Artificial Immune System Applied to Job Shop Scheduling

Steven Kosasih, Cecilia E. Nugraheni, and Luciana Abednego
Parahyangan Catholic University, Bandung, Indonesia
Email: 7315050@student.unpar.ac.id, cheni@unpar.ac.id, luciana@unpar.ac.id

***Abstract*—Job Shop Scheduling is a problem to schedule n number of jobs in m number of machines with a different order of processing. Each machine processes exactly one job at a time. Each job will be processed in every machine once. When a machine is processing one particular job then the other machine can't process the same job. Different schedule's order might produce different total processing time. The result of this scheduling problem will be total processing time and schedule's order. This paper uses clonal selection as the algorithm to solve this problem. The clonal selection algorithm comes from the concept of an artificial immune system. It's developed by copying a human's immune system behavior. A human's immune system can differentiate foreign objects and eliminate the objects by creating an antibody. An antibody will go to a cloning process and will mutate to further enhance itself. Clonal selection algorithm applies this cloning and mutation principle to find the most optimal solution. The goal is to find the best schedule's order and makespan. Taillard's benchmark is used to verify the quality of the result. To compare the result, we use two values: the upper bound and the lower bound. The upper bound is used to describe the best result of a scheduling problem that has been conducted using a certain environment. On the contrary, the lower bound shows the worst. Experiments on changing the algorithm's parameters are also conducted to measure the quality of the program. The parameters are the number of iterations, mutations, and clone numbers. According to the experiment's results, the higher the number of iteration, mutation rate, and clone number, the better solution for the problem. Clonal selection algorithm has not been able to keep up with upper bound or lower bound values from Taillard's case. Therefore, parameters need to be increased significantly to increase the chance to produce the optimum result. The higher number of parameters used means the longer time needed to produce the result.**

***Index Terms*—job shop scheduling, artificial immune system, clonal selection, Tailard's benchmark**

I. Introduction

Scheduling is a problem of assigning resources to a range of time so that some objectives functions are optimized. In general, scheduling is needed in many fields. With scheduling, we can measure what we can realistically achieve in a given time period and forecast the finish time of a given job in the queue [1]. In this paper, we discuss the job shop scheduling problem which will be explained in more detail in the following section. We've done some research related to the scheduling problem [2]-[8]. In this paper, clonal selection algorithm is used to solve the job shop scheduling problem.

II. Job Shop Scheduling Problem

This research's goal is to solve the job shop scheduling problem. It's an optimization problem to assign **n** jobs to **m** number of machines with different order of processing. Given n jobs: $J = \{J_1, J_2, J_3, \ldots, J_n\}$ with various processing times to be assigned on m machines $M = \{M_1, M_2, M_3, \ldots, M_m\}$. The order of the process can be various for each job. Each job is processed in every machine exactly once. Each job (J_i) consist of a set of operations $O_{i1}, O_{i2}, O_{i3}, \ldots, O_{in}$. Each operation has different processing time PT_{ij}, which denotes the processing time of job i at machine j. The goal is to find a schedule with minimum makespan. Makespan is the total time needed until all of these jobs are finished to be processed in these machines [9].

Job shop scheduling can be visualized using a gantt chart to manually calculate the makespan. Fig. 1 shows an example of a gantt chart that processes 3 jobs at 3 machines with different processing orders. As can be seen at Fig. 1, the processing order for job 1 (J1) is: machine 2 - machine 1 - machine 3. Job 2 (J2) is processed with different orders: machine 3 - machine 2 - machine 1. Job 3 (J3) is processed in machine 1 - machine 3 - machine 2. Notice there are some gaps between J2 in machine 3 and J3 in machine 3. The reason J3 cannot be started at machine 3 at time 1 is because it hasn't finished yet to be processed in machine 1. J3 just finished to be processed in machine 1 at time 4, then it can be continued to the next machine (machine3) at time 4.

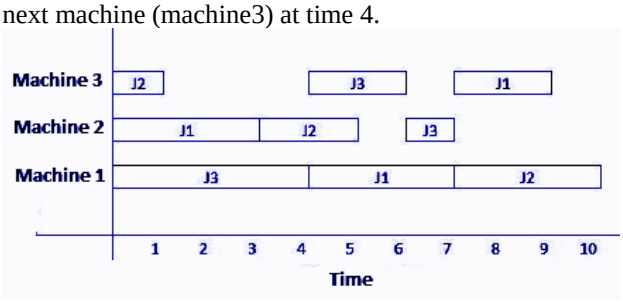

Figure 1. Illustration of Job Shop Scheduling Problem.

Table I shows the same example in a tabular form. This table shows the order of machines for each job with

Manuscript received January 22, 2021; revised May 28, 2021.

doi: 10.18178/jiii.9.1.15-22

each processing time. Each row in the table shows the processing order for each job.

TABLE I. PROCESSING TIME TABLE OF FIGURE 1

Job	Operation i1		Operation i2		Operation i3	
	Time	Machine	Time	Machine	Time	Machine
J1	3	2	3	1	2	3
J2	1	3	2	2	3	1
J3	4	1	2	3	1	2

III. CLONAL SELECTION

A. Artificial Immune System

Artificial immune system is a collection of techniques that mimics natural immune responses that defend the body from foreign pathogens (nonself-antigen) [10]. T-cell and B-cell are part of lymphocytes that respond to specific antigens invading the body. T-cell detects foreign pathogens and gives an order to B-cell to eliminate them. Inspired by immunology, artificial immune systems have been applied to real-world science and engineering problems, such as job shop scheduling.

Fig. 2 shows an illustration of the antigen elimination process, which is called clonal selection. After antigens have been detected by T-cell, B-cell creates antibodies that destroy nonself-antigen. B-cell multiplies the production of identical clones of antibodies by combining each structure with the others. This concept is called the antibody library which is used in the artificial immune system.

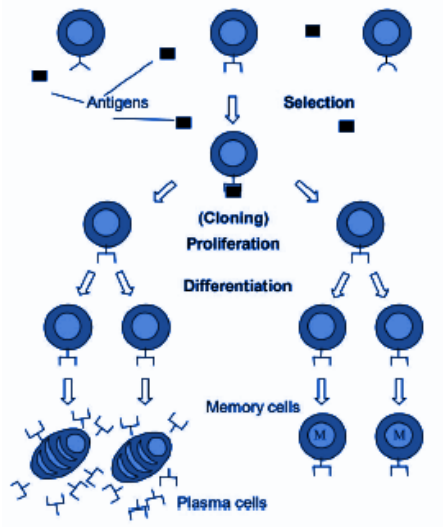

Figure 2. Clonal selection process.

B. Antibody Library

The antibody library is a collection of partial solutions that forms the antibody (see Fig. 3). An antibody library consists of several components (denotes as C_i). Each component consists of the same number of genes for each library. These genes are randomly generated and have different values for each component. The length of a component is usually obtained by dividing the length of an antibody with a certain integer number. A gene in a component can model any kind of information, such as a number or a letter [11].

Fig. 3 (bottom) shows a new antibody that is formed by selecting some random components from the library. In Fig. 3 (bottom), an antibody with length 15 is divided into 3 parts with equal length. The length of each part is equal to the length of a library component. In this figure, a new antibody is a combination of the fourth component of library-1 (L1C4), the third component of library-3 (L3C3), and the first component of library-2 (L2C1).

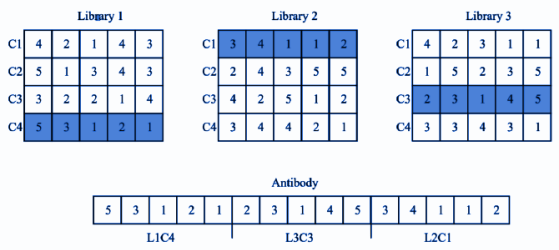

Figure 3. Antibody libraries (top) and a new formed antibody (bottom).

C. Clonal Selection Algorithm

Clonal selection algorithm is a technique that adapts artificial immune systems. It uses two populations: antibody population and antigen population. Antibody population is the current best population. Antigen population is defined by the environment and formed by randomizing antibody solutions. After the population is formed, the population with the highest fitness will be cloned and mutated.

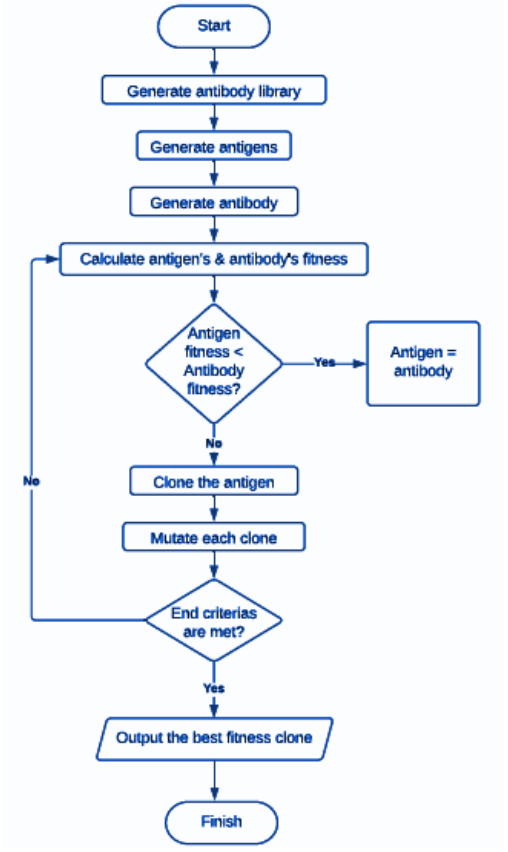

Figure 4. Clonal selection algorithm flowchart.

Fig. 4 shows the flowchart of the clonal selection algorithm applied in this paper [12], [13]. At first, this algorithm generates antibody libraries once. The number of libraries is equal to the length of an antibody divided by an integer number. This makes sure the component length of each library gets the equal portion in the new antibody. The next step is to generate antibodies and antigen. An antibody is generated by randomly picking any library's components. If a library has chosen, it can't be used again in the next step. The process is continued until a new solution is fully generated. Antibody libraries are formed based on the antigen. It is inspired by the human immune system that is built antibody libraries based on the foreign pathogen (antigen). Antigen itself is formed randomly.

In this algorithm, the fitness value of the antigen will be compared with the fitness value of the antibody. The antigen helps to find the best solution because in each iteration the value of the antigen is replaced by the best antibody found so far. In the next step, only the best solutions from the previous steps will be cloned.

At the comparison process, antigen with worse fitness than the antibody will be replaced by the whole genes from the antibody. This antigen will then be cloned. Some clones are formed based on the previous best antigens. These clones will then be mutated. The fitness for each mutated clone will be calculated, and the best-mutated clones will be the new solution candidate. This process is continued until some final criteria are met. The best clone from the final iteration will be the final solution for the clonal selection algorithm.

D. Fitness

Fitness value is an evaluation function to measure the performance of the algorithm. It is calculated with the following formula:

$$fitness = \frac{1}{makespan} \quad (1)$$

Makespan is the total processing time needed until all the jobs are finished to be processed in all machines. The formula is designed so that the longer the makespan, the smaller the fitness value for the candidate solution. On the other hand, the smaller makespan will make the fitness value bigger.

E. Hypermutation

Hypermutation type in the clonal selection that is used in this paper is inverse mutation. It is the most commonly used in clonal selection algorithms. It works by investigating the sequence of the solution from j to i index. As an example, if there is a sequence of 1-2-3-4-5 and i=2 and j=5, then by applying the inverse mutation, the sequence becomes: 1-5-4-3-2. The makespan of the original sequence will then be compared to the makespan of the new sequence. If the makespan of the new sequence is less than the original sequence, then this sequence will replace the original sequence. The new sequence mimics the antibody clone after mutation, while the original sequence mimics the antibody clone before the mutation.

F. Mutation Rate

Mutation rate or probability is the rate that determines how many times a candidate solution should be mutated in a cloning process. In this paper, the mutation rate is calculated by the following formula:

$$mutation\_rate = lengthOfAntibody \times 3 \quad (2)$$

where

- ***mutation_rate:*** An integer value that shows how many times the mutation process will be executed.
- ***Length of antibody***: The length of an antibody

G. Taillard Benchmark

As a benchmark problem, we use Taillard [14] set problem to measure the performance of the clonal selection algorithm. The problem size corresponds to the real dimensions of industrial problems. The upper bound and lower bound makespans show the best and the worst cases of makespan for each test case. We conduct some experiments and compare the makespan results with the upper and lower bound in the Taillard's dataset.

Each dataset consists of the information about:

1. Number of machines
2. Number of jobs
3. The job processing order for each machine
4. Processing time for each job at each machine.

```
Nb of jobs, Nb of Machines, Time seed, Machine seed, Upper bound, Lower bound
      20        20 1035939303  773961798      2100      1711
Times
 64 57 81 98 59 87 93 62 20 14 85 45 47  9 94  9 15 66  1 94
 39 96 88 83 77 58 83  3 78 68 64 97 33 25 47 44  7 60 42 91
 96 66 88 60 22 92 62 14 89 39 94 66 10 53 26 15 65 82 10 27
 93 92 96 70 83 74 31 88 51 57 78  8  7 91 79 18 51 18 99 33
  4 82 40 86 50 54 21  6 54 68 82 20 39 35 68 73 23 30 30 53
 94 58 93 32 91 30 56 27 92  9 78 23 21 60 36 29 95 99 79 76
 93 42 52 42 96 29 61 88 70 16 31 65 83 78 26 50 87 62 14 30
 18 75 20  4 91 68 19 54 85 73 43 24 37 87 66 32 52  9 49 61
 35 99 62  6 62  7 80  3 57  7 85 30 96 91 13 87 82 83 78 56
 85  8 66 88 15  5 59 30 60 41 17 66 89 78 88 69 45 82  6 13
 90 27  1  8 91 80 89 49 32 28 90 93  6 35 73 47 43 75  8 51
  3 84 34 28 60 69 45 67 58 87 65 62 97 20 31 33 33 77 50 80
 48 90 75 96 44 28 21 51 75 17 89 59 56 63 18 17 30 16  7 35
 57 16 42 34 37 26 68 73  5  8 12 87 83 20 97 20 85 61  9 36
 63 11 45 10 33  5 41 47  9 74 33 35 78 12 22 44  8 97 10 86
 33 60 21 96 69 34 94 15 23 84 16 55 50  5 59 35 12 57 11 51
 72 42  4 62 15 27 16 34  8 50 85 12 48  5 25 40 81 46 67 25
 83 92 25 40 21  4 43 38 60 24  3 28 86 68 55 91 97 19 73 20
 28 81 46 98 46 29 96 12 71 32 64 39 16 97 99 49 75  7 79 80
 71  9 11  8  4 47 93 82  6 49  7 24 92 13 86 80 34 75 35 29
Machines
  7  2 16  3 20 14 17 19  4 13 15  6 11  8  9 10  1 18 12  5
  9  7 11 10 20  2  1 18  3 12  8 14 19 16  5 15  6  4 17 13
  2  4  5 13  1 15  8 20 11 12  6 19 14 16 17  9 10  3 18  7
 19 16 11  2 20 10  3 13  5  1 14 17 15 12  4  6  9 18  7  8
  3 11 13 16  8 15 18  6  9  2 14  1  4 17  5 10 12 19  7 20
  1 17 11 10 18  9  6  4 20 12 16  3 15  5 13 14  2  7  8 19
 12  1  2 20 10 11  9 18 19 17  5  4 13  8  7 16  6 15  3 14
 17  4  9 11 19  8  2 15 20  3 13  7  6 12 18  5 14 10 16  1
 18 19 16 12 11 13 15  4  2  1 17 10  7 20  8  9  5  3 14  6
 10  6  4  3 12  5 17 15  8 14  2 19  9 16 13  7 18 20  1 11
  6 10 19  1  4 17  7  2  3 12 16 20  8  9  5 11 13 18 15 14
 12  6 10  5  4 16 18  8 19 20 14  2  7 13 11  1 15 17  3  9
 16  4  5  8 19 14 20 10  3 11 12 15  7 18 13 17  1  9  2  6
 14 11 17  9  5  2  3 20 15 13 19 18  1  6 10  8  7  4 12 16
  1  6  5 20 15  9  2 17 16 11 13  3 12  7  8 19 14  4 18 10
 12 17 18 11 16 13  1  6  5  9  4  8 14  2 19 10 20  7 15  3
  8  9  6 14 20 17 12  2 15 19 11  7  4 16  5  3  1 10 18 13
 18 11 16  6  5 15  9 19 14  7 20  4  2 10 12  3 13 17  8  1
 15  4 17  1 11 19  5  2  6 20 13 10 12 16 14  8 18  9  7  3
 14 11  4 15  2  9  1 17 20  5  7 10 12  6  3  8 18 13 16 19
```

Figure 5. An example of Taillard's benchmark problem.

Fig. 5 shows an example of a job shop problem in Taillard's dataset. In this case, there are 20 jobs to be processed in 20 machines, with the upper bound (the worst makespan) = 2100, and the lower bound (the best makespan) = 1711. The data is followed by Times, which show the processing times of each job in each machine. Each row shows each job. Because there are 20 jobs, in this case, there are 20 rows in Times. The first row shows

the processing times of jobs 1 in all 20 machines. So, there are 20 columns in each row. Each column shows the processing time of job 1 in each machine. The next block of data is called Machines. Each row in this block shows the machine order for each job. Because there are 20 jobs in this case, then there are 20 rows in this block. The first line shows the machine order to process job 1: start from machine 7 - 2 - 16 - 3 - 20 - … -5. This block relates to the previous block, Times. It can be seen that the first job goes to machine 7 for 64 times, then continues to machine 2 for 57 times, etc. until the last machine, machine 5 for 94 times.

IV. Experiments

Fig. 6 shows an example of the program result to schedule 6 jobs in 4 machines. The makespan for this example is 13.

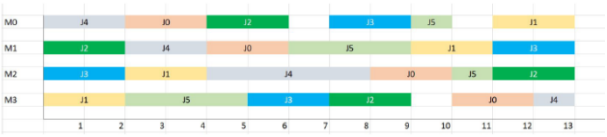

Figure 6. Program run to schedule 13 jobs in 4 machines.

Some experiments with different program parameters are conducted to find factors that affect the program results. These parameters are number iterations, number of mutations, and number of clones. Each experiment has a different number of jobs and machines (see Table II).

TABLE II. Processing Time Table of Figure 1

Experiment no.	Number of machines	Number of jobs
1	15	15
2	20	20
3	15	20

The results can be seen from Table IV - Table IX, and Fig. 7 – Fig. 17 show the chart representations of some of these experiments. The makespan results have been compared with the lower bound and the upper bound of Taillard's benchmark problems.

A. Experiment with Number of Iterations

In this experiment, different numbers of iterations were tested to see its effect on the results (see Table III).

TABLE III. Setting of Experiment 1

Table	Number of				
	Iterations	Clones	Mutations	Machines	Jobs
IV	1,000	10	10	15	15
V	10,000		10		
VI	1,000	10	10	20	20
VII	10,000	10	10	15	20
VIII	1,000				
IX	10,000				

TABLE IV. Experiment with 1,000 Iterations, 15 Machines, 15 Jobs

Case	Lower Bound	Upper Bound	Makespan
1	1005	1231	1507
2	953	1244	1589
3	1036	1222	1579
4	973	1181	1666
5	940	1233	1671

TABLE V. Experiment with 10,000 Iterations, 15 Machines, 15 Jobs

Case	**Lower Bound**	**Upper Bound**	**Makespan**
1	1005	1231	1494
2	953	1244	1531
3	1036	1222	1540
4	973	1181	1673
5	940	1233	1655

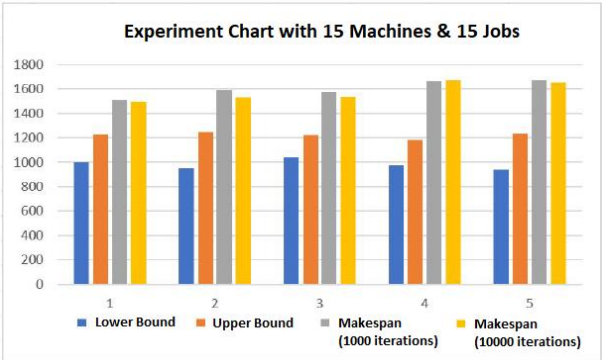

Figure 7. Experiment Chart with 15 Machines & 15 Jobs.

TABLE VI. Experiment with 1000 Iterations, 20 Machines, 20 Jobs

Case	Lower Bound	Upper Bound	Makespan
1	1217	1663	2120
2	1314	1626	2141
3	1248	1574	2018
4	1284	1660	2075
5	1256	1598	2031

TABLE VII. EXPERIMENT WITH 10000 ITERATIONS, 20 MACHINES, 20 JOBS

Case	Lower Bound	Upper Bound	Makespan
1	1217	1663	2084
2	1314	1626	2105
3	1248	1574	1983
4	1284	1660	2066
5	1256	1598	2017

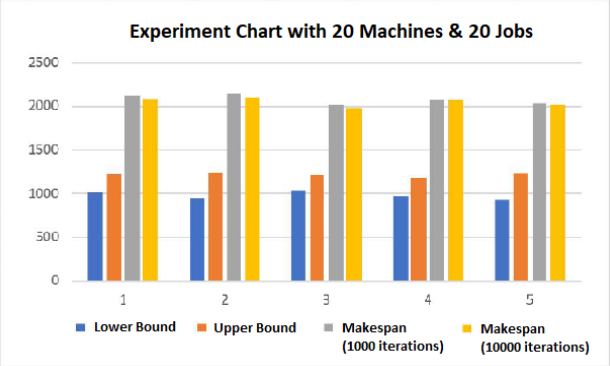

Figure 8. Experiment Chart with 20 Machines & 20 Jobs.

TABLE VIII. EXPERIMENT WITH 1000 ITERATIONS, 15 MACHINES, 20 JOBS

Case	Lower Bound	Upper Bound	Makespan
1	1254	1376	1835
2	1244	1267	1801
3	1243	1367	1785
4	1329	1345	1694
5	1163	1366	1784

TABLE IX. EXPERIMENT WITH 10000 ITERATIONS, 15 MACHINES, 20 JOBS

Case	Lower Bound	Upper Bound	Makespan
1	1254	1376	1780
2	1244	1267	1787
3	1243	1367	1739
4	1329	1345	1673
5	1163	1366	1759

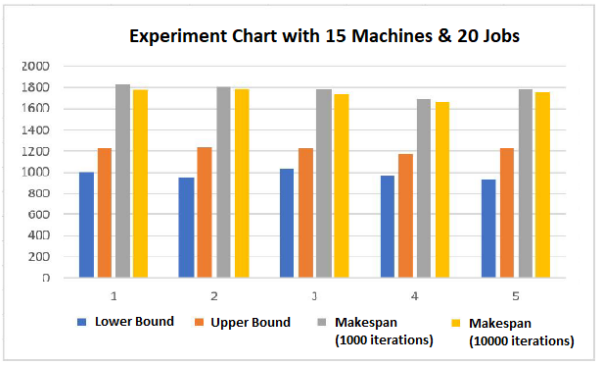

Figure 9. Experiment Chart with 20 Machines & 20 Jobs.

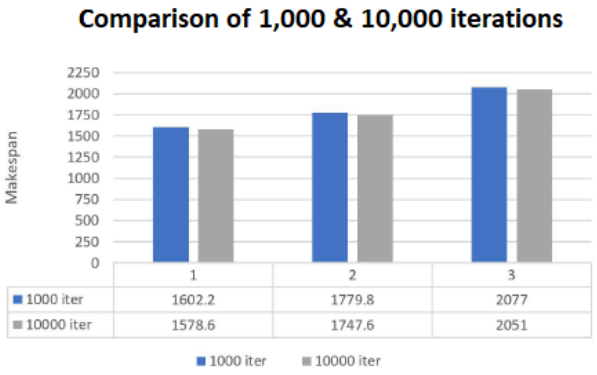

Figure 10. Comparison of 1,000 and 10,000 iterations

From these experiments, the larger size of iterations, the lower the makespan (see Fig. 10).

B. Experiment with Number of Mutations

In this experiment, different numbers of mutations were tested to see its effect on the results (see Table X). The results of these experiments can be seen in Fig. 11 – Fig. 13.

TABLE X. SETTING OF EXPERIMENT 2

Fig.	**Number of**				
	Mutations	**Clones**	**Iterations**	**Machines**	**Jobs**
11	10	10	10,000	15	15
	100				
12	10	10	10,000	20	20
	100				
13	10	10	10,000	15	20
	100				

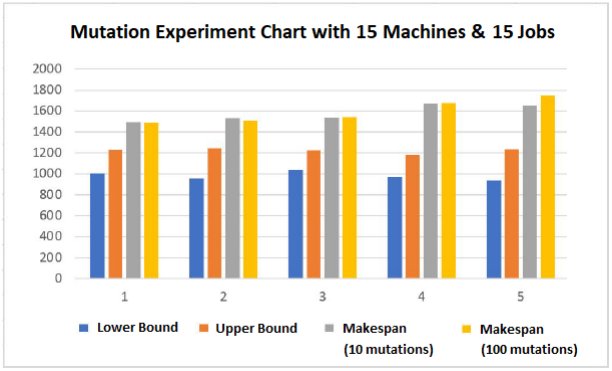

Figure 11. Mutation experiment chart with 15 machines & 15 Jobs

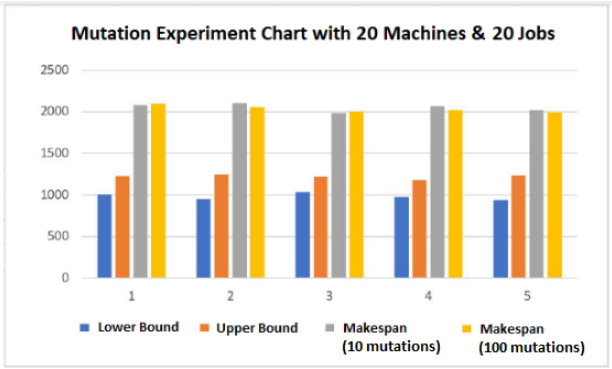

Figure 12. Mutation experiment chart with 20 machines & 20 Jobs.

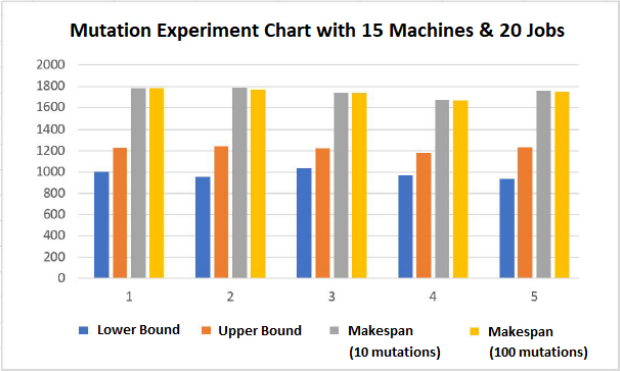

Figure 13. Mutation experiment chart with 15 machines & 20 Jobs

From these experiments, the larger size of mutations, the lower the makespan (see Fig. 14).

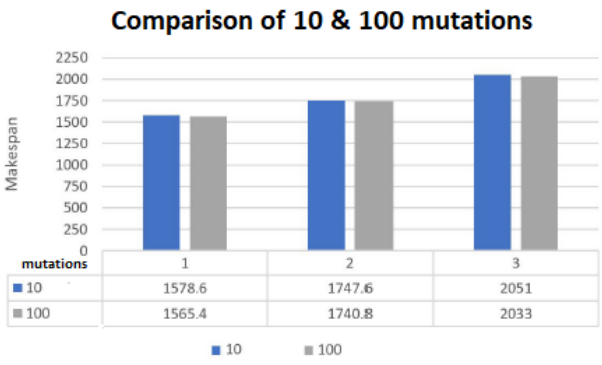

Figure 14. Comparison of 10 and 100 mutations

C. Experiment with Number of Clones

In this experiment, different numbers of mutations were tested to see its effect on the results (see Table XI). The results of these experiments can be seen in Fig. 15 – Fig. 17.

TABLE XI. SETTING OF EXPERIMENT 3

Fig.	Number of				
	Clones	Mutations	Iterations	Machines	Jobs
15	10	10	10,000	15	15
	100				
16	10	10	10,000	20	20
	100				
17	10	10	10,000	15	20
	100				

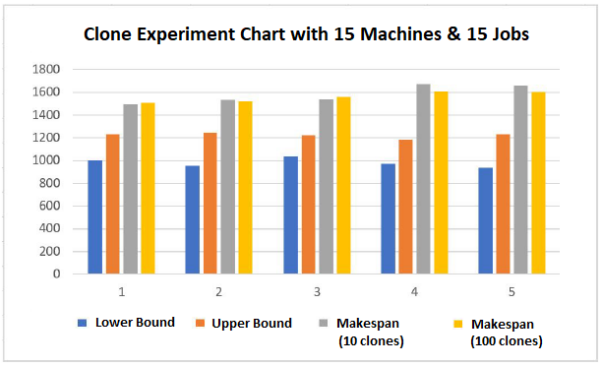

Figure 15. Clone experiment chart with 15 machines & 15 jobs.

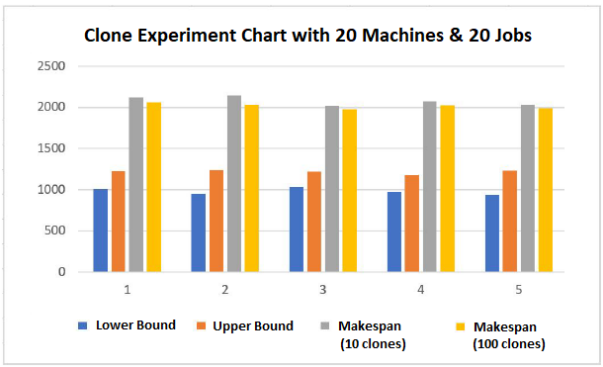

Figure 16. Clone experiment chart with 20 machines & 20 jobs

From the experiment results, it can be observed that the larger the number of clones, the lower the makespan will be (see Fig. 18).

The number of clones makes a larger makespan difference than the number of iterations and mutations. The larger number of clones and iterations increase significantly the processing time of the program. The change of mutation parameter doesn't make a significant

difference in the processing time but also results in a slightly different makespan.

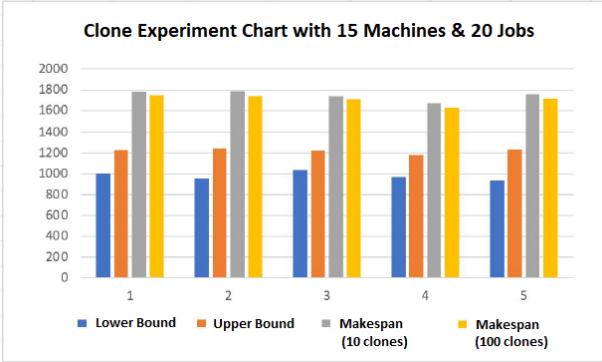

Figure 17. Clone experiment chart with 15 machines & 20 jobs

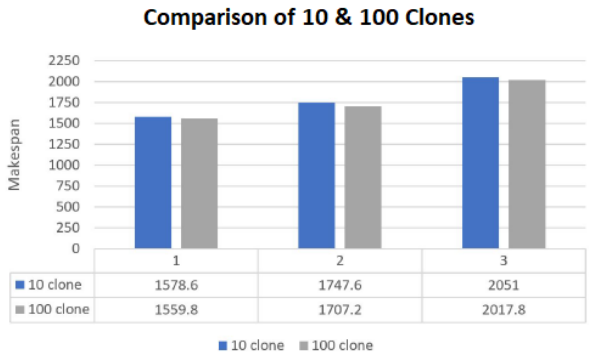

Figure 18. Comparison performance of 10 and 100 clones

V. Summary

From the experiment results, it can be observed that changing the program parameter, such as the number of iterations, mutation rate, and the number of clones can change the length of makespan. The results of the program are bigger than the upper bound of the Taillard benchmark dataset, which means that it is not as good as the benchmark data. This paper uses maximum 10,000 iterations, 100 mutation rate, and 100 clones, while Taillard uses minimum 10^6 iterations and the number of clone minimum10^3. The bigger the number of iterations, mutation rate, and the number of clones, the better the result for the clone selection algorithm. The clonal selection algorithm needs huge parameter setting values to perform better.

Acknowledgment

This research was supported by the Ministry of Research, Technology, and Higher Education of Republic Indonesia under the scheme of PTUPT Research Grant (2020) no.: III/LPPM/2020-04/105-PE-S.

References

[1] K. R. Baker and D. Trietsch, Principles of Sequencing and Scheduling, 1st edition. Wiley Publishing, Canada, 2009.
[2] C. E. Nugraheni and L. Abednego, "A tabu-search based constructive hyper-heuristics for scheduling problems in textile industry," *Journal of Industrial and Intelligent Information*, vol. 5 no. 2, 2017.
[3] C. E. Nugraheni and L. Abednego, "Multi agent hyper-heuristics based framework for production scheduling problem," in *Proc. International Conference on Informatics and Computing* (ICIC), , 2016, pp. 309-313.
[4] C. E. Nugraheni and L. Abednego, "On the development of hyper heuristics based framework for scheduling problems in textile industry," *International Journal of Modeling and Optimization*, vol. 6, no. 5, 2016.
[5] C. E. Nugraheni and L. Abednego, "A comparison of heuristics for scheduling problems in textile industry," *Jurnal Teknologi*, vol. 78, 2016.
[6] C. E. Nugraheni and L. Abednego, "A survey on heuristics for scheduling problem in textile industry," *Proceeding of ICEAI*, 2015.
[7] C. E. Nugraheni and L. Abednego, "A combined meta-heuristic with hyper-heuristic approach to single machine production scheduling problem," *International Journal of Computer and Information Engineering*, vol. 8, no. 8, 2014.
[8] C. E. Nugraheni and L. Abednego, "Collaboration of multi-agent and hyper-heuristics systems for production scheduling problems," *International Journal of Computer and Information Engineering*, vol. 7, no. 8, 2013.
[9] D. Grimes and E. D. Hebrard, Models and Strategies for Variants of the Job Shop Scheduling Problem, 2011.
[10] J. Timmis, T. Knight, L. D. Castro, and E. Hart, "An overview of artificial immune systems," *An Overview of Artificial Immune Systems*, 2004.
[11] S. Al-Sharhan, "Artificial immune systems as models, algorithms, and applications," *International Journal of Research and Reviews in Applied Sciences*, 2010.
[12] C. A. C. Coello, D. C. Rivera, and N. C. Cortés, "Use of an artificial immune system for job shop scheduling," in *Artificial Immune Systems*, E. J. D. Bentley, J. Peter, and Hart, Eds., Berlin Heidelberg, Springer, 2003, pp. 1–10.
[13] B. Cai, S. Wang, H. Hu, "Hybrid artificial immune system for job shop scheduling problem," *International Journal of Economics and Management Engineering*, vol. 5, pp. 1606–1611, 2011.
[14] E. Taillard, "Benchmark for basic scheduling," *European Journal of Operational Research*, vol. 264, pp. 278–285, 1993.

Steven Kosasih, S. Kom. received his bachelor degree from Dept. of Informatics, Parahyangan Catholic University, Bandung, Indonesia. His research interest includes optimazition and intelligent systems.

Dr. Cecilia E. Nugraheni received her bachelor degree (1993) and master degree (1995) from Dept. of Informatic Engineering, Bandung Institute of Technology (ITB), Bandung, Indonesia. She has received PhD Degree (2004) from Dept. of Informatics, Ludwig Maximilians Universität, Munich, Germany. Her research interest includes formal methods, intelligent systems, machine learning, meta-heuristic and hyper-heuristic techniques.

Luciana Abednego, M. T. received her bachelor degree from Dept. of Informatics, Parahyangan Catholic University, Bandung, Indonesia. She has done her Master in Informatics from Bandung Institut of Technology, Bandung, Indonesia. Currenty, she is working as a lecturer at the Dept. of Informatics, Parahyangan Catholic University. Her research interest includes machine learning and intelligent systems.

