Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Application of Simple Genetic Algorithm to U-Shaped Assembly Line Balancing Problem of Type II

Venkatesh, Jonnalagedda*. Balaji, Dabade**

** Shri Guru Gobind Singhji Institute of Engineering & Technology, Nanded (MS), India (Tel: +919657720072; e-mail: meghavenkatesh@gmail.com).*
*** Shri Guru Gobind Singhji Institute of Engineering & Technology, Nanded (MS), India (e-mail:bmdabade@gmail.com)*

Abstract: Process of assigning tasks to workstations arranged along a U-shaped assembly line is known in literature as U Assembly Line Balancing Problem (UALBP). Maximizing the line efficiency by way of minimizing number of workstations (m) for a given cycle time (c) leads to UALBP-I while targeting the same objective through minimization of 'c' for given 'm' is known as UALBP-II. Although, quite a good amount of research has been reported on UALBP-I since the first published work on U lines in 1994; very little work has been reported on type II. Type E (*minimizing 'c' and 'm' together*) and type F (*finding feasible line balance for given 'c' and 'm'*) problems which represent other two types of U line balancing problems also have not received any attention. This paper reports the initial efforts towards application of metaheuristics for solving UALBP-II problem which is encountered when the line already exists.

Keywords: Genetic Algorithms, Assembly Line Balancing, U-shaped Lines,

1. INTRODUCTION

The purpose of solving an assembly line balancing problem (ALBP) is to assign a set of tasks (inter-related through precedence relationships) to a set of workstations arranged along some material handling system. The workstations were traditionally arranged in a straight line may be along a moving conveyor. Owing to the success of JIT, U shaped assembly lines became more and more popular. Assembly line balancing problem in its simplest form is itself a computationally NP hard problem. Generalizing it even for a single additional aspect such as consideration of assignment restrictions, assembling multiple models on single line, changing the line layout, etc further complicates the problem. U-shaped Assembly Line Balancing Problem (UALBP) is one such complex problem which has aroused a lot of interest among researchers. In simple ALBP, a task can be assigned to a workstation only after all its predecessors have already been assigned. Whereas, a task becomes assignable on a U shaped line either when all its predecessors or when all its successors have already been assigned. Solutions for U line layout can be expected to provide better line efficiency for the same values of number of workstations (or cycle time) as against serial lines because there are more options of assignable tasks for any particular workstation on U lines than serial lines.

1.1 Literature Review

UALBP research started only about two decades ago after its introduction by Miltenburg and Wijngaard (1994). However, major focus of UALBP research has been on type I wherein cycle time (c) is given and number of stations (m) is to be minimized (for example: Miltenburg and Wijngaard, 1994; Urban, 1998; Scholl and Klein, 1999; Ajenblit and Wainwright, 1998; Erel et al., 2001, etc). Studies on certain variants of UALBP-I have also been reported. For example, multiple U lines have been considered by Miltenburg (1998) and Sparling (1998); Sparling and Miltenburg (1998) reported a study of mixed model U line balancing problem; Miltenburg (2000) analyzed the effect of breakdowns on efficiency of U lines and reports that U lines are better when buffers are arranged at all contact points between stations; Urban and Chiang (2002) dealt with stochastic task times; Mustafa et al. (2010) considered a two sided U-shaped assembly line balancing problem. Jaydeep et al. (2009) examine and modify 13 single pass heuristics for solving UALBP-I. Fathi et al. (2011) propose a new heuristic based on 'Ranked Positional Weight (RPW) method' of solving assembly line balancing problems and 'Critical Path Method (CPM)' of solving project management problems for balancing of U shaped lines for given 'c'. Metaheuristcs have also been proposed for solving UABLP-I by various researchers such as Genetic Algorithm by Ajenblit and Wainwright, (1998), Simulated Annealing by Erel et al. (2001), Ant Colony Optimization by Nuchsara et al. (2007), etc. Nourmohammadi et al. (2010) propose the use of another metaheuristic namely Imperialist Competitive Algorithm but for a multi objective UALBP. However, even in their work cycle time is considered as given and number of workstations is to be minimized along with minimization of workload variations.

Surprisingly, not much literature can be found on UALBP-II. Nakade et al. (1997) reported their work on stochastic UALBP-II. They developed bounds and approximations for cycle time. Scholl and Klein (1999) developed ULINO procedure based on their exact procedure SALOME-1 for

simple assembly line balancing problem of type I (SALBP-I) and applied to UALBP-II also. To best of the knowledge of authors of this paper, no other work has been reported on UALBP-II besides the above referred two works. UALBP-II instances arise out of situations wherein the assembly lines are already installed. The number of workstations is fixed and one needs to balance the line for minimization of cycle time. More particularly when the design or demand changes for the product, reconfiguration or rebalancing calls for solving an UALBP-II. No application of heuristics or metaheuristics could also be found by the authors of this paper for solving the said problem. Complexity of UALBP-II makes obtaining optimal solution in polynomial time very unlikely and hence heuristics or metaheuristics are obvious options. As per Erel et al. (2001), it would be interesting to use metaheuristics for solving UALBP.

1.2 Scope of this Work

This paper reports the initial efforts made towards solving UALBP-II using Genetic Algorithm (GA). The authors consider simple UALBP-II as defined by the following assumptions:

i Only one product is assembled on the line.

ii Task times are deterministic.

iii There is a single U-shaped assembly line and operators can perform their work only on one side of the line.

iv The line is a paced one.

v Each operator is associated with a single workstation only.

vi Task assignment to workstations must follow precedence relations as modified to suit a U-shaped line.

vii Number of workstations is known and cycle time is to be minimized.

Besides, a simple GA is used to solve the problem instances as the objective of the work reported here is to identify the suitability of GA for solving UALBP-II. The idea here is to model the problem in such a way that relevant domain information of U lines is passed on to the GA which then attempts to find the best possible solution.

2. MODELLING UALBP-II FOR APPLICATION OF GA

A sample test problem (Jackson's 11 task problem) is represented by its precedence diagram in figure 1. A possible solution to UALBP-II for 4 workstations is shown in terms of the layout in figure 2 wherein workstation numbers have been written in the middle of dashed boxes while tasks assigned to particular workstation have been written over and below the workstation number to denote when a particular task shall be performed. For example, task 1 and 2 shall be performed at the beginning and task 11 shall be done at the end on workstation number 1 (WS#1). As can be observed, tasks at the start and end of a precedence diagram can be assigned to WS#1. In general, a task becomes assignable to a workstation for U-shaped line, when either all its predecessors or all its successors have been assigned to that or earlier workstation.

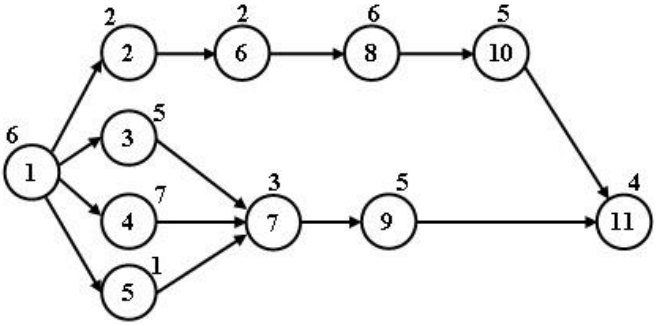

Figure 1: Precedence Diagram of sample test problem

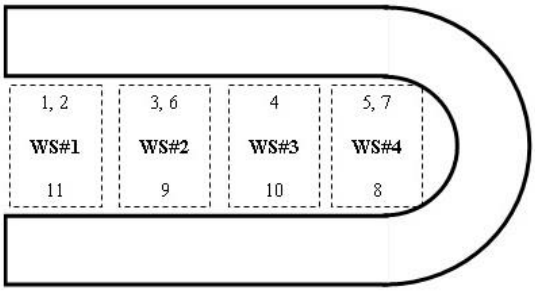

Figure 2: A possible solution for Jackson's test problem

2.1 Representing UALBP-II solution as a chromosome

Genetic Algorithm requires the solution to any problem to be coded in the form of a chromosome. For modelling UALBP-II, we use "standard coding" in which there shall be as many genes in a chromosome as the number of tasks and each gene in the chromosome shall take as its allele the workstation number to which a particular task (represented by the gene's location) is assigned. Thus a solution to UALBP-II instance is a set of workstation numbers to which the tasks have been assigned. Figure 3 represents the solution of figure 2 in terms of a chromosome.

WS	**1**	**1**	**2**	**3**	**4**	**2**	**4**	**4**	**2**	**3**	**1**
Task	1	2	3	4	5	6	7	8	9	10	11

Figure 3: Representation of a solution as Chromosome

Genetic Algorithm starts with a population of randomly generated chromosomes (solutions) and with the help of some selection criteria and various genetic operators like crossover, mutation, etc iterates towards optimal solution from generation to generation. Thus GA can be called as a directed random search approach. So, what we need now is a set of pre-decided number of chromosomes that are randomly generated. Generation of such chromosomes requires random generation of workstation numbers for each task which means random generation of an allele for each location of the genes in the chromosome. Researchers can be found quoting that a feasible solution for SALBP is also feasible for UALBP. As an extreme, an exact reverse order of a feasible SALBP solution can also be feasible for UALBP. That is, a sample SALBP solution namely {1, 1, 2, 2, 1, 1, 3, 3, 3, 4, 4} and exact reverse order solution {4, 4, 3, 3, 4, 4, 2, 2, 2, 1, 1} are feasible UALBP-II solutions for the test problem of figure 1. The true benefits of a U-shaped line such as employee rotation, team work, etc cannot be obtained through such

solutions but still they represent feasible solutions in mathematical terms. Permitting such extreme feasibility means that the first task (on which all subsequent tasks might possibly depend) can be assigned to any workstation between 1 through 'm'. Similarly many other tasks (including the last one in the precedence diagram) can also be assigned to any workstation between 1 through 'm'. One approach for generating workstation numbers can, therefore, be to generate any number between 1 and the maximum number (m) of workstations for each task. There is a major problem in such representation of a solution besides loosing the real benefits of U-shaped lines that the possible number of solutions that could be generated will be very high and most of them would be infeasible.

In this work, an approach: on the lines of finding earliest and latest workstation for task assignment; is proposed for determining the workstation numbers to which the tasks can be assigned to maintain the assembly line's shape as close to an 'U' as possible. The algorithmic representation of the proposed approach is as follows:

1. Determine the earliest and latest workstation numbers to which each task can be assigned in the forward direction using:

$$E_{if} = \left\lceil \frac{t_i + \sum_{j \in P_i} t_j}{c_{th}} \right\rceil \quad \text{and} \quad L_{if} = m - \left\lfloor \frac{t_i + \sum_{j \in S_i} t_j}{c_{th}} \right\rfloor \qquad (1)$$

where

t_i - task time of the i^{th} task,

P_i - set of all predecessors of i^{th} task,

S_i - set of all successors of i^{th} task,

c_{th} - theoretical minimum cycle time = $\left\lceil \sum_{\forall i} t_i \Big/ m \right\rceil$

2. Determine the earliest and latest workstation numbers to which each task can be assigned in the backward direction using:

$$E_{ib} = \left\lceil \frac{t_i + \sum_{j \in S_i} t_j}{c_{th}} \right\rceil \quad \text{and} \quad L_{ib} = m - \left\lfloor \frac{t_i + \sum_{j \in P_i} t_j}{c_{th}} \right\rfloor \qquad (2)$$

3. Set E_i (earliest workstation for task 'i') equal to the smallest of E_{if} and E_{ib} if the absolute difference (AD_i) between the work content of all its predecessors taken together and the work content of all its successors taken together is greater than the total work content (TWC) of the assembly multiplied by a factor (f_d). Otherwise, set E_i equal to the largest of E_{if} and E_{ib}.

4. Set L_i (latest workstation for task 'i') equal to the smallest of L_{if} and L_{ib} if AD_i is greater than TWC multiplied by f_d. Otherwise, set L_i equal to the largest of L_{if} and L_{ib}.

5. If E_i and L_i become equal for some tasks then either increase L_i by 1 or decrease E_i by 1.

Step 3 and 4 help in assigning smaller values of earliest and latest workstations to the task which fall at either ends of the precedence diagram while they help push the middle tasks (in terms of preceding work content and succeeding work content) to larger workstation numbers. Such a criterion helps in assigning the tasks strictly as required for a U-shaped line although random generation of workstation number as a value between E_i and F_i leads to some infeasibilities. The factor 'f_d' is used to vary the amount of difference between preceding and succeeding work content permitted for such demarcation of tasks as obtained by steps 3 and 4. A value of 0.2 was assigned to f_d for the work reported here. For some tasks it has been observed that the values of E_i and L_i become equal due to steps 3 and 4. For example, the earliest and latest workstations for tasks 1 and 11 become equal to 1 for the example problem of figure 1. This puts additional restrictions on task assignment in the sense that it forces the tasks to be strictly adhering to U-shape. Step 5 relaxes this restriction by permitting some flexibility to task assignment.

2.2 Dealing with infeasibilities

Infeasibilities arising out of random generation of workstation number for the tasks in the sense that some of the tasks are assigned to a workstation up to which at least one of the predecessors and at least one of the successors of the task under consideration are not assigned. Such infeasibilities are removed by simply shifting the task to a workstation (say p) up to which either all the predecessors or to a workstation (say q) up to which all the successors have been assigned. Further, it has been imposed that an infeasible task whose AD_i is greater than (TWC×f_d) is shifted to an earlier workstation out of p and q and an infeasible task whose AD_i is less than or equal to (TWC×f_d) is shifted to a later workstation out of p and q.

3. SIMPLE GENETIC ALGORITHM

The flowchart (figure 4) depicts the simple genetic algorithm used in this work. Following is the parameter setting for the GA:

Population Size	100
Tournament Size	4
Crossover probability	0.50
Mutation probability	0.02
Stopping Criterion	50 generations

The simple GA used in this work uses tournament selection with a tournament size of 4. Best chromosome among the four is transferred to the intermediate generation. Selection contributes to 50% of the chromosomes in the intermediate generation while remaining 50% of the chromosomes are contributed by crossover. Two point crossover is used for this work wherein the portion between two crossover sites of the two chromosomes involved in crossover operation is swapped. Mutation replaces the value of an allele by a number of workstation generated randomly between the E_i

and L_i values for the corresponding task. It is quite possible that some of the chromosomes obtained after crossover and mutation represent infeasible solutions. The same logic as given in section 2.2 is used for removing such infeasibilities. The intermediate generation is taken as the next generation for further processing. It has been observed for most of the test problems that termination criterion of 50 generations is quite good.

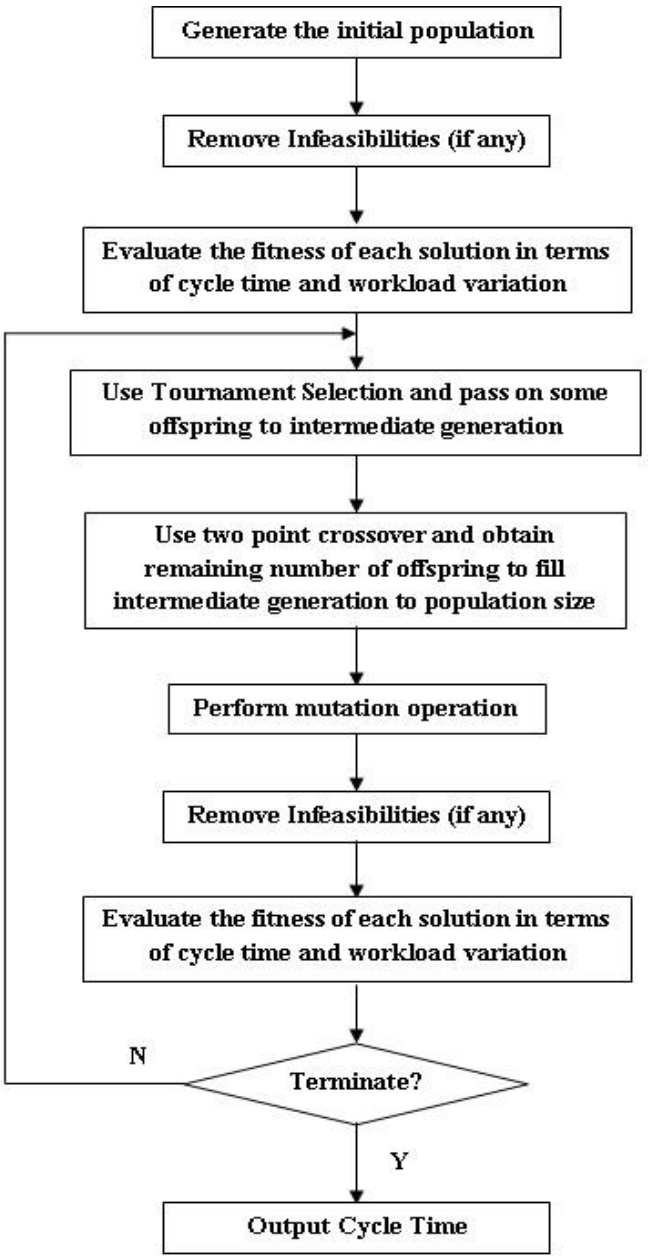

Figure 4: Simple Genetic Algorithm used for UALBP-II

4. AN ILLUSTRATION

The example problem given in figure 1 is taken here for illustrating the application of proposed approach of finding the earliest and latest workstation for each task. Total work content (TWC) for the 11 task problem is equal to 46 and the number of workstations (m) is taken as 4. The theoretical minimum cycle time is equal to 12 i. e. the smallest integer larger than the ratio of TWC and m. Next the values of E_{if}, L_{if}, E_{ib} and L_{ib} are computed using equations (1) and (2). Subsequently, E_i and L_i values are obtained following steps 3, 4 and 5. The computed values are tabulated below in table 2 after sample calculations for task 1 and task 5 (table 1).

Table 1: Sample Calculations for finding earliest and latest workstation

Task 1	Task 5
$E_{1f} = \left\lceil \frac{6+0}{12} \right\rceil = 1;$	$E_{5f} = \left\lceil \frac{1+6}{12} \right\rceil = 1;$
$L_{1f} = 4 - \left\lfloor \frac{6+40}{12} \right\rfloor = 1;$	$L_{5f} = 4 - \left\lfloor \frac{1+12}{12} \right\rfloor = 3;$
$E_{1b} = \left\lceil \frac{6+40}{12} \right\rceil = 4;$	$E_{5b} = \left\lceil \frac{1+12}{12} \right\rceil = 2;$
$L_{1f} = 4 - \left\lfloor \frac{6+0}{12} \right\rfloor = 4;$	$L_{5f} = 4 - \left\lfloor \frac{1+6}{12} \right\rfloor = 4;$
$AD_1 = \lvert 0 - 40 \rvert = 40;$	$AD_5 = \lvert 6 - 12 \rvert = 6;$
$TWC \times f_d = 46 \times 0.2 = 9.2$	$TWC \times f_d = 46 \times 0.2 = 9.2$
$AD_1 > TWC \times f_d$ so	$AD_1 < TWC \times f_d$ so
$E_1 = \min(E_{1f}, E_{1b}) = 1;$	$E_5 = \max(E_{5f}, E_{5b}) = 2;$
$L_1 = \min(E_{1f}, E_{1b}) = 1;$	$L_5 = \max(E_{5f}, E_{5b}) = 4;$
As $E_1 = L_1$ and as $L_1 < 4$, L_1 is incremented to 2.	As $E_1 \neq L_1$, no change in their values.

Note that although the search space gets enlarged due to the relaxation provided by step 5, it actually helps in getting more options for task combinations for assignment to each station. Therefore, strict allocation of some tasks to particular workstations is avoided leading to better balancing solutions. Tasks 1 and 11 in table 2 represent such examples.

Table 2: Earliest and Latest Workstation for all tasks

Task	E_{if}	L_{if}	E_{ib}	L_{ib}	E_i	L_i
1	1	1	4	4	1	2
2	1	3	2	4	1	3
3	1	3	2	4	2	4
4	2	3	2	3	2	3
5	1	3	2	4	2	4
6	1	3	2	4	2	4
7	2	3	1	3	1	3
8	2	3	2	3	2	3
9	3	4	1	2	1	2
10	2	3	1	3	1	3
11	4	4	1	1	1	2

5. RESULTS AND DISCUSSION

The approach proposed in this work has been applied to a number of test problems found in earlier literature. A total of 25 test problems have been considered by varying the number of workstations for the test problems of Jackson, Mitchell, Heskia, Sawyer, Kilbridge and Tonge. Test problems with number of tasks less than or equal to 70 have been selected for testing the proposed approach considering the fact that U-shaped assembly lines are generally implemented wherever the number of tasks are fewer. Table 3 provides the results obtained in terms of the cycle time (c) and the corresponding

line efficiency (η) achieved. Line efficiency is the ratio of total work content (TWC) to the product of number of workstations 'm' (given) and cycle time (c) obtained using the simple Genetic Algorithm. It can be observed that GA provides quite good results for most of the test problems. As mentioned earlier, this paper reports initial efforts towards application of Genetic Algorithm for solving UALBP-II and hence it was not attempted to check results of the proposed approach for optimality. However, the approach can be seen to have provided optimal solution to a few test problems such as Jackson's 11 task problems with 4 and 3 workstations, Mitchell's 21 task problem with 6 and 5 workstations, etc while it provided near optimal solution for others such as Heskia's 28 task problem with 5 stations, Kilbridge's 45 task problem with 3 stations, Tonge's 70 task problem with 7 stations, etc.

Table 3: Results for various test problems

Problem	Tasks	TWC	M	c	η (%)
Jackson	11	46	4	12	95.83
			3	16	95.83
Mitchell	21	105	8	15	87.50
			7	16	93.75
			6	18	97.22
			5	21	100.00
Heskia	28	1024	8	144	88.89
			7	159	92.00
			6	177	96.42
			5	208	98.46
Sawyer	30	324	11	35	84.16
			9	42	85.71
			7	49	94.46
			5	68	95.29
Kilbridge	45	552	10	63	87.62
			8	75	92.00
			6	96	95.83
			4	140	98.57
			3	185	99.46
Tonge	70	3510	10	382	91.88
			9	419	93.08
			8	457	96.00
			7	521	96.24
			6	607	96.38
			5	724	96.96

The proposed approach has been found to perform exceedingly well for lower values of number of workstations 'm'. Invariably it has been observed that as 'm' is reduced for a test problem, the efficiency obtained by the proposed approach steadily increases. Possible reasons for the inadequate performance for higher number of workstations could be insufficient inclusion of domain knowledge into the definition of UALBP-II solution, vagueness of the factor "f_d" and / or the setting up of parameters for the GA used for this work. Efforts made to improve the definition of UALBP-II solution for its representation as a chromosome may help in circumventing the requirement of "f_d".

Nonetheless, further efforts in the direction of applying GA in particular and metaheuristics in general are definitely warranted for obtaining better solutions to UALBP-II. This is further emphasized by the convergence obtained by the simple GA for all the test problems considered. Figures 5 and 6, for example, exhibit how the GA performed in converging to the solution over generations for sample test problems namely Kilbridge's 45 task problem with 10 workstations and Tonge's 70 task problem with 7 workstations.

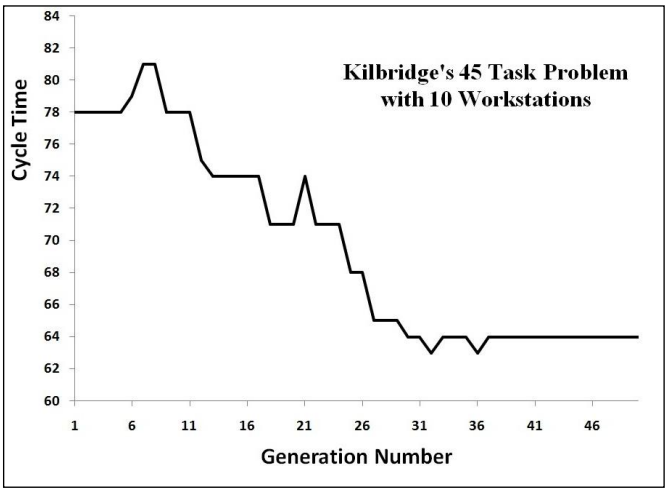

Figure 5: GAs output vis-a-vis generations for Kilbridge's test Problem

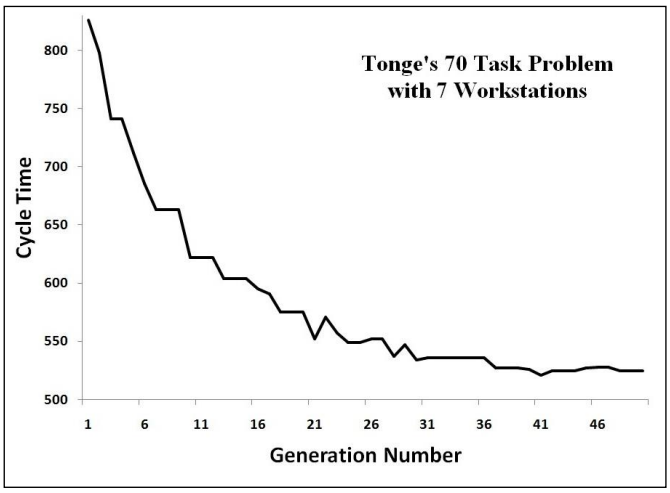

Figure 6: GAs output vis-a-vis generations for Tonge's test Problem

REFERENCES

Ajenblit, D.A., Wainwright, R.L., 1998. Applying genetic algorithms to the U-shaped assembly line balancing problem, Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, 96-101.

Erel, E., Sabuncuoglu, I., Aksu, B.A., 2001. Balancing of U-type assembly systems using simulated annealing, International Journal of Production Research 39, 3003-3015.

Jaydeep Balakrishnan, Chun-Hung Cheng, Kin-Chuen Ho, Kum Khiong Yang, 2009, The application of single pass heuristics for U lines, Journal of Manufacturing Systems, 28, 28-40.

Miltenburg, J., 1998. Balancing U-lines in a multiple U-line facility, European Journal of Operational Research 109, 1-23.

Miltenburg, J., 2000. The effect of breakdowns on U-shaped production lines, International Journal of Production Research 38, 353-364.

Miltenburg, J., Wijngaard, J., 1994. The U-line line balancing problem, Management Science 40, 1378-1388.

Mustafa Fatih Yegul, Kursad Agpak, Mustafa Yavuz, 2010, A new algorithm for U-shaped two-sided assembly line, Transactions of the Canadian Society for Mechanical Engineering, Vol. 34, No. 2, 225-241.

M. Fathi, M. J. Alvarez, V. Rodriguez, 2011, A New Heuristic Approach to Solving U-shape Assembly Line Balancing Problems Type-, World Academy of Science, Engineering and Technology, 59, pp. 413-421.

Nakade, K., Ohno, K., Shanthikumar, J.G., 1997. Bounds and approximations for cycle times of a U-shaped production line, Operations Research Letters 21, 191-200.

Nourmohammadi, A., Zandieh, M., Tavakkoli-Moghaddam, R., 2010, An Imperialist Competitive Algorithm for multi-objective U-type assembly line design, *Journal of Computational Science* (2010), doi:10.1016/j.jocs.2012.09.001.

Nuchsara Kriengkorakot, Nalin Pianthong and Rapeepan Pitakaso, 2007, The ANT algorithm for U-shaped assembly line balancing, ICEE, 89-93.

Scholl, A., Klein, R., 1999. ULINO: Optimally balancing U-shaped JIT assembly lines, International Journal of Production Research 37, 721-736.

Sparling, D., 1998. Balancing JIT production units: The N U-line balancing problem, Information Systems and Operational Research 36, 215-237.

Sparling, D., Miltenburg, J., 1998. Mixed-model U-line balancing, International Journal of Production Research 36, 485-501.

Urban, T.L., 1998. Note. Optimal balancing of U-shaped assembly lines, Management Science 44, 738-741.

ACKNOWLEDGEMENT

This paper is outcome of a part of the research work supported by All India Council for Technical Education (AICTE) under its Research Promotion Scheme (*F. No. 8023/ BOR/RID/RPS-82/2009-10 dated 31/3/10*).