Optimization of Part Type Selection and Machine Loading Problems in Flexible Manufacturing System Using Variable Neighborhood Search

Wayan Firdaus Mahmudy, *Member, IAENG*

***Abstract*—This paper addresses optimization of the integrated part type selection problem and machine loading problem that are considered as NP-hard problems in production planning of flexible manufacturing system (FMS) and strongly determine the system's efficiency and productivity. The integrated problems are modelled and solved simultaneously by using Variable Neighbourhood Search (VNS). A new neighbourhood structure is designed to enable the VNS produces near optimum solutions in a reasonable amount of time. The proposed VNS improves the FMS performance by considering two objectives, maximizing system throughput and maintaining the balance of the system. The resulted objective values are compared to the optimum values produced by branch-and-bound method. The numerical experiments show the effectiveness of the proposed VNS for several test-bed problems.**

***Index Terms*—Flexible manufacturing system, production planning, part type selection problem, machine loading problem, variable neighborhood search**

I. Introduction

FLEXIBLE manufacturing system (FMS) is designed to address rapid changing customer needs on various high quality products. Flexibility is the main feature of FMS where a variety of products in low to medium volumes can be produced by using computer numerically controlled (CNC) machines and automatic transfer lines [1]. However, building such systems using high technology equipment requires a high initial investment. To enable early return on investment, an optimum resources utilization and system productivity must be achieved. These objectives can be achieved by establishing a good production planning [2] and a good scheduling to minimize energy consumption that lead to production cost reduction [3].

As a tool to survive in hard competition of global market, FMS are implemented in many manufacturing areas. The areas include defense industries, aerospace, metal-cutting machining, metal forming, various automotive part, and plastic injection molding [4].

The flexibility of FMS may be used to optimize the utilization of the production resources and also reduce the production time [5, 6]. The flexibility of FMS refers to its ability to produce various products by using same resources (machines and tools). There are two categories of FMS flexibility which may be divided into several sub categories. The first is *machine flexibility*. By using high technology, machines setting can be reconfigured easily to produce new type of products for different market segments [7]. The second is *routing flexibility* that means one product may be produced by a number of alternative machining sequences. This flexibility enables manufacturers to increase and balance their machines utilization and decrease processing time [5].

There are several sub-problems in the production planning of FMS such as part type selection problem, machine grouping problem, production ratio problem, resource allocation problem, and machine loading problem [8]. However, due to the specific FMS environments, not all the sub-problems simultaneously exist.

This study considers a plant that manufactures various models of products. Various types of components are assembled to build a product. For generality, the component is called a part type. The various part types are produced by using a same machines set called FMS. A process called *aggregate production planning* produces a master planning contains the number of each model of product that must be manufactured. Each model of product requires several part types. Thus, the quantity of parts types that must be produced can be calculated.

As the FMS have technological constraints such as limited machines availability, limited tool magazines capacity of each machine, and limited number of tools, the part types must be produced in several batches. This decision process is called the *part type selection*. Specifically, the part type selection is concerned with selection of a set of part types (products) from a number of part types in the production order into a production batch. The selected part types in the batch will be manufactured immediately.

The next production step is called the *machine loading* problem that deals with allocation of operations for the selected part types and loading required tool types to the machines magazine. The next stage is *scheduling* that determine order of operations of the selected part types in the machines. The scheduling also determines the starting time of each operation in the machine. To perform the final product, a process called *assembly operations* is carried out

Manuscript received June 30, 2014; revised May 11, 2015.

W. F. Mahmudy is with Faculty of Computer Science, Universitas Brawijaya (UB), Indonesia (email: wayanfm@ub.ac.id).

to join related part types.

The overall stages in FMS environment are depicted in Fig. 1. Problems addressed in this paper are shown in grey areas. The scheduling and assembly operations problems can be solved after solutions of the part type selection and machine loading problems are obtained.

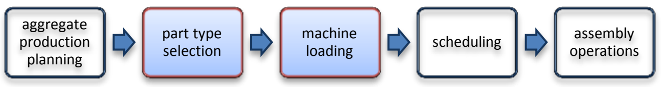

Fig. 1. Manufacturing processes

Even if the part type selection and machine loading problems can be solved hierarchically in separated stages, solving them simultaneously will produce better solutions that are indicated by higher throughput and balance machines' workload [9]. Moreover, a solution produced by the part type selection in the previous stage may become infeasible for the machine loading the next stage [4]. Thus, this paper focuses on the integrated part type selection and machine loading problems.

The part type selection problem and the machine loading problem are strongly related problems and exist in most FMS environments. The problems also heavily determine the system's efficiency [10]. Higher throughput and efficient allocation of production resources of the FMS will be achieved by simultaneously solving part type selection and machine loading problems [11].

The part type selection and machine loading problems are considered as strongly NP-hard problems with a very large search space and the optimum solution may not be obtained by exact methods or complete enumeration in a reasonable amount of time [7]. For a medium size problem with 36 part types and the average possible machining routes of 5, the total number of possible solutions for the integrated part type selection and machine loading problems is about 5.4×10^{66}. For this kind of problem, a branch-and-bound method run on personal computer equipped with Intel® Core™ i3-380 processor required more than 150 hours to get the optimum solution [4]. This computational time cannot be accepted for daily operation of FMS. Thus, a good approach to achieve near optimum solutions on reasonable amount of time is required. While Mahmudy, et al. [12] proved that their real-coded genetic algorithm (RCGA) could effectively exploring a huge search space of the problems, a series of experiments was required to obtain the best parameters of the approach. In this paper, an efficient variable neighbourhood search (VNS) is proposed.

Variable neighbourhood search (VNS) is meta-heuristic technique that manages a local search (LS) technique. Here, the LS is systematically iterated to explore larger neighbourhood until termination condition is achieved. The neighbourhood structure is designed to enable the LS exploring the search space from new starting points [13]. Thus, these properties enable the VNS to escape from local optimal areas and obtain optimum or near optimum solution. As a simple and effective method, the VNS have been successfully implemented to solve a various combinatorial problems such as maximum satisfiability problem [14], job scheduling [15], location routing problem with capacitated depots [16], transportation and distribution [17], bin packing problem [18], and a number variants of travelling salesman problem [17, 19, 20].

To get more powerful VNS, the VNS is improved in various ways. For example, Parallel Variable Neighborhood Search (PVNS) was developed by running several instances of VNS on different processors [21]. The study tested various levels of parallelization strategies. The low level was developed by parallelizing the local search of neighborhoods. In the high level, parallelizing was done by running several VNSs and equipping a mechanism of cooperation (exchanging of information) among VNS instances. Parallel VNS was also developed by Eskandarpour, et al. [22] to solve the multi-objective sustainable post-sales network design problem. As previous work, the parallelization was done by running several instances of VNS to find better Pareto optimum points. Moreover, parallelizing the local search of neighborhoods in the sequential VNS was done in the study of S´anchez-Oro, et al. [23].

The VNS is also hybridized with other algorithms in several studies. For example, Li, et al. [24] combined the VNS with the chemical-reaction optimization (CRO) and the estimation of distribution (EDA) to solve the hybrid flow shop scheduling problem. Moreover, the VNS was hybridized with the greedy randomized adaptive search procedures to solve the targeted offers problem in direct marketing campaigns [25].

This paper attempts to develop a new neighbourhood structure for the VNS to effectively explore a large search space of the integrated part type selection and machine loading problems. This effort will enable the VNS produces near optimum solutions in a reasonable amount of time.

II. Related Works

As the part type selection and machine loading problems have an important role in determining the productivity and efficiency of FMS, an extensive research has been conducted in these areas. Various methods used in the literature to solve the problems will be briefly discussed in this section. The methods include mathematical programming based approach, heuristic and meta-heuristic approach, and hybrid approach.

Mathematical programming-based approaches were used in few earliest studies. For instance, Stecke [8] applied a nonlinear integer programming to solve machine grouping and loading problems. Another study was conducted by Mgwatu [26] that formulated two mathematical models to solve the integrated part type selection, machine loading, and machining optimization. A commercial package software for mathematical programming was used to obtain solutions of the integrated problems. A similar approach was developed by Bilge, et al. [27] to solve the part type selection and machine loading problems in FMS with flexible process plans.

Exact algorithms and analytical and mathematical-programming-based methods are robust in applications [28]. However, even if the articles discussed in this section reported promising results, their approaches are only suitable and were used for small and medium size problems. They tend to become impractical when the problem size increases and the optimum solution may not be achieved in a reasonable amount of time [4]. Therefore, Stecke [8] suggests to use an efficient heuristic algorithm to solve a larger and more complex problem.

Due to their efficient computational time, heuristic based approaches were frequently used in the production planning of FMS. However, the heuristics were only successful in solving problems of limited complexity. They used only small size problems in the experiment. For example, Kim, et al. [29] developed two-stages heuristic to address the machine loading problem. In the first stage, a modified bin-packing algorithm was used to produce an initial solution. In the second stage, the initial solution was improved by implementing a simple search algorithm. The study used test-bed problems that had maximum number of part type of 15. By using complete enumeration [30] or branch-and-bound methods [31] that produce the optimum solution, the small size problems may be easily and quickly solved. Therefore, even though they reported good results, their heuristics are not likely to produce promising results in large scale problems.

To address the limitations of heuristic methods, meta-heuristic approaches were proposed in several studies. The meta-heuristic simultaneously manages several heuristic procedures to deal with the large space of the integrated part type selection and machine loading problems. The approaches included genetic algorithms [2, 31-33], particle swarm optimization [7, 34, 35], ant colony optimization [36], immune algorithm [37, 38], harmony search algorithm [39], and symbiotic evolutionary algorithm [40]. The studies reported that the meta-heuristic approaches were efficient and produced near-optimum solutions.

As powerful population-based algorithms, genetic algorithms and particle swarm optimization were frequently used to solve the production planning problem in FMS. For example, a study by Mahmudy, et al. [2] equipped a specialized genetic algorithm with various crossover and mutation operators to enable exploring the very large search space of the integrated part type selection and machine loading problems. Genetic algorithm was also used in a study by Abazari, et al. [31]. In this study, a mixed-integer linear mathematical programming model was firstly developed and then integer-based chromosome was used to represent the solution.

Other type of genetic algorithms was implemented by Yusof, et al. [32]. The study developed constraint-chromosome genetic algorithm to solve the machine loading problem. The chromosome was designed to produce only feasible solution. This effort reduced a high computational time that is required to repair infeasible solutions. Furthermore, knowledge-based genetic algorithm was developed to solve machine loading problem [33]. The proposed approach exloited tacit and explicit knowledge that was obtained from the problem. The knowledge was used in the stage of generating initial population and also in applying genetic operators such as crossover, mutation, and selection. Thus, the genetic algorithm could explore the large search space more efficient.

Particle swarm optimization was used in several studies. For example, Biswas and Mahapatra [7] modified particle swarm optimization to solve the part type selection and machine loading problems. Modification was done by adding a mechanism to prevent early convergence. They reported that the mechanism was effective to obtain better solutions. A similar approach was proposed by Mahmudy [34] that adopted chromosome representation of genetic algorithm in [12] for his particle swarm optimization.

Another type of population-based algorithm, immune algorithm, was also implemented to solve the part type selection and machine loading problems. For instance, Prakash, et al. [37] modified immune algorithm by adding a new hypermutation operator to improve the driving forces of the immune algorithm. A similar improvisation was done by Dhall, et al. [38] that developed several special operators for immune algorithm.

The complexity level of the integrated part type selection and machine loading problems become a reason for the researchers to develop more powerful approaches by combining two methods. For example, Yusof, et al. [41] combined genetic algorithm and harmony search algorithm. The study used test-bed problems that had maximum number of part type of 8. Furthermore, hybrid genetic algorithm with simulated annealing was developed by Yogeswaran, et al. [1]. Genetic algorithm also combined with a local search as shown in works by Basnet [30] and Mahmudy, et al. [10].

To produce satisfactory results, preliminary experiments are required by heuristic and meta-heuristic methods to determine their optimum parameters [42, 43]. Parameters of several methods are shown in Table 1. The table clearly shows that VNS has fewest parameters compare to other methods. Thus, it will reduce computational time required in the preliminary experiments. Fewer parameters also enable to give more efforts on designing the best neighbourhood structure for VNS.

TABLE I
PARAMETERS OF HEURISTIC AND META-HEURISTIC METHODS

method	parameters
genetic algorithms	- population size - crossover rate - mutation rate - termination condition
simulated annealing	- initial temperature - cooling factor - inner iteration - probability of accepting worse solution - termination condition
particle swarm optimization	- number of particles - inertia vector (w) - self-recognition component (c_1) - social component (c_2) - termination condition
tabu search	- size of tabu list - termination condition
variable neighbourhood search	- number of neighborhoods - termination condition

The integrated part type selection and machine loading problems is known as NP-hard problems and finding reasonable solutions is more difficult if other flexibilities are addressed. An example of such flexibility is the possibility of processing an operation in alternative machines with different tool types [7, 10].

Due to the complexity of the problems, several simplicities were adopted in the existing researches. For example, Yusof, et al. [32], Basnet [30], and Biswas and

Mahapatra [7] ignored the tool allocation problem as the integral part of the machine loading problem. They did not mention specific tool types and its availability and only mentioned the number of slots needed by the tools. This paper attempts to fill these knowledge gaps by addressing the machine flexibility and the occurrence of specific tool types for the specific operations.

III. Problem Formulation

A FMS under consideration is arranged by a number of computer numerically controlled (CNC) machines. Each machine has a tool magazine with limited tool slot capacity. Thus, only limited number of tools can be attached to the machine.

Production requirement of each part type is stated as sequence of operations. Different set of tool types are required by different operations. This paper considers the flexibility of machining operations where several alternative machines with several alternative tool types are available for each operation. Different processing times are required for these alternative machines.

A production process is started when production orders that consist several part types are arrived. The system must select which part type are loaded to the current batch due to the limited availability of tools attached in the machines. This approach is known as batching approach as several production batches is required to produce all part types [4].

Several assumptions are made as follows:

- All machines are available at time 0 and never breakdown.
- Machines are independent from each other.
- Part types are independent from each other and there are no precedence constraints among operations of different part types.
- A machine can only execute one operation at a given time.

A. Subscripts and parameters

Several subscripts are used in the model as follows:

$p = 1,\ldots,P$	part type
$o = 1,\ldots,O_p$	operation of part type p
$t = 1,\ldots,T$	tool type
$m = 1,\ldots,M$	machine type

Parameters of the model are defined as the following:

C_m = tool slot capacity of machine m

N_t = number of tools type t

S_t = number of slots required by tool type t

B_p = batch size of part type p

V_p = value (price) of part type p

μ_{po} = set of possible machines on which operation o of part type p can be performed

$\tau_{pomt} = \{1,0\}$: 1 if tool type t is required for processing operation o of part type p on machine m, 0 otherwise

t_{pom} = processing time of operation o of part type p on machine m

B. Decision variables

Two objectives of the model are defined as follow:

$X_p = \{1,0\}$: 1 if part type p is selected in the current batch, 0 otherwise

$Y_{pom} = \{1,0\}$: 1 if machine m is selected for operation o of part type p, 0 otherwise

The depending variable for this model is stated as follow:

$Z_{mt} = \{1,0\}$: 1 if tool type t is loaded to the machine m, 0 otherwise

The value of depending variable is determined once the values of the decision variables are obtained.

C. Objectives

To measure the performance of FMS production planning, a number of objectives were used such as minimizing part movement [40] and minimizing tool changeovers [40]. However, most of the studied addressed two common objectives, maximizing system throughput and maintaining the balance of the system. The objectives may contribute to the other criteria of system's performance such as the completion time of all part types' operations [9].

Maximizing system throughput is obtained by maximizing the value (price or profit) of selected part types as expressed in (1).

$$\text{Maximize: } \sum_{p=1}^{P} X_p B_p V_p \tag{1}$$

Maintaining the balance of the system is obtained by minimizing system unbalance as shown in (2). W_m is workload of machine m and $\overline{W}$ is the average machine workload. Here, length of scheduling period for each machine (L_m) is predetermined and overloading of the machines is allowed.

$$\text{Minimize: } \sum_{m=1}^{M} |L_m - W_m| \tag{2}$$

$$\text{where } W_m = \sum_{p=1}^{P} \sum_{o=1}^{O_p} Y_{pom} t_{pom} B_p$$

$$\text{and } \overline{W} = \frac{\sum_{m=1}^{M} W_m}{M}$$

The two objective functions in the problem must be converted to a single objective function which is used to measure the goodness of the solution produced by VNS. Equation (2) should be converted into (3) to produce value between 0 and 1.

$$f_1 = \frac{\sum_{p=1}^{P} X_p B_p V_p}{\sum_{p=1}^{P} B_p V_p} \tag{3}$$

Minimizing system unbalance in (2) can be converted as maximizing (4) as follow:

$$f_2 = 1 - \frac{\sum_{m=1}^{M} |L_m - W_m|}{\sum_{m=1}^{M} L_M} \tag{4}$$

Finally, the objective function can be formulated as in Equation (5). The values of α_1 and α_2 can be determined according to the decision maker's preference.

$$\text{Maximize: } F = \alpha_1 f_1 + \alpha_2 f_2 \tag{5}$$

$$\alpha_1 \alpha_2\text{: weigthed parameter}$$

D. Constraints

While considering the objectives of the system, several constraints must be satisfied as follows:

$$\sum_{o=1}^{O_p} \sum_{m=1}^{M} Y_{pom} = O_p X_p, \qquad \forall p \tag{6}$$

$$\sum_{m\in\mu_{po}} Y_{pom} = X_p\ , \qquad \forall p,\ \forall o \tag{7}$$

$$Z_{mt} = Y_{pom}\tau_{pomt}\ , \qquad \forall p,\ \forall o, \forall m, \forall t \tag{8}$$

$$\sum_{m=1}^{M} Z_{mt} \le N_t\ , \qquad \forall\ t \tag{9}$$

$$\sum_{t=1}^{T} Z_{mt}S_t \le C_m\ , \qquad \forall\ m \tag{10}$$

Constraint (6) ensures that all operations of the selected part types are performed on proper machines. Constraint (7) guarantees that each operation of the selected part types must be completed on only one machine. The machine must be determined as the operation can be processed on several alternative machines. Constraint (8) is used to ensure that all required tools are attached to the selected machine for an operation. Constraint (9) guarantees that number of tools assigned to the machines must not exceed its availability. Constraint (10) ensures that the number of tool slots occupied on a machine must not exceed the machine's tool slot capacity.

IV. Numerical Example

A simple problem set is developed as an example of the problem formulation. The FMS have 3 different machines which have tool slot capacity of 15, 20 and 25 respectively. The machines have length of scheduling period (L_m) equal to 2500. Here, overloading of the machines is permitted. Moreover, there are 10 different tool types and each tool type has several instances (copies) and occupies a number of tool slots on machines' magazine as shown in Table 2.

A production orders that consist seven part types are arrived. Each part type has specific production requirements as shown in Table 3. For example, part type 3 has 3 operations. Operation 1 can be executed on machines 2 or 3. Machine 2 needs 30 unit times for processing and requires tool types 6, 7 and 8. Different processing times and tool types are required if machine 3 is chosen. In this case, machine 3 needs 40 unit times and requires tool types 8, 9 and 10. Thus, it shows the occurrence of machine and tool flexibility in the production planning problem.

TABLE II
THE AVAILABILITY OF TOOL TYPES

tool type	1	2	3	4	5	6	7	8	9	10
availability	2	3	2	3	2	3	2	3	2	3
number of slot needed	3	3	4	4	5	5	5	3	3	3

TABLE III
PRODUCTION REQUIREMENT OF PART TYPES

part type	batch size	value $	op	mac	time	tools		
1	20	5	1	2	20	2	3	5
			2	1	30	4	5	
			3	2	30	3	4	
				3	30	5		
2	20	3	1	1	30	1	3	
			2	2	20	3	4	
			3	2	30	4	6	7
3	40	2	1	2	30	6	7	8
				3	40	8	9	10
			2	2	20	1	10	
				3	40	2	10	
			3	1	20	1	2	
4	20	1	1	2	30	9	10	
				3	20	9	10	
			2	2	30	6	7	
				1	40	6	7	
			3	1	30	3	4	
5	30	4	1	2	40	1	2	3
			2	1	40	7	8	
				2	30	3	4	
6	30	3	1	3	20	7	8	
			2	2	50	9	10	
			3	3	10	2		
7	30	5	1	1	50	1	2	3
				2	40	7	9	10
			2	3	30	4	6	

op:operation; mac:machine; time: unit time; tools: required tool types

V. Modeling Using VNS

A. Solution Representation

A solution representation is required by VNS to determine how solutions can be obtained from the problem. A solution of VNS for the problem in Section 4 is shown in Table 4. Here, the solution is represented as an array of record contained with part type and machines sequence for all operation of the part type.

TABLE IV
AN EXAMPLE OF SOLUTION

record	part type	machines
1	7	1 3
2	3	3 2 1
3	5	2 2
4	1	2 1 3
5	4	3 1 1
6	6	3 2 3
7	2	1 2 3

Table 4 shows that the first operation of part type 7 is performed on machine 1 and the second operation is processed in machine 3. Furthermore, part type 3 is processed in machines 3, 2, and 1 sequentially.

After determining machines for operations, required tool types are attached to the machines. At this stage, all constraints such as the availability of tools and number of empty slots on the machines are checked. For example, after selecting part types 7, 3 and 5 according to the part type sequence as shown in second column of Table 4, adding part type 1 to the solution will not satisfy the constraints. Thus, the VNS solution states that only part types 7, 3, and 5 are selected for the current batch and the objective functions of the problem are calculated based on the selected part types. The other part types will be produced in the next batches.

The calculation of system throughput is presented in Table 5. Here, the system throughput is obtained by summing all part types' value.

TABLE V
CALCULATION OF SYSTEM THROUGHPUT

part type	batch size	value $	total value
7	30	5	150
3	40	2	80
5	30	4	120
		throughput	**350**

Machines' workload and assigned tool types are provided in Table 6. Here, used slot does not exceed the number of slot in each machine.

TABLE VI
MACHINES WORKLOAD

mac	workload	unbalance	number of slots	used slot	tools assigned
1	2300	200	15	10	1 2 3
2	2900	400	20	17	1 2 3 4 10
3	2500	0	25	18	4 6 8 9 10
System unbalance		600			

B. *Neighborhood Structure*

Variable neighborhood search (VNS) works by iterating a local search (LS) technique. In each iteration, the LS explores the search space from a new starting point. The starting point is determined using a mechanism called neighborhood structure [13]. As the main feature of VNS, the neighborhood structure is used to produce new candidate solutions by changing initial/current solution. The changing mechanism involves swap, insert, and exchange operations.

The neighborhood structures N_k (k=1,...,k_{max}) is adopted and N_k (x) is defined as the set of solutions in the k^{th} neighborhood of x. N_k (x) is obtained by randomly changing order of k part types in the solution representation. k_{max} is determined according to the size of problems used in experiments. A large value of k_{max} will enable the VNS to get better solutions. However, higher computational time is required.

A pseudo code for the VNS is shown in Fig. 2. The initial solution of the VNS is randomly generated. Thus, different instances of the VNS may produce different solutions. The value of k will be increased if there is no improvement in the best solution. The increase of k will drive the VNS to explore different area of the search space. However, the value will set to 1 if a better solution is found. This strategy will enable the VNS to exploit local optimum area.

```
PROCEDURE VariableNeighborhoodSearch
Input:
      curr:     current/initial solution
      kMax:     number of neighbourhoods
Output:
      best:     the best solution

best ← curr
k ← 1
WHILE k<=kMax DO
   // change order of k part types
   curr ← ChangeOrder (best, k)
   // find local optimum
   bestLocal ← LocalSearch (curr)
   IF Fitness(bestLocal)> Fitness(best) THEN
      best ← bestLocal
      k ← 1
   ELSE
      k ← k + 1
   END IF
END WHILE
END PROCEDURE
```

Fig. 2. Pseudo code of the VNS

The local search works by randomly replacing machine for each operation with other possible machines as shown in Fig. 3. If the new solution has better fitness value then it replaces the current solution.

```
PROCEDURE LocalSearch
Input:
      curr:     current solution
Output:
      best:     the best solution

best ← curr
// check each operation of the part type
FOR EACH operation $O_i$ IN curr DO
   // Change a machine for $O_i$ with other possible
      machine
   curr ← ChangeMachine (curr, $O_i$)
   IF Fitness(curr)> Fitness(best) THEN
      best ← curr
   END IF
END FOR
END PROCEDURE
```

Fig. 3. Pseudo code of the local search

An example of changing order of part types to provide a new starting point is depicted in Fig. 4. Here, k=3 so three records (records 2, 4, and 6) are randomly chosen as highlighted in the left part of the figure. Contents of the selected records are randomly exchanged and the result is depicted in the right part of the figure.

record	part type	machines	record	part type	machines
1	7	1 3	1	7	1 3
2	5	2 1	2	3	2 3 1
3	6	3 2 3	3	6	3 2 3
4	3	2 3 1	4	1	2 1 3
5	2	1 2 2	5	2	1 2 2
6	1	2 1 3	6	5	2 1
7	4	2 1 1	7	4	2 1 1
before changing			after changing		

Fig. 4. An example of changing order of 3 part types

An example of changing machines of operations by the local search is provided in Fig. 5. Here, second record is selected and machines for operations of part type 3 are replaced by other possible machines.

record	part type	machines	record	part type	machines
1	7	1 3	1	7	1 3
2	3	2 3 1	2	3	3 3 1
3	6	3 2 3	3	6	3 2 3
4	1	2 1 3	4	1	2 1 3
5	2	1 2 2	5	2	1 2 2
6	5	2 1	6	5	2 1
7	4	2 1 1	7	4	2 1 1
before changing			after changing		

Fig. 5. An example of changing machines of operations

VI. RESULT AND DISCUSSION

A. *Test-Bed Problems*

The performance of the proposed VNS is evaluated by using twelve test-bed problems taken form [12] available at 'http://lecture.ub.ac.id/anggota/wayanfm/ data_test/'. The optimum solutions for all test-bed problems are also provided. The characteristics of the problems are shown in Table 7. The test-bed problems are divided into three different classes in terms of the number of part types, the number of machines, the number of tool types, the length of scheduling period, and the level of flexibility. Small size problems are represented by problems 1 to 4. Problems 5 to 8 represent medium size problems whereas problems 9 to 12 represent large size problems. All machines for each problem in the same class have an equal scheduling period (L_m) as shown in the last column in Table 7.

TABLE VII
TEST-BED PROBLEMS

problem	num. of part types	num. of machines	num. of tool types	scheduling period (L_m)
1	8	4	20	4000
2	8	5	25	4000
3	10	4	20	4000
4	10	5	25	4000
5	16	4	20	7000
6	16	5	25	7000
7	18	4	20	7000
8	18	5	25	7000
9	24	4	20	10000
10	24	5	25	10000
11	26	4	20	10000
12	26	5	25	10000

B. *Experimental Design*

The VNS is implemented in Java and experiment is carried out on personal computer equipped with AMD Quad-Core processor working at speed 2.8 GHz and 4GB DDR3 memory. The first stage of the experiment is determining a proper value for k_{max}. For this purpose, problem 6 is used. VNS is a stochastic method and different solution is obtained in each run, so the VNS is run 10 times for each value k_{max} ranging from 1 to 16 (number of part-types in this problem).

TABLE VIII
OBJECTIVE VALUE OF THE PROBLEM OVER DIFFERENT k_{max} VALUES

k_{max}	time	F
1	0.0024	0.640342
2	0.0034	0.782617
3	0.0038	0.802161
4	0.0050	0.883821
5	0.0048	0.937525
6	0.0048	1.051203
7	0.0060	1.015235
8	0.0064	1.030236
9	0.0074	1.048757
10	0.0154	1.116403
11	0.0070	1.032010
12	0.0086	1.136945
13	0.0100	1.210845
14	0.0078	1.181168
15	0.0106	1.167585
16	0.0104	1.365191

The result is provided in Table 8. Columns ‘time’ depicts the time required in seconds to complete the VNS’ cycle. Column ‘F’ denotes the objective value of the problem that is calculated using Equation 5. Table 8 and also Fig. 6 clearly show that higher value of k_{max} will produce better results. Higher k_{max} will enable the VNS to explore wider area of the search space. However, the best result achieved in this stage is far below of the optimum result. By terminating the iteration of the VNS until a variable k reaches k_{max}, the cycle of the VNS run only less than 1 second. Therefore, the cycle of the VNS that is depicted in Fig. 2 is modified.

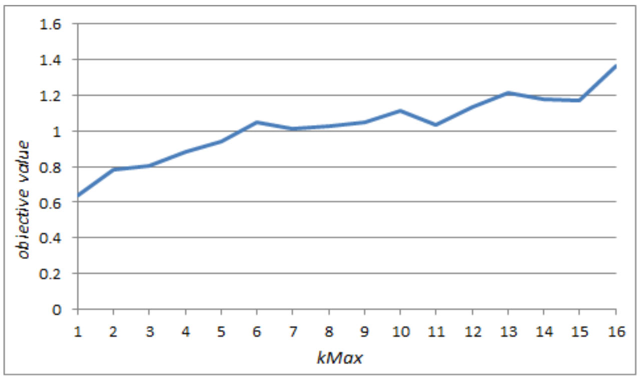

Fig. 6. Objective Value of the Problem over Different k_{max} Values

The modified version of the VNS is presented in Fig 7. Here, rather than terminating the iteration until k reaches k_{max}, the cycle is stopped until predetermined running time. k_{max} is set equal to the number of part-types in the problem. If k reaches k_{max}, it value will be set to 1.

```
PROCEDURE VariableNeighborhoodSearch
Input:
      curr:     current/initial solution
      kMax:     number of part-types
Output:
      best:     the best solution
best ← curr
k ← 1
WHILE NOT termination_condition DO
   // change order of k part types
   curr ← ChangeOrder (best, k)
   // find local optimum
   bestLocal ← LocalSearch (curr)
   IF Fitness(bestLocal)> Fitness(best) THEN
      best ← bestLocal
      k ← 1
   ELSE
      k ← k + 1
   END IF
   IF k=kMax THEN
      k ← 1
   END IF
END WHILE
END PROCEDURE
```

Fig. 7. Pseudo code of the Modified VNS

By determining 20 seconds of running time, the VNS is run again 10 times and the results are presented in Table 9. The comparison of the VNS and the modified VNS is depicted in Fig. 8. Table 9 and Fig. 7 clearly show that the modified VNS produce better and more stable results.

TABLE IX
OBJECTIVE VALUE OF THE PROBLEM OVER DIFFERENT k_{max} VALUES OF MODIFIED VNS

k_{max}	F
1	1.849487
2	1.847529
3	1.849819
4	1.849487
5	1.849819
6	1.863265
7	1.849819
8	1.872933
9	1.873597
10	1.885486
11	1.873431
12	1.885652
13	1.897209
14	1.896877
15	1.908766
16	1.908600

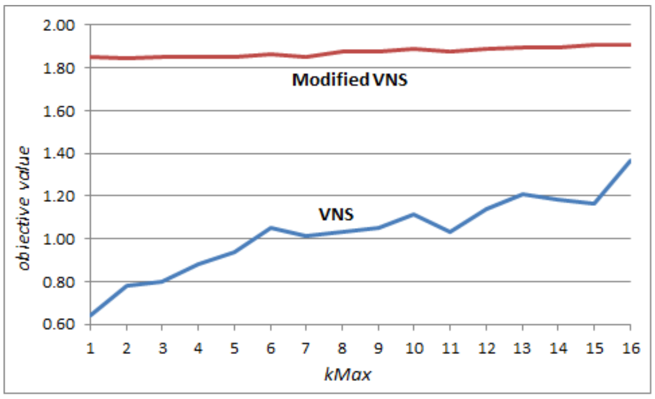

Fig. 8. Objective Value of the Problem over Different k_{max} Values of Modified VNS

The next stage of the experiment is determining a proper running time of the VNS. The time should be determined so that the VNS have most likely achieved their convergence and have a very low chance to obtain better solution in the next iterations. For this purpose, the biggest problem for each class is chosen. For the small size problems, problem 4 is chosen. Firstly, the VNS is run for 1 second. To obtain a fair result, the VNS is run 10 times and the average of the objective function (F) is calculated. Next, running of the VNS is repeated for 2 seconds, 3 seconds, and so on until there is no significance improvement of the average of F. The complete results are presented in Table 10 and their graph is depicted in Fig. 9. The table and the graph clearly show that for problem number 10 the VNS achieve it convergence in around 10 seconds. Thus, the VNS will be run 10 seconds for all small size problems.

TABLE X
THE AVERAGE OF THE OBJECTIVE VALUE OVER DIFFERENT RUNNING TIME FOR PROBLEM 4

time	average of F
1	2.5102
2	2.5166
3	2.5298
4	2.5298
5	2.5315
6	2.5281
7	2.5315
8	2.5298
9	2.5298
10	2.5315
11	2.5315
12	2.5315

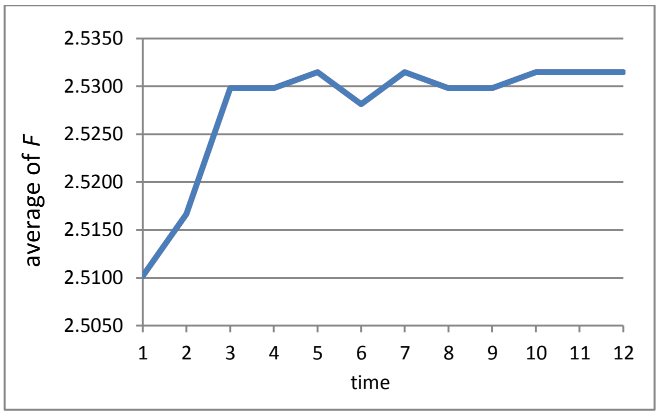

Fig. 9. The Average of the Objective Value over Different Running Time for Problem 4

For the medium size problems, problem number 8 is chosen. The average of the objective value over different running time for problem 8 is depicted in Fig 10. The graph clearly shows that the VNS achieve it convergence in around 40 seconds. Thus, the VNS will be run 40 seconds for all medium size problems

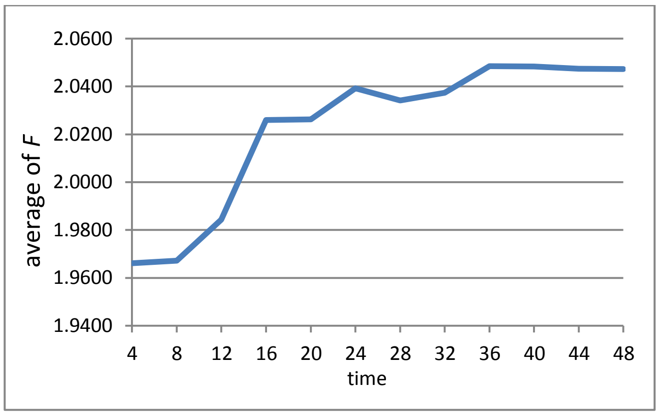

Fig. 10. The Average of the Objective Value over Different Running Time for Problem 8

For the large size problems, problem number 12 is chosen. The average of the objective value over different running time for problem 12 is depicted in Fig 11. The graph shows that the VNS achieve it convergence in around 80 seconds. After 80 seconds the VNS cannot obtain better results. Thus, the VNS will be run 80 seconds for all large size problems

TABLE XI
COMPUTATIONAL RESULTS

problem	optimum values			VNS				
	F	*TH*	*SU*	*NOS*	*F*	*TH*	*SU*	*DEV(%)*
1	2.545	1,616	803	20	2.545	1,616	803	0.00
2	2.926	2,591	9,838	20	2.926	2,591	9,838	0.00
3	2.972	3,058	6,858	20	2.972	3,058	6,858	0.00
4	2.531	2,196	3,233	20	2.531	2,196	3,233	0.00
							average	0.00
5	2.156	2,676	3,738	17	2.148	2,649	3,613	0.40
6	1.968	2,605	7,126	8	1.897	2,508	8,077	3.60
7	2.458	3,595	5,529	2	2.404	3,258	2,722	2.23
8	2.088	2,871	4,768	9	2.041	2,787	5,164	2.25
							average	2.12
9	2.349	4,150	4,204	2	2.201	3,814	5,424	6.30
10	1.809	3,212	10,879	7	1.765	3,083	11,043	2.46
11	2.305	4,417	5,519	2	2.169	4,134	7,253	5.88
12	2.018	3,937	9,291	5	1.909	3,837	13,192	5.38
							average	5.01

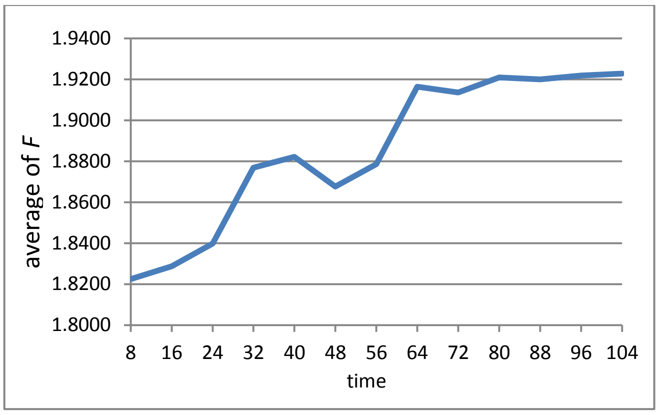

Fig. 11. The Average of the Objective Value over Different Running Time for Problem 12

The final stage of the experiment is running the VNS that has been modified for all test-bed problems. As obtained in the previous stage of experiment, the VNS is run 10, 40, and 80 seconds for small, medium, and large size problems respectively. To obtain a fair result, the VNS is run 20 times for each test-bed problem and obtain results of value of objective function (*F*), system throughput (*TH*), and system unbalance (*SU*).

The performance of the proposed VNS is measured by using number optimum solutions (*NOS*) and deviation of objective values resulted by the VNS to its optimum values (*DEV*). The optimum solutions are calculated by using a branch-and-bound method and are obtained from [12]. Equation (11) shows the deviation of average objective values from 20 runs of the VNS to optimum objective value. F_{opt} is objective value obtained by branch-and-bound method. $FVNS_r$ is objective value obtained by the VNS in run *r*.

$$DEV = \left| \frac{F_{opt} - \left(\sum_{r=1}^{20} FVNS_r\right) / 20}{F_{opt}} \right| \times 100\% \tag{11}$$

C. Numerical Results

The complete computational result is provided in Table 11. Columns 'F', 'TH' and 'SU' below column 'VNS' depict the average of fitness value, throughput and system unbalance obtained from 20 runs of the VNS

Based on the empirical results of Table 11, perfect results are obtained by the proposed VNS in all small size problems (problems 1 to 4). Here, the VNS could achieve optimum solution in all runs so the value of *DEV* is 0%. The results indicate that the VNS may explore all possible solutions and the best solution is obtained easily.

In the medium size problems (problems 5 to 8), the best result is obtained in problem 5 with *DEV* of 0.40% and the worst solution is occurred in problem 6 with *DEV* of 3.60%. In addition, the VNS could produce optimum solutions in several test-bed problems and the best result is achieved in problem 5 with NOS of 17. The average of *DEV* in the medium size problems is only 2.12% that are obtained in only 40 seconds of computational time.

The VNS also obtains optimum solutions in several runs in all large size problems. The best result is achieved in problem 10 with *DEV* of 2.46% and the worst solution is obtained in problem 11 with *DEV* of 5.88%. Overall, in larger problems, *DEV* values tend to increase as the search space becomes very large and it is impossible for the VNS to explore all areas in limited computational time. Note that all *DEV* values in large size problems are below 7% which may be regarded as good results considering these results are achieved in only 80 seconds.

Promising results in this research are achieved by using only simple VNS. The neighbourhood structure is designed to enable the VNS produces near optimum solutions in a reasonable amount of time. Other approaches may be supported by complex strategies to achieve good results which may need excessive computation time. Examples of such approaches are combining genetic algorithm with harmony search algorithm [41], enhancing the power genetic algorithm by combining with simulated annealing [1] equipping genetic algorithm with local search [30], and hybridizing particle swarm optimization with local search methods [7].

VII. CONCLUSION AND FUTURE STUDY

A model for the integrated part type selection and machine loading problems is developed in this paper. The model considers the flexibilities of operations which involves the availability of alternative machines and alternative tool types. The integrated model of the NP-hard problems is solved by using VNS. The proper neighbourhood structure could produce promising results in reasonable amount of time. By using 12 test bed problems available in the literature, the proposed VNS improves the FMS performance by considering two objectives, maximizing system throughput and maintaining the balance of the system. To measure the effectiveness of the proposed VNS, the obtained results are compared to the optimum values produced by branch-and-bound method. The numerical experiments prove that the proposed VNS could reach near optimum solutions in reasonable amount of time.

More complex problem will be addressed in the next work. It includes the existence of alternative production plans which refer to possibility of producing part on alternative operation sequence and solving the problem for multiple batches. Here, new solution representation and neighbourhood structure for the VNS must be developed. Thus, a more powerful of VNS is required. Developing new local search methods and combining the VNS with other heuristics methods will be considered.

REFERENCES

[1] M. Yogeswaran, S. G. Ponnambalam, and M. K. Tiwari, "An efficient hybrid evolutionary heuristic using genetic algorithm and simulated annealing algorithm to solve machine loading problem in FMS," *International Journal of Production Research,* vol. 47, no. 19, pp5421-5448, 2009.

[2] W. F. Mahmudy, R. M. Marian, and L. H. S. Luong, "Modeling and optimization of part type selection and loading problems in flexible manufacturing system using real coded genetic algorithms," *International Journal of Electrical, Computer, Electronics and Communication Engineering,* vol. 7, no. 4, pp251-260, 2013.

[3] C. Duerden, L.-K. Shark, G. Hall, and J. Howe, "Minimisation of energy consumption variance for multi-process manufacturing lines through genetic algorithm manipulation of production schedule," *Engineering Letters,* vol. 23, no. 1, pp40-48, 2015.

[4] W. F. Mahmudy, "Optimisation of Integrated Multi-Period Production Planning and Scheduling Problems in Flexible Manufacturing Systems (FMS) Using Hybrid Genetic Algorithms " Ph.D. Thesis, School of Engineering, University of South Australia, 2014.

[5] H. T. N. I. K. Nejad, N. Sugimura, K. Iwamura, and Y. Tanimizu, "Integrated dynamic process planning and scheduling in flexible manufacturing systems via autonomous agents," *Journal of Advanced Mechanical Design, Systems, and Manufacturing,* vol. 2, no. 4, pp719-734, 2008.

[6] A. H. R. Zaied, "Quantitative models for planning and scheduling of flexible manufacturing system," *Emirates Journal for Engineering Research,* vol. 13, no. 2, pp11-19, 2008.

[7] S. Biswas and S. Mahapatra, "Modified particle swarm optimization for solving machine-loading problems in flexible manufacturing systems," *The International Journal of Advanced Manufacturing Technology,* vol. 39, no. 9, pp931-942, 2008.

[8] K. E. Stecke, "A hierarchical approach to solving machine grouping and loading problems of flexible manufacturing systems," *European Journal of Operational Research,* vol. 24, no. 3, pp369-378, 1986.

[9] W. F. Mahmudy, R. M. Marian, and L. H. S. Luong, "Hybrid genetic algorithms for multi-period part type selection and machine loading problems in flexible manufacturing system," in *IEEE International Conference on Computational Intelligence and Cybernetics*, Yogyakarta, Indonesia, 2013, pp. 126-130.

[10] 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 & results," in *5th International Conference on Knowledge and Smart Technology (KST),* Chonburi, Thailand, 2013, pp. 81-85.

[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," *ECTI Transactions on Computer and Information Technology (ECTI-CIT),* vol. 8, no. 1, pp80-93, 2014.

[12] W. F. Mahmudy, R. M. Marian, and L. H. S. Luong, "Solving part type selection and loading problem in flexible manufacturing system using real coded genetic algorithms – Part II: optimization," in *International Conference on Control, Automation and Robotics*, Singapore, 2012, pp. 706-710.

[13] P. Hansen, N. Mladenović, and J. Moreno Pérez, "Variable neighbourhood search: methods and applications," *Annals of Operations Research,* vol. 175, no. 1, pp367-407, 2010.

[14] N. Bouhmala, K. Hjelmervik, and K. I. Øvergaard, "A generalized variable neighborhood search for combinatorial optimization problems," *Electronic Notes in Discrete Mathematics,* vol. 47, no. 0, pp45-52, 2015.

[15] M. Yazdani, M. Amiri, and M. Zandieh, "Flexible job-shop scheduling with parallel variable neighborhood search algorithm," *Expert Systems with Applications,* vol. 37, no. 1, pp678-687, 2010.

[16] B. Jarboui, H. Derbel, S. Hanafi, and N. Mladenović, "Variable neighborhood search for location routing," *Computers & Operations Research,* vol. 40, no. 1, pp47-57, 2013.

[17] N. Mladenović, D. Urošević, S. d. Hanafi, and A. Ilić, "A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem," *European Journal of Operational Research,* vol. 220, no. 1, pp270-285, 2012.

[18] N. Dahmani, S. Krichen, and D. Ghazouani, "A variable neighborhood descent approach for the two-dimensional bin packing problem," *Electronic Notes in Discrete Mathematics,* vol. 47, no. 0, pp117-124, 2015.

[19] N. Mladenovic, R. Todosijevic, and D. Urosevic, "An efficient general variable neighborhood search for large travelling salesman problem with time windows," *Yugoslav Journal of Operations Research* vol. 23, no. 1, pp19-31, 2013.

[20] N. Mladenović, R. Todosijević, and D. Urošević, "Two level general variable neighborhood search for attractive traveling salesman problem," *Computers & Operations Research,* vol. 52, Part B, no. 0, pp341-348, 2014.

[21] T. Davidovic and T. G. Crainic, "Parallelization strategies for variable neighborhood search," *Research Report for Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada,* 2013.

[22] M. Eskandarpour, S. H. Zegordi, and E. Nikbakhsh, "A parallel variable neighborhood search for the multi-objective sustainable post-sales network design problem," *International Journal of Production Economics,* vol. 145, no. 1, pp117-131, 2013.

[23] J. u. S´anchez-Oro, M. Sevaux, A. e. Rossi, R. Mart´i, and A. Duarte, "Solving dynamic memory allocation problems in embedded systems with parallel variable neighborhood search strategies," *Electronic Notes in Discrete Mathematics,* vol. 47, pp85–92, 2015.

[24] J.-q. Li, Q.-k. Pan, and F.-t. Wang, "A hybrid variable neighborhood search for solving the hybrid flow shop scheduling problem," *Applied Soft Computing,* vol. 24, no. 0, pp63-77, 2014.

[25] T. A. Oliveira, V. N. Coelho, M. J. F. Souza, D. L. T. Boava, F. Boava, M. Coelho, *et al.*, "A hybrid variable neighborhood search algorithm for targeted offers in direct marketing," *Electronic Notes in Discrete Mathematics,* vol. 47, pp205–212, 2015.

[26] M. I. Mgwatu, "Integration of part selection, machine loading and machining optimisation decisions for balanced workload in flexible manufacturing system," *International Journal of Industrial Engineering Computations,* vol. 2, pp913–930, 2011.

[27] U. Bilge, E. Albey, U. Besikci, K. Erbatur, and A. N. Aslan, "Mathematical models for FMS loading and part type selection with flexible process plans," *European Journal of Industrial Engineering,* 2014.

[28] T. Hasuike, "Exact and explicit solution algorithm for linear programming problem with a second-order cone," *IAENG International Journal of Applied Mathematics,* vol. 41, no. 3, pp213-217, 2011.

[29] H.-W. Kim, J.-M. Yu, J.-S. Kim, H.-H. Doh, D.-H. Lee, and S.-H. Nam, "Loading algorithms for flexible manufacturing systems with partially grouped unrelated machines and additional tooling constraints," *The International Journal of Advanced Manufacturing Technology*, vol. 58, no. 5, pp683-691, 2012.
[30] C. Basnet, "A hybrid genetic algorithm for a loading problem in flexible manufacturing systems," *International Journal of Production Research*, vol. 50, no. 3, pp707–718, 2012.
[31] A. M. Abazari, M. Solimanpur, and H. Sattari, "Optimum loading of machines in a flexible manufacturing system using a mixed-integer linear mathematical programming model and genetic algorithm," *Computers & Industrial Engineering*, vol. 62, no. 2, pp469-478, 2012.
[32] U. K. Yusof, R. Budiarto, and S. Deris, "Constraint-chromosome genetic algorithm for flexible manufacturing system machine-loading problem," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 3A, pp1591-1609, 2012.
[33] A. Sadrzadeh, "Knowledge-based genetic algorithm for dynamic machine–tool selection and operation allocation," *Arabian Journal for Science and Engineering*, vol. 39, no. 5, pp4315-4323, 2014.
[34] W. F. Mahmudy, "Optimasi part type selection and machine loading problems pada FMS menggunakan metode particle swarm optimization (Optimization of part type selection and machine loading problems in FMS using particle swarm optimization)," in *Konferensi Nasional Sistem Informasi (KNSI)* STMIK Dipanegara, Makassar, 2014, pp. 1718-1723.
[35] W. F. Mahmudy, "Improved particle swarm optimization untuk menyelesaikan permasalahan part type selection dan machine loading pada flexible manufacturing system (FMS) (Improved particle swarm optimization for solving part type selection and machine loading problems in flexible manufacturing system)," in *Konferensi Nasional Sistem Informasi*, Universitas Klabat, Airmadidi, Minahasa Utara, Sulawesi Utara, 2015, pp. 1003-1008.
[36] M. H. M. A. Jahromi, R. Tavakkoli-Moghaddam, S. A. Jazayeri, R. Jafari, and A. Shamsi, "Ant colony optimization for multi-objective machine-tool selection and operation allocation in a flexible manufacturing system," *World Applied Sciences Journal*, vol. 15, no. 6, pp867-872, 2011.
[37] A. Prakash, N. Khilwani, M. K. Tiwari, and Y. Cohen, "Modified immune algorithm for job selection and operation allocation problem in flexible manufacturing systems," *Adv. Eng. Softw.*, vol. 39, no. 3, pp219-232, 2008.
[38] P. R. Dhall, S. S. Mahapatra, S. Datta, and A. Mishra, "An improved artificial immune system for solving loading problems in flexible manufacturing systems," presented at the Industrial Engineering and Engineering Management (IEEM), 2010 IEEE International Conference on, 2010.
[39] U. K. Yusof, R. Budiarto, and S. Deris, "Harmony search algorithm for flexible manufacturing system(FMS) machine loading problem," presented at the 2011 3rd Conference on Data Mining and Optimization (DMO), Selangor Malaysia, 2011.
[40] K. Seok Shin, J. O. Park, and Y. Keun Kim, "Multi-objective FMS process planning with various flexibilities using a symbiotic evolutionary algorithm," *Computers and Operations Research*, vol. 38, no. 3, pp702-712, 2011.
[41] U. K. Yusof, R. Budiarto, I. Venkat, and S. Deris, "Machine Loading Optimization in Flexible Manufacturing System Using a Hybrid of Bio-inspired and Musical-Composition Approach," in *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2011 Sixth International Conference on*, 2011, pp. 89-96.
[42] W. F. Mahmudy, R. M. Marian, and L. H. S. Luong, "Real coded genetic algorithms for solving flexible job-shop scheduling problem – Part I: modeling," *Advanced Materials Research*, vol. 701, pp359-363, 2013.
[43] W. F. Mahmudy, R. M. Marian, and L. H. S. Luong, "Real coded genetic algorithms for solving flexible job-shop scheduling problem – Part II: optimization," *Advanced Materials Research*, vol. 701, pp364-369, 2013.