# A Novel Multi-agent Evolutionary Algorithm for Assembly Sequence Planning

Congwen Zeng
School of Electronic Engineering, Xidian University
Xi'an, Shaanxi, 710071, China
cwzeng@guet.edu.cn

Tianlong Gu, Liang Chang
Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology
Guilin, Guangxi, 541004, China
changl@guet.edu.cn

Fengyin Li
School of Computer Science and Engineering, Guilin University of Electronic Technology
Guilin, Guangxi, 541004, China
changl@guet.edu.cn

***Abstract*—Many evolutionary algorithms for assembly sequence planning (ASP) have been researched. But those algorithms have lots of blind searching because individuals have little consideration about geometry and assembly process information of product in the evolutionary process. To improve individuals' intelligence and decrease blind searching, motivated by the self-assembly computing and multi-agent evolutionary algorithm, a novel multi-agent evolutionary algorithm for assembly sequence planning (NMAEA-ASP) is presented. In the algorithm, learning, competition and mutation are designed for each agent. Learning, competition and mutation are realized by assembly and disassembly. Some notions such as assembly-unit, power about assembly are also introduced. Experimental results show that NMAEA-ASP can find an approximate solution faster than other evolutionary algorithms.**

***Index Terms*—Assembly sequence planning, Evolutionary algorithm, Assembly, Learning, Mutation**

## I. INTRODUCTION

Assembly sequence planning is one of the most important works be done during the assembly. Since it may be costly to overlook a potential candidate assembly sequence, it is desirable to select a satisfactory sequence from the set of all feasible sequences. However, assembly sequence planning has been proven to be NP-complete in theory for large scale of assemblies. Su [1] pointed out that whole product assembly process planning is much more complicated under assembly constraints comparing with single-part process planning. The search space of assembly sequence planning is exponentially in proportional to the number of parts or components and the assembly relationships between them. It will consume a lot of computation time or memory space when the assembly is complex. Once the number of parts or components is above a threshold value, assembly sequence planning can hardly be accomplished successfully [2]. Therefore, the new efficient methods are urgently called to resolve the complex problem.

Soft computing techniques applicable or artificial intelligence (AI) has been researched and applied for ASP problem in the past few years. In general, Intelligent algorithms such as artificial neural network, simulated annealing, and genetic algorithm (GA) can improve the efficiency of the process that searches for an assembly sequence. The concept of GA to assembly planning is introduced in literature [3]. Literature [4] has proposed algorithms to search for the best assembly sequence based on simulated annealing [4]. A particle swarm optimization algorithm is proposed for a simple assembly line balancing [5]. A new PSO encoding scheme is presented by defining the assembly sequences and disassembly sequences in one position matrix of a particle. The position matrix of a particle defines an assembly sequence and a disassembly sequence in the new encoding scheme. In this way, the sequence of assembly and sequence can be simultaneously planned by optimizing the position matrix of a particle. Prediction of operating loads contribution to assembly relation and product behavior is proposed in literature [6]. Product running, operating loads will lead to change of assembly relation of product parts affecting product behavior. Based on Jacobian-Torsor method, the Jacobian-Torsor tolerance model, considering contribution of operating loads, was extended and corrected, the assembly error (assembly relation change) resulted from operating loads can be calculated. A two-level genetic structure to dynamically adjust the parameters of GA is proposed in the literature [7]. Approaches based on GA to minimize production costs for ASP with consideration of physical constraints is proposed [8,9]. An approach to multi-


doi:10.4304/jsw.8.6.1518-1525

criteria assembly sequence planning using genetic algorithms is presented[10]. Literature [11] proposed a scatter search approach to the optimum disassembly sequence problem. Bin Jiao presents a cooperative co-evolution particle swarm optimization (PSO) for flow shop [12]. Fuqing et al. propose an improved PSO algorithm with decline disturbance [13]. A novel immune algorithm simulating the biological immune system was proposed to solve the Assembly Sequence Planning(ASP) problem. Implementation methods such as appetency computation, antibody generation, immunity selection, and memory cell update were provided.The immune algorithm in sequence planning problem solving reflected characteristics such as diversity, immune self-adjustment,immune memory and distributed parallel of the immune system. The immune algorithm was superior to those genetic algorithms in both global search capability and convergence speed.As a result,the immune algorithm was a prospective and efficient way to tackle ASP[14]. Literature [15] proposed a guided genetic algorithms for solving larger constraint assembly problem. Guided-GAs are proposed wherein the proper initial population and the alternation of crossover and mutation mechanisms are covered to overcome assembly planning problems that contain large constraints. The optimal assembly sequence is obtained through the combination of Guided-GAs and the Connector-based assembly planning context as previously suggested. Literature [16] proposed a memetic algorithms with guided local search to solve assembly sequence planning. Literature [17] proposed an artificial immune systems for assembly sequence planning exploration. artificial immune systems (AIS) were proposed to help solve the assembly sequence problem. In AIS algorithm, the antibody (Ab) in the immune system is simulated to encounter one or more unknown antigens (Ags). Moreover, the clonal selection concept is employed in the immune system in which a better antibody will be selected in each generation of revolution and different antibodies will be cloned to protect the infection of the original antigen. With this mechanism, the shortcoming such as the traditional GAs to converge in local optimal solution will be overcome.Literature [18] proposed Assembly planning using a novel immune approach. Literature [19] present Application of memetic algorithm in assembly sequence planning. Wei Zhou et al. present a novel BCGA-based hybrid algorithm (BGHA) for assembly sequence planning by combining bacterial chemotaxis (BC) with genetic algorithm (GA) [20]. To improve individuals' intelligence, A multi-agent evolutionary algorithm for connector-based ASP (MAEA-ASP) is presented which is integrated with the multi — agent systems. learning 、competition and crossover -mutation are designed as the behaviors of agent which locate lattice-like structure environment[21]. MAEA-ASP is designed for the connector-based ASP and also has large blindness search.

In nowadays evolutionary algorithm for ASP, individual is encoded as a permutation order of parts(such as parts or connectors), but each individual has many parts whose position can't satisfy basic constraint conditions.

Let's take fig 1 for example. Traditional evolutionary algorithms search a solution in the space which includes all permutation sequences. Obviously, many permutation sequences including "1, 2, 6" sequence in the space aren't solutions. It is also to say that traditional genetic algorithms will generate a great deal of permutation sequences which are infeasible solutions in the evolution process, which results in the inefficiency of solution-searching process. If there is a limit that the part 1,6 must be prior to 2, then much blindness search can be avoided. Adleman described how he used traditional tools of molecular biology to solve a 7-vertex instance of hamilton path[22]. If a possible solution is searched based on self-assembly computation idea, these permutations including such as "1,2,6" sequence can be avoided effectively in fig 1. It is also to say it is possible to avoid semi-blindness of traditional evolutionary operators by using similar self-assembly computation idea. Hongchun QU and Youlan WANG also present a self-assembling approach to simulation of phototropism[23], Pan Xiaoying and Jiao Licheng present a multi-agent social evolutionary algorithm for project optimization scheduling[24]. Motivated by self-assembly computation and multi — agent systems, to improve Agents' intelligence and decrease greatly search space of problems, a novel multi-agent evolutionary algorithm for assembly sequence planning(NMAEA-ASP).

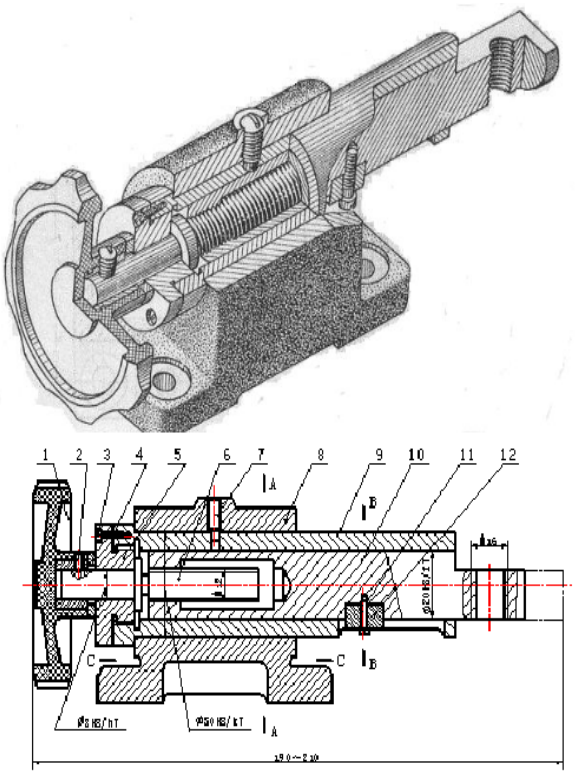



Fig 1. An example of assembly graph

The remainder of this paper is organized as follows. Section 2 discusses NMAEA-ASP. Section 3 provides experimental results compared NMAEA-ASP with other

GAs. Section 4 concludes the paper with recommendations for the algorithm.

.

## II. NMAEA-ASP

### A. Environment of Agents

According to literature [24], an agent is a physical or virtual entity. Agent has the following properties: (a) it is able to live and act in the environment; (b) it is able to sense its local environment; (c) it is driven by certain purposes; (d) it has some reactive behaviors. As can be seen, the meaning of an agent is very comprehensive, and what an agent represents is different for different problems. In general, four elements should be defined when multiagent systems are used to solve problems. The first is the meaning and the purpose of each agent. The second is the environment in which all agents live. Since each agent has only local perceptivity, so the third is the local environment. The last is the behaviors that each agent can take to achieve the purpose.

The environment is organized as a latticelike structure, which is similar to literature in the paper[24].

Latticelike structure definition: All Agents live in a latticelike environment L. The size of L is Lsize×Lsize,where Lsize is an integer. Each agent is fixed on a lattice-point and can only interact with the neighbors. Suppose that the agent located at (i, j) is represented as Li,j, i,j=1,2,…,Lsize,then Neighborsi,j are defined as fig 2.

The Latticelike structure can be described as the one in fig.2. Where each circle represents an agent, the data represent the agent's position in the Latticelike structure, and two agents can interact with each other if and only if there is a line among them.

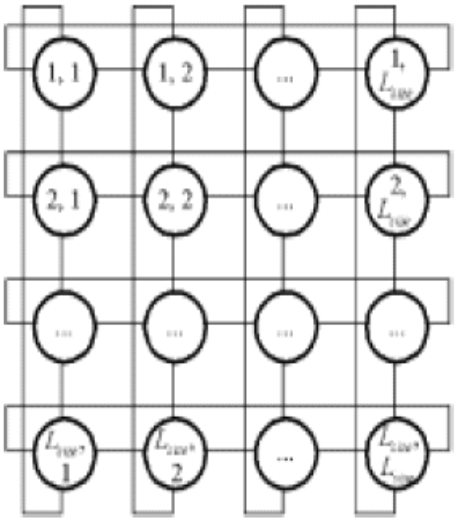
In the agent lattice, agents will compete with others so that they can gain more chance to produce offsprings to achieve their purposes. Because each agent can only sense itself local environment, the behaviors can only take place between the agent and the neighbors. An agent interacts with the neighbors so that information is transferred among them. The information is diffused to the whole agent lattice in such a manner. As can be seen, the model of the agent lattice is closer to the real evolutionary mechanism in nature than the model of the population in traditional EAs.

### B. Agent Definition

Now some notions relating with NMAEA-ASP are introduced.

Assembly-unit : The least unit of assembly. It can be described as follow:

```
Assembly-unit
{
  unit_id;
  power; // the sequence strength between assembly-
            unit and others.
  other imformation;
}
```




$$Neighbors_{i,j} = \{L_{i',j}, L_{i,j'}, L_{i'',j}, L_{i,j''}\}$$

$$Where\ i' = \begin{cases} i-1 & i \neq 1 \\ L_{size} & i = 1 \end{cases},\quad j' = \begin{cases} j-1 & j \neq 1 \\ L_{size} & j = 1 \end{cases}$$

$$i'' = \begin{cases} i+1 & i \neq L_{size} \\ 1 & i = L_{size} \end{cases},\quad j'' = \begin{cases} j+1 & j \neq L_{size} \\ 1 & j = L_{size} \end{cases}$$

Fig.2 The lattice model of agents enviorment

Unit_id is the identification of Assembly-unit , Suppose Assembly-unit _sequence is $a_1,a_2,a_3,\ldots a_i,\ldots a_k$, then power($a_i$, $a_{i+1}$) denotes the sequence strength between Assembly-unit $a_i$ and Assembly-unit $a_{i+1}(0<i<k)$. The power of sequence(i,j) is different form sequence(j,i). The power of sequence(i,i) is zero.

Agent: An agent is a physical or virtual entity. Agent is different from that of the traditional EAs because its intelligences. This paper, It is designed as follow:

```
Agent
{
  Agent_id,Agent_position,Fitness;
  Assembly-unit _sequence;
  UnAssembly-unit _set;
  Learning( );Competition( ); Mutation( );
}
```

Assembly-unit _sequence points to the part of a possible solution. But UnAssembly-unit _set is the set of unassemblied Assembly-unit . Agent_position(i, j) shows the agent location on ith row and jth column in the lattice-like structure Environment. An agent has three evolutionary actor: Learning、Competition、Mutation.

Suppose $a_1$ 、 $a_2$ 、 …$a_{k-1}$ 、 $a_k$ is Assembly-unit _sequence of Agent A。 power(ai, A) = (power($a_i$, $a_{i+1}$)+power($a_{i-1}$, $a_i$))/2 (1<i<k),power($a_k$, A) = power($a_{k-1}$, $a_k$) and power($a_1$, A)=power($a_1$, $a_2$)。Suppose Assembly-

unit b is assemblied in it's Assembly-unit _sequence in position(i), then power(b,A,i) = (power($a_i$,b) + power($a_{i+1}$,b))/2 （1<i<k）,but power(b,A,k) = power(b, $a_k$). Maxpowerposition(Assembly-unit a, indiviudal A) return the ith position which make power($a_i$, A) maxium. If maxium is NULL, then return NULL, i∈{1,2,3…k,NULL}.

Fitness shows the ability of agents' survives. Suppose the Assembly-unit _sequence of an agent is a1a2…an, it can be computed as follow:

$$fitness = \sum_{i=1}^{n-1} Power(a_i, a_{i+1})$$

### C. Assembly and Disassembly

Assembly: The process that an Assembly-unit is added into the Assembly-unit _sequence of an agent is called assembly. Suppose an agent is A, then Assembly can be described as follow :

```
Assembly(Assembly-unit a)
{
 i= Maxpowerposition(a, A);
 if (i!=NULL)
 insert a into Assembly-unit _sequence of A
    with the ith position;
 Delete the corresponding Assembly-unit a from
    UnAssembly-unit _set of A;
}
```

Disassembly: The process that an Assembly-unit is taken out of Assembly-unit _sequence of A and the corresponding Assembly-unit is added into UnAssembly-unit _set of A is called disassembly. Suppose an agent is A, then Disassembly can described as follow:

```
Disassembly(Assembly-unit a)
{
 Delete the Assembly-unit a from the Assembly-unit
    _sequence of A;
Insert a into UnAssembly-unit _set.
}
```

### D. Evolutionary Operators

- Initialization

Firstly, the information of Assembly-unit s and the assembly power between any two Assembly-unit s are inputted as origins data. Array agents[n] is created randomly and n must be the squared of some int number. The m of agent[m](m∈{0,1……n-1}) is equal to i*sqrt(n)+j. where i,j is the number of position(i,j) which is assigned by its lattice position. Assembly-unit _sequence is settled on null. UnAssembly-unit _set={0,1……,n-1}. Fitness is settled on 0.

- Learning

In NMAEA-ASP, the learning actor of an agent is most important operator. Learning actor is consisted of two parts: self-learning and neighbors-learning. The duty of self-learning is to add Assembly-unit from UnAssembly-unit _set into the Assembly-unit _sequence as many as possible. The duty of neighbors-learning is to compare itself fitness and that of neighbors. If the fitness of agent A is minimum, agent A will be replaced with one of neighbors randomly. Learning actor can be described as follow:

```
Learning( )
{
 Self-learning:
 find each Assembly-unit from UnAssembly-unit
    _set randomly.
 Do assembly(Assembly-unit );
 neighbors-learning:
 if(fitness<= the fitness of any neighbors)
 Replace the information of agent such as Assembly-
    unit _sequence、UnAssembly-unit _set、Fitness
    and so on with that of one of neighbors randomly.
}
```

- Competition

In this operator, the fitness of an agent is compared with those of neighbors. The agent can survive if the fitness is maximum; Otherwise the agent would die with probity Pc, and the child of the one with maximum fitness among the neighbors will take up the lattice-point.

Suppose that the competitive behavior is performed on the agent $L_{i,j}$ located at (i, j), and $Max_{i,j}$ is the agent with maximum fitness among the neighbors of $L_{i,j}$, namely,$Max_{i,j} \in Neighbors_{i,j}$ and $\forall Agent \in Neighbors_{i,j}$,then $Agent(F) \le Max_{i,j}(E)$. If $L_{i,j}(E) \le Max_{i,j}(E)$, then the mutation of $Max_{i,j}$ generates a child Agent, $Child_{i,j}$, to replace $L_{i,j}$ with porblity pc, otherwise $L_{i,j}$ is left untouched. The purpose of the competitive behavior is to eliminate the agents with low fitness, and give more chances to the potential agents.

- Mutation

In NMAEA-ASP, mutation is disassembly Assembly-unit from agent's Assembly-unit _sequence. Suppose the Assembly-unit number of Assembly-unit _sequence is m in an agent A, then mutation can be described as follow:

```
Mutation ( )
{
 For(i=0;i< m;i++)
 {
  If(power(Assembly-unit i,A)<avgpower)
  {
   P=random()%10;
   If(p<3)
   Disassembly(Assembly-unit i)
  }
 }
}
```

*E. The flow of NMAEA-ASP*

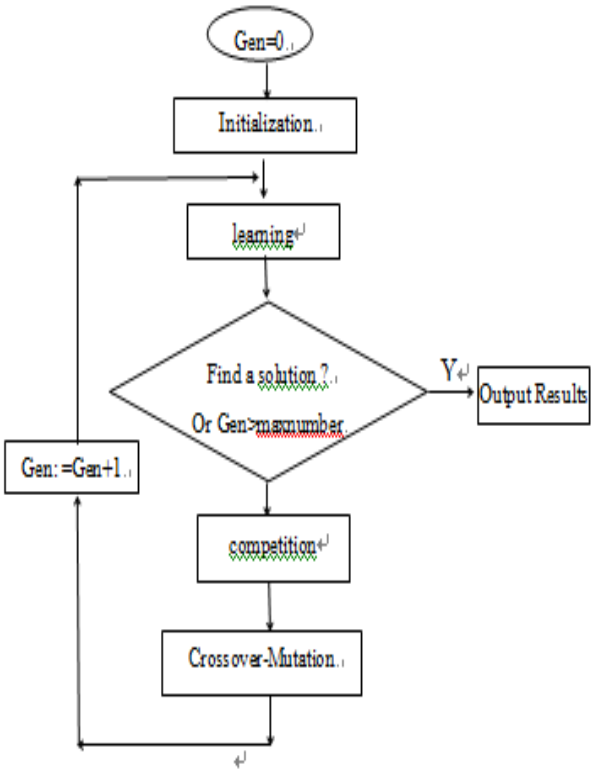



Fig 3. Flowchart of NMAEA-ASP

Suppose the Assembly-unit _sequence of agent Q = (T1, T2, ……Tm), Ti is the ith unit of Q, power (Ti,Ti+1) denotes the strenghth of border bond of Assembly-unit s. Then regard the following equation as the fitness(F) function.

$$F(Q)=\sum_{i=1}^{m-1} power\,(T_i,T_{i+1})$$

NMAEA-ASP combined the idea of self-assembly computation and multi—agent systems. But there are many differences from traditional evolutionary algorithms which embody in operators.

The flowchart of NMAEA-ASP is illustrated in fig 3.

## III. EXPERIMENTAL RESULTS

The parameters of NMAEA-ASP is setted as follow: the number of Agents is 9. Pc is set to 0.2. Fas in the table.2 and fitness in the table.4 are $F(Q)$. The Assembly-unit _sequence is consisted of part_sequence、direction_sequence、tool_sequence、type_sequence. The value of power(a, b) can be calculated by:

$$Power(a,b)=Wc\times C_{a,b}+W_d\times D_{a,b}+Wt\times T_{a,b} \quad (1)$$

Where power(a,b) represents the bond power between Assembly-unit a and Assembly-unit b; if a=b, then power(a,b)=0;

$W_c$ is the weight of the assemblytype;

$W_d$ is the weight of the assemblydirection;

$W_t$ is the weight of the assemblytool;

$C_{a,b}$ is the assemblytype between the Assembly-unit a and the Assembly-unit b. When the assemblytype if Assembly-unit $C_a$ and $C_b$ is the same, $C_{a,b}=1$; otherwise, $C_{a,b}=0$;

$D_{a,b}$ is the assemblydirection between the Assembly-unit a and the Assembly-unit b. When the assemblytype if Assembly-unit $D_a$ and $D_b$ is the same, $D_{a,b}=1$; otherwise, $D_{a,b}=0$;

Ta,b is the assemblytool between the Assembly-unit a and the Assembly-unit b. When the assemblytool if Assembly-unit $T_a$ and $T_b$ is the same, $T_{a,b}=1$; otherwise, $T_{a,b}=0$;

In contrast with the literature[14], the weight of the engineering data of the Assembly-unit in formula(1) is all settled on 1. In contrast with the literature[15][16][17][21], the weight of the engineering data of the Assembly-unit in formula(1) is all settled on formula(2).

$$W_k=\frac{n+1-k}{\sum_{l=1}^{n} l}=\frac{2(n+1-k)}{n(n+1)} \quad (2)$$

where n is the number of the item of engineering information and k is the rank of the item of engineering information[15][16][17][21].

NMAEA-ASP is written in VC++6.0. The test environment is that of a Pentium2.4 GMHz PC at 1024 MB RAM. Ten tests are executed.

*A. Experiment On Gear pump*

Fig 4 shows a gear pump consists of 22 parts, there are 16 parts by decreasing the same bolt and pin. Table I shows 16 gear pump parts、assembly tool、type of gear pump parts.

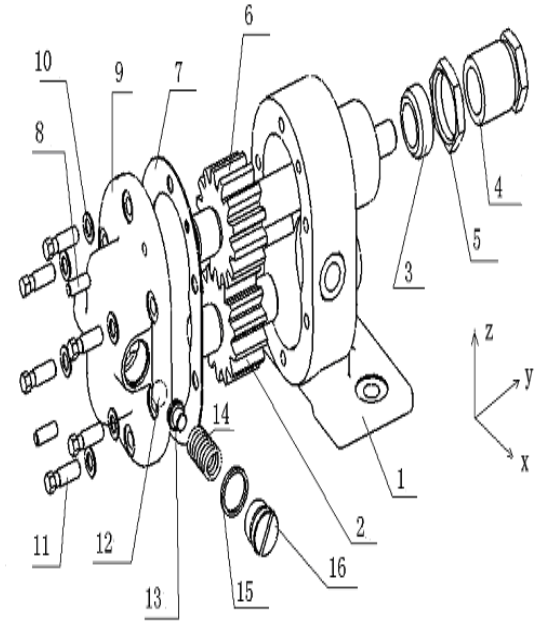



Fig 4. Gear pump

Table II shows A comparison among three EAs (Gear pump). Obviously NMAEA-ASP can find a better ASP with less time than other two GAs. Table III shows some assembly planning sequences with NMAEA-ASP.

TABLE I.
ASSEMBLY DATA OF GEAR PUMP

| partid | name | tool | type |
|---|---|---|---|
| 1 | Pump body | G1 | L1 |
| 2 | Driven gear | G1, G2 | L3 |
| 3 | wad | G1, G3, G5 | L2 |
| 4 | Wad capper | G1, G6 | L2 |
| 5 | Lock nut | G1, G6 | L2 |
| 6 | Driving Gear | G1, G2 | L3 |
| 7 | mat | G1, G2, G3 | L1 |
| 8 | pin | G2, G4 | L4 |
| 9 | Pump lid | G1 | L1 |
| 10 | Washer | G2, G3 | L4 |
| 11 | Small bolt | G6 | L4 |
| 12 | Steel ball | G2, G3 | L1 |
| 13 | Localize | G2, G3 | L1 |
| 14 | spring | G2, G3 | L1 |
| 15 | Small washer | G1, G3 | L1 |
| 16 | Big bolt | G1, G5 | L1 |

TABLE II.
A COMPARISON AMONG THREE EAS (GEAR PUMP)

| Method | Average fitness | Maximum fitness | Average Runtime(S) |
|---|---|---|---|
| GAs | 23.7 | 31 | 126 |
| Immune GAs | 63.2 | 66 | 53 |
| NMAEA-ASP | 64.2 | 66 | 2.8 |

TABLE III.
SOME ASSEMBLY PLANNING SEQUENCES

| solution | Assembly sequence | | | | | | | | | | | | | | | | $F_{AS}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Best sequence | Partid | 1 | 2 | 6 | 7 | 9 | 8 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 3 | 5 | 4 | 66 |
| | Tool | G1 | G1 | G1 | G1 | G1 | G2 | G2 | G6 | G2 | G2 | G2 | G1 | G1 | G1 | G1 | G1 | |
| | Direction | +Y | +Y | +Y | +Y | +Y | +Y | +Y | +Y | -X | -X | -X | -X | -X | -Y | -Y | -Y | |
| | Type | L1 | L3 | L3 | L1 | L1 | L4 | L4 | L4 | L1 | L1 | L1 | L1 | L1 | L2 | L2 | L2 | |
| Best sequence | Partid | 1 | 3 | 5 | 4 | 6 | 2 | 7 | 9 | 10 | 11 | 8 | 12 | 13 | 14 | 15 | 16 | 66 |
| | Tool | G1 | G1 | G1 | G1 | G1 | G1 | G1 | G1 | G2 | G6 | G2 | G2 | G2 | G2 | G1 | G1 | |
| | Direction | -y | -y | -y | -y | +y | +y | +y | +y | +y | +y | +y | -x | -x | -x | -x | -x | |
| | Type | L1 | L2 | L2 | L2 | L3 | L3 | L1 | L1 | L4 | L4 | L4 | L1 | L1 | L1 | L1 | L1 | |
| Better sequence | Partid | 1 | 3 | 5 | 4 | 6 | 2 | 7 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 8 | 63 |
| | Tool | G1 | G1 | G1 | G1 | G1 | G1 | G1 | G1 | G2 | G6 | G2 | G2 | G2 | G1 | G1 | G2 | |
| | Direction | -y | -y | -y | -y | +y | +y | +y | +y | +y | +y | -x | -x | -x | -x | -x | +y | |
| | Type | L1 | L2 | L2 | L2 | L3 | L3 | L1 | L1 | L4 | L4 | L1 | L1 | L1 | L1 | L1 | L4 | |
| Better sequence | Partid | 1 | 2 | 6 | 7 | 9 | 10 | 11 | 8 | 12 | 13 | 14 | 15 | 16 | 3 | 5 | 4 | 65 |
| | Tool | G1 | G1 | G1 | G1 | G1 | G2 | G6 | G2 | G2 | G2 | G2 | G1 | G1 | G1 | G1 | G1 | |
| | Direction | +Y | +Y | +Y | +Y | +Y | +Y | +Y | +Y | -X | -X | -X | -X | -X | -Y | -Y | -Y | |
| | Type | L1 | L3 | L3 | L1 | L1 | L4 | L4 | L4 | L1 | L1 | L1 | L1 | L1 | L2 | L2 | L2 | |
| Better sequence | Partid | 1 | 6 | 3 | 5 | 4 | 2 | 7 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 8 | 61 |
| | Tool | G1 | G1 | G1 | G1 | G1 | G1 | G1 | G1 | G2 | G6 | G2 | G2 | G2 | G1 | G1 | G2 | |
| | Direction | +Y | +Y | -y | -y | -y | +y | +y | +y | +y | +y | -X | -X | -X | -X | -X | +Y | |
| | Type | L1 | L3 | L2 | L2 | L2 | L3 | L1 | L1 | L4 | L4 | L1 | L1 | L1 | L1 | L1 | L4 | |

### *B. Experiment On Stapler*

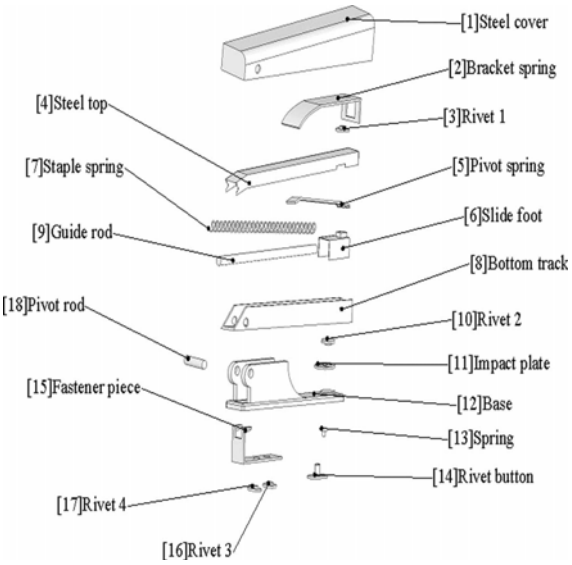



Figure 5. Stapler: diagram of parts

The stapler consists of 18 parts. According to the principle of connector settings, 8 connectors can be specified. Figure 5 illustrates the parts of the electric fan. Figure 6 shows the precedence graph for the stapler connectors[15].

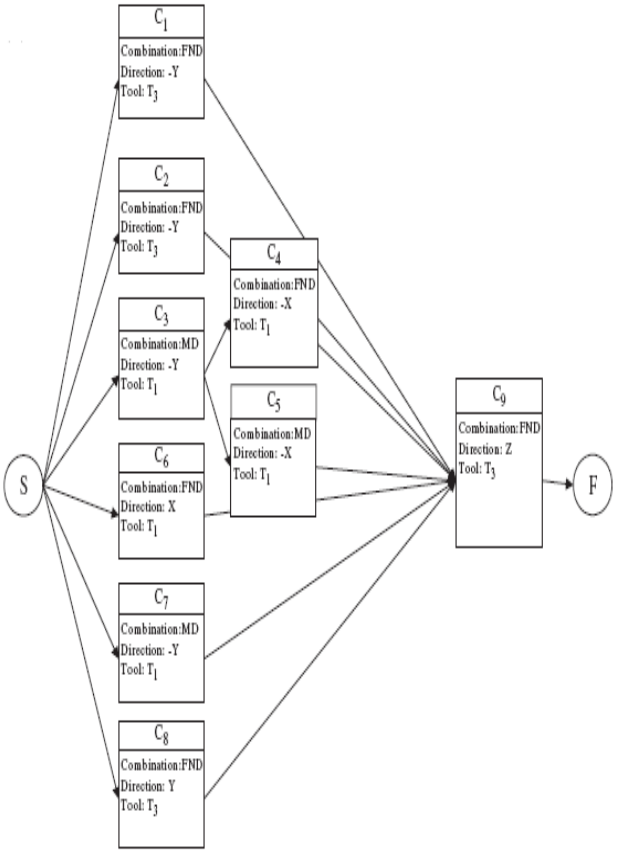



Fig 6. Stapler: connector-based precedence graph.

TABLE IV.
A COMPARISON AMONG FIVE EAS ON STAPLER

| Method | Average runtime | Average fitness | MaTimum fitness |
|---|---|---|---|
| Guided-GAs | 1.572 | 5.495 | 5.667 |
| MAs | 2.494 | 5.667 | 5.667 |
| AIS | 2.453 | 5.667 | 5.667 |
| MAEA-ASP | 0.827 | 5.667 | 5.667 |
| NMAEA-ASP | 0.363 | 5.667 | 5.667 |

The results of comparison for Guided-GAs、Memetic algorithms(MAs) 、AIS 、MAEA-ASP and NMAEA-ASP can be found in Table IV. From the results of NMAEA-ASP in Table 4, the average computation took up 0.363s. Obviously, NMAEA-ASP is undoubtedly a preferable selection for assembly planning decision.

## IV. CONCLUSION

The new idea of NMAEA-ASP is motivated by DNA molecular computation and multi-agent evolutionary algorithm whose original goal is to deal with combinatorial optimization problems by decreasing blind searching and increasing intelligence searching. At the same time, there are many evolutionary algorithms for dealing with ASP. We take a try that evolutionary operators are realized with assembly and disassembly methods. The experiment results show that algorithm has a faster speed in finding solutions.

Recently, symbolic ordered binary decision diagram (OBDD) is presented for assembly sequence planning [25]. NMAEA-ASP will be developed and tested with a symbolic OBDD assembly technology in the future study.

## ACKNOWLEDGEMENTS.

This work is supported by the National Natural Science Foundation of China (No. 61100025, 61163041, 61262030), the Natural Science Foundation of Guangxi Province (No.2012GXNSFBA053169, GXNSFAA053220) and the Science Foundation of Guangxi Key Laboratory of Trusted Software (No. KX201109).

## REFERENCES

[1] Su Q A, “hierarchical approach on assembly sequence planning and optimal sequences analyzing”, Robot Comput-Integrated Manuf 25(1), pp.224–234, 2009

[2] .Laperriere L, Eimafaghy HAGAPP, “A generative assembly process planner”, J Manuf Syst 15(4), pp.282–293, 1996

[3] Bonneville F, Perrard C, Henrioud JM, “A genetic algorithm to generate and evaluate assembly plans”, Proceedings of the IEEE Symposium on Emerging Technology and Factory Automation,pp.231–239,1997.

[4] Milner JM, Graves SC, Whitney F, “Using simulated annealing to select least-cost assembly sequences”, Proc IEEE Int Conf Robot Autom 3, pp.499–504,1994.

[5] Qi Lv, “Simple Assembly Line Balancing Using Particle Swarm Optimization Algorithm”, International Journal of Digital Content Technology and its Applications, 5(6), pp.297-304, 2011

[6] Pengzhong Li, Weimin Zhang and Can Chen, “Prediction of operating loads contribution to assembly relation and product behavior”, Journal of software, 7(2), pp.296-302, 2012

[7] Chen SF, Liu Y, “An adaptive genetic assembly sequence planner”, Int J Comput Integer Manuf 14(5), pp. 489–500,2001.

[8] Romeo MM, Lee HS, Luong KA, “A genetic algorithm for the optimization of assembly sequences”, Comput Ind Eng 50, pp.503–527,2006.

[9] Guan Q, Lin JH, Zhong YF , “A concurrent hierarchical evolution approach to assembly process planning”, Int J Prod Res 40(14), pp.3357–3374,2002.

[10] Young-Keun Choi,Dong Myung Lee and Yeong Bin Cho, “An approach to multi-criteria assembly sequence planning using genetic algorithms” 2009.

[11] Beatriz Gonzáleza, Belarmino Adenso-Díaz, “A scatter search approach to the optimum disassembly sequence problem”, Computers & Operations Research 33, pp.1776–1793, 2006.

[12] Bin Jiao et al, “A Cooperative Co-evolution PSO for Flow Shop Scheduling Problem with Uncertainty”, Journal of computers, 6(9), pp. 1955-1961, 2011

[13] Fuqing Zhao et al. An improved PSO algorithm with decline disturbance, Journal of computers,. 6(4), pp. 691-697, 2011.

[14] Ning Li-hua,Gu Tian-long, “Immune algorithm for assembly sequence planning problem”, Computer Integrated Manufacturing Systems. 13(1), pp.81-87, 2007.

[15] Tseng, H. E.,“Guided genetic algorithms for solving larger constraint assembly problem”, International Journal Production. Research, 44(3), pp.601–625,2006.

[16] Hwai-En Tseng, Wen-Pai Wang and Hsun-Yi Shih, “Using memetic algorithms with guided local search to solve assembly sequence planning”, Expert Systems with Applications, 33(2)：451-467, 2007.

[17] Chien-Cheng Chang, Hwai-EnTseng, Ling-PengMeng, “Artificial immune systems for assembly sequence planning exploration”, Engineering Applications of Artificial Intelligence, 22, pp.1218–1232, 2009.

[18] Cao, P.B., Xiao, R.B.,“Assembly planning using a novel immune approach”, The International Journal of Advanced Manufacturing Technology, 31 (7–8), pp.770–782, 2007

[19] Liang Gao, Weirong Qian ,Xinyu Li, Junfeng Wang. “Application of memetic algorithm in assembly sequence planning”, Adv Manuf Technol , 49, pp.1175–1184, 2010.

[20] Wei Zhou, Jian-rong Zheng, Jian-jun Yan , Jun-feng Wang. A novel hybrid algorithm for assembly sequence planning combining bacterial chemotaxis with genetic algorithm. Int J Adv Manuf Technol 52, pp.715–724, 2011.

[21] Congwen Zeng, Tianlong Gu, Yanru Zhong, Guoyong Cai. “A Multi-Agent Evolutionary algorIthm for Connector-Based Assembly Sequence Planning”, Procedia Engineering, 15, pp.3689-3693, 2011.

[22] Meng Da - zhi , CAO Hai – ping, “DNA computing and biological mathematics”, Acta biophysica sinica, 18(2), pp.163-174, 2002.

[23] Hongchun QU, Youlan WANG, “A Self-assembling Approach to Simulation of Phototropism”, International

Journal of Digital Content Technology and its Applications, 5(1), pp.55-62, 2011.
[24] Pan Xiaoying, Jiao Licheng, "A Multi-Agent Social Evolutionary Algorithm for Project Optimization Scheduling", Journal of Computer Research and Development,45 (6), pp. 998-1003, 2008.
[25] Zhoubo Xu, Tianlong Gu, Rongsheng Dong, "Symbolic OBDD Assembly Sequence Planning Algorithm Based on Unordered Partition with 2 Parts of a Positive Integer", IFIP Advances in Information and Communication Technology Volume, 385, pp.226-233, 2012.

**Congwen Zeng** was born in China, in 1972. He received his M.S. degree in the department of computer from China University of Mining and technology, xuzhou, China,in 2002. He is currently a Ph.D.candidate at Xi'dian University of Xi'an, China.

His research interests include evolutionary algorithm and symbolic technology. His title is associate professor.

**Tianlong Gu** received his Ph.D. degree from Zhejiang University, China in 1996. From 1998 to 2002, he was a Research Fellow at the School of Electrical & Computer Eng., Curtin University of Technology, Australia and Postdoctoral Fellow at the School of Engineering, Murdoch University, Australia.

His research interests include formal methodology and their industrial applications, Petri nets, formal specification and verification techniques. His title is professor and Ph.D supervisor.

**Liang Chang**, received his Ph.D. in computer science from the Institute of Computing Technology, Chinese Academy of Sciences in 2008.

His research interests include knowledge representation and reasoning, intelligent planning, and formal methods in computer science. He is currently a professor in the School of Computer Science and Technology at Guilin University of Electronic Technology (China).