Adapted Gravitational Search Algorithm Using Multiple Populations to Solve Exam Timetable Scheduling Problems

Antonius Bima Murti Wijaya
Informatics Engineering Department
UKRIM University
Yogyakarta, Indonesia
bimamurti@ukrimuniversity.ac.id

Febe Maedjaja
Informatics Engineering Department
UKRIM University
Yogyakarta, Indonesia
febe@ukrimuniversity.ac.id

Abstract — **Since timetable scheduling is a discrete NP-Hard problem, it becomes a complicated case to be solved by a heuristic approach. Gravitational Search Algorithm (GSA) was developed with the main objective to solve a continuous problem as long as it could be defined in a mathematical equation, and yet it has the potential to solve a discrete problem. This research shows the adaptation of GSA for a discrete environment involving multiple populations in GSA in order to enlarge the searching space in exam timetable scheduling problems. Every best solution from each population will be injected to each population to give additional potential members. The adaptation strategy could be implemented in an exam timetable scheduling problem, resulting in an improved value of fitness. The multiple populations strategy helps the development of fitness factor by providing potential solutions from other populations, thus escaping the trap of local optimum solutions.**

Keywords—Metaheuristic; GSA; Multiple Populations; Scheduling Problem

I. INTRODUCTION

Timetable scheduling problem is a discrete combinatorial case that often occurs in education organizations. Since a timetable scheduling problem is a Non-Deterministic Polynomial Hard Problem (NP-Hard), it becomes a complicated case to be solved by a heuristic approach. Linear Programming as an Exact Mathematical Solver could sometimes solve this case, but it could not give an approach if the perfect solution does not exist [1].

Heuristic base algorithms are designed to solve complex problem even if the perfect solution does not exist. One of the heuristic base algorithms is Gravitational Search Algorithm (GSA). GSA was first developed in 2009 [2]. The main objective of this algorithm is to solve a continuous problem as long as it could be defined in a mathematical equation. This algorithm is based on the Particle Swarm Algorithm, and inspired by Newton's gravitational law. The interesting part of GSA is that every iteration in GSA will move all of the solutions in a population. GSA utilizes the best solution as the main mass to attract the other solutions to come closer, although they are also distracted by other heavy solutions that affect their position and mass. This idea potentially works in a discrete environment as well. Converting such a continuous heuristic algorithm into a discrete algorithm has the difficulties of adapting while keeping the concept of the algorithm. To solve this problem binary GSA has been suggested [3][4]. GSA works by moving an object's position based on its acceleration and velocity towards the heavier object. This concept is very clear to solve in a continuous environment, but in a discrete environment what is defined as positions, accelerations, velocity and direction toward center of gravitations are not clear. Mapping the terms in GSA is needed to make this algorithm work in a combinatorial case such as timetable scheduling problem. Since a fast convergence often occurs with GSA, modifying the algorithm by developing such a strategy is needed [5][6]. In some cases, GSA is developed by combining it with other algorithms to improve their result [7][8]. GSA works for only one population and develops one best solution. In this research multiple populations would be developed to give different characteristic of the best solutions. This effort has been done using Particle Swarm Optimization (PSO) which shows a better result compared to the original algorithm. Moreover, a variety of strategies have been developed for multiple populations purposes [9][10][11][12].

Exam timetable scheduling problem has the sensitivity, that once a schedule is changed, it could affect other resources and even decrease or increase its fitness dramatically. The connection between resources gets stronger when the schedule is tighter. Combining different characteristics of a schedule is expected to give improvement in exploration ability, even if it is already in exploitation phase. This research adapts the GSA for the discrete problem of exam timetable scheduling at UKRIM University using multiple populations to improve the schedule fitness by utilizing the different characteristics produced from each population, and will measure the schedule generation performance.

II. SCHEDULLING PROBLEM

Scheduling problem is a combinatorial problem of arranging activities with limited resources. The limited resources are developed by the system boundary. This problem could be solved using iterative mathematics and heuristic search, and Linear Programming is an iterative mathematics method that is able to solve this problem. This algorithm could give the optimize schedule if the solution exists, but in a scheduling problem the solution does not always exist [1]. On the other hand, the heuristic method offers an approach as an alternative solution. This method has solved some scheduling problem such as production scheduling problem using artificial bee colony combined with genetics algorithm [13]. A development using a heuristic search method, Tabu Search, has been used to solve the lecture scheduling problem at university of Malaya by modifying the neighbor solutions [14]. Genetics Algorithm, which is a later generation of tabu algorithm, has also been developed into Cuckoo Search to solve NP-Hard problems such as the problem of university timetable scheduling [15] [16].Compared to the previous research about adapted cuckoo search[15], this research also include how the modified algorithm combined using multiple population strategy, that resulted in fitness score performance rise.

The development of GSA which is based on Particle Swarm algorithm [1] turns to be the competition for the Cuckoo Search algorithm. In recent years the performance of Cuckoo and GSA have been tested for predicting academic performance, where GSA is claimed to be inferior to Cuckoo [17]. Developing a strategy in an algorithm becomes an option in the maturing process of the algorithm to solve the scheduling problem [18].

III. Exam Time-Table Schedulling Problem

Exam timetable scheduling problem is a variant of timetable scheduling problem. Exam timetable scheduling problem consists of two constraints: soft and hard constraints, and is categorized as a constraint satisfaction problem [19]. A violation of a soft constraint results in penalty, while a hard constraint must not be violated. However, in this research, a hard constraint results in a much heavier penalty than soft-constraint to improve the combining flexibility. The problem deals with how it is modelled and how the algorithm solves the schedule [20]. Instead of using greedy algorithm and put the hard-constraint to develop the initial population, this research was intended to investigate the heuristic evolving performance.

In the case of UKRIM University, the final and midterm exams are organized into four sessions per day for two weeks. There are two different kinds of room: a classroom that could be used for theory exam, and a laboratory that could handle theory and practice exam. This research identifies the hard and soft constraints for this scheduling problem, and the rule to develop the solutions.

The soft constraints are:

1. A maximum of one subject for students in the same year is allowed in one day. (penalty score: 3)
2. A big capacity classroom should be used for a big class, reducing the use of a big capacity classroom for a small class. (penalty score: 2)
3. Saturday is an alternative day for holding an exam. (penalty score: 2)
4. Exams for difficult subjects should start in the morning. (penalty score : 2)
5. Requested time for exam should be granted. (penalty score: 5)

The hard constraints are:

1. The classroom must be enough to fit the size of class. (penalty score: 15)
2. Practicum Exam must be held in a laboratory. (penalty score: 15).
3. The parallel classes must start at the same time. (penalty score : 15)

The rule to develop the solutions is that only one subject can be held per room per session.

IV. Gravitational Search Algorithm

GSA is a metaheuristic algorithm based on gravity law to solve searching problem, firstly introduce in 2009 by Rashedi [2]. GSA is based on Particle Swarm Optimization idea. The application of this algorithm varies in searching problems, such as scheduling problem in Air Traffic Control (ATC) to optimize air plane fuel, ticket price and the ATC load itself [21], in Allocation Energy problem [22], and in Production problem where GSA is said to be better than Genetics and Particle Swarm [23]. The problem of searching for Multiple Objectives is another problem that can be solved by GSA [24].

In GSA the fitness of a solution is referred to as the particle mass. Therefore, a best solution is a particle with the biggest mass which attracts other objects around it toward itself by changing their acceleration, velocity and distance.

Gravitational force of two particles is directly proportional to their mass product and inversely proportional to the square of distance.

$$F = G\frac{M_1 M_2}{R^2} \tag{1}$$

Every particle/solution (i) is represented by

$$X_i = x_i^1, \dots, x_i^d, \dots, x_i^n$$

Where, i stands for the order of the particle/solution, d is the order of particle's dimension, and n is the number of dimensions. Therefore the value of force in 't' times, against 'i' and 'j' mass is

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t)+\varepsilon}\left(x_j^d(t) - x_i^d(t)\right) \tag{2}$$

Where,

M_{aj} is the active gravitation mass connected with particle j,

M_{pi} is the passive gravitational mass connected with particle i,

G(t) is a constant gravitation at time t; the value of the first G is initialized at the beginning, and it is subtracted over time,

ε is a small constant, and

$R_{ij}(t)$ is the Euclidian distance between two particles.

$$R_{ij} = \sqrt{\sum (x_i(t) - x_j(t))^2} \tag{3}$$

In order to give a stochastic nuance, the value of total force at particle 'i' and dimension 'd' becomes

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N} rand_j F_{ij}^d(t) \tag{4}$$

Random function is given with interval 0 and 1, and based on Newton's Second Law of Motion, the acceleration function becomes:

$$a_i^d(t) = \frac{F_i^d(t)}{Mii(t)} \tag{5}$$

where $M_{ii}(t)$ is the inertia weight from particle 'i'.

The velocity and its new position become:

$$V_i^d(t+1) = rand_i \times v_i^d(t) + a_t^d(t) \quad (6)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (7)$$

The value of $M_i(t)$ is calculated by dividing average mass in a particle with all particles:

$$m_i(t) = \frac{fit_{i(t)} - worst(t)}{best(t) - worst(t)} \quad (8)$$

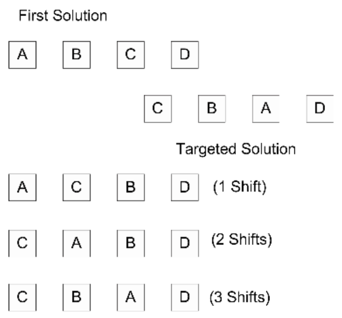
$$M_i(t) = \frac{m_i(t)}{\sum_j^N m_j(t)} \quad (9)$$

V. Adapting GSA in Combinatorial Case

GSA is based on Particle Swarm Optimization (PSO) Algorithm, while the main objective of PSO is to solve problems in a continuous environment. Hence, an adaptation is needed to convert the algorithm to fit in discrete environments. This adaptation needs to maintain the main idea of the algorithm. A similar effort to adapt continuous metaheuristic for discrete cases had also been done by Goldbarg and Yan [25][26]. Goldbarg focused on how to generate new particle in PSO by the new velocity and distance, while on the other hand Yan focused on the algorithm and strategy for the particles to be able to avoid worst solutions by using the experience of other particles. This research adapts the algorithm to edit the function and uses the Goldbarg model to generate new particle but in GSA and scheduling case.

Firstly in this exam timetable scheduling problem the solutions or particles is a combination between exam subject, room, and session which fulfill the hard-constraints, and then the fitness is calculated based on the soft constraints that have been violated. Mass is inversely proportional against fitness; with a higher violation against the soft constraints, a higher fitness is gained. In the position shift model that is proposed by Goldbarg, every element will be shifted closely to target solution with minimum cost.

Figure 1. Path Relinking (Goldbarg Et Al, 2008)



In the swarm concept the first solutions will not be directed 100% to target solutions, so the third shift on Figure 1 which shows the shifting phases in a Travelling Salesman Problem will not be executed by algorithm. The velocity and the distance will determine how many shifting will happen between two particles. In GSA the particle does not go to the best solution directly but slightly deviates due to other particles that also have a gravitational force.

The value of Euclidean distance for this case becomes the difference between one exam schedule solution to another solution. The distance has integer value since it counts the difference between two schedule solutions. Therefore, the gravity force from each particle signifies the ability of a solution to move into another position and is most affected by the heaviest mass. The multiplication operator in Formula 2 becomes the multiplication of normalized weight, and the difference between x_j with x_j is the distance between solutions in the same dimension. Furthermore, the value of F is a decimal value that shows how many elements will be exchanged in a solution. The total force (Formula 4) will summarize all gravitational force in a particle, and to gain the acceleration this total force value will be divided by its inertia mass. Acceleration in this context means acceleration to shift.

A particle should not go directly into the particle with the biggest mass. It therefore needs to develop a temporary target solution by combining a part of the biggest mass particle with a smaller part of other particles that affect this particle. After obtaining the force from all particles, then all the force values will be normalized. The force that reaches the threshold will be used to affect the temporary target solutions. The particle will not be converted into temporary target solutions 100%. The acceleration will be converted into velocity using Formula 6 and converted to distance using Formula 7. The value of distance will determine how much shift or crossover will happen from a particle to it is target solutions, and the path-relinking on Figure 1 can be implemented. Because it is not possible to convert the position in this combinatorial case into a decimal value, in this research the subtraction in Formula 1 is removed.

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon} \quad (10)$$

VI. The Adaptation Strategy In Schedulling Problem

The steps in the strategy to convert the traditional GSA to solve exam timetable scheduling problem are as follows:

Step 1 Generate the Population

Generate n random solutions without considering the hard and soft constraints. The solutions will then move or shift flexibly, but consequently the size of searching space becomes bigger.

Step 2 Calculate the fitness or mass

Calculate the fitness of the schedule by giving score to every schedule solution that has been generated based on the soft and hard constraints. There are two kinds of calculations: giving penalty for every constraint violation, and giving reward for every fulfilled constraint. This research prefers to use penalty calculation in order to monitor violations with the aim to reduce it. The calculation, however, is reversed by subtracting the high value that is not far away from best reward with its penalty. This is necessary to maintain the mass concept in GSA, that the bigger mass has better fitness.

Step 3 Initialize the GSA variable

Initialize the value of velocity. The G constant (Formula 2) will be removed from the calculation because in a combinatorial case it does not make any big impact to the shifting phase of solutions.

Step 4 Calculate the Force, the Inertia Mass and Accelerations

Calculate the gravitational force for every solutions (*j*) against every other solutions (*i*) using Formula 10. If *i* equals with *j* the distance is 0. Then calculate the inertia mass using Formula 8, and calculate the acceleration using Formula 5. If the inertia mass is less than 0 then the inertia mass is assigned with small decimal (0.01), and if the inertia mass is equal with 1 then the acceleration becomes 0.

Step 5 Determine the number of shifting

To determine the number of shifting for every solution (*j*) against every other solution (*i*) this strategy considers to use the velocities of every solution (*i*) in solution (*j*). Every solution (*i*) has the right to shift from 0 to n combinations, while a bigger n could lead the solution into conformity with the target particle quickly. Then convert the velocity into shift combinations from 0 to *n* by mapping it into a defined range for every number. As an example, shift is 0 if the velocity value is between 0-1, and shift is 1 if the value of velocity is between 1-5. This shifting number will be used to develop the deviated solutions. This step will be repeated for every solution (*j*). Furthermore, the solution that has the biggest mass will have less move, unless it has worthy competitors.

Step 6 Develop the deviated solutions as addressed solutions

Get *n* random parts from the exam schedule from every solution (*j*) that will affect the current solution (*i*). This step uses one-way evaluations, because the current solutions do not interfere the other solutions to select their part. Then, put the chosen part to the deviated schedule without redundancy.

This step will be repeated for every solution (*j*).

Step 7 Shift the current solution toward the deviated solution.

The deviated solution that has been developed will replace the corresponding part in the current solutions. This step will be repeated for every solution (*j*).

After step 7 this algorithm will be repeated from step 4 until the maximum iteration or the aimed fitness/mass value has been reached.

The fitness graph in Figure 2 shows how the fitness develop from 15 solutions for 30 iterations. The highest number of fitness is reached in the latest iteration. The peak is not always found in the last iteration, but it could also be found before. This happens because the population is still able to move the biggest mass, as long as it finds a worthy competitor, and the shifting could lead into higher or lower fitness. In the other word the fitness graph does not always have an upward trend. Furthermore, increasing the number of solutions and iterations cannot guarantee the finding of a better fitness. It does, however, improve the chance of it. Figure 3 shows the fitness development with more initial populations and iterations. The fitness or the mass score develops to more than 1700. The highest fitness is not always the same, but varies depending on the initial populations. This algorithm adaptation strategy will move every solution based on the solutions that have already been generated in the populations. The solutions would not move their exam subject if no exam subject is plotted in the resource such as room and session. As an addition, if there is no subject being plotted in the resource then there is no experience or assessment. These conditions could be an advantage or disadvantage, because every chance should be tried in the case of combinatorial problems. In the concept of GSA there is no evidence that a particle will move without being interfered by another particle.

The different results for every execution caused by the random generation in the initial population leads this research to extend an additional strategy by using multiple populations.

Figure 2 .Fitness graph with 15 solutions and 30 iterations

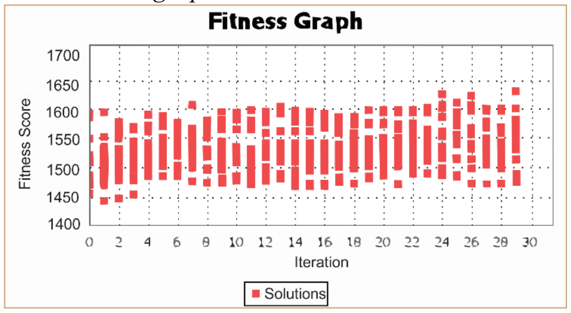
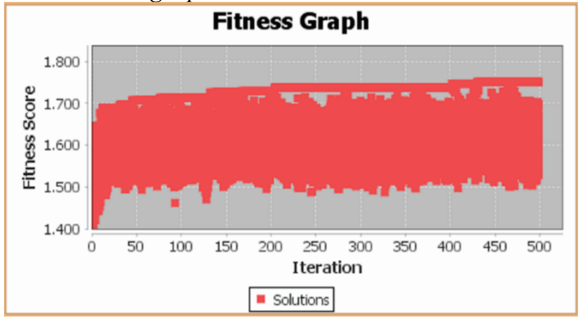


Figure 3. Fitness graph with 70 Solutions and 500 iterations



VII. THE MULTIPLE POPULATION STRATEGY

All particles in a population will affect each other, and they develop the biggest mass or fitness together. The iterations create the development pattern and have less move for the highest solutions in the last iteration (see Figure 3). However, the populations need potential input from different patterns as a different point of view to develop the solutions. This strategy is adopted from Multi Swarm Optimization that uses more than 1 swarm to develop the searching space.

Step 1 The strategy start with develop *p* population and each population will develop *n* solutions.

Step 2 Execute every population using GSA algorithm.

Step 3 When all the populations have been executed and already in convergence, inject the best solution from each population to all of the populations as additional solutions

Step 4 execute every solution again using GSA

Step 5 Get the best solutions between all of the populations

The second execution could use a different maximum iteration depending on the value of *p*. The best solution from *p* populations is the final result. This process is shown in Figure 4.

Figure 5 shows the result with 10 populations (from 0 to 9) that have been optimized using GSA, and the next populations are the 10 initial population that have been injected with the best solutions from every initial population, then executed again using GSA. The result shows a gradual increase of fitness score at the end of the process, from 1742 points as the best solution from the initial population to 1793 points as global best. Since the other populations cannot find more than 1779 points in that state, the best solution is steady. The Global Best was resulted in 9 hard-constraints violations. These violations happen as consequences for letting the hard-constraint behave like soft-constraint with bigger penalty.

The graph in Figure 6 shows the best solutions before and after being injected by the best solutions in each populations in 15 experiments. The graph shows increasing trend of fitness in most of experiments after being injected. Yet, there is an anomaly: the score did not change in the second experiment, this is due to the populations could not find the best, even after injection.

This method requires more time and computation effort, because it will execute more populations in order to escape from local optimum. If there is 10 populations in the initialization state, then it needs 10 times computation effort than single GSA with the same number of iterations, plus computation effort to execute again after being injected by best solutions from another solution. However, after reach the convergence state, GSA will get the same result for the rest of the iterations, then with the same big number of iteration between GSA and MPGSA, MPGSA will resulted in higher fitness score.

Figure 4 Flowchart of Multi Populations GSA

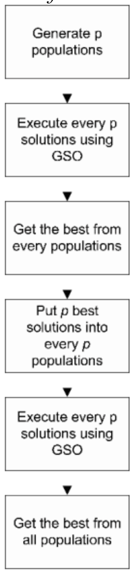

Generate p populations
Execute every p solutions using GSO
Get the best from every populations
Put *p* best solutions into every *p* populations
Execute every p solutions using GSO
Get the best from all populations

Figure 5. Fitness graph with 10 populations, 70 solutions each, 500 iteration each

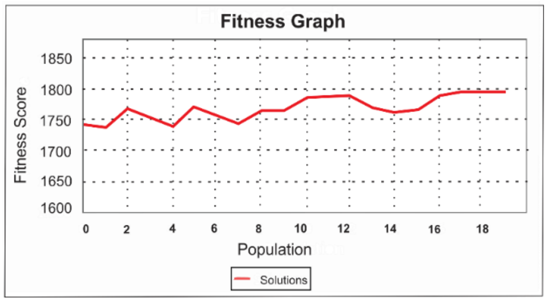
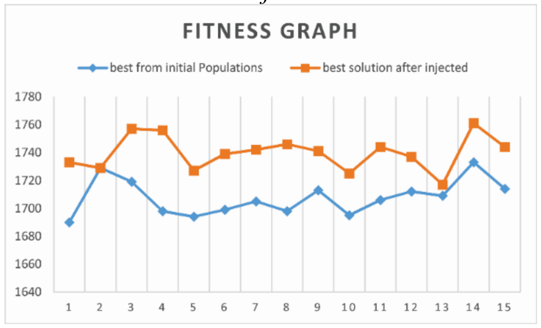


Figure 6. Fitness graph Comparison between before and after injection



VIII. CONCLUSION AND FURTHER RESEARCH

The adaptation strategy could be implemented in the exam timetable scheduling problem and resulted in improving the value of fitness/mass. Every solution will be involved with the gravitational rule based on the mass of every solution to develop the best solutions. The solutions will move their particle based on the last path of other worth population's particles and will not move to the unexplored area Although the key parts lay in the unexplored area, however, if the solutions is not good enough, it will be ignored.

The multiple populations strategy helps the development of fitness factor by providing potential solutions from other populations, and resulted in improving fitness (with an average of 32.26 points according to Figure 6), but definitely need more computation effort.

There are also some aspects that need to be examined for further research:

a. In the algorithm section, adding a strategy to move the solutions without considering another neighbor solution could be conducted in order to explore the unexplored area, even if this violates the conceptual of GSA.
b. There is a need to explore the strategy to develop offspring solutions in GSA.
The strategy of multiple populations in this research could be developed using multithreading technique, which allows boosting by using General Processing Unit (GPU). Developing another strategy and comparing it with current strategy is also needed to find the best fitness.
c. This research also could be developed to investigate which adaptation strategy is the best between the novel algorithm such cuckoo search or the modified mature traditional algorithm such as genetics algorithm.

REFERENCES

[1] Schimmelpfeng K and Helber S,"Application of a real-world university-course timetabling model solved by integer programming", *ORSpectrum* **29,** 783–803, 2007

[2] Rashedi E, Nezamabadi-pour H and Saryazdi S , "GSA: A Gravitational Search Algorithm". *Elsevier* Information Science, **179** 2232-2248, 2009

[3] Goldbarg E F G, Givanaldo R de Souza and Goldbarg M C, "Particle Swarm for the Traveling Salesman Problem*",* ed Gottlieb J and Raidl G R (Berlin Heidelberg: Springer Verlag) EvoCOP 2006, LNCS 3906 pp 99–110, 2006

[4] Razavi S F and Sajedi H, "Cognitive discrete gravitational search algorithm for solving 0-1 knapsack problem", *Journal of Intelligent & Fuzzy Systems* **29(5)** 2247-2258, 2015

[5] Weijia C U I and Yuzhu H E, "A novel hybrid approach based on a chaotic cloud gravitational search algorithm to complicated image template matching", *Turkish Journal of Electrical Engineering & Computer Sciences,* vol **25,** 4545-4557, 2017

[6] Aziz N A A, Ibrahim Z, Mubin M and Sudin S, "Adaptive switching gravitational search algorithm: an attempt to improve diversity of gravitational search algorithm through its iteration strategy", *Sadhana* Indian Academy of Sciences, vol **42(7)**, pp 1103–1121, 2017

[7] Fei S W, "Fault Diagnosis of Bearing by Utilizing LWT-SPSR-SVD-Based RVM with Binary Gravitational Search Algorithm", *Shock and Vibration*, 2018*

[8] Mirhosseini M, "A clustering approach using a combination of gravitational search algorithm and k-harmonic means and its application in text document clustering". *Turkish Journal of Electrical Engineering & Computer Sciences, Vol* **25(2),** 1251-1262, 2017

[9] Liu Q, Zhou B, Li S, Li A P, Zou P and Jia Y, "Community detection utilizing a novel multi-swarm fruit fly optimization algorithm with hill-climbing strategy", *Arabian Journal for Science and Engineering, Vol* **41(3),** 807-828, 2016

[10] Serraji M, El Amine D O and Boumhidi J, "Multi swarm optimization based adaptive fuzzy multi agent system for microgrid multi-objective energy management", *International Journal of Knowledge-based and Intelligent Engineering Systems, Vol* **20(4),** 229-243, 2016

[11] Alfarisy G A F, Mahmudy W F and Natsir M H,"Optimizing Laying Hen Diet using Multi-Swarm Particle Swarm Optimization", *TELKOMNIKA (Telecommunication Computing Electronics and Control), Vol* **16(4),** 2018

[12] Liang J J, Pan Q K, Tiejun C and Wang L,"Solving the blocking flow shop scheduling problem by a dynamic multi-swarm particle swarm optimizer", *The International Journal of Advanced Manufacturing Technology, 55.5-8, pp 755-762,* 2011 *

[13] Bulut O and Tasgetiren M C, "An artificial bee colony algorithm for the economic lot scheduling problem", *International Journal of Production Research* **52(4)** 1150–1170, 2014

[14] Shakir A, Belal A K, Shaker K and Jalab H, "The Effect of Neighborhood Structures on Tabu Search Algorithm in Solving University Course Timetabling Problem", *International Conference on Quantitative Sciences and Its Application. AIP Conf. Proc.* **1635** 657-664, 2014

[15] Teoh C K, Wibowo A and Ngadiman M S, "An Adapted Cuckoo Optimization Algorithm And Genetic Algorithm Approach To The University Course Timetabling Problem", *International Journal Of Computational Intelligence And Applications, Imperial College Press* Vol **13(1),** 1-13, 2014

[16] Yang X S and Deb S, "*Cuckoo search via Lévy flights",* World Congress on Nature & Biologically Inspired Computing IEEE Publications, pp 210–214, 2009

[17] Chen, Jeng-Fung Dan Quang Hung Do, "Training Neural Networks To Predict Student Academic Performance: A Comparison Of Cuckoo Search And Gravitational Search Algorithms", International Journal Of Computational Intelligence And Applications, Vol. 13, No. 1,Pp 1450005, 2014

[18] Alzaqebah M and Abdullah S,"Hybrid bee colony optimization for examination timetabling problems", *Computers & Operations Research, Vol* **54,** 142-154, 2015

[19] Ho W K, Lim A and Oon W C, "Maximizing paper spread in examination timetabling using a vehicle routing method *Proc", 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)* IEEE, 359-366, 2001

[20] Elsaka T,"Autonomous generation of conflict-free examination timetable using constraint satisfaction modelling", *2017 International Artificial Intelligence and Data Processing (IDAP) Symp.,* IEEE 1-10, 2017

[21] Dastgerdi K, Mehrshad N and Farshad M,"A new intelligent approach for air traffic control using gravitational search algorithm", *Sadhana* Indian Academy of Sciences, Vol **41(2),** 183–191, 2016

[22] Rahman I, Vasant P M, Singh B S M and Wadud M A A, "Intelligent Energy Allocation Strategy for PHEV Charging Station Using Gravitational Search Algorithm", *3rd International Conference on Fundamental and Applied Sciences* AIP Publishing , pp 52-59, 2014

[23] Pei J, Liu X and Pardalos P M," Application of an effective modified gravitational search algorithm for the coordinated scheduling problem in a two-stage supply chain", *International Journal of Advanced Manufacturing Technology* Springer, Vol **70,** 335–348, 2014

[24] Abbasian M A and Pour H N, "A Clustering Based Archive Multi Objective Gravitational Search Algorithm", *Fundamenta Informaticae* Ios Press, Vol **138,** 387–409, 2015

[25] Goldbarg E F G, Goldbarg M C and Souza G R, "*Particle Swarm Optimization Algorithm for the Traveling Salesman Problem",* Traveling Salesman Problem ed Federico Greco ISBN: 978-953-7619-10-7, 2008

[26] Yan X, Wu Q, Fan Y, Liang Q and Liu C, "An Improved Particle Swarm Optimization Algorithm for Traveling Salesman Problems", *International Journal of Control and Automation,* SERSC, Vol, **10(2)** 187-200, 2017