# Assembly Sequence Planning with Fireworks Algorithm

Jun-Ying Li and Cong Lu

***Abstract*—This paper investigates an assembly sequence planning approach with fireworks algorithm. To solve the problem of assembly sequence planning, the relevant operations of fireworks algorithm are discussed, and the frequency of assembly orientation change, the frequency of assembly tool change, and the frequency of operation change are taken into account in the fitness function of assembly sequence planning. The steps of fireworks algorithm for assembly sequence planning are investigated and finally a case study is given to verify the algorithm.**

***Index Terms*—Fireworks algorithm, assembly sequence planning, multi-objective optimization.**

## I. Introduction

Assembly planning consists of three parts, including assembly modeling, assembly sequence planning, and assembly path planning. The key of the assembly planning is the assembly sequence planning, which directly affects the assembly cost of the products. As the number of the assembly sequence increases exponentially when the components of the product increase, so the complex products which have a lot of components will cause the combinatorial explosion problem about assembly sequence planning. Therefore, the method of assembly sequence planning will fall into the class of nondeterministic polynomial bounded hard problems [1].

The traditional methods [2], [3] are difficult to solve assembly sequence planning for complex products. In recent 20 years, many scholars have applied the intelligent optimization algorithms to the assembly sequence planning, such as genetic algorithm (GA) [4], particle swarm optimization (PSO) [5], simulated annealing algorithm and ant colony algorithm (AC) [6], which provided the powerful methods to obtain the assembly sequence of the complex products. The efficiency of the process to search feasible assembly sequence is proved by the intelligent optimization algorithm.

Fireworks algorithm was firstly proposed by Tan and Zhu according to the natural phenomenon of the spark fireworks explosion in 2010 [7]. Fireworks algorithm with the ability of local search and global search has good optimization performance, and the searching mechanism is different with the other algorithms, which makes it searching the solution space more thoroughly. In this paper, to solve the problem of assembly sequence planning, the firework algorithm is applied in assembly sequence planning for the first time, to find the optimal or near-optimal assembly sequence of the product.

Manuscript received May 3, 2016; revised June 1, 2016.

The authors are with the School of Mechatronics Engineering, University of Electronic Science and Technology of China, Chengdu, China (e-mail: conglu@uestc.edu.cn).

## II. Principle of Fireworks Algorithm

In the fireworks algorithm [7], n initial locations are first selected, where n fireworks are set off. After the explosion of the fireworks, the locations of sparks can be obtained, and the quality of the locations can be evaluated. The algorithm should stop when the optimal location is found; otherwise, n current locations should be selected from the fireworks and sparks for the next generation of explosion. Some key technologies of the fireworks algorithm are discussed as follows:

$$s_i = m \cdot \frac{y_{\max} - f(x_i) + \xi}{\sum_{i=1}^{n}(y_{\max} - f(x_i)) + \xi} \tag{1}$$

where $m$ represents a parameter which controls the total number of sparks generated by the $n$ fireworks, $y_{\max} = \max(f(x_i))\ (i = 1, 2, \cdots, n)$ stands for the firework with the maximum (worst) value of the objective function, and $\xi$ represents the smallest constant in the computer, which is used to avoid zero-division-error.

The explosion amplitude of each firework is calculated as (2):

$$A_i = \hat{A} \cdot \frac{f(x_i) - y_{\min} + \xi}{\sum_{i=1}^{n}(f(x_i) - y_{\min}) + \xi} \tag{2}$$

where $\hat{A}$ is a parameter that represents the maximum amplitude of explosion, $y_{\min} = \min(f(x_i))\ (i = 1, 2, \cdots, n)$ stands for the firework with the minimum (best) value of the objective function., and $\xi$ is the same as defined in (1).

The sparks generating process of the *i*-th firework is as follows:

Suppose the sparks can be generated in the explosion from random *z* directions, then *z* can be represented as (3):

$$z = round(d \cdot rand(0,1)) \tag{3}$$

where, *d* is the dimensionality of the location of the firework $x_i$, rand (0,1) is a random number following an uniform distribution over [0,1].

Then initialize the location of the j-th spark $\tilde{x}_j$, randomly select z dimensions of $\tilde{x}_j$, for each dimension k, k $\in$ {random selected z dimensions of $\tilde{x}_j$ }, the kth dimensional coordinate of the j-th spark $\tilde{x}_j$ in the i-th firework $x_i$ can be calculated

DOI: 10.7763/IJMO.2016.V6.526

as (4):

$$\tilde{x}_j^k = \tilde{x}_j^k + A_i \cdot rand(-1,1) \tag{4}$$

where $rand(-1,1)$ is a uniform distribution over [-1, 1].

In order to increase the diversity of the sparks, Gaussian explosion is used as another method of sparks generating in the fireworks algorithm. With this method, the specific sparks will be generated in each explosion generation as follows:

Suppose the sparks can be generated in the explosion from random z directions as (3), then initialize the location of the *j*-th spark $\hat{x}_j$, randomly select *z* dimensions of $\hat{x}_j$, for each dimension *k*, $k \in$ {random selected z dimensions of $\hat{x}_j$}, the kth dimensional coordinate of the *j*-th spark in the *i*-th firework can be obtained as (5):

$$\hat{x}_j^k = \hat{x}_j^k \cdot Gaussian(1,1) \tag{5}$$

where $Gaussian(1,1)$, which is the coefficient of the explosion, is a random number following a Gaussian distribution with mean 1 and standard deviation 1.

Based on the locations of the sparks obtained with the above methods, n sparks are selected from all of the sparks and the fireworks based on the concentration principle, for the next generation of explosion.

In fireworks algorithm, elite strategy [8] is used to update the location of the new sparks with (6):

$$X_i^{'} = \eta X_i + (1-\eta) bestX, \ (i = 1,2,\cdots,n) \tag{6}$$

where *bestX* is the best location of the spark in the current generation, $\eta$ is a random number generated in the range [0, 1].

More detailed description of the fireworks algorithm can be referred to the literature [7].

## III. Application of Fireworks Algorithm In Assembly Sequence Planning

### A. Modeling the Assembly Sequence with Fireworks Algorithm

The solution space of assembly sequence planning is a discrete integer solution space, and the assembly sequence planning can be regarded as the fireworks explosion process, then the assembly sequence can be modeled with the fireworks algorithm as follows:

For a product including n components, an assembly sequence of the product can be modeled as the location of the firework or the spark, which is defined as (7):

$$X_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,j}, \cdots x_{i,n}) \tag{7}$$

where $x_{i,j}$ is the number of the part, $x_{i,j} \in \{1,2,\cdots,n\}$, and any two elements in vector $X_i$ are not equal.

The fireworks algorithm in the literature [7] solves the minimum value of the fitness function, but the fitness function in this paper solves the maximum. The explosion amplitude of each firework and the number of the sparks are improved as (8) and (9).

$$A_i = \hat{A} \cdot \frac{y_{\max} - f(x_i) + \xi}{\sum_{i=1}^{n}(y_{\max} - f(x_i)) + \xi} \tag{8}$$

$$s_i = m \cdot \frac{f(x_i) - y_{\min} + \xi}{\sum_{i=1}^{n}(f(x_i) - y_{\min}) + \xi} \tag{9}$$

The parameters in the (8) and (9) are the same as (2) and (1).

In the fireworks algorithm, $n$ fireworks or sparks should be selected for the each firework explosion generation. The operational approach is referred to the literature [7].

### B. Building the Fitness Function of Assembly Sequence Planning

Assembly sequence planning is the typical multi-objective optimization problem. In this paper, in order to reduce the assembly instability and the assembly costs, the number of assembly orientation changes, the number of the assembly operation changes and the number of assembly tool changes are taken into consideration [8]. If the assembly sequence is feasible, the fitness function of the assembly sequence is given as (10):

$$F = 2S - w_t n_t - w_{or} n_{or} - w_{op} n_{op} \tag{10}$$

where, $S$ stands for the total number of components in a product or an assembly, $n_{or}$ is the number of assembly orientation changes, $n_{op}$ is the number of the assembly operation changes, $n_t$ is the number of assembly tool changes; $w_t, w_{or}, w_{op} \in [0,1]$ are the weights of the above three objectives, respectively, and $w_t + w_{or} + w_{op} = 1$.

However, if the assembly sequence is infeasible, the fitness function of the assembly sequence is given as (11):

$$F = (2S - w_t n_t - w_{or} n_{or} - w_{op} n_{op}) / (2m) \tag{10}$$

where *m* stands for the interference times during one assembly sequence.

Based on the above definition, the fitness function value of the fireworks can be concluded with (10) or (11).

The fitness function ensures that the higher fitness values, the better assembly sequence, the more probability to be selected. On the contrary, the lower fitness values, the worse assembly sequence, the less probability to be selected.

### C. Framework of Assembly Sequence Planning with Fireworks Algorithm

The framework of assembly sequence planning with fireworks algorithm is proposed as follows:

Step 1 Set the number of components in assembly sequence

as $n$; set the weights of the fitness function; build the interference matrix; set the number of initial fireworks as $n_{pop}$; set the parameter m controlling the total number of sparks generated by the fireworks; set the maximal amplitude of explosion $\hat{A}$; set the constant parameters $a$ and $b$; set the iteration number T=1 and the maximal iteration number $T_{\max}$;

Step 2 Randomly generate the initial locations of the fireworks, which are represented as the assembly sequence with (7);

Step 3 For each firework, calculate the explosion amplitude $A_i$ and the number of sparks $S_i$, then update the locations of the ordinary sparks with (4);

Step 4 Calculate the fitness function value of each firework and spark, conclude the best location of the current firework and sparks (optimal assembly sequence), update the location of the new sparks with elite strategy with (6);

Step 5 If T< $T_{\max}$, go to step 6; else, go to step 8;

Step 6 $\hat{m}$ sparks which are the specific sparks, are generated in each explosion generation by updating the location of the specific sparks with (5);

Step 7 Select $n$ optimal sparks from all current fireworks and sparks as the new fireworks for the next generation of explosion, T=T+1, go to step 3;

Step 8 Output the best assembly sequence and its fitness function value.

## IV. Case Study

In this section, an assembly product [8] shown in Fig. 1 is used to verify the fireworks algorithm for solving assembly sequence planning problem, and the algorithm is programmed with Matlab2014a.

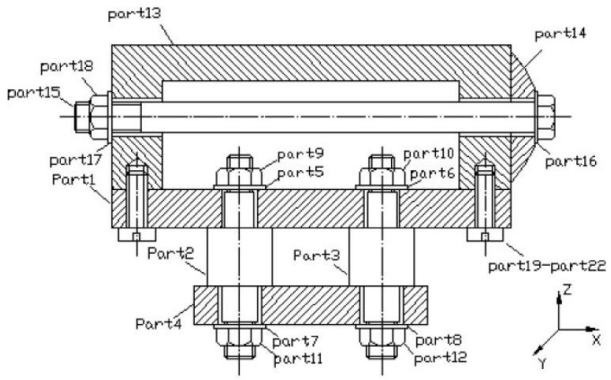


Fig. 1. An assembly consisting of 22 parts.

In this case, given the number of the component $S=22$, the weights of the number of assembly tool changes, the number of assembly orientation changes, the number of the assembly operation changes are given as $w_t=0.3$, $w_{or}=0.4$, $w_{op}=0.3$, respectively, set the maximal amplitude of explosion $\hat{A}=100$, the maximal iteration number $T_{max}=100$, the constant parameters $a=0.1$ and $b=0.9$.

In order to find the influence of the initial fireworks size, the different number of initial fireworks n are set, all programs run 20 times to get the value of optimal or near optimal sequences. The results show that the number of the optimal or near optimal sequences obtained increases when the number of the initial fireworks increases. The result when the number of initial fireworks n is set as 100 is shown in TABLE I, where $n_e$ is the program running times, $n_s$ is the number of the optimal or near optimal sequences obtained when the program runs $n_e$ times.

TABLE I: The Number of Optimal or Near Optimal Sequence

| $n$ | $n_e$ | $n_s$ |
|---|---|---|
| 100 | 20 | 16 |

In order to investigate the influence of the number of the sparks, the different number of sparks m are set, all programs run 20 times to get the value of optimal or near optimal sequences. The results show that with different setting of the number of the sparks, the higher number of the sparks, the more number of the optimal or near optimal sequences can be obtained. The result when the number of sparks m is set as 200 is shown in TABLE II, where the number of initial fireworks is set as 10, the parameters $n_e$ and $n_s$ are the same as that in the previous test.

TABLE II: The Number of Optimal or Near Optimal Sequence

| $m$ | $n_e$ | $n_s$ |
|---|---|---|
| 200 | 20 | 3 |

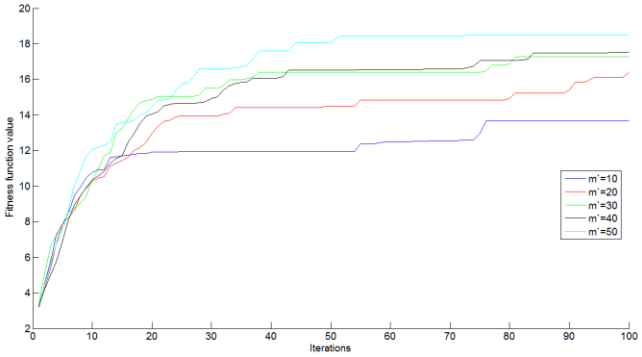


Fig. 2. Evolution performance of the fireworks algorithm with different specific sparks.

To investigate the influence of specific sparks, the number of the specific sparks is set as 10, 20, 30, 40, 50, respectively. The number of initial fireworks is set as 10, and the number of sparks is set as 50. For different number of specific sparks, the program runs 20 times to get the mean value of the fitness function. Fig.2 shows the evolution performance of the fireworks algorithm with different number of specific sparks. It can be seen that the larger number of the specific sparks, the better mean value of the fitness function.

In order to prove the effectiveness of the fireworks algorithm, the results in [9] are used for the comparison between the fireworks algorithm and the discrete particle swarm optimization algorithm.

Fig. 3 shows the curve of the fitness function value of assembly sequence planning based on fireworks algorithm when the program runs 20 times. Fig. 4 [9] shows the curve of

fitness function value of assembly sequence planning based on the discrete particle swarm optimization algorithm when the program runs 20 times. Through comparison, it can be seen that with the fireworks algorithm, more optimal or near optimal assembly sequences can be obtained than that with the discrete particle swarm optimization algorithm.

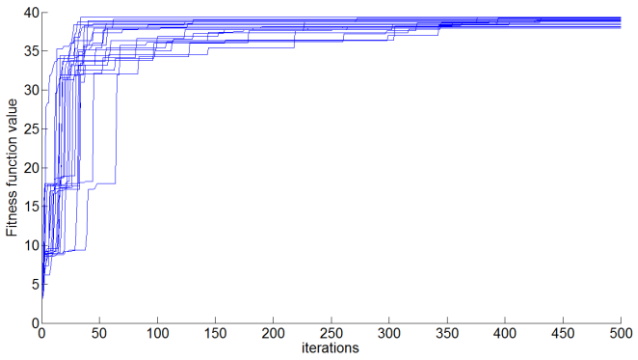


Fig. 3. Fireworks algorithm.

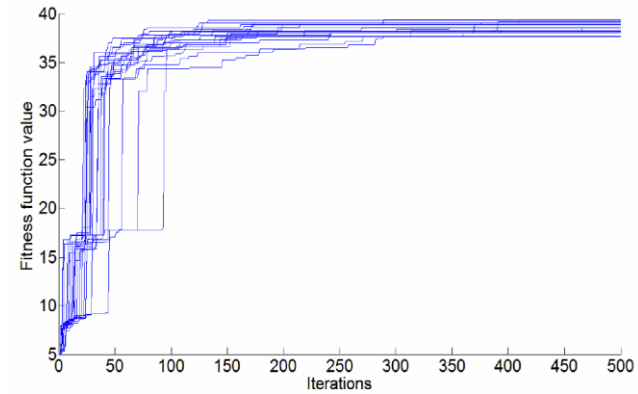


Fig. 4. Discrete particle swarm optimization algorithm.

## V. Conclusion

This paper investigates an assembly sequence planning approach with the fireworks algorithm. Through modeling the assembly sequence with fireworks algorithm and building the fitness function of assembly sequence planning, the framework of the fireworks algorithm for assembly sequence planning is proposed. With case study, it is verified that the fireworks algorithm is more effective in solving assembly sequence planning than the discrete particle swarm optimization algorithm.

## Acknowledgment

The authors are grateful for the financial support from National Natural Science Foundation of China [grant No. : 51575089].

## References

[1] Y. Wang, J. H. Liu, and L. S. Li, "Assembly sequences merging based on assembly unit partitioning," *The International Journal of Advanced Manufacturing Technology*, vol. 45, no. 7-8, 2009, pp. 808-820.

[2] H. L. S. de Mello and S. A. C., A correct and complete algorithm for the generation of mechanical assembly sequences, vol. 7, no. 2, 1991, pp. 228-240.

[3] R. Chen, K. Lu, and P. Tai, "Optimizing assembly planning through a three-stage integrated approach," *International Journal of Production* Economics, vol. 88, no. 3, 2004, pp. 243-256.

[4] W. Hui, X. Dong, and D. Guanghong, "A genetic algorithm for product disassembly sequence planning," *Neurocomputing*, vol. 71, no. 13-15, 2008, pp. 2720-2726.

[5] H. G. Lv and C. Lu, "A discrete particle swarm optimization algorithm for assembly sequence planning," in *Proc. 8th International Conference on Reliability, Maintainability and Safety*, Chengdu, China, 2009, pp. 1119-1112.

[6] H. Shan, S. Zhou, and Z. Sun, "Research on assembly sequence planning based on genetic simulated annealing algorithm and ant colony optimization algorithm," vol. 29, no. 3, 2009, pp. 249-256.

[7] Y. Tan, Y. Shi, K.C. Tan, Y. Tan, and Y. Zhu, "Fireworks algorithm for optimization," in *Proc International Conference on Swarm Intelligence (ICSI 2010)*, Lecture Notes in Computer Science, 2010, pp. 355-364.

[8] C. Lu, J. Y. H. Fuh, and Y. S. Wong, "An enhanced assembly planning approach using a multi-objective genetic algorithm," in *Proc. the Institution of Mechanical Engineers*, 2006, vol. 220, no. 2, pp. 255-272.

[9] H. Lv and C. Lu, "An assembly sequence planning approach with a discrete particle swarm optimization algorithm," *The International Journal of Advanced Manufacturing Technology*, 2010, vol. 50, pp. 761-770.

**Jun-Ying Li** is currently a master student of School of Mechatronics Engineering, University of Electronic Science and Technology of China.

Her research interests include CAD/CAM, advanced manufacturing.

**Cong Lu** obtained the PhD in mechanical engineering from National University of Singapore (NUS) in Jan 2007. His research interests include CAD/CAM, advanced manufacturing, concurrent engineering.

He is currently an associate professor in the School of Mechatronics Engineering, University of Electronic Science and Technology of China (UESTC). He worked as a research fellow in Department of Mechanical Engineering of NUS from June 2006 to June 2007, and he joined in UESTC since June 2007.