Journal of Information Technology and Computer Science
Volume 4, Number 2, September 2019, pp. 166-176
Journal Homepage: www.jitecs.ub.ac.id

Hybrid Real-Coded Genetic Algorithm and Variable Neighborhood Search for Optimization of Product Storage

Nindynar Rikatsih[1], Wayan Firdaus Mahmudy[2], Syafrial[3]

[1,2]Faculty of Computer Science, Brawijaya University, Indonesia
[3]Faculty of Agriculture, Brawijaya University, Indonesia

Received 20 May 2019; accepted 29 September 2019

Abstract. Agricultural product storage has a problem that need to be noticed because it has an impact in gaining the profit according to the number of products and the capacity of storage. Inappropriate combination of product causes high expenses and low profit. To solve the problem, we propose genetic algorithm (GA) as the optimization method. Although GA is good enough to solve the problem, GA not always gives an optimum result in complex search spaces because it is easy to be trapped in local optimum. Therefore, we present a hybrid real-coded genetic algorithm and Variable Neighborhood Search (HRCGA-VNS) to solve the problem. VNS is applied after reproduction process of GA to repair the offspring and improve GA exploitation capabilities in local area to get better result. The test results show that the optimal *popsize* of GA is 180, number of generations is 80, combination of *cr* and *mr* is 0.7 and 0.3 while optimum K_{max} of VNS is 40 with number of iterations 50. Even though HRCGA-VNS need longer computational time, HRCGA-VNS has proven to provide a better result based on higher fitness value compared with classical GA and VNS.

1. Introduction

Product storage is one of the knapsack problems that is needed to be noticed. Knapsack problem is a well-known NP-hard combinatorial optimization problem that is often encountered in many application [1], [2]. Various kind of knapsack problems have been applied such as budgeting project, cargo loading, bin packing, selection of items, material, cost-effective development, etc. [3], [4]. The aim of doing product storage is to maximize number of items in the storage which have the highest total profit with an appropriate capacity so that the items do not exceed the capacity of storage.

Many method has been proved that can be used to solve knapsack problems such as local search, heuristics method and hybridization methods [1]. Genetic algorithm (GA) is easy to use and a simple method. It also has a wide search area and has been proved that can be used in many applications [5], [6], [7], [8]. GA also can be applied to solve knapsack problem [3], [4]. It is because GA is a heuristic method with population-based solution that can gives many possible solutions. So, in this research we propose GA as the basic method to solve the knapsack problem.

GA is one of the heuristic methods which delivers optimal solutions by implementing the natural selection and the mechanism of biological evolution [9]. There are several steps of GA that consists of (i) initializing the population (ii) producing new solutions by reproduction process that consists of crossover and

mutation (iii) doing the selection to get the best result [10].

GA is good enough to solve the optimization problem [8]. However, GA not always gives an optimum result in complex search spaces. In finding the optimum solution, hybridization can improve the accuracy and efficiency of its performance [11]. GA in its implementation also does not always fulfill the expectation because of its inability to increase the variety of solution that makes it being trapped in local optimum and causing premature convergence [12]. GA is difficult to get out of local optimum when the genetic operators does not produce offspring with the higher quality then the parent [13]. Many researches have been done to deal with premature convergence in GA. Several researches combine GA and local search. This approach balances GA exploration capability in global area and local search exploitation capabilities in local area [14]. The other research use hybrid GA to improve GA performance such as combining GA with Simulated Annealing, Tabu Search and other local search-based algorithms. Moreover, doing random injection also can be used to avoid premature convergence in GA [15], [16].

We propose hybrid GA and VNS with real-coded representation in order to fix GA weakness. VNS can improve exploitation capabilities in local area [17] so we set VNS in the evaluation process to increase variety of solution and get the better results. VNS performs local searches by moving from one solution to another until it reaches the optimum local solution. Furthermore, VNS changes the structure of the environment so that it can find a better solution. VNS is applied to repair selected individuals at certain intervals in the cycle of Genetic Algorithms. Therefore, in this study we propose hybrid GA-VNS to solve knapsack problem of agricultural product storage.

2. Problem Description

This research addresses the optimization of agricultural product storage by determining appropriate combination of product quantities to gain maximum profit when the product is sold by the trader to the customer. Every product has purchase and selling cost that are used to calculate the revenue. The product is stored in storage container. This makes there are several things from storage container that has to be noticed to gain profit from storage process. It consists of expenses for warehouse maintenance and labor cost.

Trader should minimize number of items in the storage and the total profit without exceed the capacity of storage. Trader should combine the number of each product to know the most valuable combination in gaining profit among other combinations. An example of agricultural product storage is showed in table 1.

Table 1 Combination of product storage

No.	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6	Product 7	Product 8
1	100	150	100	50	80	55	120	200
2	150	100	200	80	1000	100	80	120
3	200	150	300	120	250	100	400	180
4	50	80	100	1000	250	100	200	350
5	1000	2000	50	80	100	120	2000	150

Table 1 shows the combination of product quantities in kilogram. There are five combinations where first combination shows the quantity of product 1 is 100 kg, product 2 is 150 kg, product 3 is 100 kg and so on. Each combination gives different

profit. Trader should find the appropriate combination but there is no actual method to determine the best combination. Nowadays, the only way trader can do is only predicting the combination by their experience so it does not give the appropriate result that affects their profit.

3. Model Formulation

To solve that problem, the formulation is applied to calculate the profit from the combination of product. The profit is gained from total revenue and total cost where to get high profit, total revenue (TR) should be higher than total cost (TC). The formulation of total cost has shown in (1) while the formulation of total revenue is in (2) and formulation to get total profit has shown in (3).

$$TC = (\sum_{i=1}^{n} Px_{1i}Q_i) + Px_2 + Px_3 \quad (1)$$

$$TR = \sum_{i=1}^{n} Py_iQ_i \quad (2)$$

$$TPF = TR - TC \quad (3)$$

Parameters of the formulations are defined as follows:
TC = total cost
TR = total revenue
TPF = total profit
i = product identity
Px_{1i} = purchase cost of each product
Px_2 = cost of warehouse maintenance
Px_3 = labor cost
Py_i = selling cost of each product
Q_i = the quantity of product

The example of profit calculation is applied in Table 2. We choose combination 1 from Table 1 as the example of profit calculation. From Table 2 we calculate profit (TPF) by looking for the deviation between total revenue (TR) and total cost (TC). Total revenue is total income obtained from the total selling price of all products while total cost is gained from total purchase cost of each product and expenses of warehouse and labor. From Table 2 we know that the profit gained from combination 1 is 200000. It is not the best result because the other combination can give higher profit.

Table 2 Example of profit calculation

Product	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6	Product 7	Product 8
Product quantity	100	150	100	50	80	55	120	200
purchase cost per Kilogram	3000	2000	1000	5000	2500	3000	3500	1500
Purchase cost of each product	300000	300000	100000	250000	200000	165000	420000	300000
Expenses of warehouse and labor	120000							
TC	2035000+320000 = 2355000							
Selling cost per kilogram	3500	2500	1500	6000	3500	4000	4000	2000
Selling cost of each product	350000	375000	150000	300000	280000	220000	480000	400000
TR	2555000							
TPF	200000							

Table 2 shows that the profit gained from combination without considering the capacity of storage where the total of product quantity may not to exceed storage capacity. The relation between product quantity and storage capacity is showed in (4) and (5).

$$max \ = \sum_{i=1}^{n} b_i \ . \ X_i \tag{4}$$

$$where \ \sum_{i=1}^{n} w_i \ . \ X_i \ \leq \ W_i \tag{5}$$

The aim of (4) is to maximize the quantity of product in storage where *i* represent each product w_i is weight, b_i is value of the product and X_i total quantity of all products. Formula (5) shows that total quantity of all product may not to exceed the capacity of storage. Hence, there should be an optimization method to solve that kind of problem.

4. Proposed Method

4.1 Genetic Algorithm

GA is one of the heuristic methods which delivers optimal solutions by implementing the natural selection and the mechanism of biological evolution [9]. There are several steps of GA that consists of (i) initializing the population (ii) producing new solutions by reproduction process that consists of crossover and mutation (iii) doing the selection to get the best result [10].

4.2 Variable Neighborhood Search

VNS was first proposed in 1997 as a metaheuristic method by Mladenovic and Hansen [18]. VNS is successfully applied in several research such as scheduling problems [19], [20], optimization of high-performance concrete structure [21]. Machine

loading problems [22] etc. Several steps are applied in VNS procedure that starts from shaking, then doing local search and the last is movement phase (move or not) [23]. VNS is simple because it uses only two parameters that consist of the number of neighborhood and termination condition [21], [22].

4.3 Hybrid RCGA-VNS

We proposed a classical GA and VNS to solve knapsack problem in product storage. By considering GA exploration capability and the exploitation ability of VNS, we present hybrid GA-VNS in this section. This approach is purposed to give a solution as a recommendation of a set of items that will be kept in the storage. This method uses GA as the main algorithm and runs VNS to improve individual in population. More details of proposed method are described in the following subsection. The steps of proposed method are shown in Figure 1.

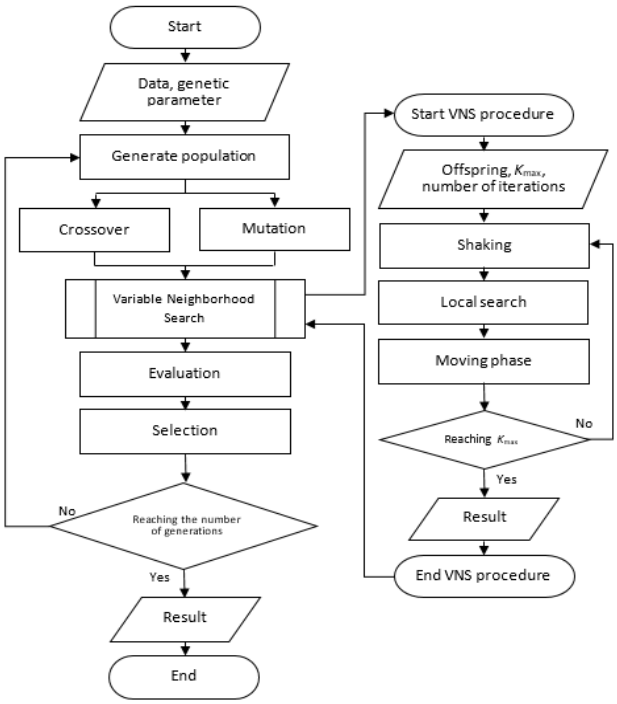

Fig. 1. Scheme of GA-VNS

The hybridization process begins by putting data and genetic parameters. Next, an initialization of the population consists of chromosome representation and calculation of fitness values. The population consists of a set of individuals represented by a number of chromosomes. Chromosomes are generated randomly as many as population size (*popsize*) [10]. Each chromosome represents a solution. This research uses real-coded representation. Each gen in chromosome represents quantity of the product. On each chromosome, fitness values are calculated to determine the quality of the solutions [24]. Fitness function of product storage optimization has shown in (6).

$$fitness = TPF - (overcapacity \times 1000) \quad (6)$$

The example of chromosome representation and fitness value has shown in Figure 2.

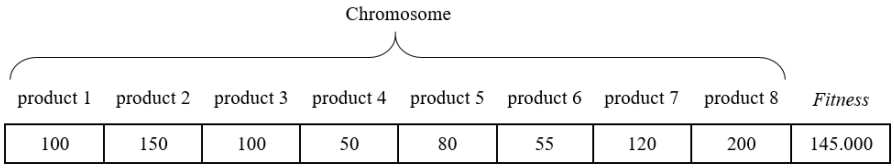

Fig. 2. Chromosome representation

Reproduction process has two genetic operators that consist of crossover and mutation. It is carried out to get offspring from individuals in the population. Crossover is part of the reproductive process in genetic algorithms that requires some strategy to select two parents from previous generation. These two chromosomes are crossed to produce new chromosome members [25]. Mutation is also an important part of genetic algorithms where there are few modifications randomly executed in each chosen chromosomes [26] to gain new chromosome variations. The process of crossover and mutation will produce a combination of various features in order to reach different directions in the search space [27]. The parameter used in the crossover operator is the crossover-rate (*cr*) while the parameter in mutation operator is the mutation-rate (*mr*). *Cr* and *mr* are used to determine the number of new chromosomes with $cr \times popsize$ to produce offspring from the crossover process and $mr \times popsize$ to get the offspring from the mutation process [5]. Reproduction methods used in this research are one-cut point crossover that is illustrated in Figure 3 and insertion mutation shown in Figure 4.

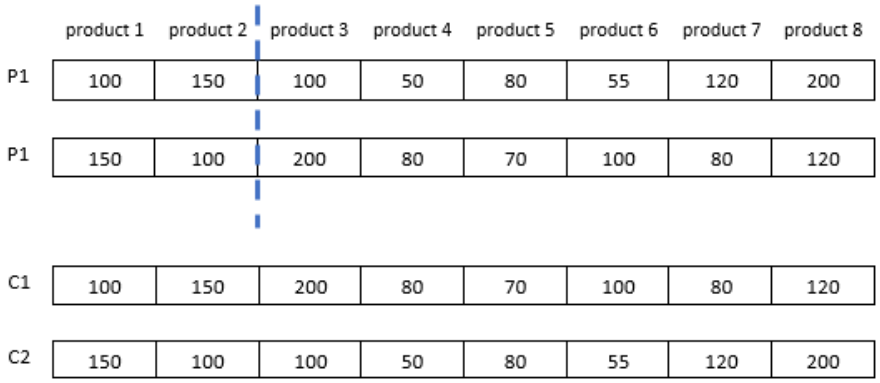

Fig. 3. Crossover

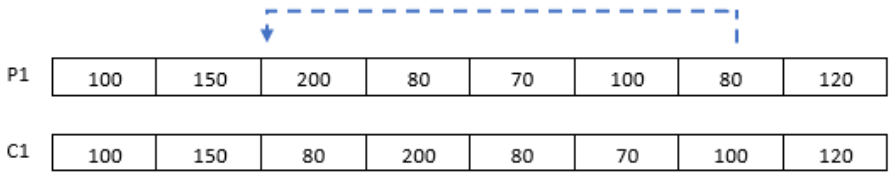

Fig. 4. Mutation

After that both the offspring and the parent chromosome will be evaluated for further improvement using the VNS method. After repairing these chromosomes, they will enter the VNS process. In the first step of VNS procedure, we set K_{max} and the process will start from K=1 until it reaches K_{max}. In shaking step, we randomly generate new solution variety from offspring that we got from crossover and mutation process. In the next step, local search method is applied by searching a new candidate solution from result that we got from shaking process as initial solution. The local search is done iteratively as many as iteration number of local searches. After it got a new solution, it enters the moving phase that will decides whether it will replace the incumbent solution or not. If it has better fitness value, then it will move to incumbent and it is not, the incumbent will stay.

Selection is done to select individuals from a set of population that consist of parent chromosomes and the offspring from reproduction process. In the selection process, the next number of chromosomes is selected based on the fitness value as many as the number of population sizes [10].

5. Computational Experiment

This section shows the experimental result of proposed method that consist of GA, VNS and hybrid GA-VNS parameter testing. First, we did genetic parameter testing of GA that is divided into three testing (i) *popsize* testing which the result is showed in Figure 5 (ii) *cr* and *mr* combination testing that is showed in Figure 6 (iii) generation number testing that is in Figure 7. The second is testing of VNS procedure that consist of two operator that are K_{max} which is showed in Figure 8 and number of iterations in local search phase that showed in Figure 9.

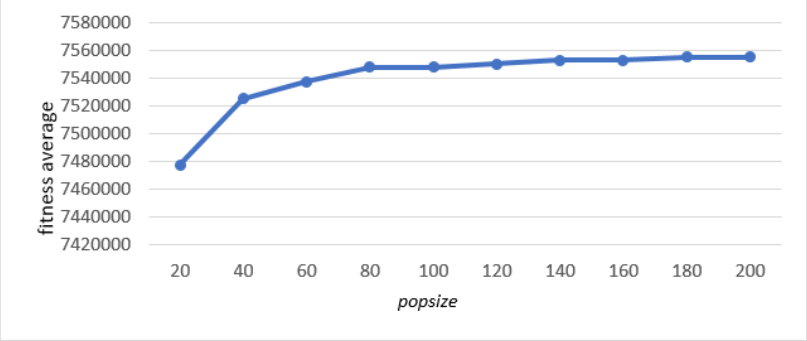

Fig. 5. *Popsize* Testing

Popsize testing starts from 20 population as the lowest *popsize* with average fitness value 7477500. Figure 5 shows that the best popsize of classical GA for solving this problem is 180 with average fitness value 7555000. The bigger population size may not necessarily produce a better solution. This can be seen from the testing above 180 population that does not provide a significant increment of fitness value average.

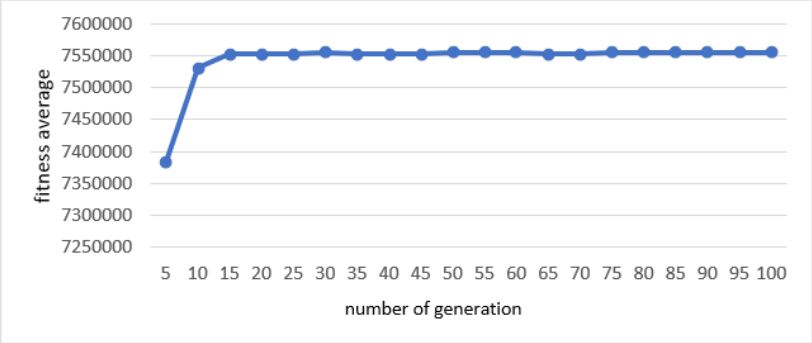

Fig. 6. Generations Number Testing

Generations number testing start from 6 generations as the lowest *popsize* with average fitness value 7383500 and continue with the range of 5 generations. Figure 3 shows that the optimum number of generations is 80 with average fitness value 7555000. The result shows that after 80 generations there is no significant increment of fitness value average.

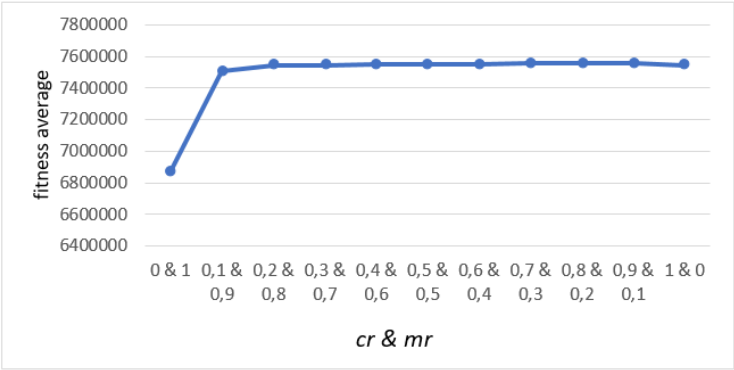

Fig. 7. Combination of *cr* and *mr* Testing

Based on Figure 7, the highest point of *cr* and *mr* combination is on the combination of 0.7 and 0.3 with an average fitness value of 7555000. In solving this problem, *cr* value affects more than *mr* value. It is showed by the lowest fitness value average 6905000 with combination of *cr* and *mr* 0 and 1.

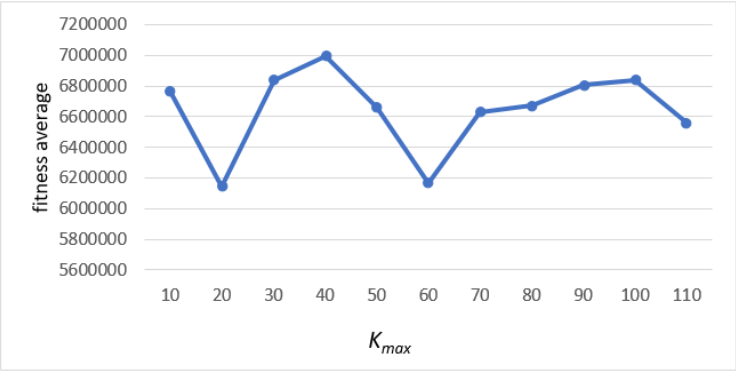

Fig. 8. K_{max} Testing

Based on Figure 8 the best K_{max} is found in K of 40 with fitness value average 6992500. The bigger K value does not necessarily provide better result than the previous K value. It is indicated by a lower fitness value average after 40 K that does not give a better result.

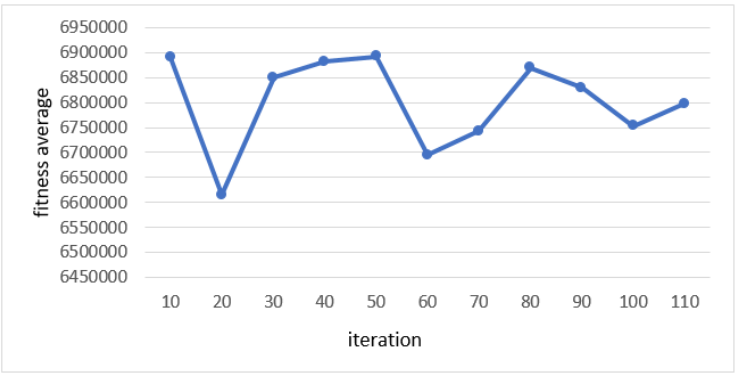

Fig. 9. Iteration Testing

The optimum fitness value average has found in 50 iterations with fitness value average 6892500. More iteration does not necessarily produce better result. Figure 9 shows that in the 60 generations there is lower fitness value average that is 6695000.

After classical GA and VNS testing, we test parameter of hybrid GA-VNS by doing the same thing as we did in GA and VNS testing. The comparation of GA, VNS and hybrid GA-VNS is performed in Table 3. From Table 3 it is found that Hybrid GA-VNS can be used to solve the problem by giving the best result among the three method. Solution of Hybrid GA-VNS is better than two other methods by reaching average of fitness value 7555000 in its best parameters while GA reaches average of fitness value 7547500 and VNS 6615000. However, the best computational time refers to GA.

Table 3. Comparation of GA, VNS and hybrid GA-VNS

GA		VNS		Hybrid GA-VNS	
Fitness Average	Computational Time	Fitness Average	Computational Time	Fitness Average	Computational Time
7547500	364.4 ms	6615000	437.7 ms	7555000	2212 ms

6. Conclusion and Future Work

Based on the experiment that has been done, total profit of product storage can be gained by classical GA and VNS. It still does not reach the optimum profit because it can be gained by hybrid GA-VNS with higher total profit of product storage showed by fitness value that is higher than classical GA and VNS as shown in Table 3. Therefore, we can conclude that the hybridization of GA and VNS can increase local search capability of VNS and global search capability of GA to increase efficiency of solutions searching. The result shows that even hybrid GA-VNS need more computational time, it performs better by gaining a better fitness value than GA and VNS itself.

Future work, usage of crossover and mutation rate can be modified as adaptive. It can change *cr* and *mr* adaptively so the individual variety is not restricted to one combination of *cr* and *mr*. GA also can be modified in the part of initializing population. It can be done by distribute population into several part to keep the individual diversity.

References

1. P. . Chu and J. . Beasley, "A Genetic Algorithm for the Multidimensional Knapsack Problem," *J. Heuristics*, vol. 86, no. 4, pp. 63–86, 1998.
2. J. Puchinger, G. R. Raidl, and U. Pferschy, "The Core Concept for the Multidimensional Knapsack Problem," *Eur. Conf. Evol. Comput. Comb. Optim.*, pp. 195–208, 2006.
3. M. Gupta, "A Fast and Efficient Genetic Algorithm to Solve 0-1 Knapsack Problem," *Int. J. Digit. Appl. Contemp. Res.*, vol. 1, no. 6, 2013.
4. V. Yadav and S. Singh, "Genetic Algorithms Based Approach to Solve 0-1 Knapsack Problem Optimization Problem," *Int. J. Innov. Res. Comput. Commun. Eng.*, pp. 8595–8602, 2016.
5. W. F. Mahmudy, R. M. Marian, and L. H. S. Luong, "Real Coded Genetic Algorithms for Solving Flexible Job-Shop Scheduling Problem - Part I: Modelling," *Adv. Mater. Res.*, vol. 701, pp. 359–363, 2013.
6. V. I. Herrera, H. Gaztañaga, A. Milo, A. Saez-de-ibarra, and T. Nieva, "Optimal Energy Management of a Battery- Supercapacitor based Light Rail Vehicle using Genetic Algorithms," *IEEE Energy Convers. Congr. Expo.*, pp. 1359–1366, 2015.
7. G. Arunkumar, I. Gnanambal, S. Naresh, P. C. Karthik, and J. K. Patra, "Parameter Optimization of Three Phase Boost Inverter Using Genetic Algorithm for Linear Loads," *Energy Procedia*, vol. 90, no. December 2015, pp. 559–565, 2016.
8. Y. Amini, M. B. Gerdroodbary, M. R. Pishvaie, R. Moradi, and S. M. Monfared, "Optimal Control of Batch Cooling Crystallizers by Using Genetic Algorithm," *Case Stud. Therm. Eng.*, vol. 8, pp. 300–310, 2016.
9. Y. Yan Song & Guoxing, "A Genetic Algorithm of Test Paper Generation," no. Iccse, pp. 897–901, 2013.
10. A. Rahmi and W. F. Mahmudy, "Regression Modelling for Precipitation Prediction Using Genetic Algorithms," *TELKOMNIKA*, vol. 15, no. 3, pp. 1290–1300, 2017.

11. W. F. Mahmudy, R. M. Marian, and L. H. S. Luong, "Hybrid Genetic Algorithms for Part Type Selection and Machine Loading Problems with Alternative Production Plans in Flexible Manufacturing System," vol. 8, no. 1, pp. 80–93, 2014.
12. A. Rahmi, W. F. Mahmudy, and S. Anam, "A Crossover in Simulated Annealing for Population Initialization of Genetic Algorithm to Optimize the Distribution Cost," *Accept. to Int. J. Intell. Eng. Syst.*, vol. 9, no. 2, pp. 177–182, 2016.
13. C. Science and S. Engineering, "Preventing Premature Convergence in Genetic Algorithm Using DGCA and Elitist Technique," vol. 4, no. 6, pp. 410–418, 2014.
14. M. Lozano, F. Herrera, N. Krasnogor, and D. Molina, "Real-Coded Memetic Algorithms with Crossover Hill-Climbing Real-Coded Memetic Algorithms with Crossover Hill-Climbing," *Evol. Comput.*, vol. 12, no. 3, pp. 273–302, 2004.
15. W. F. Mahmudy, R. M. Marian, and L. H. S. Luong, "Optimization of Part Type Selection and Loading Problem with Alternative Production Plans in Flexible Manufacturing System using Hybrid Genetic Algorithms-Part 1 : Modelling and Representation," *5th Int. Conf. Knowl. Smart Technol.*, pp. 75–80, 2013.
16. W. F. Mahmudy, R. M. Marian, and L. H. S. Luong, "Optimization of Part Type Selection and Loading Problem with Alternative Production Plans in Flexible Manufacturing System using Hybrid Genetic Algorithms – Part 2 : Genetic Operators and Results," *Int. Conf. Knowl. Smart Technol.*, pp. 81–85, 2013.
17. N. Mladenovic, D. Urosevic, and D. Perez-Brit, "Variable Neighborhood Search for Minimum Linear Arrangement Problem," *Yugosl. J. Oper. Res. 26*, vol. 26, no. 1, pp. 3–16, 2016.
18. N. Mladenovic and P. Hansen, "Variable Neighborhood Search," *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, 1997.
19. C. Exp and J. M. Moreno-vega, "Variable Neighbourhood Search for the Quay Crane Scheduling Problem," pp. 463–468, 2011.
20. R. A. Aziz, M. Ayob, and Z. Othman, "The Effect of Learning Mechanism in Variables Neighborhood Search," *2012 4th Conf. Data Min. Optim.*, no. September, pp. 109–113, 2012.
21. C. Torres-Machi, V. Yepes, J. Alcala, and E. Pellicer, "Optimization of high-performance concrete structures by variable neighborhood search," *Int. J. Civ. Eng.*, vol. 11, no. 2, 2013.
22. W. F. Mahmudy, "Optimization of Part Type Selection and Machine Loading Problems in Flexible Manufacturing System Using Variable Neighborhood Search," *Int. J. Comput. Sci.*, 2015.
23. P. Hansen and N. Mladenovi, "Variable neighborhood search : Principles and applications c," *Eur. J. Oper. Res.*, vol. 130, no. 3, pp. 449–467, 2001.
24. P. Smirnov, M. Melnik, and D. Nasonov, "Performance-aware scheduling of streaming applications using genetic algorithm," *Procedia Comput. Sci.*, vol. 108, no. June, pp. 2240–2249, 2017.
25. H. D. Mathias and V. R. Ragusa, "An Empirical Study of Crossover and Mass Extinction in a Genetic Algorithm for Pathfinding in a Continuous Environment," *IEEE Congr. Evol. Comput.*, pp. 4111–4118, 2016.
26. A. H. Beg and M. Z. Islam, "Novel Crossover and Mutation Operation in Genetic Algorithm for Clustering," *IEEE Congr. Evol. Comput.*, pp. 2114–2121, 2016.
27. M. Z. Sarwani, A. Rahmi, and W. F. Mahmudy, "An Adaptive Genetic Algorithm for Cost Optimization of Multi-Stage Supply Chain," *J. Telecommun. Electron. Comput. Eng.*, vol. 9, no. 2, pp. 155–160, 2017.