VOL. 10, NO. 22, DECEMBER, 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

AN ASSEMBLY SEQUENCE PLANNING APPROACH WITH A MULTI-STATE GRAVITATIONAL SEARCH ALGORITHM

Ismail Ibrahim[1], Zuwairie Ibrahim[1], Hamzah Ahmad[1] and Zulkifli Md. Yusof[2]
[1]Faculty of Electrical and Electronics Engineering, Universiti Malaysia Pahang, Pekan, Pahang, Malaysia
[2]Faculty of Manufacturing Engineering, Universiti Malaysia Pahang, Pekan, Pahang, Malaysia
E-Mail: pee12001@stdmail.ump.my

ABSTRACT

Assembly sequence planning (ASP) becomes one of the major challenges in product design and manufacturing. A good assembly sequence leads to reduced costs and duration in the manufacturing process. However, assembly sequence planning is known to be a classical NP-hard combinatorial optimization problem; assembly sequence planning with many product components becomes more difficult to solve. In this paper, an approach based on a new variant of the Gravitational Search Algorithm (GSA) called the multi-state Gravitational Search Algorithm (MSGSA) is used to solve the assembly sequence planning problem. As in the Gravitational Search Algorithm, the MSGSA incorporates Newton's law of gravity and the law of motion to improve solutions based on precedence constraints; the best feasible sequence of assembly can then be determined. To verify the feasibility and performance of the proposed approach, a case study has been performed and a comparison has been conducted against other three approaches based on Simulated Annealing (SA), a Genetic Algorithm (GA), and Binary Particle Swarm Optimization (BPSO). The experimental results show that the proposed approach has achieved significant improvement in performance over the other methods studied.

Keywords: combinatorial optimization problem, assembly sequence planning, meta-heuristics, multi-state gravitational search algorithm.

INTRODUCTION

The costs of assembly processes are determined by assembly plans. Assembly sequence planning, which is an important part of assembly process planning, plays an essential role in the manufacturing industry. Given a product-assembly model, assembly sequence planning (ASP) determines the sequence of component installation to shorten assembly time or save assembly costs [1]. ASP is regarded as a large-scale, highly constrained combinatorial optimization problem because it is nearly impossible to generate and evaluate all assembly sequences to obtain the optimal sequence, either with human interaction or through computer programs.

Historically, the typical combinatorial explosion problem requires experienced assembly technicians to determine assembly plans. This manual assembly planning approach thus requires significant time investments and does not allow quantitative analysis of assembly costs before production begins. Thus, many studies in the last two decades have focused on geometric reasoning capabilities and full automatism to locate more efficient algorithms for automated ASP. The approaches used for assembly sequence planning can be categorized into four groups, which are graph-based representation [2-6], lingual representation [2], an ordered list representation [7], and meta-heuristics based representation [8-10].

The implementation of meta-heuristics in solving discrete optimization problems, particularly in the ASP problem, lead to significant reductions in computation times, which in turn sacrifices the guarantee of finding exact optimal solutions [7, 11]. However, these approaches typically obtain acceptable performance at acceptable costs in a large number of possible assembly sequences; thus, these approaches have a capacity to find good solutions to large-sized problems.

In the past few years, there has been increasing interest in algorithms inspired by Newton's Law of Universal Gravitation, which states that all objects attract each other with a force of gravitational attraction. Rashedi *et al.* [12] proposed a stochastic population-based meta-heuristic algorithm based on Newton's law called the Gravitational Search Algorithm. The conventional GSA was originally designed to solve problems in continuous-value space. The GSA was successfully applied to a variety of problems including feature selection [13], data clustering and classification [14], image processing [15], data clustering and classification [14], power system [16], filter design [17], and machining process [18]. Later, Rashedi *et al.* [19] reworked the conventional GSA to create the binary gravitational search algorithm (BGSA) to allow the GSA to operate in discrete binary variables.

In this paper, a new variant of the GSA called the multi-state GSA (MSGSA) that represents each agent's vector as a state is introduced to solve discrete combinatorial optimization problems. The MSGSA is applied to generate and optimize assembly sequences of mechanical products. The purpose is to investigate the applicability of an alternative intelligent approach to the ASP.

Gravitational search algorithm

The computation of the GSA requires a set of N agents that are randomly positioned in the search space during initialization. The position of agents, which are the candidate solutions to the problem, are represented as:

$$X_i = (\boldsymbol{x}_i(1),..., \boldsymbol{x}_i(d)..., \boldsymbol{x}_i(n)) \text{ for } i = 1,2,3,...N \quad (1)$$

where $\boldsymbol{x}_i(d)$ presents the position of the i^{th} agent in the d^{th} dimension, and n is the space dimension. Figure-1 portrays the principle of the GSA. Initially, all agents are assigned a velocity $\boldsymbol{v}_i(t,d)$ that is equal to zero, where t represents the iteration number. Next, the fitness of agent i at t, $fit_i(t)$ for each agent is evaluated with respect to $\boldsymbol{x}_i(t,d)$. The gravitational constant $G(t)$ is then updated using:

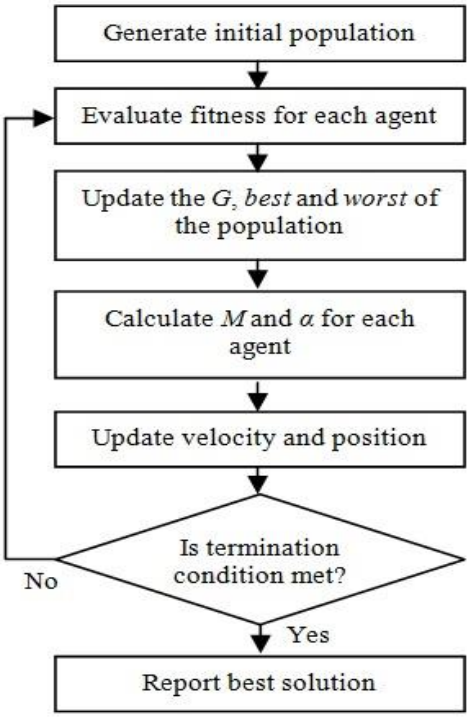

Figure-1. General principle of GSA.

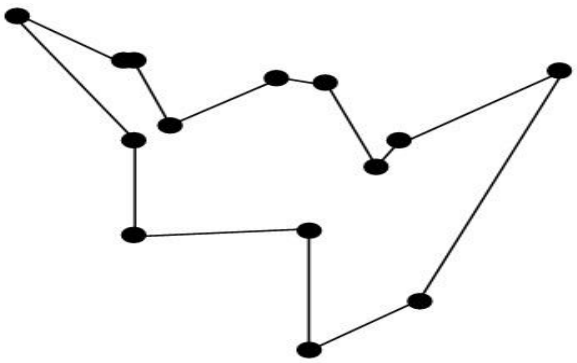

Figure-2. Burma14 benchmark instance of the travelling salesman problem (TSP).

$$G(t) = G_0 e^{-\beta \frac{t}{T}} \quad (2)$$

where T is the number of maximum iteration, G_0 and β are constant values. The gravitational constant is a decreasing function of time where it is valued to G_0 at the beginning and it is exponentially decreased towards zero as the iteration increases to control the search accuracy.

Next, *best*(t) and *worst*(t) are calculated. For the minimization problem, the definition of *best*(t) and *worst*(t) are given in Equation (3) and Equation (4):

$$best(t) = \min_{j \in \{1,....,N\}} fit_j(t) \quad (3)$$

$$worst(t) = \max_{j \in \{1,....,N\}} fit_j(t) \quad (4)$$

For the maximization problem, the definition of *best*(t) and *worst*(t) are modified to Equation (5) and Equation (6):

$$best(t) = \max_{j \in \{1,....,N\}} fit_j(t) \quad (5)$$

$$worst(t) = \min_{j \in \{1,....,N\}} fit_j(t) \quad (6)$$

The gravitational and inertial mass are then updated as:

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \quad (7)$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \quad (8)$$

where $M_i(t)$ is the inertial mass of the i^{th} agent. The acceleration, α, of mass i at t in the d^{th} dimension is calculated as:

$$\boldsymbol{\alpha}_i(t,d) = \frac{\boldsymbol{F}_i(t,d)}{M_i(t)} \quad (9)$$

where the force acting $\boldsymbol{F}_i(t,d)$ is calculated as:

$$\boldsymbol{F}_i(t,d) = \sum_{j=1, j \neq i}^{N} rand_j \, \boldsymbol{F}_{ij}(t,d) \quad (10)$$

$$\boldsymbol{F}_{ij}(t,d) = G(t) \frac{M_j(t) M_i(t)}{R_{ij}(t) + \varepsilon} \left(\boldsymbol{x}_j(t,d) - \boldsymbol{x}_i(t,d) \right) \quad (11)$$

where ε is a small constant, $R_{ij}(t)$ is the Euclidean distance between agent i and j, and $rand_j$ is a random number uniformly distributed between 0 and 1.

Afterwards, the next velocity of the agents, as given in Equation (12), are calculated as a fraction of their associated current velocity added to their associated acceleration and their next position of the agents are calculated by using Equation (13)

$$\boldsymbol{v}_i(t+1,d) = rand_i \times \boldsymbol{v}_i(t,d) + \Delta t \, \boldsymbol{\alpha}_i(t,d) \quad (12)$$

$$\boldsymbol{x}_i(t+1,d) = \boldsymbol{x}_i(t,d) + \Delta t \, \boldsymbol{v}_i(t+1,d) \quad (13)$$

where the time step Δt between the distinct time instants is assumed to be equal to unity and $rand_i$ is a random number selected from a uniform distribution between 0 and 1. The algorithm iterates until the stopping condition is met: usually the maximum number of iterations is reached or a sufficiently good fitness is obtained.

MULTI-STATE GRAVITATIONAL SEARCH ALGORITHM

The MSGSA follows a similar general principle to the original GSA with a few modifications in updating the velocity and position of each agent and formulating the calculation of force for each agent. The MSGSA is proposed to solve discrete combinatorial optimization problems.

Each agent's vector in the MSGSA is represented by a state, which is neither a continuous nor discrete value. To elaborate this state representation, the Burma14 benchmark instance of the Travelling Salesman Problem (TSP) is used as an example, as shown in Figure-2. All cities in the Burma14 benchmark instance can be represented as a collective of states, in which the states are represented by small black circles, as presented in Figure-3. A centroid of the circle shows the current state, and the radius of the circle represents the next velocity of the current state. These three elements occur in each dimension for each agent. The updating velocity and position in form of state in the MSGSA are performed after the inertial mass M and acceleration α are calculated.

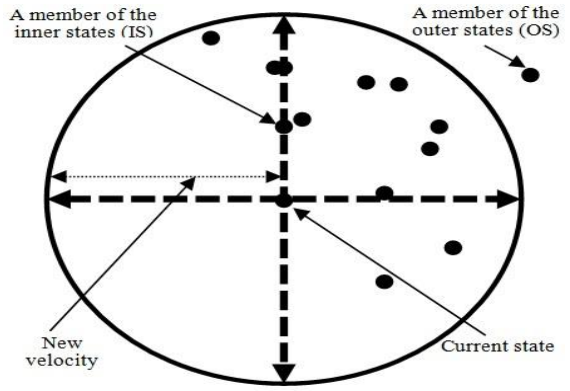

Figure-3. Illustration of the multi-state representation in the MSGSA for the Burma14 benchmark instance of TSP. Each agent's vector shows a similar representation.

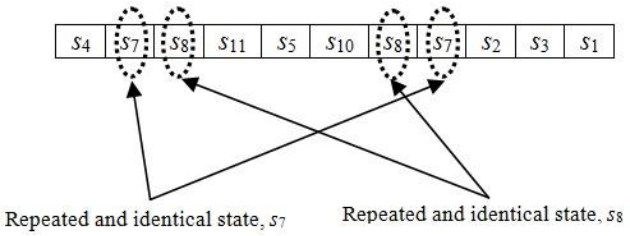

Figure-4. Example of a solution that consists of the repeated states.

In the MSGSA once the velocity is updated, the process of updating the current state to the next state for each dimension of each particle is executed. We define the current state as a centroid and the updated velocity as a radius, thus creating a circle. Any state that is located in the area of the circle is defined as a member of the inner states (IS) group. Given a set of j IS members, $\boldsymbol{I}_i(t,d) = (\boldsymbol{I}_{i_1}((t,d),...,\boldsymbol{I}_{i_j}(t,d)))$. Any state that is located outside of the area of the circle is then defined as a member of the outer states (OS) group. Given a set of l OS group members is $\boldsymbol{O}_i(t,d) = (\boldsymbol{O}_{i_1}(t,d),...,\boldsymbol{O}_{i_l}(t,d))$. Based on the current state and the new velocity of the current state, the next state can be selected as:

$$\boldsymbol{x}_i(t+1,d) = \text{random}(\boldsymbol{I}_{i_1}((t,d),..., \boldsymbol{I}_{i_j}(t,d))) \quad (14)$$

To update each position that is a state in the MSGSA, a random function as derived in Equation (14) is applied. This equation may lead to the existence of many repeated states in the updated solution. Let us consider a solution for an agent at a particular iteration that consists of 11-dimensional vector which is {s_5, s_3, s_{11}, s_2, s_8, s_9, s_{10}, s_1, s_4, s_6, s_7}. This solution has no any repeated state. Each dimension of this solution is then updated, for instance, to be a 11-dimensional vector which is {s_4, s_7, s_8, s_{11}, s_5, s_{10}, s_8, s_7, s_2, s_3, s_1}, as illustrated in Figure-4. It seems that the state in the 2nd, 3th, 7th, and 8th dimension occurs more than once. However, each state should occurs just once in each solution as in the combinatorial optimization problems such as the TSP and ASP.

To overcome the limitation of the MSGSA, an additional procedure is implemented after the velocity and state for each dimension of each agent is updated to ensure that each agent is represented by the unrepeated states. An archive is used to store components that were not used during this evolution period of the updated solution of each agent that consists of the unrepeated states. The maximum archive size is a fixed value according to how many dimensional vectors should be considered. The archive is updated by removing each state that has been previously selected.

A subtraction operation, as presented in Equation (11), is executed to calculate the difference of the vector value between the positions of two agents $\boldsymbol{x}_j(t,d)$ and $\boldsymbol{x}_i(t,d)$ for each vector and iteration, resulting in a numerical value. However, in the MSGSA, each vector's position of each agent is represented as a state. Because a state is not associated with any value, the subtraction operation in Equation (11) cannot be used to find the difference between these two positions. To accommodate the calculation of force, $\boldsymbol{F}_{ij}(t,d)$ in the MSGSA, a *cost* function $C(.)$ is introduced and incorporated into the force formulation as derived in Equation (15):

$$\boldsymbol{F}_{ij}(t,d) = G(t)\frac{M_j(t)M_i(t)}{R_{ij}(t)+\varepsilon} C\left(\boldsymbol{x}_j(t,d), \boldsymbol{x}_i(t,d)\right) \quad (15)$$

Cost may be defined as the distance and time for the ASP and TSP, respectively [10, 20]. The cost between the two states is a positive number given by $C\left(\boldsymbol{x}_j(t,d), \boldsymbol{x}_i(t,d)\right)$. In this force formulation, $R_{ij}(t)$ is the difference in fitness between agents i and j.

ASSEMBLY SEQUENCE PLANNING (ASP)

The primary objective of the ASP is to generate a feasible assembly sequence in which it will take less time to assemble, thereby reducing assembly costs. The most important factor in reducing assembly time and costs include setup time, which includes transfer time, the number of tool changes, and proper fixture selection.

In this paper, assumptions for the ASP include;

1. The setup time and the actual assembly time for each part and component are given.
2. The transfer time between workstations is included in the setup time.
3. The downtime of machines and workstations is negligible.

A precedence matrix (PM) is used to show the relationships between the components in the assembly using precedence constraints. These relationships include the nature of the connection (i.e., free or assembled components) and the relative assembly precedence between two components (i.e., *a* and *b*). If component *a* must be assembled after component *b*, $\mathrm{PM}(P_a, P_b) = 1$; otherwise $\mathrm{PM}(P_a, P_b) = \emptyset$, where (P_a, P_b) is a pair of components with geometric information in which P_a must be assembled without interfering with P_b. To decide which pair is feasible, precedence constraints for a product should be described using a PM. Assuming that γ is the set of components that have been assembled before component *a*, and the union of the PM is a feasible assembly sequence $\mathrm{FAS}(P_a, P_b)$ with constraints, then:

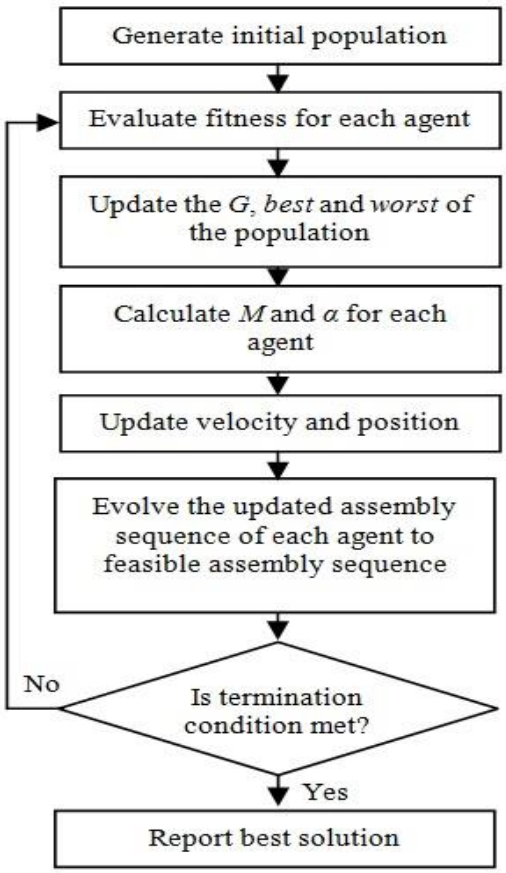

Figure-5. Outline of the proposed approach.

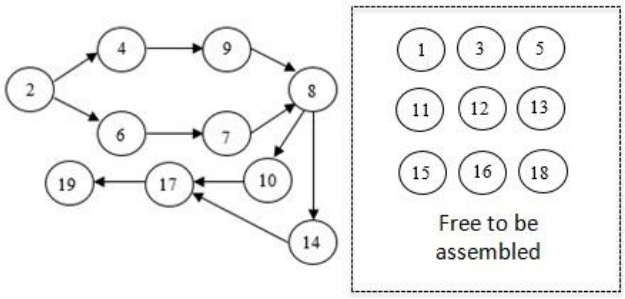

Figure-6. Assembly precedence diagram for the case study.

2	3	1	5	4

Figure-7. Example of an assembly sequence represented by an agent.

$$FAS(P_a, P_b) = \cup \mathrm{PM}(P_a, P_b), P_b \in \psi \quad (16)$$

The generation of feasible assembly sequences is explained in [10] for details.

SOLVING ASSEMBLY SEQUENCE PLANNING PROBLEM USING MULTI-STATE GSA

Figure-5 shows the outline of the proposed approach based on the MSGSA. To search an optimal solution, each agent must be evaluated to measure its fitness value. The evaluation of fitness is performed after the initial population is generated and the PM, coefficient table and actual assembly times are loaded. The gravitational constant *G*, the *best* and the *worst* of the population are then updated. The mass and acceleration for each agent is then calculated. Next, the velocity and position for each agent is updated. The updated assembly sequence of each agent is then evolved to feasible assembly sequence. Occasionally, some respective assembly components cannot be integrated into a feasible assembly sequence. The determination of the assembly components that do not correspond to a feasible assembly sequence is achieved by satisfying all PM constraints between the components in the assembly, which are determined earlier, either from CAD or a disassembly analysis [4]. As a result, each agent produces a feasible assembly sequence. The optimum sequence is then selected from the feasible assembly sequences by evaluating the fitness of each agent using Equation (3) because the ASP is a minimization problem. The best of the population is the sequence that is more optimal up until the stopping condition is met. After the stopping condition is met, the performance of the proposed approach based on the MSGSA can be investigated.

The assembly of a hypothetical product with 19 components is considered according to [8-9] and its associated coefficient table is outlined in [10]. Figure-6 shows the precedence diagram. In this diagram, the components that are free to be assembled are the

components that can be placed regardless of any part of a sequence.

During initialization, the initial population is randomly generated. Second, the generated initial population is subjected to a validity check using the PM described in the previous section to ensure the initial population is valid and feasible. The vector representation corresponding to the assembly components of an agent is represented in Figure-7; in this case, the sequence is 2-3-1-5-4. The length of a string depends on the total number of components used in the assembly process.

The evaluation procedure is performed to improve the objective value. Hence, for each iteration, each agent represented by a feasible assembly sequence is evaluated. The agent with the highest objective value for current iteration is then compared with the agent with the highest objective value for previous iterations. If the agent with the highest objective value for the current iteration is better than the agent with the highest objective value for previous iterations, the agent with the highest objective value for the current iteration is updated to be the agent with the highest objective value for all iterations.

To evaluate the fitness for each agent, the total assembly time should be found. The total assembly time is the combination of the setup time and the actual assembly time. It is assumed that regardless of the assembly sequence, the actual assembly time is constant, and a proper tool and setup for each component to be assembled is required. These two items depend on the geometry of the component itself and the components assembled up to that point. The setup time for a component can be predicted using Equation (17) [9]:

$$Time_{Setup}(a) = p_{a0} + \sum_{b=1}^{c} p_{ab} q_{ab} \quad (17)$$

where (a) is the component to be assembled; p_{a0} is the setup time with product (a) being the first component; p_{ab} is the contribution to the setup time due to the presence of part (b) when entering part (a); and $q_{ab} = 1$ if component (b) has already been assembled and $q_{ab} = 0$ otherwise for a = 1, 2,…, c. where c is the number of components in the ASP.

The total assembly time is the summation of the setup time and the actual assembly time. Because the objective in this work to minimize assembly costs and time, the fitness function for minimizing the assembly time should be calculated by:

$$\text{Min } Time_{Assembly} = \sum_{b=1}^{c} \left(Time_{Setup}(a) + A_a \right) \quad (18)$$

where A_a is the assembly time for component a. The calculation of time is in time units.

To confirm the production of a feasible assembly sequence of each agent, the updated assembly sequence of each agent produced by the updating position process is evolved to a feasible assembly sequence. To assemble the components of the product in a valid manner, only the feasible assembly sequences should be used. The feasibility of the sequences can be determined by referring to the PM.

The PM gives information which position of each sequence should be swapped randomly (an infeasible component and a feasible component). For instance, the updated assembly sequence and the swap position between two components are shown in Figure-8. The swapping process ends when each component occurs in an assembly sequence in which the sequence is now feasible.

EXPERIMENTAL RESULTS

Table-1 demonstrates the best results and theirs associated assembly sequences of the proposed approach based on the MSGSA and the approach based on SA [8], GA [9], and BPSO [10]. To simplify the understanding of this work, fitness or objective value is now called total assembly time and feasible assembly sequence is the solution.

The success of the MSGSA is heavily depend on setting of control parameters namely; constant β, initial gravitational constant G_0, number of agents *NOA* and number of iteration T. These control parameters should be carefully selected when using the MSGSA in order to know the best parameters, so a successful implementation of the algorithm can be achieved. A series of experiments are carried out to tune the MSGSA best parameters for the assembly sequence planning problem. It is clear from results shown in Table-1 that the best parameters for constant β, initial gravitational constant G_0, number of agents *NOA* and number of iteration T are β = 20, G_0 = 100, *NOA* = 30, T = 500 respectively. The best objective value obtained for these parameters is 508.3. The assembly sequence generated for the best objective value using these parameters is 1-2-4-3-9-12-13-5-16-15-18-11-6-7-8-14-10-17-19. The result clearly shows that the MSGSA successfully provides the best assembly sequence compared to SA, GA, and BPSO. Convergence pattern of the best assembly sequence obtained by the MSGSA is then portrayed in Figure-9. It seems that the MSGSA converges fast at iteration 228.

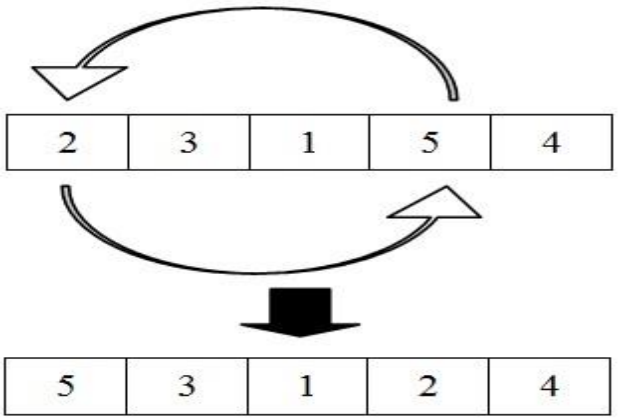

Figure-8. Swap position between two components in a sequence.

Table-1. Best results and their associated assembly sequences of the proposed approach based on the MSGSA and the approach based on SA, GA, and BPSO.

Algorithm	Total Assembly Time	Assembly Sequence
MSGSA	508.3	1-2-4-3-9-5-12-13-15-16-18-11-6-7-8-14-10-17-19
SA	528.7	2-1-4-9-3-12-13-16-5-15-18-6-11-7-8-10-14-17-19
GA	524.1	2-18-3-12-1-13-16-5-11-15-4-6-9-7-8-10-14-17-19
BPSO	514.4	16-2-13-4-1-15-11-9-6-5-18-7-8-14-12-10-3-17-19

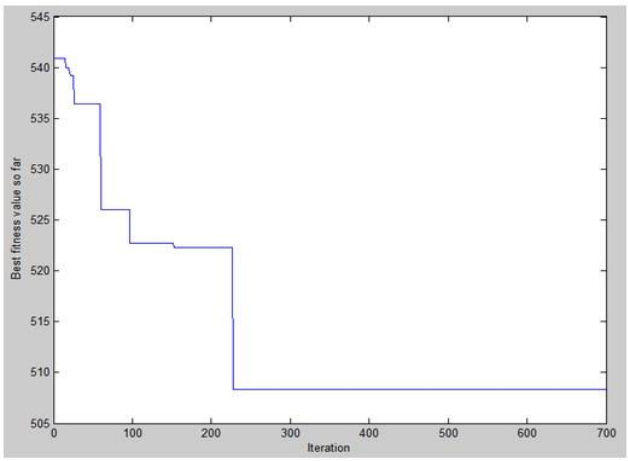

Figure-9. Convergence pattern of the best assembly sequence for the MSGSA in solving the ASP.

CONCLUSIONS

This paper presents an approach based on the multi-state gravitational search algorithm to solve the assembly sequence planning problem. Three new modifications of the original GSA (i.e., updating the velocity, updating the position, and the force formulation) work efficiently together with the general principle of the original GSA to find the optimal or nearly optimal assembly sequence of a mechanical product. To show the relations between components in the ASP, the PM assembly has been used. To evaluate the performance of the proposed approach, a case study of the ASP consisting of 19 components is examined, and the performance of the proposed approach based on the MSGSA is evaluated against three different approaches that use SA, GA, and BPSO. The experimental results obtained shows that the proposed approach outperforms the three other approaches in obtaining the best assembly sequence. In the future, we may examine the performance of this approach with other constraints in assembly sequence planning, including assembly stability, machine and workstation assignment, and workload.

ACKNOWLEDGEMENTS

This work is financially supported by the Ministry of Higher Education Malaysia through the Fundamental Research Grant Scheme (RDU140114) and Universiti Malaysia Pahang's Post Graduate Research Scheme (GRS140364).

REFERENCES

[1] Lv H. G. and Lu C. 2010. An assembly sequence planning approach with a discrete particle swarm optimization algorithm. The International Journal of Advanced Manufacturing Technology. Vol. 50, pp. 761–770.

[2] L. S. H. d. Mello and C. D. Arthur. 1990. And/ or graph representation of assembly plans. IEEE Transaction on Robotics and Automation. Vol. 6, pp. 188–199.

[3] W. Zhang. 1989. Representation of assembly and automatic robot planning by petri net. IEEE Transaction on System, Man, and Cybernetics. Vol. 19, pp. 418–422.

[4] S. Lee and Y. G. Shin. 1990. Assembly planning based on geometric reasoning. Computers and Graphics. Vol. 14, No. 2, pp. 237–250.

[5] E. K. Moore, G. As¸ kıner and M. G. Surendra. 2001. Petri net approach to disassembly process planning for products with complex and/or precedence relations. European Journal of Operational Research. Vol. 135, No. 2, pp. 428–449.

[6] X. F. Zha. 2000. An object-oriented knowledge based petri net approach to intelligent integration of design and assembly planning. Artificial Intelligence in Engineering. Vol. 14, No. 1, pp. 83–112.

[7] W. Garrod and L. J. Everett. 1990. A.S.A.P.: automated sequential assembly planner. In: Computers in Engineering Conference. G. L. Kinzel Smrophe (Ed.). pp. 139–150.

[8] S. Motavalli and A. Islam. 1997. Multi-criteria assembly sequencing. Computers and Industrial Engineering. Vol. 32, No. 4, pp. 743–751.

[9] Y. K. Choi, D. M. Lee and Y. B. Cho. 2008. An approach to multi-criteria assembly sequence planning using genetic algorithms. International Journal of Advanced Manufacturing Technology. Vol. 42, pp. 180–188.

[10] J. A. A. Mukred, Z. Ibrahim, I. Ibrahim, A. Adam, K. Wan, Z. M. Yusof and N. Mokhtar. 2012. A binary particle swarm optimization approach to optimize

assembly sequence planning. Advanced Science Letter. Vol. 13, No. 1, pp. 732–738.

[11] C. Blum and A. Roli. 2003. Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Computing Surveys. Vol. 35, No. 3, pp. 268–308.

[12] E. Rashedi, H. Nezamabadi-pour and S. Saryazdi. 2009. GSA: a gravitational search algorithm. Information Sciences. Vol. 179, No. 13, pp. 2232–2248.

[13] M. Sheikhan and M. S. Rad. 2012. Gravitational search algorithm–optimized neural misuse detector with selected features by fuzzy grids–based association rules mining. Neural Computing and Applications. Vol. 7, No. 23, pp. 2451–2463.

[14] Y. Kumar and G. Sahoo. 2014. A review on gravitational search algorithm and its applications to data clustering and classification. International Journal of Intelligent Systems and Applications. Vol. 6, No. 6, pp. 79–93.

[15] V. Kumar, J. K. Chhabra and D. Kumar. 2014. Automatic cluster evolution using gravitational search algorithm and its application on image segmentation. Engineering Applications of Artificial Intelligence. Vol. 29, pp. 93–103.

[16] R. K. Sahu, U. K. Rout and S. Panda. 2013. Automatic generation control of multi-area power system using gravitational search algorithm. In: Swarm, Evolutionary Memetic Computing: 4th Joint International Conference. B. K. Panigrahi, P. N. Suganthan, S. Das and S. S. Dash (Eds.). pp. 537-546.

[17] S. K. Saha, R. Kar and D. Mandal. 2013. Design and simulation of FIR band pass and band stop filters using gravitational search algorithm. Memetic Computing. Vol. 4, No. 5, pp. 311–321.

[18] M. Hrelja, S. Klancnik, J. Balic and M. Brezocnik. 2014. Modelling of a turning process using the gravitational search algorithm. International Journal of Simulation Modelling. Natural Computing. Vol. 13, No. 1, pp. 30–41.

[19] E. Rashedi, H.Nezamabadi-pour and S. Saryazdi. 2010. BGSA: Binary gravitational search algorithm. Natural Computing. Vol. 9, No. 3, pp. 727–745.

[20] A. Stentz. 1994. Optimal and efficient path planning for partially-known environments. In: Robotics and Automation: IEEE International Conference. M. H. Hebert, C. Thorpe, and A. Stentz (Eds.). pp. 3310–3317.