Vector Evaluated Particle Swarm Optimization Approach to Solve Assembly Sequence Planning Problem

Jameel A. A. Mukred[1], Ruzairi A. Rahim[2]*, Elmy Johana Mohamad[1], Adeeb Salh[1], Adel Y.I. Ashyap[1], Qazwan Abdullah[1], Najib Al-Fadhali[1], S. Al-Ameri[1], Rania Al-Ashwal.
[1] Universiti Tun Hussein Onn Malaysia, Parit Raja, 86400, Johor, Malaysia.
[2*] Universiti Teknologi Malaysia, Skudai, 81310, Johor, Malaysia.
[3] Universiti Tun Hussein Onn Malaysia, Pagoh, Muar, Johor, Malaysia.

Corresponding author* email: jameel@uthm.edu.my
Available online 20 June 2022

ABSTRACT

Multi-criteria Assembly Sequence Planning (*ASP*) is known as large scale, time consuming combinatorial problem. *Vector Evaluated Particle Swarm Optimization (VEPSO)* is a computational approach modified from Particle Swarm Optimization (*PSO*) to solve the multi-objective problems optimization. Multi-objective optimization problem consists of several objectives that needed to be achieve at the same time. This problem occurs in many applications with the simultaneous optimization of time, cost, orientation, and etc where these factor sometimes compete and/or incommensurable.

Keywords: Assembly sequence planning, vector evaluated, multi-objective, parito optimal, pareto front.

1. Introduction

MULTI-CRITERIA Assembly sequence planning (ASP) is known as large scale, time consuming combinatorial problem. Production scheduling is a complex combined optimization problem and the optimization method of which is not perfect [1]. The product order of assembly is the main focus of ASP to determine, which is subject to precedence constraint matrix (PM) that is to be strictly followed in the assembly line to shorten the assembly time and hence save the assembly cost. Refs. [2, 3] proposed the concept of Assembly Precedence Relations (APRs), which is applied to determine the precedence relations among the liaisons in the product. Cut-set analysis method by which the number of queries can be reduced by 95% [4]. More efficient queries is proposed in ref. [5]. When number of parts increase the problem became more complex. Heuristic methods developed to overcome this complicity. It is more efficient but it may stick in local optima, no guarantee that global optima may be found. Some heuristic methods may use Neural Network (NN), which need system training before start searching. Meta-heuristic method is able to escape the local optima. Simulated Annealing (SA) is used where search is done in sequence basis and to solve optimization problems. Ref. [6] used (SA) approach, which is based on searching via all the feasible sequences. This disadvantage is overcome by an improved cut-set [7, 8]. Generation and evaluation of assembly plans, when the number of parts is large their planer is slow [9]. Genetic Algorithm (GA), where the genes in chromosomes represents the components of the product [10, 11]. An integrated approach such that liaison graph represents the physical connections between two components [15]. An extension to previous work is proposed in [16]. Finding a method to determine global optima or near global optima more reliably and quickly [17]. The definition of genes and evaluation criteria here are based on the connector concept [18]. The complete or partial automation of assembly of products in smaller volumes and with more rapid product changeover and model transition has enabled through the use of programmable and flexible automation. AI is increasingly playing a key role in such flexible automation systems [19].

The total assembly time is the combination of the preparation time and the practical assembly time. The practical assembly time is assumed to be constant regardless of the assembly sequence. Each component to be assembled requires the proper tool and initialization, which depends upon the geometry of the component itself and the components assembled to that point. The preparation time of a component can be calculated by using the following formula:

$$D_{prep}(i) = n_{i0} + \sum_{j=1}^{r} n_{ij}\, m_{ij} \quad (1)$$

Where:

(i) : Number of component to be assembled, for i=1, …r.

n_{i0} : Preparation time for product (i) being the first component, for i=1, …r.

n_{ij} : Contribution to preparation time due to a presence of part (j) when entering part (i).
m_{ij} = 1 if component j has already assembled, for i=1, ...,r
= 0 otherwise, for i = 1,...,r

The total assembly time is the summation of the preparation time and the practical assembly time.

$$D_{assm(\min)} = \sum_{j=1} (n_{i0} + \sum_{j=1} n_{ij}\, m_{ij} + K_i) \quad (2)$$

Where K_i is the assembly time for component i.

The precedence relationship between the components is one of the constraints factor in the assembly design. Therefore, the precedence constraints of the assembly design must be satisfied for a feasible assembly plans. The precedence constraints of 19 components can be represented by a directed graph as shown in Figure 1.1. From the *CAD* or disassembly analysis, the precedence constraints were determined earlier as shown in Figure 1.2. Figure 1.3 shows the setup time for each component with respect to other components. The first column of Figure 1.3 represents the components to be assembled and the first row represents the components already assembled.

A precedence matrix (*PM*) is proposed to describe the precedence constraints between the parts in assembly. A *PM* will describe all precedence constraints for the product, as shown in Figure 1.2, the precedence constraints between part i and j, PM (i, j). If part i needed to be assembled after part j, PM (P_i, P_j) =1, otherwise PM (P_i, P_j) = $\emptyset$. Let Ω be the set of the parts already assembled before part i. The union of PM is defined as a feasible assembly sequence FA (P_i) with the given constraints. Thus,

$$FA\,(P_i, P_j) = \bigcup PM\,(P_i, P_j), \qquad P_j \notin \Omega \quad (3)$$

Therefore, part i can be assembled if $FA\,(P_i, P_j)$ is an empty set because all part must be assembled before part i were already assembled. But if $FA\,(P_i, P_j)$ is not an empty set, then part i cannot be assembled in the given order and the given assembly plans is invalid.

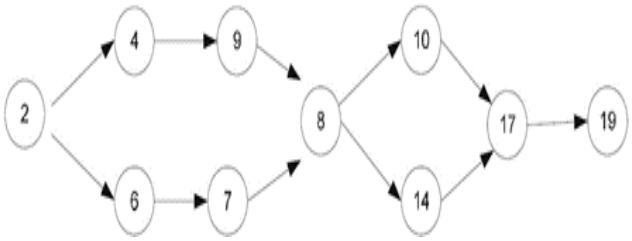

Fig. 1.1. The assembly precedence diagram

Component to be assembled (i)	Component (j)																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1																			
2																			
3																			
4		1																	
5																			
6		1																	
7		1				1													
8		1		1		1	1		1										
9		1		1															
10		1		1		1	1	1	1										
11																			
12																			
13																			
14		1		1		1	1	1	1										
15																			
16																			
17		1		1		1	1	1	1	1				1					
18																			
19		1		1		1	1	1	1	1				1			1		

Fig. 1.2. Precedence Matrix

Component to be assembled	Component *i*																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	10	1	2	3	4	5	6	7	8	9	3.2	4.3	7	6.1	1.2	3.4	0	0	7.4
2	1.5	10	2	2	2	2	2	2	2	2	0	3.1	6	4.3	2.7	4.8	0	3	0.5
3	1	2.3	10	0	4	5	0	4	2.3	4.3	9.8	2.4	5	1.2	3.4	4.5	5.6	3.4	3.1
4	0	2	3.4	10	4.5	0	4	0	8	0	3.4	5.6	5	0	0	3.4	0	0	9.8
5	1.2	1	2	3	10	7.9	8.9	0	1.2	2	2.3	0	3	0	3.6	0	2.8	9.8	0
6	9.8	4.5	0	1.2	3.6	10	3.4	4	0	2.3	4.6	5.6	0	4	3	2	0	0.4	3.2
7	0.5	1.4	2.3	0.5	1.9	1	10	13.4	1.2	4	2.3	0	3	5.7	8.30	2	0.1	0	0.5
8	0	0	0	0	0	1.8	9.8	10	2.3	3	8.9	2.3	0	0	2.3	0.5	9.8	0	2.3
9	1	3	4.5	2.3	4.6	9.8	7.5	6.8	10	6	2.3	3.4	5	12.3	3.4	5.61	1	0	0
10	2.3	4.5	2.3	0	2.3	0	2.1	0	4.5	10	1.1	2.3	2	0	0	2.1	1.2	5.4	9.2
11	1	1	2	3	4	5	6	7	8	9	10	4.5	3	6.1	1.2	3.4	0.3	0	1.3
12	1.5	0	2	2	2	2	2	1	2	2	11.2	10	6	4.3	2.7	4.8	0	3	0.5
13	1	2.3	0	0	4	5	0	4	2.3	4.3	9.8	2.4	10	1.2	2.4	4.5	1.6	2.4	3.1
14	0	2	3.4	0	4.5	0	4	0	8	0	3.4	5.6	5	10	2.1	1.4	1	0	2.8
15	1.2	1	2	3	0	7.9	8.9	0	1.2	2	1.3	4	3	1.4	10	1.3	9.8	9.8	2
16	9.8	4.5	0	1.2	3.6	0	3.4	4	0	2.3	4.6	3.6	0	4	3	10	1.5	0	3.2
17	1	3	4	5	0	5	4	3.4	1.2	4	1.3	0	2	3.7	4.3	2.3	10	3.8	10
18	0.6	0.5	3.4	1.2	3	2	9.8	2	2.3	3	5.9	2.3	0	1.0	2.3	0.5	9.8	10	2.3
19	1	3	4.5	2.3	4.6	9.8	7.5	6.8	0	6	3.3	3	2	3.3	4.4	2.6	0.3	2.5	10

Fig. 1.3. Coefficient of various components in the assembly

For example, if there are three components, 4,7 and 9 already assembled in the product and component 1 needing to be assembled, the total setup time can be determined by the equation (4).

$$D_{prep}(1) = n_{10} + n_{14}m_{14} + n_{17}m_{17} + n_{19}m_{19} \quad (4)$$
$$= 10 + 3 + 6 + 8$$
$$= 27 \text{ time units}$$

Where, $n_{14} = n_{17} = n_{19} = 1$

2. Basic Concept of Multi-Objective Optimization

Let the function

$$f(x) = [f_1(x),\ f_2(x), \ldots, f_k(x)]^{\mathrm{T}} \quad , \mathbb{R}^{\mathrm{n}} \rightarrow \mathbb{R}^{\mathrm{k}}. \quad (5)$$

The objective functions may be in conflict; therefore, in most of the cases it is not possible to obtain the global minimum at the same point for all the objectives. The goal of the multi-objective function is to provide a set of Pareto optimal solution to the multi-objective problems Eq. (5).

2.1. Pareto Optimality

Let $u=(u_1,\ldots,u_k)$,and $v=(v_1,\ldots,v_k)$ be two vectors, u will dominates v if and only if $u_i \leq v_i$, $i = 1, \ldots, k$, and $u_i < v_i$ for at least one component. The properties mention is known as Pareto dominance, and it is used to define the Pareto optimal points. Therefore, the solution x of the multi-objective problem is said to be Pareto optimal if there is no other solution, says *f(y)* to dominate the *f(x)*. A set of Pareto optimal is denoted as Pareto front as shown in figure 1.4 below [20, 21].

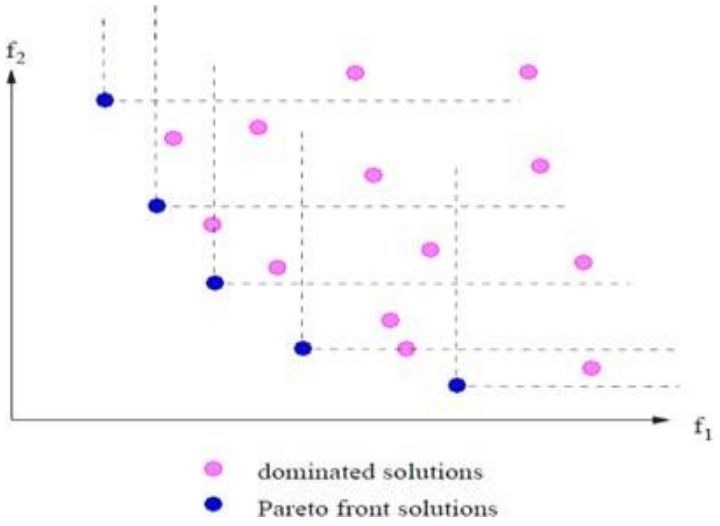

Fig. 1.4. Pareto Front Solution

2.2. Particle Swarm Optimization

Particle Swarm Optimization (*PSO*) is stochastic optimization approach, which was modeled based on the social/swarm behavior of bird flocks. *PSO* is a population based search process where the individual, which is called as 'particles', are grouped into a swarm, then each particle i is assigned with random position, $X_i = (x_{i,1}, x_{i,2}, \ldots x_{i,j})$ and velocity, $V_i = (v_{i,1}, v_{i,2}, \ldots v_{i,j})$ [21].

In PSO, each particle is 'flown' through the multidimensional search space. At each time steps, the particle will adjust its position in search space according to its own experience and that of neighboring particles. The best position (fitness) of that particle i have been achieved so far is stored as $Pbest_i = (pbest_{i,1}, pbest_{i,2}, \ldots pbest_{i,j})$, while the best position that have been achieved by the group of the swarm is stored as the $Gbest_i = (gbest_{i,1}, gbest_{i,2}, \ldots gbest_{i,j})$ [22, 23].

By comparing the pbest and the gbest values, the particle will move to the optimum solution. The performance of each particle is measured according to a predefined fitness function which is related to the problem being solved.

The standard *PSO* algorithm can be describe in the following equations (6) and (7):

$$v_{ij}(t+1) = \omega v_{ij}(t) + c_1 rand_1\ (pbest_{ij}(t) - x_{ij}(t)) + c_2 rand_2 (gbest_{ij}(t) - x_{ij}(t)) \quad (6)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (7)$$

Where,
ω = inertia weight factor
c_1 = cognition weight factor
c_2 = social weight factor
v_{ij} (t) = j^{th} dimension velocity of particle *i* at time *t*
x_{ij} (t) = j^{th} dimension position of particle *i* at time *t*
$rand_1$ & $rand_2$ = random number uniformly range from [0,1]
$pbest_{ij}(t)$ = jth dimension of the own best particle *i*.
$gbest_{ij}(t)$ = jth dimension of the best particle *i*.

The algorithm above will continue searching until the convergence has reached. Normally, the *PSO* algorithm is executed for a fixed number of iteration, or fitness function is evaluated, we call these factors as stopping criteria. In the other words, *PSO* algorithm can be terminated if the velocity changes are close to zero for all the particles, in which case there will be no further changes in particle's position.

2.3. Vector Evaluated Particle Swarm Optimization

VEPSO is a multi-swarm variant of *PSO*. *VEPSO* is inspired by the concept of the Vector Evaluated Genetic Algorithm (*VEGA*). In *VEPSO*, each swarm is evaluated using only one of the objective functions of the problem. The information that it possesses for this objective function is communicated to the other swarms through the exchange of their best experience. The best position attained by each particle separately as well as the best among these positions are the main guidance mechanisms of the swarms. Through the exchanging information, the Pareto Optimal points can be obtained [24].

The standard *VEPSO* algorithm can be describe in the following equation (8):

$$v_{ijs}(t+1) = \omega v_{ijs}(t) + c_1 rand_1(t)[\hat{x}_{ijs}(t) - x_{ijs}(t)] + c_2 rand_2(t)[x^*_{jm_s}(t) - x_{ijs}(t)] \quad (8)$$

Where,
$v_{ijs}(t)$ velocity of the jth dimention of the ith particle of swarm '*s.*' at time '*t.*'
$x_{ijs}(t)$ position of the jth dimention of the ith particle of swarm '*s.*' at time '*t.*'
$\hat{x}_{ijs}$(t) *pbest* of the jth dimention of the ith particle of swarm '*s.*' at time '*t.*'
$x^*_{jm_s}(t)$ jth dimension of the global best of swarm '*s.*' at time '*t.*'

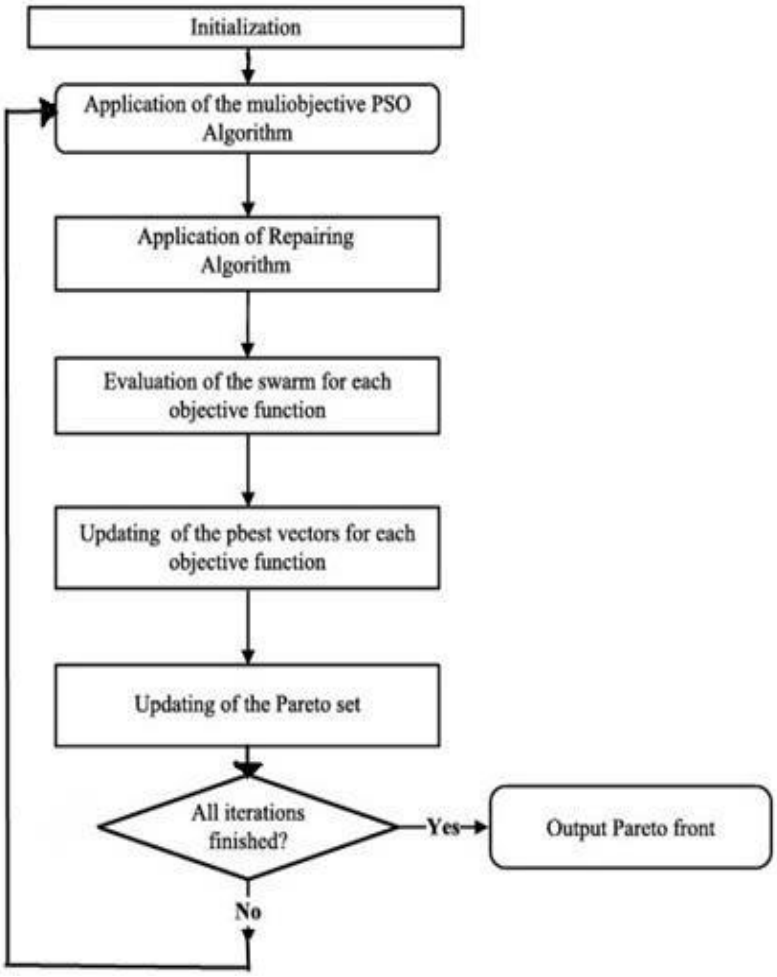

Fig. 1.5. Flow chart of VEPSO algorithm

3. Result and Discussion

The project involved some research on some criteria that will lead to a better solution of Pareto Front, which minimized the total tool change as well as the assembly time. In order to get the Pareto Front, some criteria have to be fixed,

The inertia weight, ω_{max} and ω_{min}
ω_{max} = 0.9
ω_{min} = 0.4
The correction factor, c_1 and c_2
$c_1 = c_2 = 1.42$

After the criteria have been fixed, then the result can be obtained through the varying the number of iterations and number of particles. To investigate the effect of the number of iterations of the algorithm to the result obtained, the number of particles is firstly fixed to 40, 60, 80 and 100 particles and run with 1000, 3000, 5000 and 8000 iterations. To make sure the result obtained is the minimized value, each iteration is run for 10 times and the best sequence is obtained through the results in the following tables (1-5).

Table 1. Result for 40 particles with different iterations

Total tool change	Total assembly time
7	513.2
9	510.0
10	508.7
11	506.6
12	502.2
13	500.6

Table 2. Result for 60 particles with different iterations

Total tool change	Total assembly time
7	511.8
9	511.8
10	506.1
11	505.6
12	500.4

Table 3. Result for 80 particles with different iterations

Total tool change	Total assembly time
7	512.6
8	512.1
9	510.6
10	504.1
12	501.2
15	500.0

Table 4. Result for 100 particles with different iterations

Total tool change	Total assembly time
6	524.9
7	513.6
8	511.7
9	509.7
10	505.3
11	504.6
12	500.2

Table 5. Result for different particles with different iterations

Total tool change	Total assembly time
6	524.9
7	511.8
8	511.7
9	509.7
10	504.1
12	500.2
15	500.0

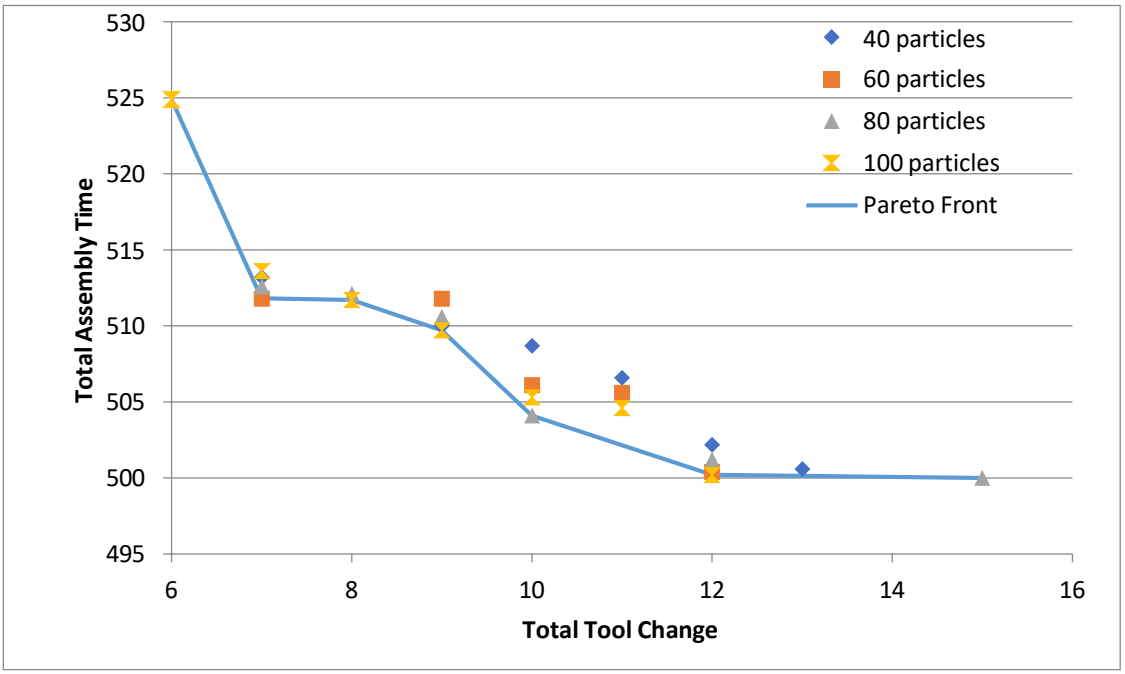

Fig. 1.6. Pareto Front of different particles and different iterations

Through the graph that connect the Pareto Front, it shows that the line connects mostly from the result of 80 and 100 particles. From the result obtained, it shows that with the increases in the number of iterations and particles, a better result can be obtained. Table 6 shows the sequences for the Pareto Front solutions.

Table 6. Sequences for the Pereto Front solutions

Sequence																			Tool change	Assembly time
1	18	13	16	11	5	15	12	2	4	3	9	6	7	8	10	14	17	19	6	524.9
1	3	13	12	15	5	11	16	2	4	18	6	9	7	8	14	10	17	19	7	511.8
1	2	13	16	12	3	6	5	15	11	18	7	4	9	8	10	14	17	19	8	511.7
1	15	2	4	3	12	13	9	16	5	18	11	6	7	8	14	10	17	19	9	509.7
2	1	3	6	11	15	5	18	16	7	13	12	4	9	8	14	10	17	19	10	504.1
1	15	18	5	16	12	13	3	2	4	6	9	7	8	10	14	11	17	19	12	500.2
1	11	13	12	3	18	15	2	6	4	16	7	5	9	8	14	10	17	19	15	500.0

4. Conclusion

The Assembly Sequence Planning of 19 components has been solved using the Vector Evaluated Particle Swarm Optimization at which two swarms representing two objective functions; one is the total assembly time and the other is the number of tool changing while assemble the product. The paper contribution was in finding the Pareto front on which the optimal location at which the factory can operate to achieve the best time to produce the product with two conflicting objectives. With the increased in the number of iterations and number of particles in the experiment, the better solution can be obtained. Pareto front is used in this project to determine the non-dominated solutions in the problem space. The better the non-dominated solution that can be obtained through the Pareto Front means that the industry can save more time in the production line and hence reduced the product cost of assembly.

Acknowledgment

Authors would like to express their appreciations to the people who are directly or indirectly involved in this research for their suggestion and valuable comments. A special thanks to Universiti Tun Hussein Onn Malaysia (UTHM) and Ministry of Higher Education Malaysia (MOHE) for the support in term of facilities of Research University Grant (Tier1-H910) Universiti Tun Hussein Onn Malaysia.

References

[1] A. Liu, Y. Yang, Q. Xing, H. Yao, Y. Zhang, and Z. Zhou, Adv. Sci. Lett. 4, 2180-2183 (2011)
[2] Q. Liu, M. Dong, Adv. Sci. Lett. 4, 2369-2373 (2011)
[3] P. Yao, C. Tan, and L. Zhou, Adv. Sci. Lett. 4, 2352-2355 (2011)
[4] M. Mohammad jafari, S. Ahmed, S. Md. Dawal, H. Zayandehroodi, Adv. Sci. Lett., 4, 2513-2516 (2011)
[5] H. Lv and C. Lu, "An assembly sequence planning approach with a discrete particle swarm optimization algorithm," Int J Adv Manuf Technol, 2010, 170-010-2519-4
[6] A. Bourjault, "Contribution une approche methodologique del'assemblage automatise: elaboration automatique des sequences operatiores," France: Thesis d'Etat Universite de Franche-Comte, Besancon,1984.
[7] L. S. Homem de Mello, A. C. Sanderson, IEEE Trans. Robot. Autom (1991)
[8] D. F. Baldwin, T.E. Abell, M-C Lui, T.L. de Fazio, D. E. Whitney, IEEE Trans. Robot. Autom. (1991)
[9] J. M. Milner, S. C. Graves, D. E. Whitney, Proceedings of the IEEE International Conference on Robotics and Automation, 2058–2063 (1994).
[10] R. H. Wilson, IEEE Trans. Robot. Automn., 11(2): 308–311, (1995)
[11] S. Motavalli, A. Islam, Computational Industrial Eng., 32(4): 743–751 (1997)
[12] F. Bonneville, C. Perrard, J. M. Henrioud, "A genetic algorithm to generate and evaluate assembly plans," Proceedings of the IEEE Symposium on Emerging Technology and Factory Automation, 1995, 231–239.
[13] M. F. Sebaaly, H. Fujimoto, Japan / USA Symposium on Flexible Automation, 2:1235–1240, (1996)
[14] M. F. Sebaaly, H. Fujimoto, IEEE Conference on Evolutionary Computation, 401–406, (2000)
[15] P. De Lit, P. Latinne, B. Rekiek, A. Delchambre, International Journal Production Res., 39 (16): 3623–3640, (2001)
[16] S. F. Smith (Chen), Y. J. Liu, Journal Industrial Technology 17(4), (2002)

[17] S. F. Smith (Chen), Y. J. Liu, "The application of multi-level genetic algorithm in assembly planning," Journal Industrial Technology 17(4)
[18] H. E. Tseng, J. D. Li, Y. H. Chang, "Connector-based approach to assembly planning using a genetic algorithm," International Journal Prod Res; 2004; 42(11):2243–2261
[19] R. M. Marian, H. S. Luong, K. Abhari, "A genetic algorithm for the optimization of assembly sequence," Computational Industrial Eng ; 2006; 50:503–527
[20] K. E. Parsopoulos, D. K. Tasoulis, M. N. Vrahatis, "Multiobjective Optimization Using Parallel Vector Evaluated Particle Swarm Optimization," *Proceedings of the 2004 IASTED International Conference on Artificial Intelligence and Applications (AIA 2004)*, innsbruck, Austria, vol. 2, pp 823-828.
[21] J. Kennedy and R.C. Eberhart, "Particle Swarm Optimization," *In Proc. IEEE International Conference on Neural Networks*, 1995. pp 1942-1948.
[22] P. A. Engelbrecht, "Particle Swarm Optimization" second edition, In Pretoria, Computational Intelligence An Introduction, 2007, pp 289-357, Wiley, England.
[23] R. C. Eberhart, Y. Shi, "Particle Swarm Optimization: Developments, Applications and Resources," 0-7803-6657-3/01,2001,pp 81-86.
[24] J. D. Schaffer, "Multiple Objective Optimization With Vector Evaluated Genetic Algorithms," PhD thesis, Vanderbilt University, Nashville, TN, USA, 1984.