# A Multi-objective Particle Swarm Optimization for Assembly Line Design with Station Paralleling

Jianping Dou
School of Mechanical Engineering
Southeast University
Nanjing, China
djpseu@gmail.com

Xia Zhao
Center for Food Security and Strategic Studies
Nanjing University of Finance and Economics
Nanjing, China
txzhaoxia@163.com

**Abstract—To solve bi-objective assembly line design problem (ALDP) considering station paralleling and equipment selection, a Pareto-dominance-based method is presented. The two objectives are minimization of investment cost and maximization of availability of the assembly line. A multi-objective particle swarm optimization (MoPSO) is proposed to obtain a set of Pareto solutions through combining the techniques of crowded distance and external Pareto solution archive. The developed solution representation and relating updating mechanism of MoPSO ensures each particle to be a feasible solution. The performance of MoPSO was compared with that of NSGA-II against two cases. The comparison results show the effectiveness of the MoPSO. The computation results also indicate that the MoPSO is superior to the NSGA-II for the ALDP with respect to solution quality and computational efficiency.**

***Keywords- assembly line design; multi-objective optimization; partticle swarm optimization; station parallel;Pareto solutions***

## I. INTRODUCTION

Assembly lines are production systems composed of a succession of stations, connected by a conveyor, performing a set of tasks on the product passing through them. Each product unit remains at each station for a fixed time called the cycle time, *C*. When alternative automated equipments are used for assembly tasks, the assembly line design problem (ALDP) becomes very important. The design in this context consists of selecting the type and amount of paralleling equipment for the stations and assigning tasks to stations.

Bukchin [1] and Nicosiaa [2] have shown that the ALDP is more difficult than the assembly line balancing problem (ALBP) which is known to be a NP-hard combinatorial optimization problem. Due to the NP-hard essence of ALDP, traditional exact methods such as branch and bound [1] [3-4], and dynamic programming [2] are computational intractable for large scale problems. Recent years, some population based meta-heuristics such as genetic algorithms (GAs) [5-7], particle swarm optimization (PSO) [8-9] are utilized to solve ALDPs. However, the aforementioned works only consider cost related single objective. In practice, multiple objectives such as equipment cost, workload smoothness, and availability are needed to conceive for assembly line design. As for multiple-objective optimization for ALDP, the optimization methods can be categorized into two classes [10]: aggregative methods and Pareto-dominance-based methods. The aggregative method for ALDP [11-12] is to convert multiple objectives into single objective based on the assumption that the preferences of the decision-maker can be known a priori. On the contrary, the Pareto dominance-based method is to provide a list of interesting trade-offs between the objectives rather than a lone solution supposing that the preferences of the decision-maker are unknown. The Pareto dominance-based methods for ALDP with station paralleling and equipment selection are seldom. Rekiek [13] developed a grouping GA for ALDP considering equipment selection. Goyal [7] utilized the NSGA-II for ALDP with station paralleling and equipment selection. Saif [14] developed a Pareto based artificial bee colony algorithm for multi-objective assembly line balancing with uncertain task times. However, we are aware of no Pareto based multi-objective PSO (MoPSO) for ALDP. Nearchou [15] proposed a dynamic weighted PSO for multi-objective assembly line balancing problem. Chutima [16] presented a MoPSO for two-sided mixed-model assembly line balancing. To authors' best knowledge, the MoPSO for ALDP considering station paralleling and equipment selection is absent.

In this paper, we extend our previous work [9] to solve multi-objective ALDP and developed an efficient MoPSO for ALDP with station paralleling and equipment selection. The effectiveness of the proposed MoPSO is illustrated by the performance comparison between MoPSO and NSGA-II against two cases.

## II. PROBLEM STATEMENT

In a make-to-order environment, the most important objective for assembly line design is to minimize the cost (e.g., fixed cost) given the capacity derived from the custom's order. Minimizing the fixed cost (investment cost) is as the first objective of the concerned ALDP. Another important objective is to meet the custom's demand reliably.

The ability of a production system to satisfy production demands depends on its availability [17]. Thus, maximizing availability of the assembly line is as the second objective for the addressed ALDP.

As aforementioned, we deal with the ALDP considering station paralleling (equipment paralleling) and equipment selection. Under such conditions, the ALDP is to determine the number of stations, select the number of paralleling equipments and equipment type to be placed to each station as well as assign assembly tasks to each station, satisfying capacity constraints (cycle time limitation) and observing precedence constraints among tasks. In practice, the space limitation should also be satisfied [6].

In our previous work, a 0-1 integer programming model for the single objective ALDP was presented [9]. The model of the multi-objective ALDP is identical to the model in [9] except for adding the below objective:

$$\text{Min } 1.2\text{-}fa(\text{AL}) \qquad (1)$$

where AL represents the designed assembly line, $fa(\bullet)$ is the function for computing the availability. The UGF method [17] is adopted to calculate the availability of AL. Eq.(1) is to maximize the availability of assembly line.

Since the ALDP belongs to NP-hard problems, a Pareto based multi-objective meta-heuristics named MoPSO is developed for solving the addressed problem.

## III. MULTI-OBJECTIVE PSO FOR ALDP

### A. Introdution to PSO

Particle swarm optimization (PSO) developed by Kennedy and Eberhart [18] is inspired by the social behavior of a flock of migrating birds trying to reach an un-known destination. In PSO, each solution is a 'bird' in the flock and is referred to as a 'particle'. A particle in the population evolves their social behavior and accordingly their movement towards a destination.

The evolution process of PSO is initialized with a group of random particles (solutions). The *i*th particle is represented by its position as a point in a N-dimensional space, where N is the number of variables. Throughout the process, each particle i monitors three values: its current position ($X_i(t)$); the best position it reached in previous cycles ($P_i$); its flying velocity ($V_i(t)$). In each time interval (generation), the position ($P_g$) of the best particle *g* is calculated as the best fitness of all particles. Accordingly, each particle updates its velocity $V_i(t+1)$ and position $X_i(t+1)$ to catch up with the best particle *g*, as follows:

$$\begin{cases} V_i(t+1) = \omega V_i(t) + C_1 r_1 (P_i - X_i(t)) + C_2 r_2 (P_g - X_i(t)) & (a) \\ X_i(t+1) = X_i(t) + V_i(t+1), \quad -V_{\max} < V_i(t+1) < V_{\max} & (b) \end{cases} \qquad (2)$$

where $C_1$ and $C_2$ are two positive constant, namely, learning factors, $r_1$ and $r_2$ are two random real number in the range [0, 1]. Usually $C_1$ and $C_2$ are set to be 2.0. $V_{max}$ is an upper limit on the maximum change of particle velocity, and $\omega$ is an inertia weight. To apply PSO to ALDP, suitable solution representation and particle's updating method need to be developed.

### B. Solution Representation

The representation of assembly line by a particle is composed of two components as shown in Fig .1. One component represents the feasible operation assignment (FOA), another represents the machine type and number of stations (machines). In Fig.1(a), the FOA records the assignment of tasks along the assembly line. The FOA is determined by the assignment of each task (operation) of feasible operation sequence (FOS). An operation sequence satisfying the precedence relations is named feasible operation sequence (FOS). It is clear that a feasible assignment of tasks along an assembly line must be a FOS. In Fig .1, the indirect encoding of FOS [6] is adopted. The basic idea is to represent a FOS by recording the selection priority of each operation. Then, a zero-indegree topological sort [6] is used to de-code the permutation of priority number, i.e, 5-7-1-3-2-6-4 in Fig .1, to a FOS, i.e., OP1-OP2-OP7-OP4-OP5-OP6-OP3. After the FOS is decided, the assigned station of each operation of FOS is recorded by an element of a particle in Fig .1(a). In Fig .1(a), OP2 is assigned to station 1 and OP5 is assigned to station 2.

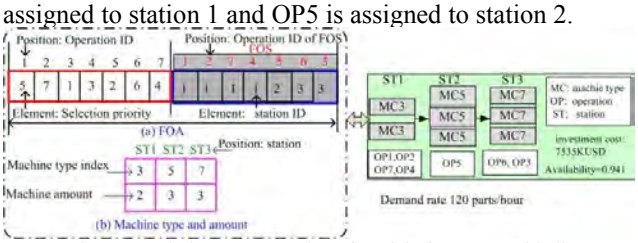


Figure 1. Solution representation of particle for an assembly line

To fully represent an assembly line, we have to record the machine type and machine amount via a particle. As shown in Fig .1(b), for each station, the information of machine type and machine amount are recorded directly by the corresponding elements of a particle. To avoid the unfeasibility of this direct encoding, this part is constructed in light of the FOA in Fig .1 (a) using the below procedure.

Step 1. Let the index of station j:=1;

Step 2. For station j,

Using the FOA information, get the assigned task set $AJ_p=\{1,\ldots,p_j\}$;

Identify the eligible machine type set $AM_j=\{1,\ldots,m_j\}$ for task set $AJ_p$ ;

Find the minimum cost machine type m from $AM_j$ and calculate the minimum number of corresponding machine type $m_j$;

Let the machine type be m and the number of parallel machine be a random integer in the range $[m_j, M]$.

Step 3. Let j:=j+1;

Step 4. If j=J, end; otherwise go to Step 2.

where the maximum parallel number of station is *M*, the maximum length of assembly line is *J* . For a product with N tasks and J stations, the maximum dimension of the particle is 2(N+J). According to above description, the developed solution representation ensures that each particle corresponds to a feasible solution.

### C. Particle Updating

The updating of a particle is realized by changing the velocity and position of a particle through learning from the $P_g$ and $P_i$. The updating mechanism of original PSO [18] is for real-encoding. Nevertheless, the integer encoding is used in the developed PSO. Thus, new updating mechanism is needed.

For the permutation encoding of FOS, the updating method for permutation encoding in our previous work [19] is adopted. For non-decreasing integer encoding for assignment of FOS (right part of Fig .1(a)), the mutation operator [20] is used to update this part. As aforementioned, the integer encoding of machine type and amount does not updating independently but updates in light of the relating FOA.

### D. Multi-Objective PSO

As aforementioned, the Pareto based method is adopted in this paper. The proposed MoPSO utilizes the concepts of Pareto dominance and Pareto solutions (See [21] for details). For multi-objective optimization, the comparison of two solutions is not easy. Therefore, the selection of $P_i$ and $P_g$ is far more difficult than single objective optimization. Based on the concept of crowded distance [22] and external Pareto solution archive [21], the procedure of MoPSO is shown in Fig .2. The main procedure of MoPSO is the same as the basic PSO. However, the difference lies in the selection of $P_i$ and $P_g$ as well as the computation of particle's fitness.

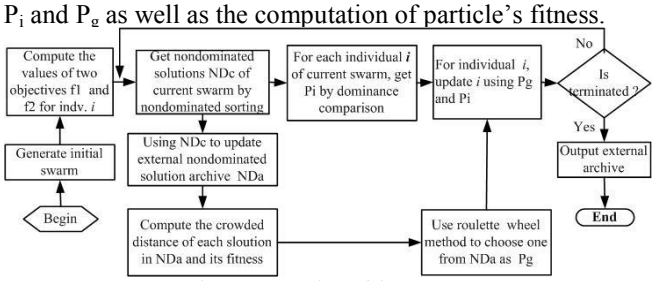


Figure 2. Procedure of the MoPSO

To select the personal best $P_i$, if current personal best $P_i(t+1)$ dominates its previous one $P_i(t)$ (denoted by $P_i(t+1) \succ P_i(t)$), let $P_i$ equal $P_i(t+1)$. If $P_i(t) \succ P_i(t+1)$, let $P_i$ equal $P_i(t)$. If $P_i(t+1)$ is incomparable with $P_i(t)$, select $P_i(t+1)$ or $P_i(t)$ randomly with identical chance as $P_i$. To identify the $P_g$, we use the roulette wheel method to choose one from external Pareto solution archive $ND_a$. It is clear that the individual within $ND_a$ with big fitness will be chosen. The procedure of obtaining and updating $ND_a$ is expounded below.

The construction and maintenance of external Pareto solution archive are crucial to MoPSO. The purpose of constructing external Pareto solution archive is to store the non-dominated solutions found so far and is as candidate of global best ones. In addition, the final output of the MoPSO is just the archive solutions. Theoretically speaking, the more the size of archive is, the better the performance of MoPSO is. Considering the computation burden of maintaining big size archive, let the archive size NA be the same with the swarm size *NP*. Let the maximum iterative number of MoPSO be *NM*. The procedure of updating $ND_a$ is as follow.

Step1. For the first iterative t:=1, find the non-dominated solutions NDc and let the $ND_a = \Phi$;

Step2. For $i \in NDc$,

If $\exists j \in ND_a, i \succ j$, then *i* substitutes *j* and enters $ND_a$;

Otherwise, if $\forall j \in ND_a, \neg\exists j \succ i$, then *i*enters $ND_a$ and |NDa|:=|NDa|+1;

Step3. If $|ND_a| > NA$,

Compute the crowded distance and fitness of each individual within $ND_a$;

Sort the individuals in $ND_a$ as descending order in light of their fitness;

Reserve the fear *NA* individuals as $ND_a$.

Step4. Update the crowded distance and fitness of each individual within $ND_a$;

Step5. If t=NM, end; otherwise t:=t+1, and go to Step2.

From above equation, it is clear that the individual with big value of crowded distance will enter the $ND_a$ with high probability. The use of Eq.(3) is to maximize the spread of solutions found, so that we can have a distribution of vectors as smooth and uniform as possible. The computation of crowded distance is the same with the method in [22].

According to above description of MoPSO, the parameters of MoPSO is as follows: the swarm size *NP*, the maximum iterative number *NM*, the archive size *NA*, leaning factors $C_1$ and $C_2$, inertia weight $\omega$.

## IV. CASE STUDY

Two cases derived from literature [6] are used to illustrate effectiveness of our approach. Two parts ANC-90 (Part A) and ANC-101 (Part B) will be produced. The demand rates of parts A and B are 120 and 180 parts/hour, respectively. The precedence graphs of two parts are shown in Fig .3. The machines and their capacity are provided in Table 1. The objective is to find the solutions with tradeoff between fixed cost and availability.

To verify the effectiveness of the MoPSO, the MoPSO and NSGA-II [22] are employed to solve two cases. The code of NSGA-II was downloaded from http://www.egr.msu.edu/~kdeb/codes/nsga2/nsga2code.ta.For the ALDP, corresponding functions for computing cost and availability were programmed with C++. Both the MoPSO and NSGA-II were implemented by Visual C++ 6.0 on a PIV 3.2GHz PC with 2GB memory. The configuration of MoPSO is as follows: *NM*=1000, $C_1=C_2=2.0$, $\omega = 0.9$, *NA*=*NP*. The parameters of genetic operators for NSGA-II are set as the values recommended by [22]. The population sizes of MoPSO and NSGA-II are the same, *NP*=20 for part A and *NP*=30 for part B. The maximum iterative number of NSGA-II is also 1000. For each case, five replicates (runs) were implemented.

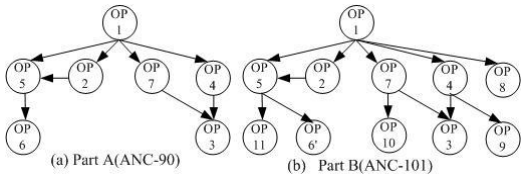


Figure 3. Precedence graphs for parts A and B

Three metrics are used to measure the performance of MoPSO and NSGA-II. The first metric is the error rate (ER) [21], which indicates the percentage of solutions (the non-dominated vectors found so far) that are not members of the true Pareto optimal set. It is clear that ER=0 indicates an ideal behavior. The second is the spacing (SP). A value of zero of SP indicates all members of the Pareto front currently available are equidistantly spaced [21]. The third metric is the computational time (CT) required by the replicate, which is used to measure the computational efficiency. The computational results of two algorithms against two cases are listed in Table 2~3. A solution for case 1 is shown on the right in Fig .1.

TABLE I MACHINES AND THEIR CAPICITY (PARTS/HOUR)

| Type | MC1 | MC2 | MC3 | MC4 | MC5 | MC6 | MC7 | MC8 | MC9 |
|---|---|---|---|---|---|---|---|---|---|
| cost/KUSD | 860 | 1140 | 1420 | 1700 | 1010 | 385 | 555 | 725 | 895 |
| Availability | 0.92 | 0.90 | 0.88 | 0.86 | 0.90 | 0.94 | 0.92 | 0.90 | 0.88 |
| OP1 | 120 | 240 | 360 | 480 | 120 | - | - | - | - |
| OP2 | 180 | 360 | 540 | 720 | 180 | - | - | - | - |
| OP3 | 120 | 240 | 360 | 480 | 120 | 120 | 240 | 360 | 480 |
| OP4 | 180 | 360 | 540 | 720 | 180 | - | - | - | - |
| OP5 | - | - | - | - | 60 | - | - | - | - |
| OP6 | 40 | 80 | 120 | 160 | 40 | 40 | 80 | 120 | 160 |
| OP6' | 30 | 60 | 90 | 120 | 30 | 30 | 60 | 90 | 120 |
| OP7 | 200 | 400 | 600 | 800 | 200 | - | - | - | - |
| OP8 | - | - | - | - | 180 | - | - | - | - |
| OP9 | - | - | - | - | 90 | - | - | - | - |
| OP10 | - | - | - | - | 200 | - | - | - | - |
| OP11 | 150 | 300 | 450 | 600 | 150 | - | - | - | - |

From the ER metric listed in Table 2~3, it can be seen that MoPSO is superior to the NSGA-II for two cases. This means that MoPSO outperforms the NSGA-II with respect to the solution quality. According to the SP metric shown in Table 2~3, it is clear that MoPSO is better than NSGA-II for two cases. This result indicates that the solutions obtained by MoPSO distribute more uniformly than those found by NSGA-II. As for the CT metric, it is clear that the MoPSO is dramatically superior to NSGA-II. The reason may be that the mechanism of MoPSO is simpler than that of the NSGA-II. In a whole, the MoPSO outperforms NSGA-II for solving the addressed ALDP.

TABLE II COMPARISON OF COMPUTATIONAL RESULTS FOR CASE 1(PART A)

| Metrics | Algorithm | Run1 | Run2 | Run3 | Run4 | Run5 | Mean |
|---|---|---|---|---|---|---|---|
| ER | MoPSO | 1/9 | 4/9 | 3/8 | 0/8 | 1/8 | 21.1% |
| | NSGA-II | 3/4 | 4/5 | 4/5 | 3/4 | 4/5 | 78.0% |
| CT /sec. | MoPSO | 0.125 | 0.125 | 0.125 | 0.11 | 0.125 | 0.122 |
| | NSGA-II | 1.281 | 1.297 | 1.275 | 1.297 | 1.266 | 1.283 |
| SP | MoPSO | 0.302 | 0.286 | 0.302 | 0.306 | 0.308 | 0.301 |
| | NSGA-II | 0.389 | 0.423 | 0.437 | 0.439 | 0.399 | 0.418 |

TABLE III COMPARISON OF COMPUTATIONAL RESULTS FOR CASE 2(PART B)

| Metrics | Algorithm | Run1 | Run2 | Run3 | Run4 | Run5 | Mean |
|---|---|---|---|---|---|---|---|
| ER | MoPSO | 10/14 | 10/16 | 9/12 | 13/17 | 8/12 | 70.4% |
| | NSGA-II | 3/4 | 4/5 | 3/4 | 4/5 | 3/4 | 77% |
| CT /sec. | MoPSO | 0.359 | 0.422 | 0.437 | 0.485 | 0.484 | 0.437 |
| | NSGA-II | 2.235 | 2.437 | 2.422 | 2.253 | 2.125 | 2.294 |
| SP | MoPSO | 0.213 | 0.203 | 0.223 | 0.199 | 0.228 | 0.213 |
| | NSGA-II | 0.486 | 0.390 | 0.426 | 0.394 | 0.463 | 0.432 |

## V. CONCLUSION

A multi-objective PSO is presented for the ALDP with station paralleling and equipment selection in this paper. The considered objectives are to minimize investment cost of machines and maximize the availability of the assembly line. The MoPSO is based on the concepts of crowded distance and external Pareto solution archive. The developed solution representation of MoPSO ensures each particle to be a feasible solution. The effectiveness of the MoPSO is verified by case study. The computational results of comparison between MoPSO and NSGA-II against two cases show that the MoPSO is better than the NSGA-II with regards to solution quality and computational efficiency for the addressed ALDP.

Although the MoPSO is effective for ALDP, the MoPSO with advanced operator such as dynamic leaning factors and inertia weight is needed to be further investigated. In addition, the MoPSO should be compared with other multi-objective meta-heuristics against more cases in the future.

## ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No. 51105076 and 61374069).

## REFERENCES

[1]J. Bukchin and A. Rubinovitz, "A weighted approach for assembly line design with station paralleling and equipment selection," IIE Transaction, pp. 73-85, 2003.

[2]G. Nicosiaa, DarioPacciarelli, and A. Pacifib, "Optimally balancing assembly lines with different workstations," Discrete Applied Mathematics, vol. 118, pp. 99-113, 2002.

[3]A. Dolgui and I. Ihnatsenka, "Branch and bound algorithm for a transfer line design problem: Stations with sequentially activated multi-spindle heads," European Journal of Operational Research, vol. 197, pp. 1119-1132, 2009.

[4]S. Barutcuoğlu and M. Azizoğlu, "Flexible assembly line design problem with fixed number of workstations," International Journal of Production Research, vol. 49, pp. 3691-3714, 2011.

[5]O. Guschinskaya, E. Gurevsky, A. Dolgui, and A. Eremeev, "Metaheuristic approaches for the design of machining lines," The International Journal of Advanced Manufacturing Technology, vol. 55, pp. 11-22, 2011.

[6]J. Dou, X. Dai, and Z. Meng, "A GA-based approach for optimizing single-part flow-line configurations of RMS," Journal of Intelligent Manufacturing, vol. 22, pp. 301–317, 2011.

[7]K. K. Goyal, P. Jain, and M. Jain, "Optimal configuration selection for reconfigurable manufacturing system using NSGA II and TOPSIS," International Journal of Production Research, vol. 50, pp. 4175-4191, 2012.

[8]R. J. Kuo and C. Y. Yang, "Simulation optimization using particle swarm optimization algorithm with application to assembly line design," Applied Soft Computing, vol. 11, pp. 605-613, 2011.

[9]J. Dou and X. Zhao., "A Particle Swarm Optimization for Assembly Line Design with Station Paralleling and Equipment Selection," Proc. 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE 2012), Zhangjiajie,China, 2012, pp. 468-472.

[10]X. Delorme, O. Batta a, and A. Dolgui, "Multi-objective Approaches for Design of Assembly Lines," in Applications of Multi-Criteria and Game Theory Approaches, ed: Springer, 2014, pp. 31-56.

[11]N. Pekin and M. Azizoglu, "Bi criteria flexible assembly line design problem with equipment decisions," International Journal of Production Research, vol. 46, pp. 6323-6343, 2008/11/15 2008.

[12]G. Michalos, S. Makris, and D. Mourtzis, "An intelligent search algorithm-based method to derive assembly line design alternatives," International Journal of Computer Integrated Manufacturing, vol. 25, pp. 211-229, 2012.

[13]B. Rekiek, P. De Lit, F. Pellichero, T. L'Eglise, P. Fouda, E. Falkenauer, and A. Delchambre, "A multiple objective grouping genetic algorithm for assembly line design," Journal of Intelligent Manufacturing, vol. 12, pp. 467-485, 2001.

[14]U. Saif, Z. Guan, W. Liu, C. Zhang, and B. Wang, "Pareto based artificial bee colony algorithm for multi objective single model assembly line balancing with uncertain task times," Computers & Industrial Engineering, vol. 76, pp. 1-15, 2014.

[15]A. C. Nearchou, "Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization," International Journal of Production Economics, vol. 129, pp. 242-250, 2011.

[16]P. Chutima and P. Chimklai, "Multi-Objective Two-Sided Mixed-Model Assembly Line Balancing using Particle Swam Optimisation with Negative Knowledge," Computers & Industrial Engineering, vol. 62, pp. 39-55, 2012.

[17]A. M. A. Youssef, A. Mohib, and H. A. ElMaraghy, "Availability Assessment of Multi-State Manufacturing Systems Using Universal Generating Function," CIRP Annals - Manufacturing Technology, vol. 55, pp. 445-448, 2006.

[18]J. Kennedy and R. Eberhart, "Particle swarm optimization," Proc. IEEE international conference on neural networks, Perth,Australia, 1995, pp. 1942-1948.

[19]J. Dou, C. Su, and J. Li, "A discrete particle swarm optimization algorithm for assembly line balancing problem of type 1," Proc. Third International Conference on Measuring Technology and Mechatronics Automation (ICMTMA 2011), Shanghai, 2011, pp. 44-47.

[20]J. Dou, X. Dai, and Z. Meng, "Optimization for multi-part flow-line configuration of reconfigurable manufacturing system using GA," International Journal of Production Research, vol. 48, pp. 4071-4100, 2010.

[21]C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," IEEE Transactions on Evolutionary Computation, vol. 8, pp. 256-279, 2004.

[22]K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, vol. 6, pp. 182-197, 2002.