Journal of Applied Intelligent System (e-ISSN : 2502-9401 | p-ISSN : 2503-0493)
Vol. 7 No. 1, April 2022, pp. 80 – 88

A Good Relative Percentage Increase (RPI) of Variant Job Scheduling Using Artificial Bee Colony (ABC)

Riri Damayanti Apnena*[1], **Firdhani Faujiyah**[2]
[1)]Mekanik Industri dan Desain, Politeknik TEDC Bandung
Pesantren KM. 2, Cibabat, Kec. Cimahi Utara, Cimahi City, Jawa Barat, 40513
[2)]Manajemen Pemasaran Industri Elektronika, Politeknik APP Jakarta
Jl. Timbul No.34, RT.6/RW.5, Cipedak, Kec. Jagakarsa, Kota Jakarta Selatan, Daerah Khusus Ibukota Jakarta 12630
E-mail : riri.damayanti.apnena@poltektedc.ac.id[1],f.firdhani@poltekapp.ac.id[2]*
**Corresponding author*

Abstract - Artificial Bee Colony (ABC) which is a development of the intelligent swarm model and is a branch of artificial intelligence based on self-organization systems. Artificial Bee Colony (ABC) is an intelligent algorithm that is inspired by the food search process carried out by bees. This is like what is done when there are many jobs that need to find the optimal value, where each job to be processed has a specific route of operations to be performed on a set of machines, and a different flow shop and variant: all jobs follow the same machine sequences. We will focus on the latter. In this study, ABC is implemented to optimize work scheduling, in this case 7 different variations are used with mxn values between 10x3 to 40x15 on 10 to 40 jobs. To evaluate the results, the Relative Percentage Increase (RPI) has been used in the test with an achievement range between 1.9 to 18.9.

Keywords – Scheduling, Job Shop, Artificial Bee Colony (ABC)

1. INTRODUCTION

Production scheduling is a decision-making process that exists in most manufacturing and production systems, as well as in most information processing environments. Similarly, it is also applied in the transportation and distribution environment and in other types of service industries. Scheduling attempts to allocate limited resources to tasks over time. It is a decision-making process aimed at optimizing one or more objectives. Resources and tasks can take many forms, for example, a resource can be a machine in a shop, a runway at an airport, a crew at a construction site, a processing unit in a computing environment, etc. and tasks can be operations in the production process, take-off and landing at airports, stages in construction projects, execution of computer programs, etc. Each task can have a different priority level [1]–[4], among other things based on the earliest possible start time or due date. Goals can also take many forms.

One possible goal is to minimize the time for completion of the last task, and another possible goal is to minimize the number of tasks completed after a specified due date [2]. In the same way, various limitations can be considered, such as not allowing the machine to be idle while processing different jobs. These constraints model important practical situations when very expensive machines are used or technical limitations prevent these situations from occurring [5]. Moreover, despite its great relevance, this is an issue that has not been specifically addressed in the literature. That is why we cover this aspect in this work. The specific purpose of this work is solving programming problems in a flow shop environment, to minimize the completion time of the last job, or makespan, under the no idle limit, that is, inactivity in the machine is not allowed after the first job starts processing [6].

Scheduling is responsible for allocating resources to tasks that must be performed over a time horizon to achieve a production plan, and time horizons are usually less than a week. Thus, scheduling is the allocation of resources to the tasks that define the production plan [7], and its complexity and importance will grow to the extent that the use of resources conflicts. Common scheduling conditions for all production systems:

- They are complex decisions. And its complexity will continue to increase due to the current trend of increasing flexibility in customer orders, which results in increasing the complexity of the production process.
- It has a short time horizon, the life cycle of an order or schedule is usually very short.
- Despite being short-term decisions, they are usually very important for an organization's production line since they have a direct impact on production costs and delivery times, which affects the company's competitiveness (delivery time and costs).
- Because they are located in the center of production operations, the scheduling restrictions and objectives are highly dependent on the type of company.
- They are usually structured decisions, which allows information, criteria and limitations to be easily formalized.

Programming problems are generically referred to as α | β | γ [4], being α the environment to be considered. There are different types of environments:

- Single Machine (α=1). Indicates that it is a single machine environment.
- Parallel Machine. It is made up of a series of parallel machines through which any job can be processed. In turn, they can be divided into:
  - Identical Parallel machines (α=Pm). The process times do not depend on the machine.
  - Uniform Parallel machines (α=Qm). Each machine has different speeds.
  - Unrelated Parallel machines (α=Rm). The most general case, with different machines.
- Workshop-type environments, with machines arranged in series, and which differ depending on the route of their work:
  - Flow Shop (α=Fm). All jobs have the same route.
  - Job Shop (α=Jm). Jobs follow a predetermined path through machines.
  - Open Shop (α=Om). The jobs path is not defined. It is the most complex of all.

There are also hybrid environments (α=Hm), formed by the combination of several environments.

2. RESEARCH METHOD

2.1. Job Shop

The production configuration of the flow shop type consists of a group of machines that perform their tasks sequentially, as we can see in Figure 1. The first task must be done on the first machine first, then on the second machine, and so on. This figure schematically shows what a flow shop configuration looks like. Four work centers represented by four machines [5], [8]. The arrows indicate the direction of the jobs that are processed in said system. All jobs go through all machines and in the same sequence.

In a more formal sense, a flow shop system consists of a set J = $\{J1, \ldots, Jn\}$ of n jobs and a set M = $\{Mi, \ldots, Mm\}$ of m machines. Each job consists of a set Oj = $\{Oj,1, \ldots, Oj,m\}$ of m operations, where the ith operation is executed on machine Mi . Furthermore, the operation

$Oj,+1$ can start if and only if the operation Oj,i was completed. Each operation $Oj,$has a processing time pi,j ϵ ℕ. Each machine Mi can process only one operation at a time.

A job ordering σ is defined as the sequence in which each machine will process the jobs, and consequently the operations that it must perform. Two types can be distinguished [7], [9], [10]:

- If this ordering of jobs on each machine is the same, the scheduling problem is called a permutative flow shop problem (PFS).
- Otherwise it is called non-permutative flow shop problem (NPFS).

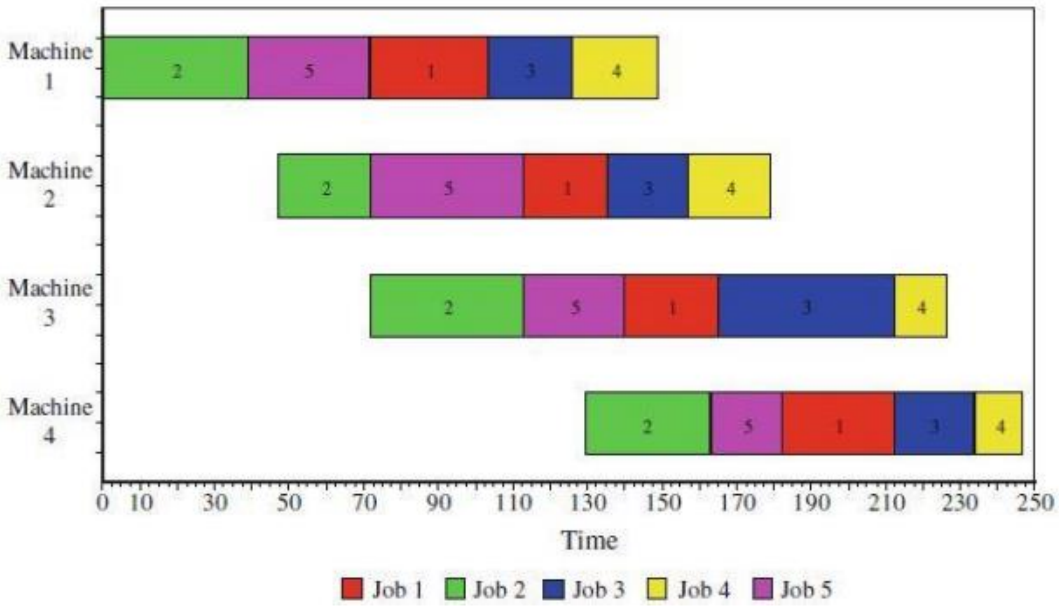

Figure 1. Flow Shop Gantt Chart with no-idle constraint

Within the Flow shop environment, the no-idle constraint is considered. An interesting situation arises when no machine downtime is allowed between jobs. This restriction models an important practical situation that arises when expensive machinery is used. Idling in expensive equipment is often undesirable or even not possible due to technological limitations [11], [12]. Some examples are the steppers used in the production of integrated circuits by photolithography. Other examples come from industries where less expensive machinery is used, but where machines cannot be easily stopped and restarted or it is completely uneconomical to do so [5], [6], [10]. Ceramic roller kilns, for example, consume a large amount of natural gas when in operation. Idling is not an option because it takes several days to stop and restart the oven due to very large thermal inertia. In all these cases, idling should be avoided.

2.2. Artificial Bee Colony (ABC)

To solve the problem described, the "Artificial Bee Colony" (ABC) algorithm has been chosen, which has shown to have good performance compared to other resolution methods for this type of problem in a Flow shop environment [11], [12]. These results are shown in the computational analysis chapter. The Artificial Bee Colony is an optimization algorithm based on swarm intelligence, proposed by Karaboga (2005) to solve modal evolutionary optimization problems [3], [7]. One of the advantages of this algorithm over others is the use of few control parameters [13], [14]. It is inspired by the intelligent behavior that bees follow when searching for food. The bees, they are divided into 3 groups:

- Employed or employed bee: It is one that is carrying out the exploitation of a food source.
- Spectator or onlooker bee: They are the bees that wait in the hive for the information given by the employed bees, in order to decide which of the current food sources should be exploited.
- Explorer or scout bee: They are those that perform a random search in order to find new food sources.

In the algorithm, each of the possible solutions to the problem considered are called food sources and are represented by a vector of dimension n, while the quality of each solution corresponds to the amount of nectar present in said food source. Like other techniques based on swarm intelligence, it is an iterative process [15], [16]. The algorithm starts with a randomly generated population of food sources, and then the following steps are followed until a termination criterion is reached:

1. The employed bees are sent to the food sources and the amounts of nectar present in them are measured.
2. Bystander bees select food sources after sharing information with employed bees.
3. Determine scout bees and send them to potential new food sources.

The parameters of the original ABC algorithm are the number of food sources (SN), which is equal to the number of employed or spectator bees [4], [17], the number of attempts after which a food source must be abandoned (limit), and the criterion of termination. The main phases of the original algorithm as shown in Figure 2.

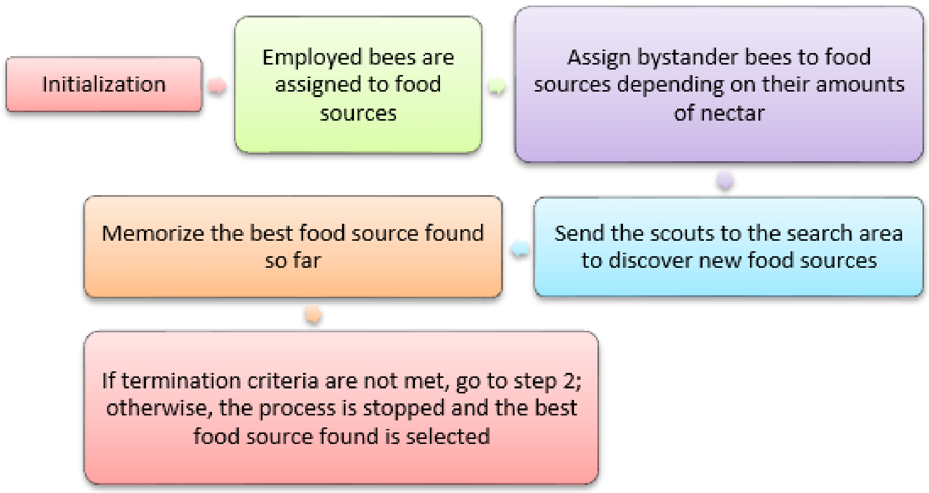
Initialization
Employed bees are assigned to food sources
Assign bystander bees to food sources depending on their amounts of nectar
Send the scouts to the search area to discover new food sources
Memorize the best food source found so far
If termination criteria are not met, go to step 2; otherwise, the process is stopped and the best food source found is selected

Figure 2. Artificia Bee Colony Steps in this research

3. RESULTS AND DISCUSSION

This section aims to adapt and use the ABC algorithm in solving the problem posed in this TFG, a problem in a flow shop type environment, considering the no-idle constraint. This algorithm has shown to have good results in solving optimization problems, compared to other similar algorithms, with the advantage of using fewer control parameters. The algorithm seeks, following the same methodology described above, to find an optimal solution to the problem. This section explains the ABC algorithm; thus, the steps to be followed by the algorithm are explained in detail, as well as the changes necessary to adapt it to our problem, since it was originally designed for the optimization of continuous functions. To guarantee a certain quality and diversity of the population, the initial sequences are generated as explained below:

- One of the sequences has been generated by assigning the SPT (Shortest Processing Time) rule to the first machine, that is, ordering the processing times of the jobs in the sequence in ascending order.
- Another of the sequences has been generated by assigning the LPT (Longest Processing Time) rule to the last machine, that is, the jobs of the sequence in descending order of processing time.
- The rest of the sequences were generated randomly, in order to ensure the diversity of the population as mentioned above.

The variations of the algorithm have been made in the spectator bee or onlooker phase, specifically when choosing the members of the population that are susceptible to change. This phase plays a very important role in the algorithm, since it decides where to focus the search efforts. Placing too much emphasis only on the best sequences, we limit ourselves to certain search areas, being able to get trapped in a local optimum, while giving opportunity to a priori unpromising sequences, the search is too dispersed, making it difficult to reach an optimal solution to the problem. That is why we must try to avoid these two extremes, reconciling global exploration with local exploitation. We have made 7 variations of the algorithm in this aspect, in all cases making SN improvement attempts on the sequences. In the first place, we have made 3 variants that consist of a tournament, in which the sequences compete with each other and only the best one is selected had been illustrated in Table 1.

Table 1. Variable of Experiment

Variant name	Explanation
ABC1	A size 2 tournament is held that corresponds to the variant used. In this tournament the best sequence is selected between 2 chosen randomly
ABC2	Same as in the previous case but with a tournament of size 3 sequences, that is, 3 sequences are chosen randomly, they are compared between them and the best one is selected.
ABC3	Each sequence is assigned a probability ϵ (0.1), entering the tournament those with a probability less than 0.4.
For the rest of the cases, among the SN=10 (see values of the parameters in Annex 1) times that the phase is carried out, instead of a tournament we directly distribute the number of attempts among the best sequences of the population. For example, (4,3,3) would be sharing the 10 trials by assigning 4 trials to the best sequence, 3 trials to the second best sequence, and the other 3 to the third best sequence. Following this notation, the remaining variants would be as follows	
ABC4	5 attempts for the best sequence and 5 for the second. (5 .5).
ABC5	2 Improvement attempts for each of the 5 best sequences. (2, 2, 2, 2, 2).
ABC6	(3, 2, 2, 1, 1, 1) Improvement attempts for the best sequences respectively.
ABC7	(3, 3, 2, 2) Improvement attempts for the best sequences respectively.

With these variants in Table 1, it is intended to address the problem mentioned above, of balancing the global search with the local exploitation in a varied way. Some variants focus only on the best sequences, such as the variant ABC4, while others involve and give opportunity to improve a larger number of them. In this block we will analyze the efficiency and results obtained when solving the problem with the proposed algorithm and its variations. As we mentioned before, it is a problem in a Flow shop environment under the no-idle restriction or inactivity in machines not allowed. We have a total of 20 different instances according to the following sizes of jobs and machines respectively, (n = 10, 20, 30, 40; m = 3, 5, 7, 10, 15). The work times are randomly generated being pi, ϵ (0.99). The algorithm parameters are as follows: SN = 10 (population members), limit = 20 (number of failed attempts before abandoning food source), pL = 0.1 (probability of performing local search), and $lmax$ = $n2$ (number of iterations in the local search). Regarding the process time, CT = $5n2m$ ms is established as termination criterion as illustrated in Figure 3.

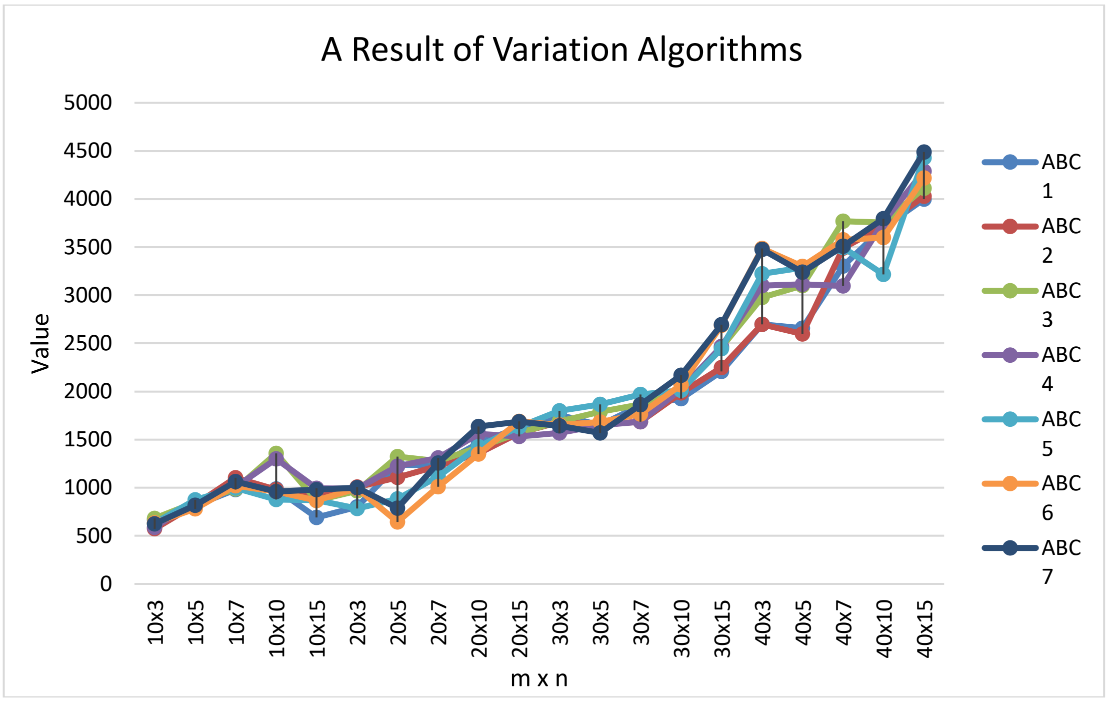

Figure 3. Variants of The Algorithm

Based on Figure 3, we calculate the Relative Percentage Increase (RPI) values for each instance and variant. The values obtained for all the instances for each variant of the algorithm and the total average of each of them are shown in Figure 3. The best overall result is from the second variant, although very different results are not observed between the different variants, unlike if we only consider more specific cases of the problem.

In the sections that follow, the RPI values divided according to the number of jobs are shown and analyzed, thus leaving 4 groups of instances of 10, 20, 30 and 40 jobs, respectively. In this way we can analyze the performance of each of the variants for specific problem cases. We can also observe these results, the mean being represented by the black bar. In addition, we can also see more clearly the RPI values for each job instance in each variant of the algorithm. In the case of instances with 20 jobs, we can see that the third variant obtains the best results, having the best average value in addition to obtaining the best value in 3 of the 5 instances considered, as we can see in Figure 3. For instances of 30 jobs, the third algorithm obtains the best average result, although not with a very significant difference, this being very similar to that obtained with other variants. Lastly, we consider the case of instances of 40 jobs, where it is the first algorithm that obtains the best result, with a greater difference compared to the other variants than in the previous case. Said algorithm obtains the best result in 3 of the considered instances. However, it does not give a good result in the specific case of 40 jobs and 7 machines. On the other hand, the fourth, fifth and sixth variants do not obtain a good average result, in addition to not finding the best solution in any of the instances.

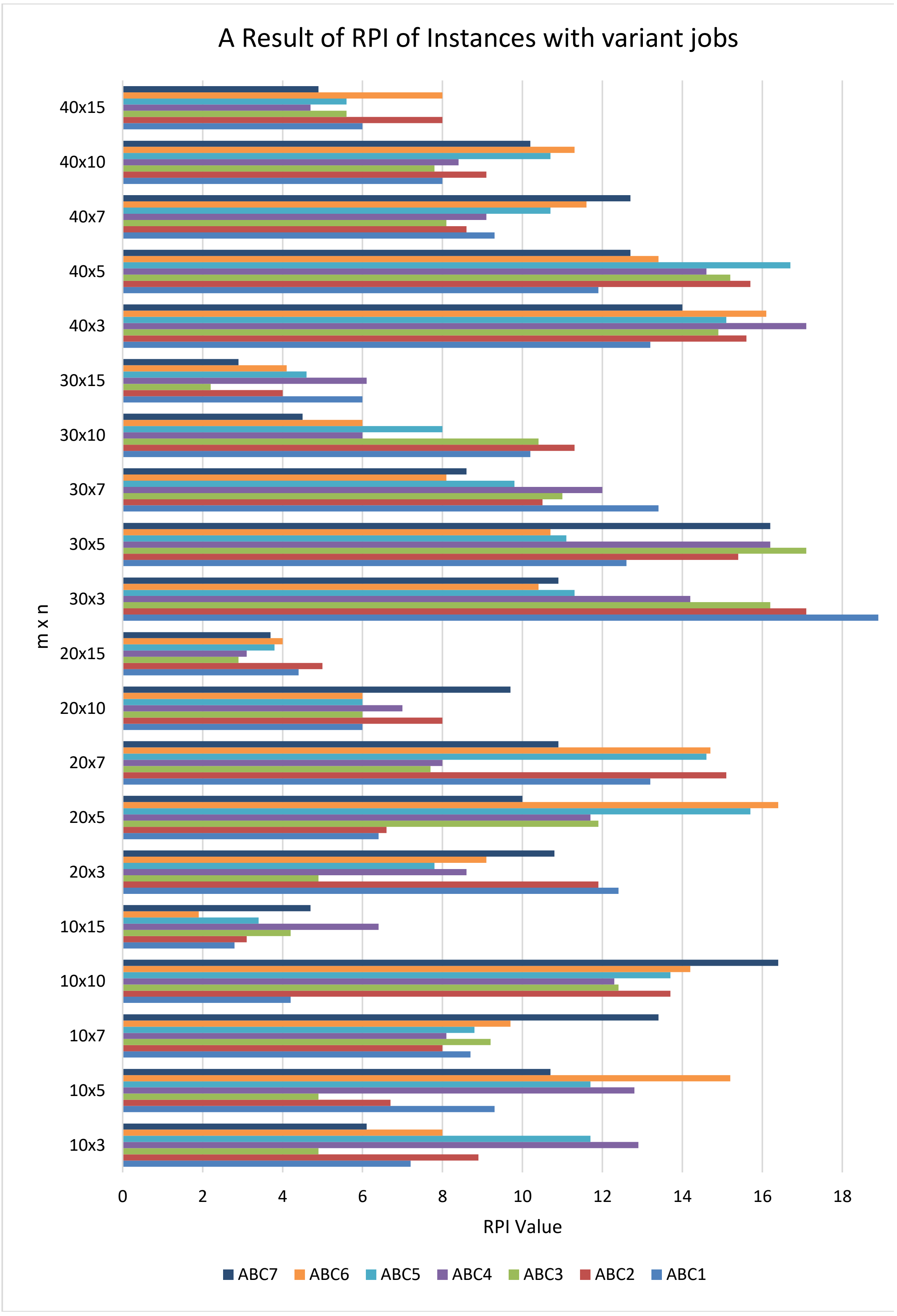

Figure 4. RPI Results of Variant Jobs

4. CONCLUSION

This work tries to address a production scheduling problem, specifically a problem in a flow shop environment with the objective of minimizing the makespan. We consider the no-idle constraint, which does not allow machines to idle once they have started processing the first job. To solve this problem, we have used an optimization algorithm, called Artificial Bee Colony (ABC), which uses swarm intelligence to find an optimal result for the problem. This algorithm simulates the real behavior of bees when searching for food sources. We have made various variants of the algorithm, specifically in the onlooker bee phase, explained earlier in this document. Based on the results obtained, the conclusions reached are the following:

- The second variant has obtained the best average results. This variant proposes the realization of a tournament of size 3 when choosing the sequences that are considered most promising and that deserve more effort to be exploited by the algorithm.
- However, other variants have shown better efficiency for specific instances among the proposals, such as the first variant for instances with 40 jobs, or the third for instances of 20 and 30 jobs.

As lines of future research, variations of the algorithm could be explored in other phases of the same, in addition to applying them to other problems with different characteristics, which have not been sufficiently studied in the literature.

REFERENCES

[1] T. Meng, Q. K. Pan, and H. Y. Sang, "A hybrid artificial bee colony algorithm for a flexible job shop scheduling problem with overlapping in operations," Int. J. Prod. Res., vol. 56, no. 16, pp. 5278–5292, 2018.

[2] M. Shahgholi Zadeh, Y. Katebi, and A. Doniavi, "A heuristic model for dynamic flexible job shop scheduling problem considering variable processing times," Int. J. Prod. Res., vol. 57, no. 10, pp. 3020–3035, 2019.

[3] G. Gong, R. Chiong, Q. Deng, and X. Gong, "A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility," Int. J. Prod. Res., vol. 58, no. 14, pp. 4406–4420, 2020.

[4] X. Long et al., "Research on job-shop scheduling problem based on bee colony algorithm," J. Phys. Conf. Ser., vol. 2033, no. 1, 2021.

[5] Y. Li, W. Huang, R. Wu, and K. Guo, "An improved artificial bee colony algorithm for solving multi-objective low-carbon flexible job shop scheduling problem," Appl. Soft Comput. J., vol. 95, p. 106544, 2020.

[6] Z. J, S. Z.H., and C. C., "An Improved Whale Optimization Algorithm for Job-Shop Scheduling Based on Quantum," Int j simul Model, vol. 18, pp. 521–530, 2019.

[7] T. K. Dao, T. S. Pan, T. T. Nguyen, and J. S. Pan, "Parallel bat algorithm for optimizing makespan in job shop scheduling problems," J. Intell. Manuf., vol. 29, no. 2, pp. 451–462, 2018.

[8] N. Xie and N. Chen, "Flexible job shop scheduling problem with interval grey processing time," Appl. Soft Comput. J., vol. 70, pp. 513–524, 2018.

[9] J. Sassi, I. Alaya, P. Borne, and M. Tagina, "A decomposition-based artificial bee colony algorithm for the multi-objective flexible jobshop scheduling problem," Eng. Optim., vol. 54, no. 3, pp. 524–538, 2022.

[10] R. Wu, Y. Li, S. Guo, and W. Xu, "Solving the dual-resource constrained flexible job shop scheduling problem with learning effect by a hybrid genetic algorithm," Adv. Mech. Eng.,

vol. 10, no. 10, pp. 1–14, 2018.

[11] S. Sundar, P. N. Suganthan, C. T. Jin, C. T. Xiang, and C. C. Soon, “A hybrid artificial bee colony algorithm for the job-shop scheduling problem with no-wait constraint,” Soft Comput., vol. 21, no. 5, pp. 1193–1202, 2017.

[12] X. Li, S. Xiao, C. Wang, and J. Yi, “Mathematical modeling and a discrete artificial bee colony algorithm for the welding shop scheduling problem,” Memetic Comput., vol. 11, no. 4, pp. 371–389, 2019.

[13] N. Sharma, H. Sharma, and A. Sharma, “Beer froth artificial bee colony algorithm for job-shop scheduling problem,” Appl. Soft Comput. J., vol. 68, pp. 507–524, 2018.

[14] K. Peng, Q. Pan, and B. Zhang, “An improved artificial bee colony algorithm for steelmaking–refining–continuous casting scheduling problem,” Chinese J. Chem. Eng., vol. 26, no. 8, pp. 1727–1735, 2018.

[15] D. Gong, Y. Han, and J. Sun, “A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems,” Knowledge-Based Syst., vol. 148, pp. 115–130, 2018.

[16] J. Xie, L. Gao, Q. K. Pan, and M. Fatih Tasgetiren, “An effective multi-objective artificial bee colony algorithm for energy efficient distributed job shop scheduling,” Procedia Manuf., vol. 39, no. 2019, pp. 1194–1203, 2019.

[17] X. Long, J. Zhang, K. Zhou, and T. Jin, “Dynamic Self-Learning Artificial Bee Colony Optimization Job Insertion,” processes, vol. 10, pp. 1–22, 2022.