawack, "The Shifting Bottleneck Procedure for Job Shop Scheduling," *Manage. Sci.*, vol. 34 (3), pp. 391–401, Mar. 1988. DOI: http://doi.org/10.1287/mnsc.34.3.391.

[15] R. M. Silva, C. Cubillos, and D. Cabrera Paniagua, "A Constructive Heuristic for Solving the Job- Shop Scheduling Problem," *IEEE Latin Am Trans*, vol. 14 (6), pp. 2758–2763, Jun. 2016. DOI: http://doi.org/10.1109/TLA.2016.7555250.

[16] M. Dell'Amico and M. Trubian, "Applying tabu search to the job-shop scheduling problem," *Ann.*

Oper. Res., vol. 41 (1-4), pp. 231–252, Sep. 1993. DOI: http://doi.org/10.1007/BF02023076.

[17] E. Nowicki and C. Smutnicki, "An advanced tabu search algorithm for the job shop problem," *J. Sched.*, vol. 8 (2), pp. 145–159, Apr. 2005. DOI: http://doi.org/10.1007/s10951-005-6364-5.

[18] C. Zhang, P. Li, Z. Guan, and Y. Rao, "A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem," *Comput. Oper. Res.*, vol. 34 (11), pp. 3229–3242, Nov. 2007. DOI: http://doi.org/10.1016/j.cor.2005.12.002.

[19] Y. K. Lin and C. S. Chong, "A tabu search algorithm to minimize total weighted tardiness for the job shop scheduling problem," *JIMO*, vol. 12 (2), pp. 703–717, Jun. 2015. DOI: http://doi.org/10.3934/jimo.2016.12.703.

[20] P. J. M. van Laarhoven, E. H. L. Aarts, and J. K. Lenstra, "Job Shop Scheduling by Simulated Annealing," *Operations Res.*, vol. 40 (1), pp. 113–125, Feb. 1992. DOI: http://doi.org/10.1287/opre.40.1.113.

[21] M. Rojas-Santiago, P. Damodaran, S. Muthuswamy, and M. C. Vélez-Gallego, "Makespan minimization in a job shop with a BPM using simulated annealing," *Int. J. Adv. Manuf. Technol.*, vol. 68 (9–12), pp. 2383–2391, Oct. 2013. DOI: http://doi.org/10.1007/s00170-013-4858-4.

[22] R. Zhang and C. Wu, "A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardinessobjective," *Comput. Oper. Res.*, vol. 38 (5), pp. 854–867, May. 2011. DOI: http://doi.org/10.1016/j.cor.2010.09.014.

[23] D. Merkle and M. Middendorf, "A New Approach to Solve Permutation Scheduling Problems with Ant Colony Optimization," *Springer Berlin Heidelberg*, 2001, pp. 484–494. DOI: http://doi.org/10.1007/3-540-45365-2_50.

[24] A. Udomsakdigool and V. Kachitvichyanukul, "Multiple colony ant algorithm for job-shop scheduling problem," *Int. J. Prod. Res.*, vol. 46 (15), pp. 4155–4175, Aug. 2008. DOI: http://doi.org/10.1080/00207540600990432.

[25] Z. Rui, W. Shilong, Z. Zheqi, and Y. Lili, "An ant colony algorithm for job shop scheduling problem with tool flow," *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.*, vol. 228 (8), pp. 959–968, Aug. 2014. DOI: http://doi.org/10.1177/0954405413514398.

[26] P. Korytkowski, S. Rymaszewski, and T. Wisniewski, "Ant colony optimization for job shop scheduling using multi-attribute dispatching rules," *Int. J. Adv. Manuf. Technol.*, vol. 67 (1-4), pp. 231–241, Jul. 2013. DOI: http://doi.org/10.1007/s00170-013-4769-4.

[27] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, Aug. 2002. DOI: http://doi.org/10.1109/mhs.1995.494215.

[28] D. Y. Sha and H.-H. Lin, "A multi-objective PSO for job-shop scheduling problems," *Expert Syst. Appl.*, vol. 37 (2), pp. 1065–1070, Mar. 2010. DOI: http://doi.org/10.1016/j.eswa.2009.06.041.

[29] T.-L. Lin *et al.*, "An efficient job-shop scheduling algorithm based on particle swarm optimization?," *Expert Syst. Appl.*, vol. 37 (3), pp. 2629–2636, Mar. 2010. DOI: http://doi.org/10.1016/j.eswa.2009.08.015.

[30] F. Zhao, J. Tang, J. Wang, and Jonrinaldi, "An improved particle swarm optimization with decline disturbance index (DDPSO) for multi-objective job-shop scheduling problem," *Comput. Oper. Res.*, vol. 45, pp. 38–50, May. 2014. DOI: http://doi.org/10.1016/j.cor.2013.11.019.

[31] T. Watanabe, H. Tokumaru, and Y. Hashimoto, "Job-shop scheduling using neural networks," *Control Eng. Pract.*, vol. 1 (6), pp. 957–961, Dec. 1993. DOI: http://doi.org/10.1016/0967-0661(93)90005-C.

[32] G. R. Weckman, C. V. Ganduri, and D. A. Koonce, "A neural network job-shop scheduler," *J. Intell. Manuf.*, vol. 19 (2), pp. 191–201, Apr. 2008. DOI: http://doi.org/10.1007/s10845-008-0073-9.

[33] D. C. Mattfeld and C. Bierwirth, "An efficient genetic algorithm for job shop scheduling with tardiness objectives," *Eur. J. Oper. Res.*, vol. 155 (3), pp. 616–630, Jun. 2004. DOI: http://doi.org/10.1016/S0377-2217(03)00016-X.

[34] J.-T. Tsai, T.-K. Liu, W.-H. Ho, and J.-H. Chou, "An improved genetic algorithm for job-shop scheduling problems using Taguchi-based crossover," *Int. J. Adv. Manuf. Technol.*, vol. 38 (9–10), pp. 987–994, Sep. 2008. DOI: http://doi.org/10.1007/s00170-007-1142-5.

[35] I. Essafi, Y. Mati, and S. Dauzere-Peres, "A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem," *Comput. Oper. Res.*, vol. 35 (8), pp. 2599–2616, Aug. 2008. DOI: http://doi.org/10.1016/j.cor.2006.12.019.

[36] L. Wang, J.-C. Cai, and M. Li, "An adaptive multi-population genetic algorithm for job-shop scheduling problem," *Adv. Manuf.*, vol. 4 (2), pp. 142–149, Jun. 2016. DOI: http://doi.org/10.1007/s40436-016-0140-y.

[37] P. Moscato and M. G. Norman, "A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems," *Proc. Parallel Comput. Transputer Appl.*, vol. 28 (1), pp. 177–186, 1992.

[38] X. Guo, Z. Wu, and G. Yang, "A Hybrid Adaptive Multi-objective Memetic Algorithm for 0/1 Knapsack Problem," *Springer Berlin Heidelberg*, 2005, pp. 176–185. DOI: http://doi.org/10.1007/11589990_20.

[39] H. Ishibuchi, Y. Hitotsuyanagi, N. Tsukamoto, and Y. Nojima, "Implementation of Multiobjective Memetic Algorithms for Combinatorial Optimization Problems: A Knapsack Problem Case Study," *Studies in Computational Intelligence-Chapter 2*, vol. 171, pp. 27–49, 2009. DOI: http://doi.org/10.1007/978-3-540-88051-6_2.

[40] A. Rezoug, D. Boughaci, and M. Badr-El-Den, "Memetic Algorithm for Solving the 0-1 Multidimensional Knapsack Problem," *Springer International Publishing*, 2015, pp. 298–304.

[41] C. Prins and S. Bouchenoua, "A Memetic Algorithm Solving the VRP, the CARP and General Routing Problems with Nodes, Edges and Arcs," *Studies in Fuzziness and Soft Computing-Chapter 4*, vol. 166, pp. 65–85, 2005. DOI: http://doi.org/10.1007/3-540-32363-5_4.

[42] J. Schönberger, "Memetic Algorithm Vehicle Routing," in *Operational Freight Carrier Planning*, Berlin/Heidelberg: Springer-Verlag, 2005, pp. 65–76.

[43] J. Nalepa and M. Blocho, "Adaptive memetic algorithm for minimizing distance in the vehicle routing problem with time windows," *Soft Comput.*, vol. 20 (6), pp. 2309–2327, Jun. 2016. DOI: http://doi.org/10.1007/s00500-015-1642-4.

[44] J.-F. Cordeau, M. Gaudioso, G. Laporte, and L. Moccia, "A Memetic Heuristic for the Generalized Quadratic Assignment Problem," *INFORMS J. Comput.*, vol. 18 (4), pp. 433–443, Nov. 2006. DOI: http://doi.org/10.1287/ijoc.1040.0128.

[45] U. Benlic and J.-K. Hao, "Memetic search for the quadratic assignment problem," *Expert Syst. Appl.*, vol. 42 (1), pp. 584–595, Jan. 2015. DOI: http://doi.org/10.1016/j.eswa.2014.08.011.

[46] T. Fischer and P. Merz, "A Memetic Algorithm for the Optimum Communication Spanning Tree Problem," in *Hybrid Metaheuristics*, Springer Berlin Heidelberg, pp. 170–184.

[47] H.-C. Cheng, T.-C. Chiang, and L.-C. Fu, "A two-stage hybrid memetic algorithm for multiobjective job shop scheduling," *Expert Syst. Appl.*, vol. 38 (9), pp. 10983–10998, Sep. 2011. DOI: http://doi.org/10.1016/j.eswa.2011.02.142.

[48] M. R. Raeesi N. and Z. Kobti, "A memetic algorithm for job shop scheduling using a critical-path-based local search heuristic," *Memetic Comp*, vol. 4 (3), pp. 231–245, Sep. 2012. DOI: http://doi.org/10.1007/s12293-012-0084-0.

[49] Y. Yuan and H. Xu, "Multiobjective Flexible Job Shop Scheduling Using Memetic Algorithms," *IEEE Trans Automat Sci Eng*, vol. 12 (1), pp. 336–353, Jan. 2015. DOI: http://doi.org/10.1109/TASE.2013.2274517.

[50] R. Mencía, M. R. Sierra, C. Mencía, and R. Varela, "Memetic algorithms for the job shop scheduling problem with operators," *Appl. Soft Comput.*, vol. 34, pp. 94–105, Sep. 2015. DOI: http://doi.org/10.1016/j.asoc.2015.05.004.

[51] E. Nowicki and C. Smutnicki, "A Fast Taboo Search Algorithm for the Job-Shop Problem," *Manage. Sci.*, vol. 42 (6), pp. 797–813, Jun. 1996. DOI: http://doi.org/10.1287/mnsc.42.6.797.

[52] L. Gao, G. Zhang, L. Zhang, and X. Li, "An efficient memetic algorithm for solving the job shop scheduling problem," *Comput. Ind. Eng.*, vol. 60(4), pp. 699–705, May. 2011. DOI: http://doi.org/10.1016/j.cie.2011.01.003.

[53] J. E. Beasley, "OR-Library: Distributing Test Problems by Electronic Mail," *J. Oper. Res. Soc.*, vol. 41 (11), p. 1069, Nov. 1990. DOI: http://doi.org/10.1057/jors.1990.166.

[54] M. A. González Fernández, *Soluciones metaheurísticas al 'job-shop scheduling problem with sequence-dependent setup times*, 2011.

[55] S. M. Kumrul Hasan, R. Sarker, D. Essam, and D. Cornforth, "Memetic algorithms for solving job-shop scheduling problems," *Memetic Comp*, vol. 1 (1), pp. 69–83, Mar. 2009. DOI: http://doi.org/10.1007/s12293-008-0004-5.