An assembly sequence planning approach with a rule-based multi-state gravitational search algorithm

Ismail Ibrahim · Zuwairie Ibrahim · Hamzah Ahmad · Mohd Falfazli Mat Jusof · Zulkifli Md. Yusof · Sophan Wahyudi Nawawi · Marizan Mubin

Received: 28 August 2014 / Accepted: 27 January 2015 / Published online: 3 March 2015

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; ASP 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 rule-based multi-state gravitational search algorithm (RBMSGSA) is used to solve the assembly sequence planning problem. As in the gravitational search algorithm, the RBMSGSA incorporates Newton's law of gravity, the law of motion, and a rule that makes each assembly component of each individual solution occur once 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.

I. Ibrahim · Z. Ibrahim (✉) · H. Ahmad · M. F. M. Jusof
Faculty of Electrical and Electronics Engineering, Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia
e-mail: zuwairie@ump.edu.my

I. Ibrahim
e-mail: pee12001@stdmail.ump.edu.my

H. Ahmad
e-mail: hamzah@ump.edu.my

M. F. M. Jusof
e-mail: falfazli@ump.edu.my

Z. M. Yusof
Faculty of Manufacturing Engineering, Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia
e-mail: zmdyusof@ump.edu.my

S. W. Nawawi
Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia
e-mail: sophan@fke.utm.my

M. Mubin
Faculty of Engineering, Universiti Malaya, 50603 Kuala Lumpur, Malaysia
e-mail: marizan@um.edu.my

Keywords Combinatorial optimization problem · Assembly sequence planning · Meta-heuristic · Multi-state gravitational search algorithm

1 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. 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 ASP can be categorized into four groups, which are as follows:

1. Graph-based representation: In this representation, the data source originates from the user or a CAD system. Using this approach, many details of the assembly analysis can be determined. Mello and Authur [1] and Zhang [2] proposed graph-based representation methods based on AND/OR and directed graph. Lee and Shin [3], Moore et al. [4], and Zha [5] proposed graph-based representation methods based on Petri nets.
2. Lingual representation: This representation uses a special language to represent subassemblies, their parts, and the relations between them. A few approaches include but are not restricted to PADL, AUTOPASS, and GDP [1].
3. An ordered list representation: This type of representation can be categorized as an ordered list of task representations, binary vectors, partitions of the set of parts, and connections. Garrod and Everett [6] represented each assembly sequence in the form of a set of list.
4. Meta-heuristic-based representation: Meta-heuristic approaches include but are not restricted to the rule-based method [7], heuristic search [8], neural networks [8–10], genetic algorithms [11–17], simulated annealing [18, 19], ant colony optimization [20], memetic algorithms [21], particle swarm optimization [22–24], and hybrid methods [25, 26].

The implementation of meta-heuristics in solving discrete optimization problems, particularly in the ASP problem, leads to significant reductions in computation times, which in turn sacrifices the guarantee of finding exact optimal solutions [27, 28]. 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. [29] proposed a stochastic population-based meta-heuristic algorithm based on Newton's law called the gravitational search algorithm. The conventional gravitational search algorithm (GSA) was originally designed to solve problems in continuous-value space. Later, Rashedi et al. [30] 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 rule-based multi-state GSA (RBMSGSA) that embeds a rule to represent each agent's vector as an unrepeated state is introduced to solve discrete combinatorial optimization problems. The RBMSGSA is applied to generate and optimize assembly sequences of mechanical products. The purpose of this algorithm and thus this study is to investigate the applicability of an alternative intelligent approach to the ASP.

The organization of this paper is as follows: Section 2 explains the original GSA; Section 3 describes the concept of the RBMSGSA; Section 4 presents the description of the problem; in Section 5, the implementation of the proposed ASP approach with the RBMSGSA is described; Section 6 provides the experimental results of the RBMSGSA compared to SA, GA, and BPSO; and Section 7 offers general conclusions.

2 GSA

In the GSA, agents are considered as objects and their associated performances are measured by their masses. All these objects attract each other by the gravity force, and this force causes a global movement of all objects, the masses unite using a direct form of connection, using gravitational force, an agent with heavy mass corresponds to good solution and moves slowly than the lighter mass.

This guarantees exploitation step of the algorithm. Gravitational and inertial masses are eventually determined using a fitness function.

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

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

where $\mathbf{x}_i(d)$ presents the position of the ith agent in the dth dimension, and n is the space dimension. Figure 1 portrays the principle of the GSA. Initially, all agents are assigned a velocity $\mathbf{v}_i(t,d)$ 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 $x_i(t)$. The gravitational constant $G(t)$ is then updated using

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

where T is the number of maximum iterations, and G_0 and β are constant values. The gravitational constant is a decreasing function with time where it is valued to G_0 at the beginning and exponentially decreases towards zero as the number of iterations increases to control search accuracy. Next, best(t) and worst(t) are calculated.

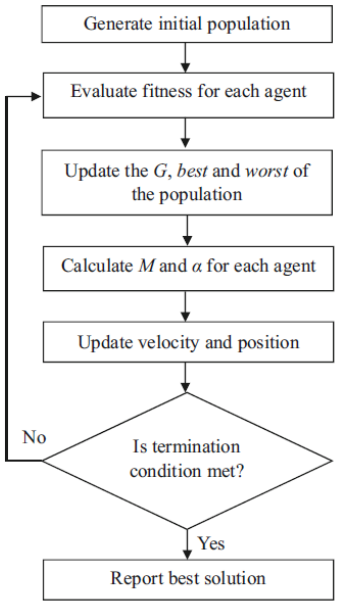

Fig. 1 General principle of GSA

For the minimization problem, the definitions of best(t) and worst(t) are given as follows:

$$\text{best}(t) = \min_{j\in\{1,\ldots,N\}} \text{fit}_j(t) \tag{3}$$

$$\text{worst}(t) = \max_{j\in\{1,\ldots,N\}} \text{fit}_j(t) \tag{4}$$

For the maximization problem, the definitions of best(t) and worst(t) are modified to the following:

$$\text{best}(t) = \max_{j\in\{1,\ldots,N\}} \text{fit}_j(t) \tag{5}$$

$$\text{worst}(t) = \min_{j\in\{1,\ldots,N\}} \text{fit}_j(t) \tag{6}$$

The gravitational and inertial masses are then updated as follows:

$$m_i(t) = \frac{\text{fit}_i(t) - \text{worst}(t)}{\text{best}(t) - \text{worst}(t)} \tag{7}$$

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

where $M_i(t)$ is the inertial mass of the ith agent.

The acceleration, α, of mass i at t in the dth dimension is calculated as follows:

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

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

$$\boldsymbol{F}_i(t,d) = \sum_{j=1,j\neq i}^{N} \text{rand}_j \boldsymbol{F}_{ij}(t,d) \tag{10}$$

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

where ε is a small constant, $R_{ij}(t)$ is the Euclidean distance between agents 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 Eq. 12) is calculated as a fraction of their associated current velocity added to their associated acceleration, and their next position of the agents is calculated by using Eq. 13

$$\boldsymbol{v}_i(t+1,d) = \text{rand}_i \times \boldsymbol{v}_i(t,d) + \Delta t\, \boldsymbol{\alpha}_i(t,d) \tag{12}$$

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

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.

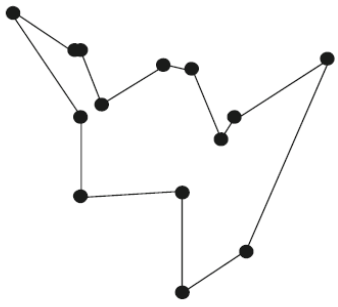

Fig. 2 Burma14 benchmark instance of the Travelling Salesman Problem (TSP)

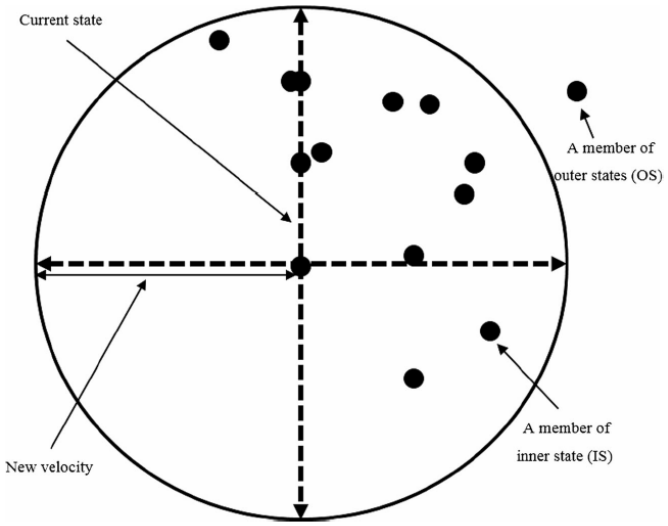

Fig. 3 Illustration of the multi-state representation in the RBMSGSA for the Burma14 benchmark instance of TSP. Each agent's vector shows a similar representation

3 RBMSGSA

The RBMSGSA is explained in this section. The RBMSGSA 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.

3.1 Updating velocity and position of each agent in the RBMSGSA

Each agent's vector in the RBMSGSA 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 Fig. 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 Fig. 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 RBMSGSA are performed after the inertial mass M and acceleration α are calculated.

In the RBMSGSA 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, $\mathbf{I}_i(t,d) = (\mathbf{I}_{i_1}((t,d), \ldots, \mathbf{I}_{i_j}(t,d)))$.. 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, $\mathbf{O}_i(t,d) = (\mathbf{O}_{i_1}(t,d), \ldots, \mathbf{O}_{i_l}(t,d))$, and a set of h selected states (SS) group members is described by $\mathbf{S}_i(t,d) = (\mathbf{S}_{i_1}(t,d), \ldots, \mathbf{S}_{i_h}(t,d))$. The SS group is defined as a group consisting of states that have been selected as the next states from the IS and OS groups. All members of the SS are not allowed to be selected as the next state because these states have been selected in previous selections.

Based on the current state and the updated velocity of the current state, the next state can be selected as:

$$\boldsymbol{x}_i(t+1,d) = \begin{cases} \text{random}\left(\boldsymbol{I}_i(t,d) - \left(\left(\boldsymbol{I}_i(t,d) \bigcap \boldsymbol{S}_i(t,d)\right)\right)\right) \\ \quad \text{if } \boldsymbol{I}_i(t,d) - \left(\left(\boldsymbol{I}_i(t,d) \bigcap \boldsymbol{S}_i(t,d)\right)\right) \neq \varnothing \\ \text{random}\left(\boldsymbol{O}_i(t,d) - \left(\left(\boldsymbol{O}_i(t,d) \bigcap \boldsymbol{S}_i(t,d)\right)\right)\right) \\ \quad \text{if } \boldsymbol{I}_i(t,d) - \left(\left(\boldsymbol{I}_i(t,d) \bigcap \boldsymbol{S}_i(t,d)\right)\right) = \varnothing \end{cases} \tag{14}$$

where $\varnothing$ is an empty set.

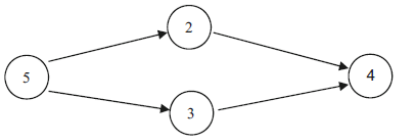

Fig. 4 Example of a precedence diagram

Table 1 Precedence matrix (PM) based on Fig. 2

Component *a*	Component *b*			
	5	2	3	4
5				
2	1			
3	1			
4	1	1	1	

3.2 Force formulation in the RBMSGSA

A subtraction operation, as presented by Eq. 11, is executed to calculate the difference of the vector value between the positions of two agents $\mathbf{x}_j(t,d)$ and $\mathbf{x}_i(t,d)$ for each vector and iteration, resulting in a numerical value. However, in the RBMSGSA, 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 Eq. 11 cannot be used to find the difference between these two positions. To accommodate the calculation of force, $\mathbf{F}_{ij}(t,d)$ in the RBMSGSA, a *cost* function $C(.)$ is introduced and incorporated into the force formulation as follows:

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

Cost may be defined as the distance and time for the ASP and TSP, respectively [23, 31]. The cost between the two states is a positive number given by $C(\mathbf{x}_j(t,d),\mathbf{x}_i(t,d))$. In this force formulation, $R_{ij}(t)$ is the difference in fitness between agents i and j. Let us say agents i and j can be represented into two sequences: 2-1-4-3-5 and 1-3-2-5-4. The fitness values of the first and second sequences are 4.5 and 5.6. The difference in fitness is 1.1, resulting from the absolute value of the subtraction between these two fitness values.

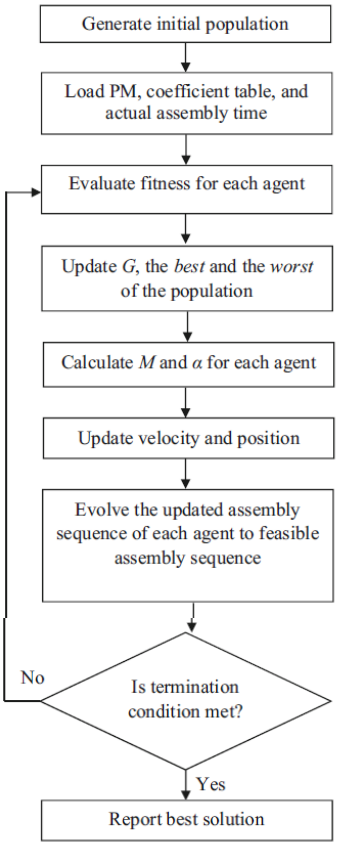

Fig. 5 Outline of the proposed approach

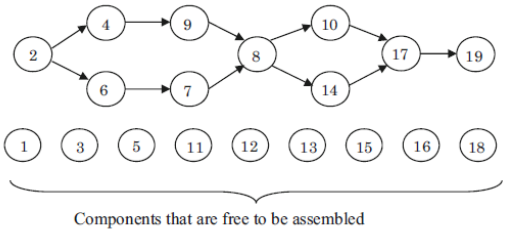

Fig. 6 Assembly precedence diagram for the case study

4 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 the following:

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.

2	3	1	5	4

Fig. 7 Example of an assembly sequence represented by an agent

Link to full text articles :

http://link.springer.com/article/10.1007/s00170-015-6857-0