8th IEEE International Conference on Automation Science and Engineering
August 20-24, 2012, Seoul, Korea

An efficient PSO for type II robotic assembly line balancing problem

J. Mukund Nilakantan, S.G.Ponnambalam

Abstract—**In this modern world of technology, robots are extensively used in an assembly line. Different robots may be allocated to the assembly tasks, and each robot requires different assembly times to perform a given task, because of its capabilities and specialization in robotic assembly line systems. Use of robots helps us to improve the productivity of the line and increases the quality of the product. To find an optimal solution for Robotic Assembly Line Balancing (rALB) problem we will have to assign robots to stations in a balanced manner to perform activities. The main objective of this work is to minimize the cycle time and maximize the production rate of the line. A particle swarm optimization method is proposed to find optimum solution for the rALB problem. Results obtained using the PSO are further improved by using a local exchange procedure. Performance of the proposed method is tested on benchmark rALB problems. The results of PSO are found to be better than the methods reported in the literature and it produce consistent results.**

I. INTRODUCTION

In the past decades, robots have been widely used in assembly systems and these are called robotic assembly lines. As there is no need of taking breaks while using robots and its ability to make it work for 24hours/7day, a robot can increase the productivity dramatically.It increases the higher rates of production which means higher production. Robots helps to produce products much faster than traditional methods there by reducing the cycle time. The use of robots creates a leaner and more efficient manufacturing cycle. Major important advantages of using robots are it increases productivity,quality of the product increases,less need of skilled labors and safety.Most important problem in this context is to how assembly lines are managed and how the assembly is balanced.

The robot could be programmed to perform a wide variety of tasks and applications. Yet, different robot types are available for use in an assembly facility, and they usually come with different capabilities and efficiencies for the various elements of assembly tasks. Hence, to allocate a proper robot for each station is critical for the performance of robotic assembly lines. The robotic assembly line balancing (rALB) problem is to assign tasks to workstations and to allocate robot for each station in order to improve the productivity. In a robotic assembly line, specific tooling is usually developed to perform the activities needed at each station. In case of manual assembly lines actual processing times for performing activities vary considerably and optimal balance is rather of theoretical importance, the performance of robotic assembly lines depends strictly on the quality of their balance and robot assignment.

Rubinovitz and Bukchin [1] first formulated the rALB problem by allocating equal amounts of work to stations on the line while assigning the most efficient robot type from the given set of available types for each station. Their objective is to minimize the number of workstations for a given cycle time. In a later work, Rubinovitz and Bukchin [2] presented a Branch and Bound (B&B) algorithm for the problem.

Levitin et al. [3] mainly dealt with a type II robotic assembly line balancing (rALB-II) problem, where robots of different capabilities and specialization were used to perform tasks and each robot had different performance times. It was assumed that any robots were available without any limitations. And cost of the purchase was not considered. The main objective is to allocate the task to the work stations and assign best robot in a systematic manner so that cycle time is minimized. Two types of genetic algorithms were presented to solve the rALB-II problem. In this when there is a new product added to the assembly line for production, the robotic assembly line should be configured properly and always help in improving the efficiency of the line and make the line always balanced. One of most important assumption in this algorithm is to have same number of robot and station. In this paper it presents a new rALB-II problem, which is to assign tasks to a fixed number of workstations and to allocate the available robots for each workstation with the objective of minimum cycle time.

Levitin et al. [3] developed a method for the robotic assembly line balancing (rALB) problem. In this method it mainly aims in achieving a balanced distribution of activities amongst the stations and to assign the best robot to the stations to perform the activities. Two methods are adapted for performing GA in this problem: a recursive and a consecutive procedure. A local exchange procedure is used to further improve the quality of solutions. In the paper the best possible combination of the procedure and GA parameters are reported by testing on the randomly generated data set and author claims the proposed method is consistent and robust.

Jie Gao et al. [4] presented a 0-1 integer programming problem for rALB and proposed hybrid genetic algorithm

*J Mukund Nilakantan is with the School of Engineering Monash University Sunway Campus,46150,PetalingJaya,Malaysia
(e-mail: mukund.janardhanan@monash,edu)
S.G.Ponnambalam is with the School of Engineering Monash University Sunway Campus,46150,Petaling Jaya,Malaysia
(e-mail: sgponnambalam@monash.edu)

(hGA) to find efficient solutions for the rALB-II problem. The genetic algorithm uses the partial representation technique, which expresses only part of the decision information about a candidate solution in the chromosome. The coding space contains only partial candidate solutions including the optimal one. New crossover and mutation operators were developed to adapt to the chromosome structure and the nature of the problem. In order to strengthen the search ability, local search procedures were also implemented by them.

The objective of this paper is to propose an efficient search heuristic to generate better solution for the rALB problems using PSO.

II. ABOUT rALB-II

In an assembly line setup, each work station is required to perform certain amount of activities to produce certain products. The precedence constraints needs to be specified and specifies h the order in which the tasks should be executed. In case of robot assembly there will be set of work stations and certain number of robots. The system needs to be configured for production of the product by assigning tasks to a particular station and assigning a particular robot to the work station in order to minimize the cycle time of the whole process.

Following assumptions as per Jie Gao [4] are used as the base for this work.

- The precedence relationship is known and they are notinvariable and assembly tasks cannot be subdivided
- The duration of an activity depends on the assigned robot.
- At a time only one robot can be assigned to a station.
- As the aim is to improve the productivity by reducing the cycle time,the number of work stations will be equal to number of robots.
- Any robot can be assigned to any station to perform certain tasks
- Material handling, loading and unloading time, as well as set-up and tool changing time are negligible, or are included in the activity time. This assumption is realistic on a single-model assembly line. Tooling on such robotic line is usually designed such that tool changes are minimized withina station. If tool change or other type of set-up activity is necessary, it can be included in the task time.
- The line is balanced for a single product with different activities to be done to get a final product.

Sample representation of the precedence graph is as shown in the Fig.1.In this paper we used recursive method [3] for finding the cycle time.

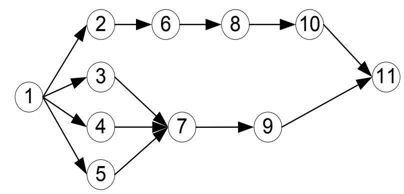

Fig.1 Example Precedence Graph

And the same recursive method is also used to divide the final sequence and assign the robots to stations to perform certain number of tasks as per the precedence constraints. Recursive method is explained in this section. The notations used are given below.

Notation:

- N_{st} total number of stations
- N_a total number of activities
- N_r total number of different types of robots
- $t_{r,j}$ time of performance of *j*th activity by robot r (if activity j cannot be performed by the robot r, $t_{r,j} = \infty$)
- τ_j average performance time for activity j
- *r (st)* number of robot assigned to station st
- T_{st} total execution time for station st
- *s* Sequence of activities represent feasible solution

This method assigns activities to the stations without disturbing the order of the activities in the sequence'*s*'. The recursive procedure was developed in order to divide the sequence into $M = N_{st}$ parts, and tries to achieve the maximal equality of total execution times for all stations.

The average performance time for each activity in the i^{th} position.

$$\tau_i = \sum_{r=1}^{N_r} t_{r,i}\delta_{r,i} \Big/ \sum_{r=1}^{N_r} \delta_{r,i} \quad (1)$$

where $\delta_{r,i} = 0$ if $t_{r,i} = \infty$ and $\delta_{r,i} = 1$

Otherwise sequence s is divided into two parts at a position i such that it satisfies the *H/Q* ratio where *H= [M/2] and Q =M –H*

To find the position '*i*' ($p1 \le i \le pr$) such that Time Ratio value (*TR*) is as close as possible to the ratio *H/Q*.

$$TR = \sum_{j=pl}^{i} \tau_{s(j)} \Big/ \sum_{j=i+1}^{pr} \tau_{s(j)} \qquad (2)$$

Using (2), position *i* divides the initial sequence into two parts, where *p1=1; pr=i* and *p1=i+1; pr=*N_a. Resulting parts is further divided into *M=H* and *M = Q* parts respectively using the same procedure iteratively until *M = 1*. At the end of the recursion, the sequence is divided based on the above conditions and the stations are fixed. After the allocation of activities to the stations robots are selected to minimize the total execution time for each station.

$$r(st) = \arg_{1 \le h \le N_r} \{T_{st}(h) = \sum_{k=p1_{st}}^{pr_{st}} t_{h,s(k)} = \min\} (3)$$

$p1_{st}$ and pr_{st} are the first and last elements of a part of a sequence corresponding to station *st*. An example of the recursive method procedure is shown in Fig.2 and performance times for the example is presented in Table 1

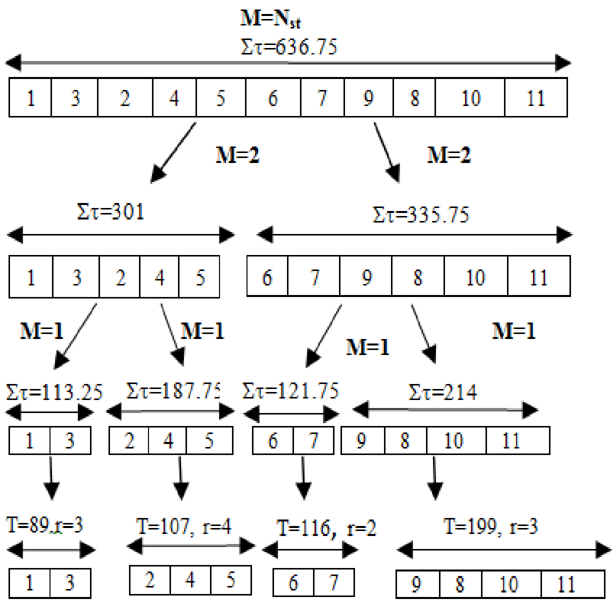

Fig. 2 Example of the recursive assignment procedure

Table I. Performance times for 11 activities by robots

Activities	*Performance Time*				
	Robot1	*Robot 2*	*Robot 3*	*Robot 4*	*Average Time(τ)*
1	81	37	51	49	54.5
2	109	101	90	42	85.5
3	65	80	38	52	58.75
4	51	41	91	40	55.75
5	92	36	33	25	46.5
6	77	65	83	71	74
7	51	51	40	49	47.75
8	50	42	34	44	42.5
9	43	76	41	33	48.25
10	45	46	41	77	52.25
11	76	38	83	87	71

III. Particle Swarm Optimization Algorithm For rALB-II

Particle Swarm Optimization algorithm is a computational method which was developed by Eberhart and Kennedy [5] in 1995 mainly based on the social behavior of bird flocking. PSO is a stochastic population based technique used for solving problems. PSO algorithm is been widely used to solve optimization problems. It is considered to be more robust compared to other algorithms. Most attractive features of PSO are its ease of implementation, robustness and very few parameters to be fine-tuned for achieving the results.

Particle Swarm Optimization algorithm starts with a population of randomly generated initialsolutions called particles (swarm). It is to be noted that the particle structure is taken as a string, which consists of tasks to be performed in rALB probelm considering the precedence constraints. After the swarm is initialized, each particle is assigned with random velcoity and length of the velocity of each particle is genrated randomly. Fitness value of each particle is evaluated and Local best and Global best are found and then the particle velocity and position are updated continuously in all iterations using the formula. Pseudocode of the PSO algorithm is shown below

A. The Pseudo code of PSO

Pseudo code of the PSO algorithm is as follows:

```
   Initialize parameters
   Initialize swarm
   Initialize velocity
Find the fitness values of each particles
Find local best and  global best
      Do{
            {     Update velocity
                  Update position
                  Evaluate particles
                   Update local best
            }
Update global best
            } ( Stopping condtion)
```

Each particle successively adjusts its velocity position towards the global optimum according to the following equations (4) and (5) respectively.

Velocity update equation:

$$v_i^{t+1} = c_1 U_1 v_i^t + c_2 U_2 ({}^{e}P_i^t - P_i^t) + c_3 U_3 (G - P_i^t) \quad (4)$$

Position update equation:

$$P_i^{t+1} = P_i^t + v_i^{t+1} \quad (5)$$

where *U1*, *U2* and *U3* are known as velocity coefficients
c1, *c2* and *c3* are known as learning coefficients
v_i^t is the initial velocity, ${}^{e}P_i^t$ is the Local best, *G* is the global best and P_i^t is the current particle position

Here we show an example of the velocity and position updation.

Let us assume the following:

c1=1 c2=1 c3=2, ${}^{e}P_i^t$= (1,2,6,3,4,5,7,8,10,9,11)
P_i^t= (1,2 ,3,6,5,4,7,8,10,9,11) G=(1,2,3,4,5,6,7,8,9,10,11)

Using (4),
v_i^{t+1}=0.5*(2,3) (4,5) + 0.8*[(1,2,6,3,4,5,7,8,10,9,11) – (1,2 ,3,6,5,4,7,8,10,9,11)]
+0.6*[(1,2,3,4,5,6,7,8,9,10,11)-(1,2 ,3,6,5,4,7,8,10,9,11)]

=0.5*(2,3) (4,5)+0.8*(2,3)(4,5) +0.6*(3,5)(8,9)
= (2,3)(4,5)(8,9)

Using (5),
P_i^{t+1}= (1,2 ,3,6,5,4,7,8,10,9,11)+(2,3)(4,5)(8,9) =
(1, 2, 6, 3, 4, 5, 8, 9, 10, 11)

B. *Initial swarm generation*

The swarm size used in this research is 24. Six particles are generated using the six heuristic rules selected from [6]; maximum rank positional weight, minimum inverse positional weight ,minimum total number of predecessors tasks,maximum total number of follower tasks,maximum and minimum task time. Table II shows set of particles formed using those methods and remaining particles are generated randomly which are satisfying the precedence condition.The precedence graph shown if Fig.1 and processing time shown in Table 1 are used to generate the information provided in Table II.

Table II. Inital swarm genration

Methods	Particle Generated										
Maximum Rank Positional Weight	1	2	6	3	4	5	7	8	10	9	11
Minimum Inverse Positional Weight	1	5	4	3	2	7	9	6	8	10	11
Minimum Total Number Of Predecessors Tasks	1	2	3	4	5	6	8	10	7	9	11
Maximum Total Number of Follower Tasks	1	2	3	4	5	6	7	8	9	10	11
Maximum Task Time	1	5	2	6	3	4	7	8	10	9	11
Minimum Task Time	1	4	3	2	5	7	9	6	8	10	11

C. *Initial velocity*

Inital velocity for the particles are randomly generated and the number of pairs of transpositions used in this research is given in Table III. The same number of pairs are used in all generation in all the test problems.

Table III. Initial Velocity Pairs

No.of ActivitIes	Velocity Pairs
0-20	4
20-40	8
40-60	10
60-80	25
80-100	30
100-120	40
120-140	50
140-200	65
200-300	75

D. *Selecting the values for the parameters*

The performance of the PSO algorithm is generally affected by the parameters used. Extensive experiments and tuning are conducted to find the optimal parameters.Five data sets of different charactersitics were chosen to find the parameters which yielded best solution.Different combinations of the parameters were tested until the best combination is achieved. Five different RALB problems were solved for all the combinations of the parameters for 10 test runs.The quality of solution was given importance com pared to the computational time in selecting the parameters. The range of values for which the parameters are tested are given below.

Following are the parameters used in this algorithm:

- Stopping conditon: 10,15,25,30
- Learning coefficients: c1=1, 2, 3;c2=1, 2, 3 and c3= 1, 2, 3

After conducting experiments with different sizes of data it is found that the best solution could be obtained in 25 generations with c1=1; c2=1; c3=2.

E. *Local Exchange procedure*

The purpose of exchange procedure used is to generate a feasible solution that guarantees towards a better solution. Initially for performing the exchange procedure considers the Gbest obtained from PSO, in which activities are assigned to the stations.

Consider f as the station with maximum cycle time and q as the adjacent station such that $T_f > T_q$. If shifting of activities from station f to q is feasible, the new execution time after shifting is as follows

$$T^*_f = T_f - t_{r,i} \tag{6}$$

$$T^*_q = T_q + t_{r,i} \tag{7}$$

The exchange is worth-while if

$$\max\{T^*_f, T^*_q\} < T_f \tag{8}$$

From the solution obtained from the above mentioned procedure we find out the station h with minimum cycle time (T_h) and the adjacent station g such that $T_h < T_g$. If shifting of activities from g to h is feasible, the new execution time is as follows

$$T^*_g = T_g - t_{r,i} \tag{9}$$

$$T^*_h = T_h + t_{r,i} \tag{10}$$

The exchange is worth-while if

$$\max\{T^*_g, T^*_h\} < T_g \tag{11}$$

These two methods are used in exchange procedure to distribute the activities amongst the stations to get a balanced cycle time between them.

The procedure is as follows:

1. The final Gbest obtained from the PSO is used as the input for the local exchange procedure.
2. Similarly look for a station with highest cycle time and try to shift an activity to the adjacent stations with lower cycle time in such a manner that activity added to the adjacent station does not exceed the cycle time of the station from where we removed.
3. Find the station with the lowest cycle time and try to shift an activity from the adjacent stations to it.
4. Repeat step 2 and step 3 until we get minimum cycle time and also check for the precedence constraints.

An example of exchange procedure is described in Fig.3 where 25 iterations were required to get better solution.

IV. RESULTS AND DISCUSSIONS

Performance Evaluation of the proposed PSO to solve rALB-II problem is coded in C++ and tested on Intel core i5 processor (2.3 GHz).In order to evaluate the performance of the proposed PSO a large set of problems are tested. We collected 7 representative precedence graphs from http://www.assembly-line-balancin.de/ which are widely used in sALB-I literature Scholl [7] .The precedence graphs considered in this paper are with tasks ranging from 25 to 148, totaling 28 test problems. All the 28 test problems are solved using the proposed algorithm. The parameters used in the algorithm are selected through series of tests until a satisfactory solution was obtained in a prescribed time span.

The average cycle time after 10 runs obtained using proposed PSO are presented in Table V. Results in Table V shows that the proposed approach is quite efficient, to find out the best solutions for all 28 data sets considered in this paper.

Task Sequence produced using PSO:1 2 3 4 8 9 5 6 7 11 12 15 17 23 13 14 16 19 20 21 25 10 24 22 18

Step1: Assignment activities to Stations using recursive method

Stations and Tasks	Robot	Cycle Time
S1: 1 2	4	59
S2 : 3 4 8	7	117
S3 :9 5 6	7	94
S4 :7 11 12	7	127
S5:15 17 23	4	114
S6 : 13 14 16 19	7	125
S7 20 21 25	8	105
S8 :10 24	4	63
S9: 22 18	7	80

Step 2: Shifting activities from the station with the highest cycle time

- Shift activity 7 from S4 to S3.

Stations and Tasks	Robot	Cycle Time
S1: 1 2	4	59
S2:3 4 8	7	117
S3:9 5 6 7	7	116
S4:11 12	7	105
S5:15 17 23	4	114
S6:13 14 16 19	7	125
S7:20 21 25	8	105
S8:10 24	4	63
S9:22 18	7	80

Step 3: Shifting activities to stations with lowest cycle time

- Shift activity 3 from S2 to S1

Stations and Tasks	Robot	Cycle Time
S1:1 2 3	4	114
S2:4 8	2	81
S3:9 5 6 7	7	116
S4:11 12	7	105
S5:15 17 23	4	114
S6 : 13 14 16 19	7	125
S7:20 21 25	8	105
S 8 10 24	4	63
S 9 22 18	7	80

Step 4: Repeat steps 2 and 3 for 25 iterations to get the solution

Stations and Tasks	Robot	Cycle Time
S1:1 2 3	4	114
S2:4 8 9	7	107
S3:5 6 7	7	91
S4:11 12	7	105
S5:15 17 23	4	114
S6:13 14 16	7	99
S7:19 20 21	7	100
S8:25 10 24	4	102
S9: 22 18	7	80

Fig.3Exchange Procedure

For the rALB-II problem addressed here, the selection of available robots helps to reduce the cycle time and in turn increases the productivity of the assembly line. Performance of PSO is compared with GA with recursive [4], GA with consecutive [4] and hybrid GA [5] are presented in Table V.

The computational time of the proposed method is tested on 28 representative rALB-II problems. It shows that the algortihm gives optimal solutions for very small-size problems in an acceptable time span. The proposed approach finds out solutions with very less computational load and is efficient and finds satisfactory solutions for large sized problems.In order to demonstrate the performance of proposed method problem with 25 activities and 9 robots is chosen. Its precedence diagram consists of 32 direct precedence relations among 25 tasks is shown in Fig.4. The processing times of 25 tasks by 9 robots are shown in Table IV. The solutions obtained for the problem is given in Fig.5.The results obtained using PSO clearly shows the better performance over other algorithms.

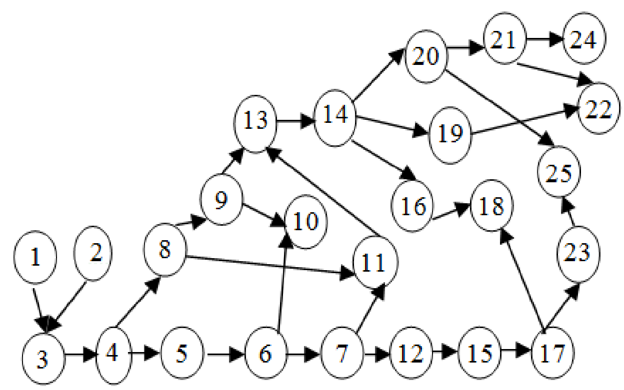

Fig.4 Precedence graph of 25 activity problem

Proposed Method

Task Sequence: 1 2 3 4 8 9 5 6 7 12 15 11 13 17 14 19 16 18 20 21 23 25 24 10 22

Stations	Robot	Activities	Cycle Time
1	4	1 2 3	114
2	2	4 8	81
3	7	9 5 6	94
4	4	7 12 15	105
5	7	11 13 17	114
6	7	14 19 16	103
7	7	18 20 21	111
8	4	23 25	79
9	4	24 10 22	99

Fig.5 Solutions obtained for rALB-II problem 25-9

V. CONCLUSION

The main intent of the work is to develop an efficient solution procedure for rALB using PSO. The solution obtained provides us the best robot assignment for the stations by distributing the activities among them. There is simultaneous improvement in the productivity with reduction in the cycle time. A local exchange procedure is implemented to improve the solution obtained from PSO.

It is observed that proposed PSO performs better than the existing methods reported in the literature.

Table IV. Task times of rALB-II

Act No:	Performance Time								
	R1	***R2***	***R3***	***R4***	***R5***	***R6***	***R7***	***R8***	***R9***
1	75	52	52	33	52	63	84	43	38
2	108	32	34	26	49	105	28	27	33
3	135	55	73	55	58	133	35	64	67
4	89	47	56	69	57	116	40	87	53
5	53	47	48	46	56	84	34	91	50
6	55	62	33	26	43	56	35	51	55
7	30	33	38	23	30	52	22	37	37
8	46	34	77	37	74	47	42	31	28
9	62	54	36	43	57	45	25	39	33
10	52	40	45	37	74	56	41	72	51
11	111	77	65	71	64	81	57	66	100
12	49	34	43	43	58	107	48	60	46
13	87	32	32	45	34	38	22	40	52
14	49	73	46	32	46	49	42	43	69
15	64	90	68	39	47	121	72	61	54
16	85	128	45	74	44	126	35	64	60
17	42	34	31	35	34	40	35	30	28
18	55	47	95	60	56	55	37	75	69
19	56	44	37	51	33	62	26	27	31
20	63	61	58	45	67	126	52	44	75
21	64	38	32	41	30	34	22	33	34
22	93	106	50	36	106	57	43	84	52
23	48	52	45	40	58	49	58	77	59
24	58	47	40	26	81	109	29	75	35
25	42	38	39	39	30	40	39	28	32

REFERENCES

[1] Rubinovitz, J., Bukchin, J., 1991. Design and balancing of robotic assembly lines.In: Proceedings of the Fourth World Conference on Robotics Research, Pittsburgh, PA.

[2] Rubinovitz, J., & Bukchin, J. (1993). RALB-a heuristic algorithm for design and balancing of robotic assembly line. Annals of the CIRP, 42, 497–500.

[3] Levitin, G.,3 Rubinovitz, J., & Shnits, B. (2006). A genetic algorithm for robotic assembly balancing. European Journal of Operational Research, 168, 811–825.

[4] Jie Gao, Linyan Sun , Lihua Wang , Mitsuo Gen (2009).An efficient approach for type II robotic assembly linebalancing problems 1065–1080

[5] Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization, *Proceedings of IEEEInternational Conference on Neural Networks-IV,* pp-1942-1948, Piscataway, NJ: IEEEservice center, Perth, Australia

[6] S. G. Ponnambalam, P. Aravindan and G. Mogileeswar Naidu(2000).A Multi-Objective Genetic Algorithm for Solving Assembly Line Balancing Problem.Int J Adv Manuf Technol (2000) 16:341–352

[7] Scholl,A. (1993).Data of assembly line balancing problems. Schriften zur Quantitativen Betriebswirtschaftslehre 16/93, Th Darmstadt.

[8] Kim, H., & Park, S. (1995). Strong cutting plane algorithm for the robotic assembly line balancing. International Journal of Production Research, 33(8), 2311–2323

Table V. Results of the 28 rALB-II problems

N_a	N_{st}	Cycle time c				CPU time (sec)
		GA+ Recursive [3]	**GA+ Consecutive [3]**	**Hybrid GA [4]**	**Proposed Method**	
25	3	518	503	503	**491**	3.5
	4	351	330	327	**294**	3.9
	6	343	234	213	**208**	4.2
	9	138	125	123	**114**	4.8
35	4	450	551	449	**347**	7.7
	5	385	352	344	**340**	7.9
	7	250	222	222	**219**	7.9
	12	178	120	**113**	115	8.3
53	5	903	565	554	**538**	22
	7	390	342	320	**304**	22.4
	10	-	251	230	**228**	22.7
	14	243	166	162	**153**	22.9
70	7	546	490	449	**448**	46.4
	10	313	287	272	**266**	47.3
	14	231	213	**204**	**204**	47.8
	19	198	167	154	**153**	48.2
89	8	638	505	494	**479**	88.7
	12	455	370	370	**345**	89.4
	16	292	246	236	**234**	92.1
	21	277	209	205	**201**	93.2
111	9	695	586	557	**551**	167.2
	13	401	339	319	**316**	167.6
	17	322	**257**	**257**	**257**	168.2
	22	265	209	192	**190**	171.2
148	10	708	638	600	**593**	381.1
	14	537	441	427	**426**	385.5
	21	404	325	300	**299**	390.4
	29	249	210	202	**200**	390.9