U-shaped assembly line balancing with Grouping Evolution Strategy (GES)

Pouria Pourmomen Davani

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
February 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işik Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Asst. Prof. Dr. Gürcü Öz
Supervisor

Examining Committee

1. Asst. Prof. Dr. Sahand Daneshvar
2. Asst. Prof. Dr. Gürcü Öz
3. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

In this research, we have applied Grouping Evolution Strategies (GES) as an alternative solution to U-shaped assembly line balancing problem (UALBP). By introduction of just-in-time (JIT) production principle, it can be proven that U-shaped assembly line system has better performance than its predecessor traditional straight line system. The parameters to compare are the number of workstations (the line efficiency) and the smoothness index of workload. Our evaluation shows by applying GES, at least same line efficiency of workstation integration can be achieved. Moreover, the variation and smoothness index of workload have been improved.

Moreover, to measure the performance validation of the proposed algorithm, a number of standard UALBPs in the literature were used to compare the proposed algorithm results with other related work results.

Simulation results show that the proposed model produced as good or even better line efficiency of workstation integration and improved the variation and smoothness index of workload.

Keywords: U-shaped assembly line balancing, Grouping Evolution Strategies, Workstation, Smoothness Index

ÖZ

Bu araştırmada U-sekilli montaj hattı dengeleme problemine (U-shaped assembly line balancing problem - UALBP) alternatif bir çözüm olarak evrim stratejilerini gruplama sistemini (Grouping Evolution Strategies -GES) uyguladık. Zamanında üretim prensibi kuramına göre (just-in-time JIT) U-şekilli montaj hattı sisteminin performansı geleneksel düz çizgi sisteminden daha iyi olduğ ispatlanmıştır. Performans ölçümünde karşılaştırılan parameteler ise, iş istasyonlarının sayısı ve iş yükü pürüzsüzlük indeksi (hat verimliliği)'dir. Değerlendirmelerimiz göstermiştir ki, belirtilen probleme GES uygulanarak istenilen iş iştasyonu entegrasyon hat verimliliği elde edilmiştir. Buna ek olarak iş yükü pürüzsüzlük indeksi de geliştirilmiştir.

Ayrıca önerilen algoritmanın performans doğrulamasını ölçmek için, literatürde bulunan bir dizi standart UALBP'ler kullanılarak önerilen algoritma sonuçları diğer ilgili çalışmalarla karşılaştırılmıştır.

Simulasyon sonuçları göstermiştir ki, önerilen model, iş iştasyonu entegrasyon hat verimliliğin ölçümünde daha iyi sonuç vermiş ve iş yükü değişimi ve pürüzsüzlük indeksini de iyileştirmiştir.

Anahtar Kelimeler: U-sekilli montaj hattı dengeleme (UALB), gruplama evrim stratejileri (GES), iş istasyonu (workstation), pürüzsüzlük indeksi (smoothness index)

DEDICATION

This thesis is dedicated to the memory of my father. I miss him every day, but I am glad he saw this process through to its completion, offering the support to make it possible, as well as plenty of friendly encouragement. I dedicate this thesis to my mother, brothers and beloved wife who are always with me in difficult situations and I cannot describe how thankful I am for all the support that I got from them.

ACKNOWLEDGMENT

First and above all, I praise God, the almighty for providing me this opportunity and granting me the capability to proceed successfully. This thesis appears in its current form due to the assistance and guidance of several people. I would therefore like to offer my sincere thanks to all of them.

Asst. Prof. Dr. Gürcü Öz, my esteemed promoter, my cordial thanks for accepting me as a Master student, your warm encouragement, thoughtful guidance, critical comments, and correction of the thesis.

My parents deserve special mention for their inseparable support and prayers. My mother Sorour Tehrani, in the first place is the person who put the fundament my learning character, showing me the joy of intellectual pursuit ever since I was a child and my dear brothers who always support me to continue my way and goals.

Words fail me to express my appreciation to my wife Mahnaz whose dedication, love and persistent confidence in me, has taken the load off my shoulder. I owe her for being unselfishly let her intelligence, passions, and ambitions collide with mine. Therefore, I would also thank Sabetghadam's family for letting me take her hand in marriage, and accepting me as a member of the family, warmly. I would like to thank everybody who was important to the successful realization of thesis, as well as expressing my apology that I could not mention personally one by one.

I will forever be thankful to my dear friend Mazyar, who assist me in this thesis. He support me and encourage me to achieve this goal.

My special thanks to my relatives Fazel family in Cyprus, who communicate with them, provided emotional atmosphere for me. Hereby, I would like to thanks them for everything.

I am grateful to Farbod for helping me in this process. I really appreciate to his support.

To all my friends, Hossein, Shahin, Kave and AmirHamzeh thank you for your understanding and encouragement in my many, many moments of crisis. Your friendship makes my life a wonderful experience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ALB	Assembly Line Balancing
ALBP	Assembly Line Balancing Problem
GA	Genetic Algorithms
GES	Grouping Evolution Strategies
JIT	Just In Time
LE	Line Efficiency
RPW	Ranked Positional Weight
SA	Simulated Annealing
SI	Smoothness Index
UALBP	U-shaped Assembly Line Balancing Problem
V	Variation
WS	Workstation

Chapter 1

INTRODUCTION

Assemble line is defined as an arrangement of some workstations where parts of a specific product get assembled. The task of changing the arrangement of workstations in a way that optimum performance/throughput (upon some specific criteria) is gained called: Assembly line balancing (ALB) [1]. Usually one of the objectives is to reduce the number of workstations as much as possible for a given cycle time.

In some cases, traditional straight assembly lines (which are serial arrangement of workstation in a line) have shown some inefficiency in line inflexibility, job monotony and large inventories. By invention of Just-in-Time (JIT), U-shaped assembly lines have become popular.

The layouts of straight assembly line and it corresponding U-shaped line shown in Figure 1.1 and Figure 1.2 with eight tasks numbered from one to eight and three workstation (operating personnel). In U-shaped line, the operating personnel stand as workstation in the U. The entry and exiting points are set to the end of U. This layout will let the operating personnel to work on both front sides for a cycle [2].

The workstation that can work on two parts, one on each side of the line in a cycle called a crossover station, as you see the station at the left end of Figure 1.2 that can works on tasks one and eight in a cycle is a crossover station. With increase in number

of crossover stations, you will have more flexible task-workstation combination. As it seen in Figure 1.2, task eight assigned to the left-end workstation, which is not possible in serial lines. In consequence, we can have a better balance while using less number of workstation and operating personnel. Other advantages are having better sight of production line and increasing in personnel dialog. In [3] it shown this leads the ability of rebalancing in fast changing demand/operating environment. Productivity improvement, reduction in work-in-process inventory, space requirement and lead-time are the other benefits of U-shaped Assembly Line [4, 5, 6, 7]

Figure 1.1: Straight assembly line

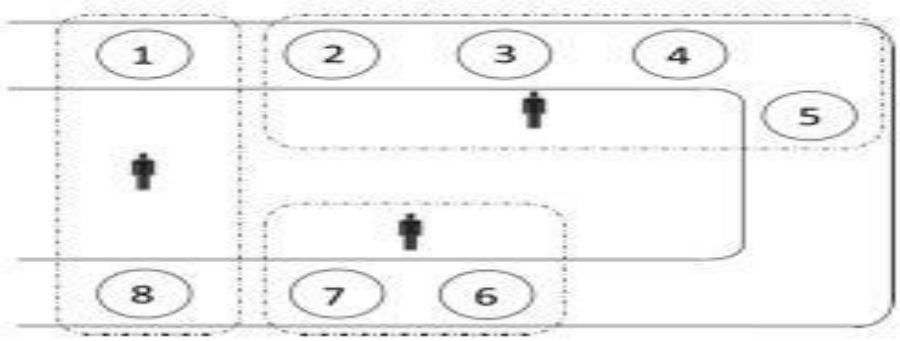

Figure 1.2: U-Shaped assembly line

Traditional straight lines are being replaced by U-shaped assembly lines to adoption of Just-in-Time philosophies in industry. In the U-shaped assembly line balancing problem (UALBP), a task can be assigned to a station after all of its predecessors or successors have been assigned to stations.

By doing adjusting modification in Grouping Evolution Strategies (GES), we used GES as an alternative solution for UALBP. Our modified strategy is applicable in single model, deterministic UALBP. The target here is to model the system with the smallest set of workstations. Genetic Algorithm (GA) [8] and Simulated Annealing (SA) [9] can be other alternatives solution. We will show our approach based on GES (which is the most recent meta-heuristic algorithm) which proposed by Kashan [10] can provide at least similar results to other solutions in a comparison.

Our aim is to improve the manufacturing throughput and reducing the required number of human resources by optimizing task assignment and resource allocation in U-shaped assembly line by using meta-heuristic method (GES).

In Chapter 2, we review the related works. Chapter 3 contains the main part of our contribution i.e. our proposal for solving UALBP. Chapter 4, we discuss the outputs and finally Chapter 5 we conclude our work and propose future topics.

Chapter 2

LITERATURE REVIEW

This section includes two parts: (1) the basic studies on grouping evolution strategies (GES) and (2) the relevant studies on the U-line balancing problem.

2.1 Grouping Evolution Strategy (GES)

By definition, grouping problems are generally concerned with partitioning a set of *V* of *n* objects into a collection of mutually disjoint subsets (groups) *Vi*, such that: $V = \cup_{i=1}^{D} V_i \; and \; V_i \cap V_j = \emptyset, i \neq j$ on the other hand, the aim in these problems is to partition the members of set *V* into *D* $(1 \leq D \leq n)$ different groups where each object is exactly in one group [11]. Normally, in grouping problems implicit that the ordering of groups is not relevant.

Some well-known grouping problems are graph coloring problem, bin packing problem, various packing/partitioning problems, timetabling problem, identical/non-identical parallel-machines scheduling problem, cell formation problem, pickup and delivery problem are some other famous grouping problems.

Grouping problems usually contain a constraint set that must be held under possible object-assignments. Hence, not all assignments are acceptable. Grouping problems are featured by an objective function upon different combination of groups. Moreover, by using evolutionary algorithms a group or a group segment is the fundamental block

that must be kept in course of search. Based on this fact scientists have used evolutionary algorithms to improve the grouping problems [11, 12, 13].

Grouping evolution strategy (GES) [10] is a kind of evolutionary algorithm that recently proposed for crisp grouping problems. It is totally compatible with the Evolution Strategy (ES) introduced by Rechenberg [14] with this distinction that ES uses Gaussian mutation during optimization whereas GES a novel comparable mutation. Further details are available in [15, 10].

2.2 U-type Assembly Line Balancing

U-line is a relatively new and promising topic in the assembly line balancing literature. The first study is due to Miltenburg and Wijngaard (1994) [16] who proposed a dynamic programming formulation to solve 21 relatively small problems (with up to 11 tasks). The authors also develop a heuristic procedure based on the maximum ranked positional weight (RPW) for large size problems. Later, Miltenburg and Sparling (1995) [17] developed three exact algorithms for the UALBP: a reaching dynamic programming algorithm, breadth- and depth-first branch-and-bound algorithms.

To handle larger problems, Scholl and Klein (1999) [18] propose ULINO (U-line optimizer); a new branch-and-bound procedure that performs a depth-first search by considering bounds and some dominance rules to solve different versions (Type-I, Type-II and Type-E) of the ULB problem. Erel et al. (2001) [9] developed an SA-based algorithm for UALBP. The proposed algorithm employs an intelligent mechanism to search the large solution space effectively. Gokcen et al. [19] proposed a shortest route formulation of ULB. Gokcen and Agpak [20], and Toklu and Ozcan

[21] developed Goal Programming formulations of ULB. Jayaswal and Agarwal [22] used Resource Dependent Task Times to solve ULB.

2.3 Proposed Method

In this thesis, regards to study of Hwang et al. [8] and Erel Et al. [9] which they used meta-heuristic methods to solve U-Shaped assembly line balancing problem we try to use grouping evolution strategy(GES) which is proposed by Kashan [10] to solve UALBP. Furthermore, GES used in the fuzzy clustering by Kashan et al. [23].

To find out enhanced result, we should use a proper technique to create our initial solution, to reach this point we used revised-RPW. To prove that the outcome of revised-RPW is more efficient than RPW, we use assumption of M. Fathi's study [24] such as their precedence diagram, and given cycle time and modified version of cycle time that they used to describe their method.

Chapter 3

RESEARCH METHOD AND PROPOSED ALGORITHM

In order to achieve the goals of this study, the hybrid algorithm including an exact algorithm to find an initial solution and a grouping meta-heuristic algorithm to improve the solution has developed. Then the proposed algorithm has coded with software MATLAB 2013a and then with use of the standard problems considered in Hwang et al. study [8] the quality of the proposed method has measured. In continue we intend to provide details of the proposed method for solving UALBP type-1.

3.1 Problem Statement

The simple case of a UALBP is one of the most discussed issues in combinational optimization. In this problem precedence graph of activities are given that activity *j* has T_j processing time unit. The objective is assigning the activities to stations considering prerequisite activities with fixed cycle time. In such a route, the point was that number of Workstations to be minimized. The proposed algorithm in this research is a two-state algorithm as follows:

1) Creating an initial solution

2) Improving initial solution to achieve to final solution (by using GES algorithm)

3.2 Generating an Initial Solution

There are several methods for determining the initial solution of the UALBP that each of them have their strengths and weaknesses. These methods are included of all accurate and heuristic methods that any of them can be considered as the initial

solution algorithm. By search in journals and literature, we found that COMSOAL method [25] is one of the well-known methods in this field.

In this method, for assigning the tasks, we start from the first node (activity) of the precedence graph and assigns the activities to the workstations randomly by considering the given cycle time.

As it is evident at the first glance, one of the strength points of this method besides the performance simplicity, it has the ability to produce different results due to the use of random selection process for the allocation of the activities in each step. Because of having much flexibility and high performance power, this method gives desired result in every run but it needs to be considered this method gives different results in every run. Hence, it is necessary to check the results of several runs to reach the best outcome to calculate the line efficiency and smoothness index.

In contrast, exist a measure to evaluate and compare a new meta-heuristic algorithm that it is one of the most important things to find the performance power of an algorithm. Hence, generating the same initial solutions for a problem with a constant parameters such as cycle time, processing times and precedence graph, seems necessary. Therefore, COMSOAL method has been used only in second state for improving the initial solution.

Leaving aside the COMSOAL method in order to find a way to create an initial solution that every time gives a constant solution for the same problem, Ranked

Positional Weight (RPW) method [26] is considered. After some necessary changes to improve result, the Revised-RPW method proposed.

3.2.1 Algorithm of the RPW to Solve UALB Problem

In our proposal we focus only on U-shape assembly line balancing problem, whereas RPW is a general solution for different forms of assembly lines. The weight discussed above must be considered in both forward direction and backward direction. [24]

The parameters used in this method are:

$T(S_i)$	Total time of each station
$T(x)$	Time of each task
CT	Given cycle time
N	Number of tasks
M	Number of workstations
S	Minimum feasible number of workstation
MCT	Minimum feasible cycle time
*CT**	Modified cycle time

To apply a priority for the tasks, we have used a precedence network calculating task's weights which we can explained as the total of activity time and times of the various succeed or progress, correspondingly. There are two criterion for assigning tasks to workstations: Firstly, succession and precedence priorities must be kept, and secondly, the workstation must have free space to handle the assigning task. In case of multiple available tasks, the one with the highest weight is assigned. Whenever all tasks have been bounded we say the assignment is complete.

In this solution, *CT* is replaced by a new symbol *CT** which is computed as below [24]:

$$S = \sum_{i=1}^{n} T(x) \,/\, CT \tag{3.1}$$

If S became non-integer value it should be rounded up.

$$MCT = \sum_{i=1}^{n} T(x)/S \tag{3.2}$$

$$CT^{*} = [(MCT + S)/2] \tag{3.3}$$

It should be kept in mind that *MCT* < *CT** <*CT*. As it can be concluded *CT* can be chosen freely in the domain *(MCT, CT)*. However, choosing *CT** as *CT* yields more appropriate outputs. To maintain the preferred circumstances, following relations must be satisfied [24]:

$$T(S_i) = \sum_{x \in S_i} T(x) \leq CT \qquad i = 1, \dots, M \tag{3.4}$$

$$if\ (x, y) \in P,\ x \in S_i\ and\ y \in S_j\ then\ i \leq j\ for\ all\ x. \tag{3.5}$$

$$if\ (y, z) \in P,\ y \in S_j\ and\ z \in S_k\ then\ k \leq j\ for\ all\ z. \tag{3.6}$$

Statement (3.4) implies that total times of tasks that there are in a station could not be more than *CT*. Statement (3.5) means that whenever task *x* comes before y and it is carried out at station number *i* while y is carried out at station number *j*, then *i* is less or equal *j*, i.e. *y* is done at in the station where *x* is done or after that. Statement (3.6) is very similar to (3.5) to guarantee the priority imperatives are fulfilled in the regressively bearing [24].

Below we describe generally the RPW rule algorithm by considering the noted explanations:

(1) Computing the least quantity of workstations *S* and the least possible cycle time *MCT* and *CT*= [(MCT+CT)/2]* which is the adjusted value.

(2) Adopting a new workstation and computing every work element's weight in 2 ways, once in forward and once in backward direction. After that, the activities which are appropriate for assigning are identified and a candidate list is generated.

(3) Arrange the weight of work elements as descending order.

(4) Assign the first activity with highest weight to first station

(5) Calculate the idle time *(IT)* for station r that has k task with below formula:

$$IT = \ CT^* - \sum_{i=1}^{k} t_{\mathrm{ir}} \tag{3.7}$$

(6) comparing time of first none assigned activity that has highest positional weight with idle time of last work station (here is first station too), then assign the activity if it has equal or less time than last work station's idle time.

(7) Assign activity to a new workstation if it's duration is bigger than the existent workstation's idle time and again implement step 5.

A sample with 12 activities and process duration *(CT)* of 12 seconds is demonstrated for delineation. The priority system of the exhibited sample is graphically indicated in Figure 3.1 [24].

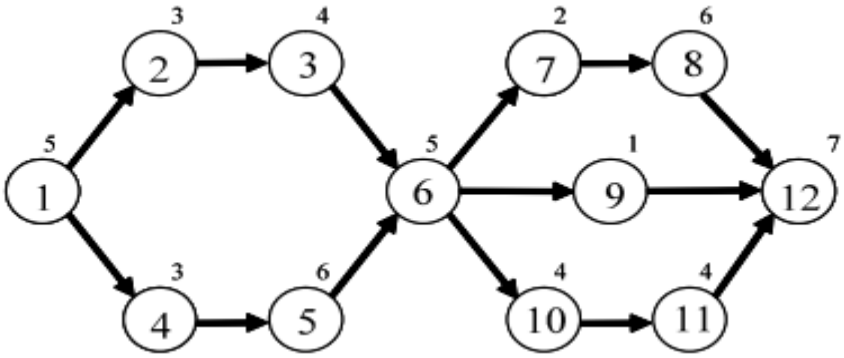

Figure 3.1: Priority chart of assembly line

We assume that *CT* equals to twelve seconds. *S*, *MTC* and *CT** [24] can be computed by the given equations. Initializing task's weights is done based on the Table 3.1.

Here with implementation of this method, the solution will be as below:

Table 3.1: Weight Computation in Forward and Backwards Direction

Task number	1	2	3	4	5	6	7	8	9	10	11	12
Backward weight	34	27	24	29	26	20	15	13	8	15	11	7
Forward weight	5	8	12	8	14	19	21	27	20	23	27	34

The summary of results of the assigning process using the RPW method is given in Table 3.2.

Table 3.2: Assigning Process for U-Shape Line Using RPW

CT = 12 , CT = 11*				
Iteration	Candidate List	Assigned Task	Station No.	Station's Idle Time
1	1,12	1	1	6
2	12, 2,4	12	2	4
3	2,4,8,9,11	4	2	1
4	2,8,9,11,5	2	3	8
5	8,9,11,5,3	8	3	2
6	9,11,5,3,7	11	4	7
7	9,5,3,7,10	5	4	1
8	9,3,7,10	3	5	7
9	9,7,10,6	10	5	3
10	9,7,6	7	5	1
11	9,6	6	6	6
12	9	9	6	5

As it shown in Table 3.2 in first iteration, we assigned task Number 1 to first workstation by considering positional weight. Then in next iteration between available tasks in candidate list, we must select task Number 12 according to its positional weight but because time of this task is more than idle time of current workstation we must assigned it to the new workstation. By following these steps, all tasks should assigned.

3.2.2 Revised Ranked Positional Weight Method

As previously mentioned each of the activities can be allocated to last station that contains all their predecessor (or successor) activities of that task or to the next stations according to this point, new algorithm solution is suggested and described as well.

In this method, from first step to the fourth one is quite similar to classified position weighting method. The fifth step in this method is revised as below:

The nomination of last station that contain predecessors (or successor) of first none assigned activity which has highest positional weighting and calculate idle time of workstation j according to the formula (step fifth). Steps six and seven are exactly similar to last method.

The steps of Revised-RPW can summarized as flowchart in Figure 3.2, (See Appendix B in order to find out source codes). Our input data are precedence diagram and given cycle time that they are provided in Fathi's study [24].

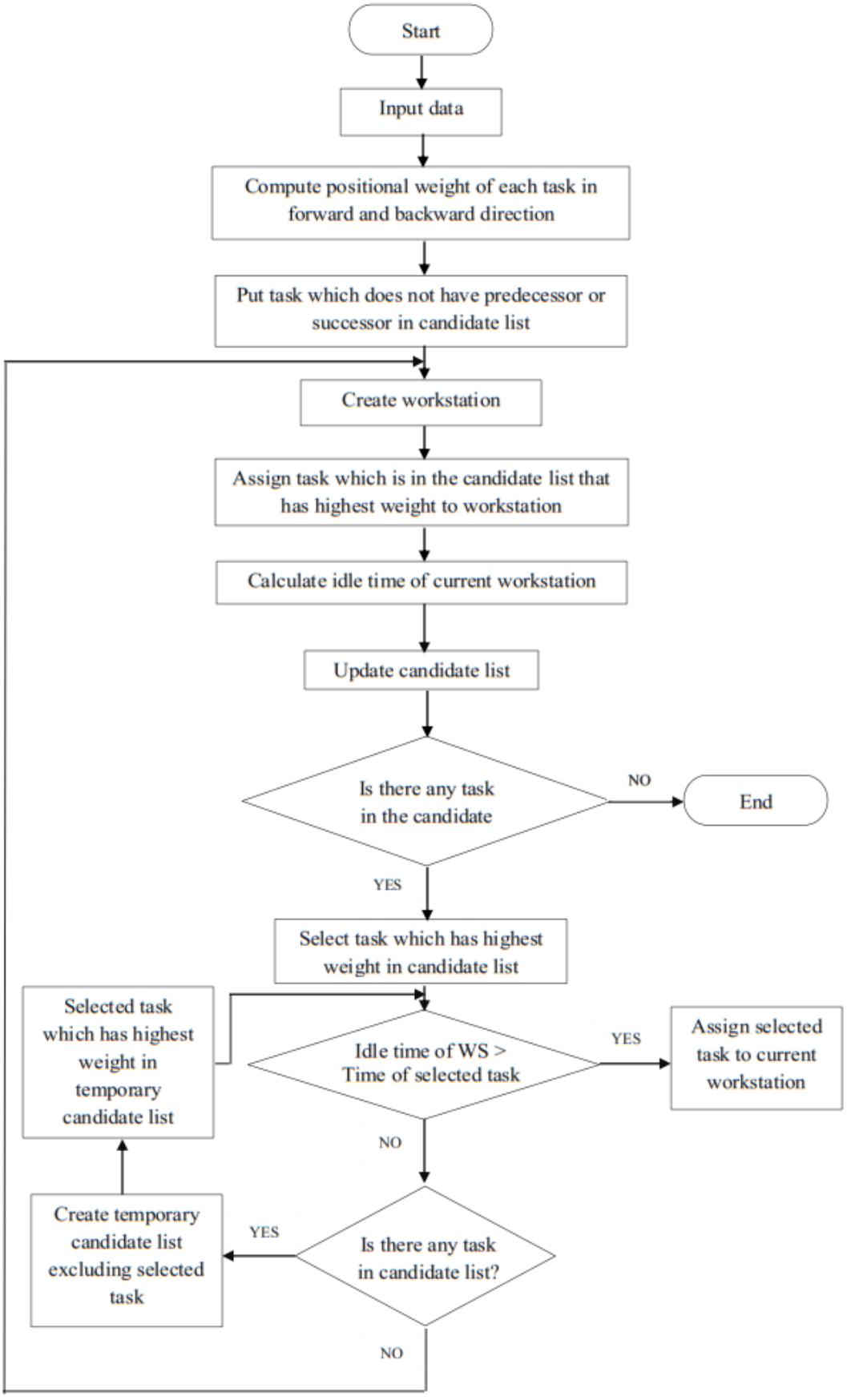

Figure 3.2: Flowchart of Revised-RPW

In this step, we implement revised method and compare it with last solution's quality.

Table 3.3: Assigning Process for U-Shape Line Using R-RPW

CT = 12 , CT* = 11				
Iteration	Candidate List	Assigned Task	Station No.	Station's Idle Time
1	1,12	1	1	6
2	12,2,4	4	1	3
3	12,2,5	2	1	0
4	12,5,3	12	2	4
5	5,3,8,9,11	11	2	0
6	5,3,8,9,10	8	3	5
7	5,3,9,10,7	3	3	1
8	5,9,10,7	9	3	0
9	5,10,7	5	4	5
10	10,7,6	10	4	1
11	7,6	7	5	9
12	6	6	5	4

In Table 3.3, notice that same as Table 3.2 at first we assigned task number 1 to first workstation, by considering it is not possible to assign task number 12 to this workstation according to the idle time, the other tasks in the candidate list will be considered. By this explanation that which has more weight must be selected and then again we will compare idle time of current workstation to time of selected task, if it is possible to assign we will do it otherwise we will check the other tasks which are available in candidate list. If none of them could assign then we must create new workstation.

For comparing two mentioned methods, we have four parameters as below:

1- Number of work stations

2- Line efficiency index: The line efficiency is an indicator for measuring the usage of line [24]

$$LE = \left(\frac{\sum_{i=1}^{m} T(S_i)}{M*CT}\right) * 100 \quad (3.8)$$

Where $T(S_i)$ is total time of tasks that there are in the station, therefore

$$T(S_i) = \sum_{j \in S_i} T_j \quad (3.9)$$

3- Smoothness index: This index is for measuring the standard deviation of work distribution between the workstations. [24]

$$SI = \sqrt{\frac{\sum_{i=1}^{m} \left(T(S_{max}) - T(S_i)\right)^2}{m}} \quad (3.10)$$

Where TS_{max} is maximum of TS_i .

4- Variation: This index is for determining the standard deviation of utilization of stations. [8]

$$V = \sqrt{\frac{\sum_{i=1}^{m} (U_i - aver)^2}{m}} \quad (3.11)$$

Where $aver = \sum_{i=1}^{m} U_i/m$ is an average utilization for all workstations and the utilization of workstation Si calculated with below formula [8]

$$U_i = T(S_i)/T(S_{max}) \quad (3.12)$$

In below table we have figures each parameters.

Table 3.4: Comparing Quality of RPW and R-RPW's Solutions

Solution algorithm	**Station's number**	**Line efficiency index**	**Smoothness index**	**Variation**
RPW	6	83.33%	6.4807	0.2055
R-RPW	5	90.91%	4.1231	0.1408

As you see in Table 3.4, according to the objective function of ALB's problems, minimum number of station is desired and the workstations in position weighting method are one unit more than the revised position weighting. With considering importance of line's capacity usage and efficiency, the results shown that efficiency index in revised position weighting is 17 percent more than this index in position weighting method.

On the other hand, according to the objective function of minimizing smoothness index and variation, it is shown that the amount of these values in revised position weighting are better than these contents in position weighing method. It is clear that the solution of revised positional weighting method will never be worse than positional weighing method. Therefore, for nomination initial solution in final algorithm, revised position weighing is used.

3.3 Initial Solution Improvement until Achieving Final Solution

In implementation of revised position weighting method, it present optimum number of workstation in some cases. However, this method does not present optimum answer. Therefore to reach this point, some optimization algorithm is used which has three sections as follows.

1- Using meta-heuristic algorithm for optimize the first solution.

2- Using a heuristic method for assign the activities again after mutation.

3- Using a method for select better solution in each step.

In this section, first we introduce grouping evolution strategy algorithm then we describe steps of improving initial solution until reaching the final answer.

3.3.1 Evolution Strategies

Darwin's theories emphasize on the principle of variation and selection which is the basis of Darwinian evolution. Rechenberg introduced evolution strategies [14] which are basically a mathematical translation of Darwinian biological evolution and applied them as general optimization technique.

$(\mu / \rho + \lambda)$ – ES presents a group of evolution strategies. All members operate with a population Π^t which contains μ individuals (time proceeds in discrete steps (generation) and is indicated by superscript t). In each generation t, a set Q^t of λ offspring solutions are produced from Π^t via recombination and mutation operators. The symbol ρ indicates the number of parental solutions involved in the creation of every single offspring solution. When ρ =1, it will be omitted. The new population Π^{t+1} is created by means of the selection schemes based on individual appropriateness [10].

Selection in ES which is shown by "$+$" is goal-directed upon individuals' appropriateness ranks. "$+$", denotes two mutually exclusive selection types. According to the selection type, selection can be either from $\Pi^t \cup Q^t$ or from Q^t.Using "+" selection, the μ best of $\mu + \lambda$ candidates in $\Pi^t \cup Q^t$ are selected from Π^{t+1}. Using selection, it is the μ best of λ candidates in Q^t that from Π^{t+1}.

New off springs are chosen by a two-step process: recombination and mutation. For recombination, ρ numbers of parents are taken randomly and their centroid point is

calculated. A point symmetric perturbation is added to the recombination output to generate minor deviations. The latter is the mutation step. The perturbation is chosen from an isotropic normal distribution. Population $\Pi^t = \{X_1^t\ ,\ X_2^t, \dots, X_\mu^t\}$ with $X_k^t = (x_{k1}^t, x_{k2}^t, \dots, x_{kD}^t) \forall\ k=1, \dots, \mu$ is a *D* dimensional candidate solution in the real-valued examine space, the set of offspring candidate solutions (Q^t) consists of vectors such as $Y_i^t = (y_{i1}^t, y_{i2}^t, \dots, y_{iD}^t) \forall\ i=1, \dots, \lambda$, where:

$$Y_{id}^t = \frac{1}{\rho}\sum_{k=1}^{\rho} X_{i_{kd}}^t + Z_d \qquad \forall d = 1, \dots, \mathrm{D}, \forall i = 1, \dots, \lambda \qquad (3.13)$$

In the above formula $z_d^t = \sigma^t N_d(0,1)$, and $N_d(0,1)$ is a normally distributed random number related to *d* (i.e. the variation source). σ^t is known as strategy parameter or mutation strength. It is computed in the time of evolution and shows how much deviation is expected between the parents' centroids and their offspring. Sometimes it is better to have a vector containing multiple strategy parameters $\sigma^t = (\sigma_1^t, \sigma_2^t, \dots, \sigma_D^t)$. The index is chosen in an independent manner with replacement and same probability from $\{1, \dots, \mu\}$.

Mutation strength σ^t must be tuned. In case of being less than a threshold, it reduces the search speed and in case of being more than a threshold, deviations can happen. Hence, in evolving solutions this tuning plays an essential role called mutation strength adaption. Rechenberg introduced 1/5-success rule in 1973 as the first method of adapting mutation strength in (1+1)-ES [14]. He proved that by setting the mutation strength at a level with success rate of 1/5, an approximate optimum performance could be achieved. The only thing to do is monitoring the portion of time an offspring candidate solution is superior to its parent over a number of time steps (i.e., estimate the success probability), if success probability comes over 1/5, σ^t is set to a higher

value and vice versa. Further discussion on ES methods is out of scope of this thesis however readers are encouraged to see Beyer and Schwefel (2002) [15], and Arnold and Beyer (2003) [27] for more detail.

3.3.2 Representation of Grouping Evolution Strategy (GES)

One of the key issues when designing an evolutionary algorithm is the solution representation that used in GES as grouping representation (Falkenauer, 1994). When optimizing a continuous function by ES, generally each solution is characterised with a vector of length *D* of real numbers (*D* is the problem dimension (i.e., the number of variables)). Similarly, for grouping problems, one can imply a solution *i* with *Di* groups as a structure which length is equal to the number of groups in this context, groups are considered as variables (look at the left part of Figure 3.3).

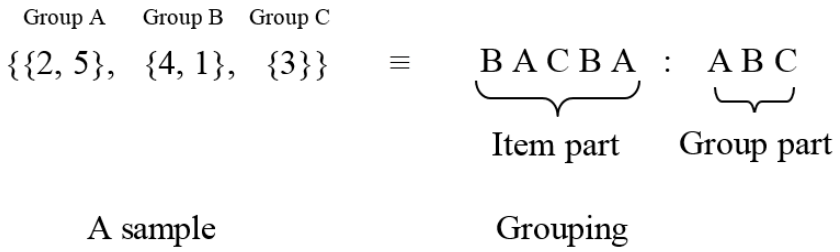

Figure 3.3: A grouping sample and its relevant group encoding

It should be kept in mind that possible solutions of grouping problems are not necessarily identical in terms of length. i.e. GES must support solutions of inconstant length. It is similar to bin packing or graph colouring problems wherein the quantity of bins/colours is inconstant. In parallel-machines scheduling problems which machines are loaded with grouped tasks, the quantity of groups is kept constant. Through Kashan study [10], they the number of groups of a possible solution *X* is shown by *DX*.

3.3.3 The GES Mutation Operator

Given the grouping representation, the reconstruction of equation 3.13 is the aim of this section, so that the new equation has the ability to work with groups rather than to work with real numbers. To reconstruct the update equation, the basic idea is using appropriate operators instead of Arithmetic's operator. Specifically, the purpose is using an appropriate operator instead of the operator -. Same as operator - which makes the vastness of difference between two real numbers small, the dissimilarity groups criteria can also make the distance / difference between the two groups low.

Suppose that the size of the two groups G, G' is shown as $|G|, |G'|$. By the quantification of the degree of similarity between the two groups, it can be seen that how the two groups are similar and how far they are from each other. Much similarity degree with multi-purpose applications are presented, some of which goes back to a century ago. Among these similarity criteria Jaccard, Simpson and Kulczynski similarity coefficient can be mentioned. One of the most widely used indicator for determining the similarity coefficient between G, G' is Jaccard similarity coefficient which can be defined as follows

$$\textit{Jaccard's Similarity}\ (G, G') = \frac{|G \cap G'|}{|G \cup G'|} \tag{3.14}$$

Index value above is equal to 1 if $G = G'$ and it will be equal to zero if $G \cap G' = \emptyset$. Based on an available similarity coefficient, the dissimilarity/ distance coefficient can be easily defined in order to calculate the distance between the two groups from each other. For example, with regard to Jaccard similarity coefficient, Jaccard distance coefficient between the two mentioned groups can be defined as follows:

$$Jaccard's\ Distance\ (G, G') = 1 - \frac{|G \cap G'|}{|G \cup G'|} \quad (3.15)$$

It is clear that $0 \leq Jaccard's\ Distance\ (G, G') \leq 1$.

Based on the concepts introduced in the previous section, the aim of this section is developing a grouping version of evolutionary strategies. Using the demonstrating scheme based on group, the main aim of this part is changing the classic ES mutation and its development based on grouping problems structure.

In order to generate Gaussian mutation it should be supposed that $\rho = 1$. This is because, it is known to work with groups that are major constituents of answer, and directly, the mean operator for the groups (sets) is not defined. Therefore, we assume that each child of a single parent is produced. Changing the classic ES mutation in the form of $y_{id}^t - x_{i_k d}^t = z_d$ and substitution of and alternative operator distance / difference (the coefficient of Jaccard distance) operator instead of -, the equation of Gaussian mutation in grouping evolutionary strategy is introduced as follows:

$$Distance\ (y_{id}^t, x_{i_k d}^t) \approx z_d \quad (3\text{-}16)$$

Where, $z_d = \sigma^t N_d(0,1)$ and $d = 1, \dots, D_{x_{i_k}^t}, i = 1, \dots, \lambda$. Indices i_k is the chosen index among $\{1, \dots, \mu\}$. Again, it should be noted that $x_{i_k d}^t$, y_{id}^t in equation 3-16, each denote a group of objects and not real numbers. Since $0 \leq Distance(y_{id}^t, x_{i_k d}^t) \leq 1$ it is located on the right side of equation 3-16 if $z_d > 1$. $z_d = 1$ and if $z_d < 0$ then $z_d = 0$ [10].

In equation 3-16, z_d can be used to make the fluctuation around zero and the vastness of this deviation will be determined based on the value of strategy parameter. If there

was an increase in σ^t, there will be more chance of large deviations and vice versa. While z_d it may be somewhat arbitrary and free of marks, the range operator distance is limited to the range [0-1]. This evidence suggests that, z_d may not be a good presenter as a source of variation production in evolutionary strategy group. Therefore, we should looking for a source for creating a diversion. As a starting point, it is desirable that the candidate source for creating distortion will be amplitude probability density function which is only in the range [0-1]. Additionally, Similar to the standard normal distribution the chance to produce a certain amount with changing the amount of σ^t changes. It is desirable that the candidate probability density function can be considered different opportunities to produce a certain amount in the range [0-1] by using different values for the control parameters. Now some probability density function that satisfies their requirements are available, including beta distribution, triangular distribution, distribution Kumaraswamy (Kumaraswamy, 1980)

The amplitude for probability density functions that have been mentioned only in the range [0-1] are defined and all of them are flexible enough. However, in grouping evolutionary strategy development, it is preferable to use the beta distribution. The reason for this choice is that, in comparison with triangular density function, beta density function has less control parameter and it models skewness in a very favorable manner. Kumaraswamy distribution is also very flexible and it has a two shape parameters. But beta distribution is much better known, and many programming software such as MATLAB are equipped with module for generating beta random numbers [10].

In statistics and probability, Beta distribution can be shown by $B(\alpha, \beta)$ with positive parameters α and β that are shape parameters. Various forms of the beta probability density function have been illustrated in Figure 3.4. As it can be clearly seen, the distribution is very flexible. So far, it can be concluded that equation 3.16 can be changed as follows:

$$Distance\left(y_{id}^{t}, x_{i_k d}^{t}\right) \approx Beta_d(\alpha^t, \beta^t), \forall d = 1, \dots, D_{x_{i_k}^t}, \forall i = 1, \dots, \lambda, i_k \in \{1 \dots \mu\} \quad (3.17)$$

In the above equation $Beta_d(.,.)$ introduces the beta random number which is producing per each d group.

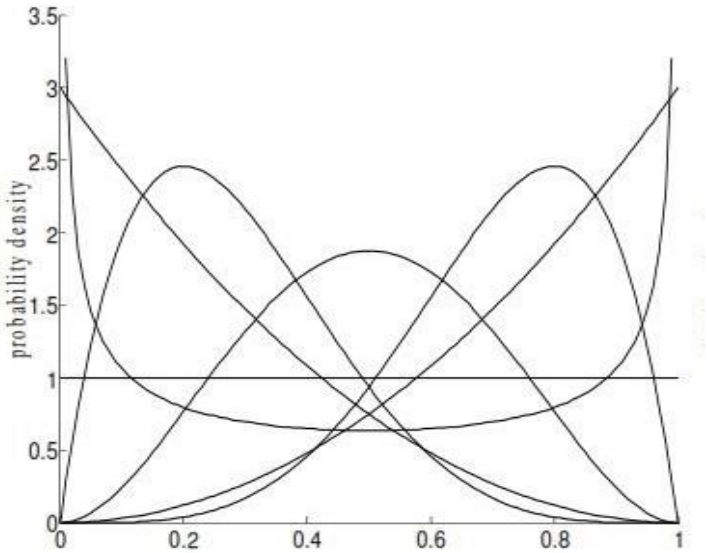

Figure 3.4: Samples of Beta PDF for different estimations of the parameters [10]

If the standard normal distribution is used, only strategy parameter σ^t needs to be available. However, by using the beta distribution, the two strategy parameters σ^t and β^t must evolve. Referring to Figure 3.4 It should be noted that there should be only form of beta distribution with a single peak and the J-shaped or bell-shaped. U-shaped distributions, distributions that are symmetric and the peak density on either side values that are zero or one. Therefore, it is expected that by U-shaped probability

density functions the chance for generating random numbers close to zero or one is the same.

This means that the chances of similarity or dissimilarity of new group of y_{id}^{t} to the present group of $x_{i_k d}^{t}$ is available, which it does not seem logical. Figure 3.5 indicates that the assumption of a constant shape parameter, for example β^{t}, at an appropriate level could be useful in modeling skewness of the probability density functions of a single peak beta.

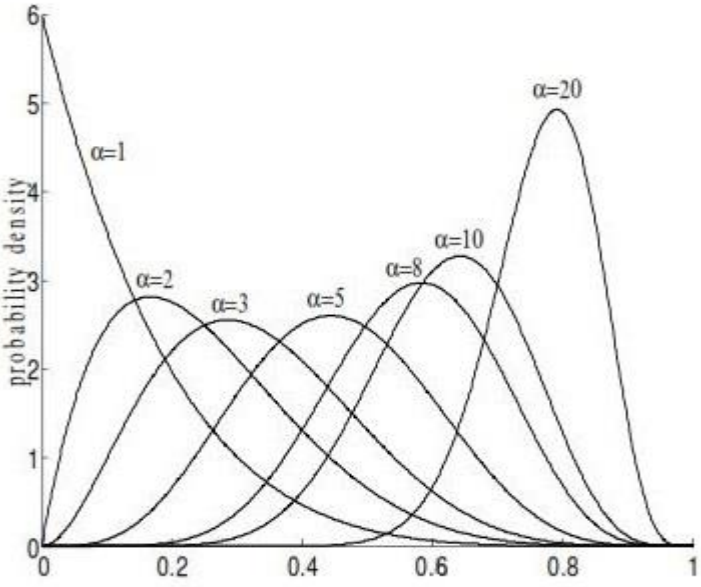

Figure 3.5: Samples of Beta PDF for different estimates of α at level of β=6 [10]

In Figure 3.6, the relationship between different values of the shape parameters of the beta distribution and the shape of the beta probability density function has been shown. As can be recognized, for $\alpha \geq 1$ or $\beta \geq 1$, the beta distribution forms all desired types. By assuming a constant value for β^{t} in $\beta \geq 1$, the parameter α^{t} can only be developed

during the search process. The final equation of mutations in a group evolutionary strategy takes the following form:

$$Distance\left(y_{id}^{t}, x_{i_k d}^{t}\right) \approx Beta_d(\alpha^t, \beta), \forall d = 1, \dots, D_{x_{i_k}^t}, \forall i = 1, \dots, \lambda, i_k \in \{1 \dots \mu\} \quad (3.18)$$

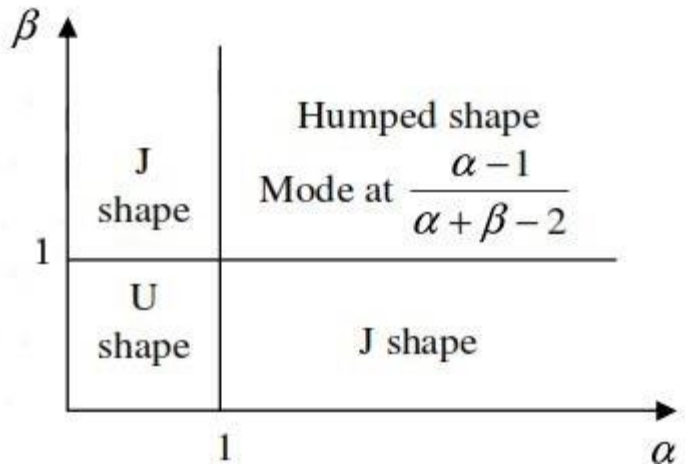

Figure 3.6: Shape of the Beta distribution as a function of α and β [10]

In grouping evolutionary strategy algorithm there is no difference for using scheme representation group based on object name or object attributes. In this algorithm the number of shared object between $x_{i_k d}^{t}$ and y_{id}^{t} can be calculated as follows:

$$n_{id}^{t} = \left\lfloor (1 - Beta_d(\alpha^t, \beta)) \left| x_{i_k d}^{t} \right| \right\rfloor \quad (3.19)$$

Selection process of evolutionary strategy algorithm is a deterministic and its similar to evolutionary strategy algorithm. Similar to those used in evolutionary strategy algorithm, $(\mu^+, \lambda) - GES$ can be used to introduce a variety of strategies to select a group evolutionary strategy. Also similar to the Law of Success $1/5$, with the initial value α^0, if the estimated probability of success is greater than the threshold P_s after G iteration, then there will be increase in α^t and otherwise, its value decreases. In this thesis, the algorithm of $(1 + \lambda) - GES$ *i*s used for performance comparisons.

3.3.4 Improving the Initial Solution

For solving ALB problems with high number of activities considering it is a hard problem, each of the meta-heuristic algorithm like genetic algorithm, tabu search algorithm, ant colony algorithm or evolution strategy can be used.

Revised-RPW rarely provides perfect solutions under some specific circumstances. Note that ALBPs are in the NP-hard kind of problems, GES algorithm is used to enhance the solution which provided by Revised-RPW.

GES algorithms to avoid static solutions apply mutation operator. First, a number of tasks of the initial solution are removed, by using a heuristic technique, the missing tasks are assigned and the solution is completed. By Figure 3.7, we describe all steps briefly.

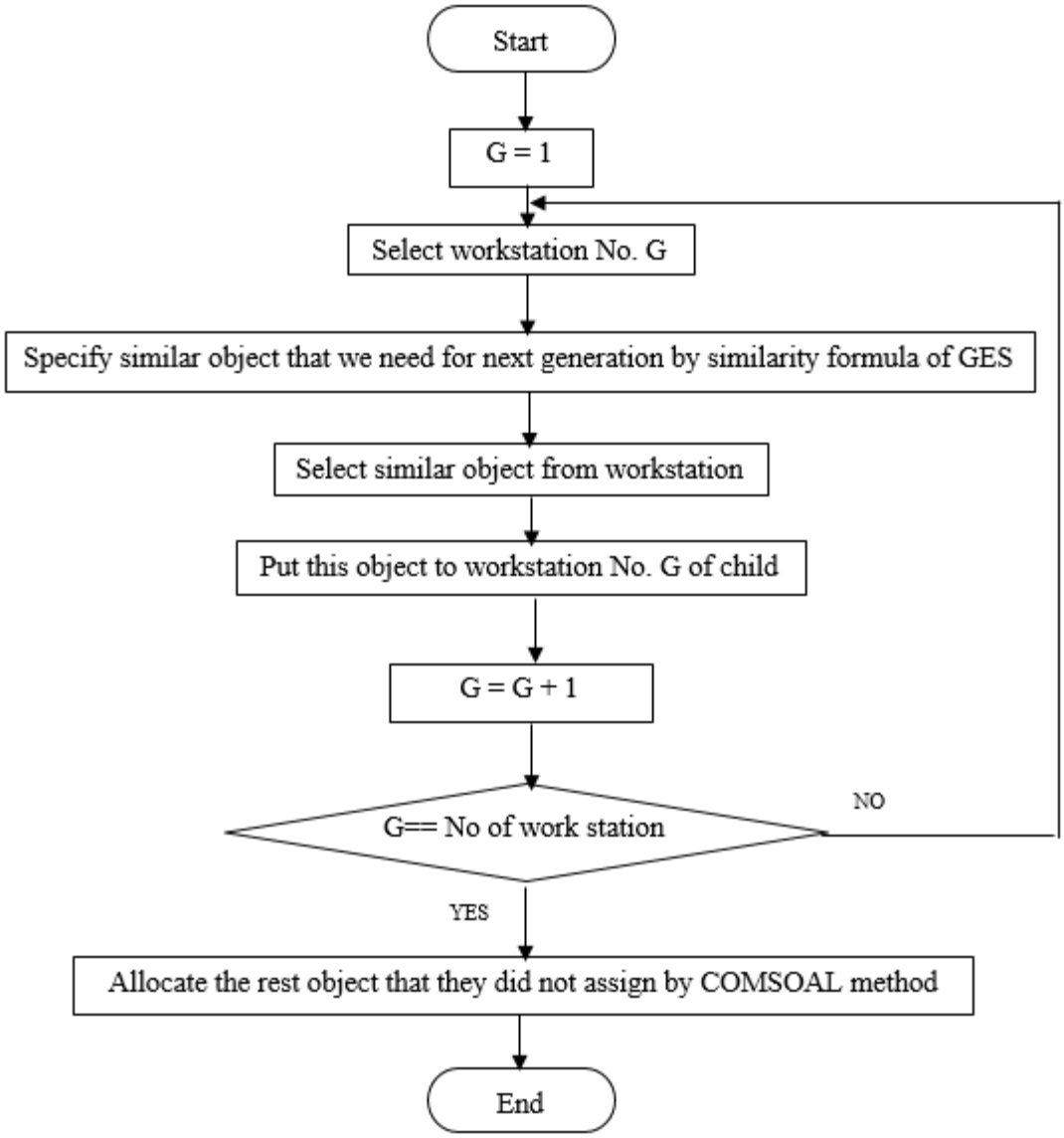

Figure 3.7: Flowchart of Mutation Operator

3.3.5 Using a Heuristic Method to Assigning Tasks after Mutation

Operator mutation is one of the most used operators in meta-heuristic algorithm, especially in grouping evolution strategy to find a better solution without any static result in research. After removing some assigned activities from related workstations according to a special pattern and allocating them again to stations by a heuristic method, we have a better solution in each step.

It should be mentioned that in this step we could not use revised-RPW because with adoption of this method we only have one solution. Considering simplicity and flexibility of COMSOAL method (Arcus, 1963) [25], we purposed this technique for solving this issue. This flexibility is because of generating different solutions by an accidental choosing process among activities that are ready to assign in each step. With doing a necessary revision, we presented Revised-COMSOAL method to create a better solution in the following.

3.3.6 Revised-COMSOAL Method

Revised-COMSOAL method in compare with classic COMSOAL by Arcus (1963) [25] has an important change according to the condition of solution algorithm.

The classical COMSOAL method for creating solution, always starts from the first node (activity) and continues with the nodes that do not have any predecessor. Then it assigns activities to workstations considering idle time of stations. However, in our case, after performing the mutation operator of GES, most of the time, a task in the middle of the precedence diagram must be assigned. Therefore, at first, the algorithm must distinguish which activities have no predecessor or successor and after that, it must ignore the assigning of those activities that have assigned before. To create these changes in the revised algorithm, there is a constraint about predecessor and successor activities, which their algorithm are as below:

Step1: Choose the activities without predecessor or successor that are ready to assign.

Step2: Finding the possible workstations which activity can be assigned.

Step3: Find the highest workstations between stations that contains predecessor activities (MPWS) and successor activities (MSWS) of chosen task. Then compare them and choose the minimum one.

Step4: Control the assumption that the chosen activity's station number must be equal or bigger than workstation number that we described in step3.

Step5: remove the assignment of chosen activity if the assumption of step4 violated.

In Figure 3.8, we will describe these steps briefly as a Flowchart (See Appendix B in order to find out source codes). In this Flowchart, we considered APWS as an appropriate workstation for assigning activities that are ready to assign.

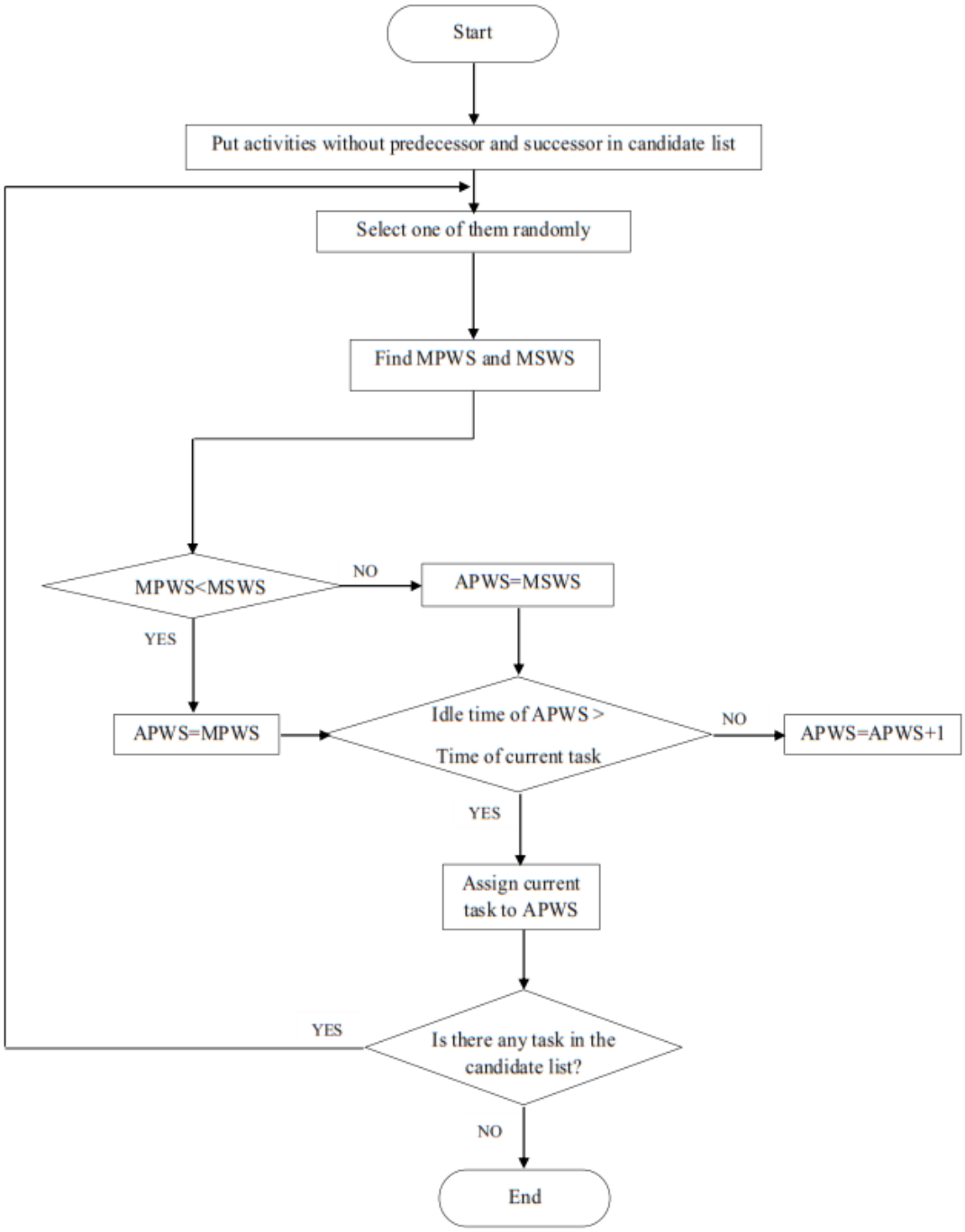

Figure 3.8: Flowchart of Revised-COMSOAL

3.3.7 Using a Method to Select the Best Solution in Every Step

After changing the solution method in proposed algorithm and improving it in every step, additional to the chosen result of the previous step, two new solutions are produced. To find the best result for producing the next solution, it is necessary to

compare all three solutions in every step. In this regard, a method where considered as selection operator, which is based on smoothness index. Its formula was presented in equation 3.10.

This none-linear objective function, speeds up the transmission of activities from low to high-pressure workstation then the chance of getting an empty workstation in the next solutions, will be increased. This lead to this idea that *SI* index was used for selecting a better solution not as an objective function.

Figure 3.9 describes how we improved the initial solution, where it was came from the input data's such as precedency diagram and given cycle time (See Appendices in order to find out pseudocodes and source codes).

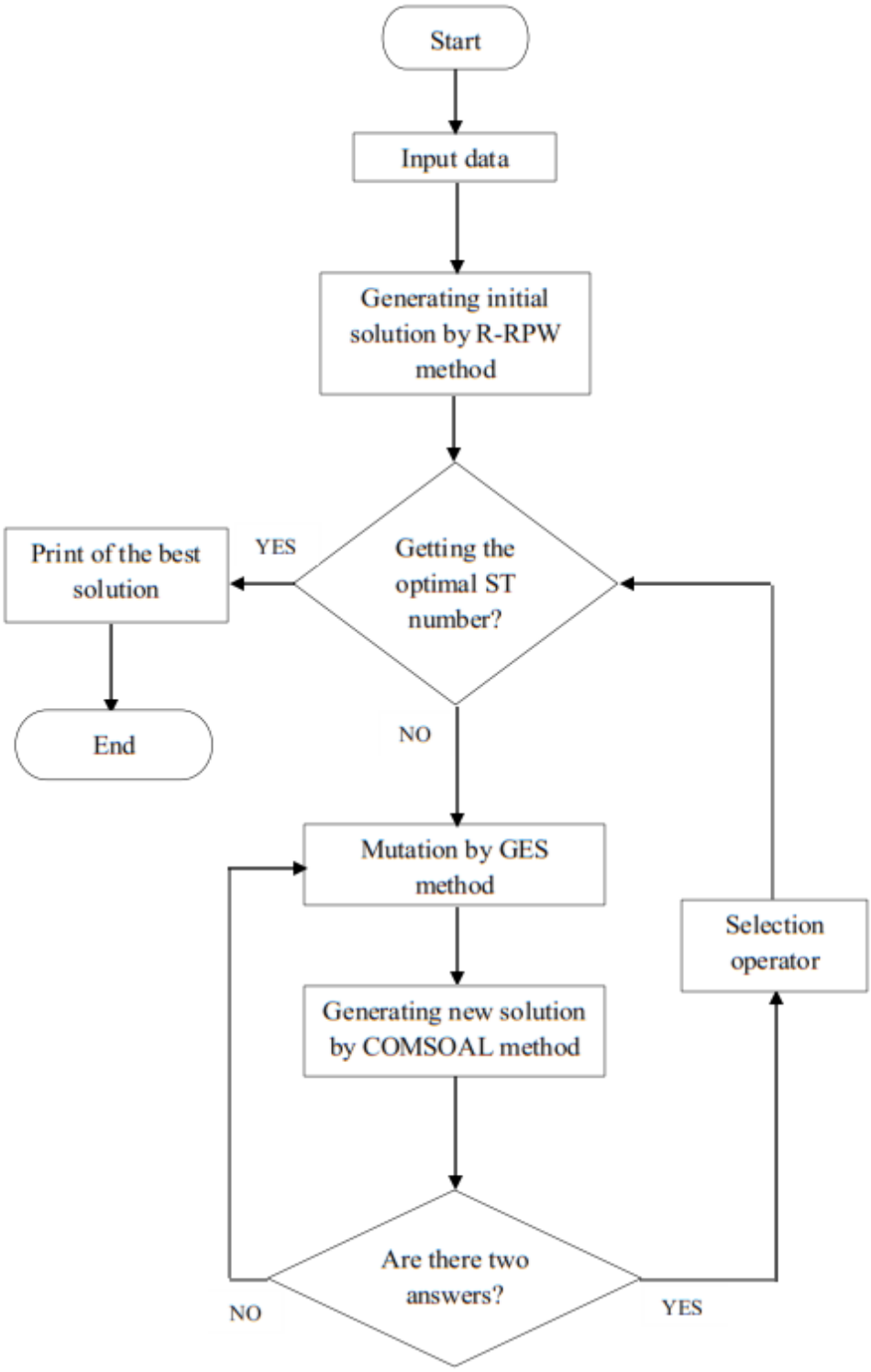

Figure 3.9: Flowchart of the proposed algorithm

Chapter 4

COMPUTATIONAL RESULT

4.1 Simulation Setup and Performance Metrics

In this chapter, the results of proposed methods compared with some well-known problems and the solutions for the elected problems are compared with the best solutions that are already obtained by Hwang et al [8]. The needed data to examine the mentioned methods and to compare the results have been acquired from their study. All procedures implemented in MATLAB 2013a software and executed on an Intel(R) Pentium(R) Dual CPU computer with 2.00GHz of CPU speed and 4.00 GB of RAM.

For the performance measurement of the proposed algorithm, the results of using the selection operator method is compared by solving well-known problems. The proposed algorithm has several parts and it is possible that the initial solution is the optimal solution.

Rest of this chapter it reveals that by using proposed method the Initial solution will be improved, then comparing this method with an existing one that was used Genetic algorithm for balancing the U-shaped assembly line [8].

Perfect balance of assembly line is attained by combination of work elements in a way that the total busy time of workstations will be same as cycle time. Since a perfect balance can rarely happens, some other metrics are used in UALBP type-1 to compare

different combinations. These metrics can evaluate the performance and efficiency of the balance. Below, we describe each:

Number of Work Station (NWS): Having less NWS means more proper task dispatch that leads to a line, which is more effective. It is clear that less number of workstations can save the budgets and working area [28, 29].

Line Efficiency (*LE*): *LE* is yield by summing up all station's time to the *CT* over the station number. It reflects the percentage of line's usage. Obviously, higher values of *LE* is more desirable with the ideal value hundred. To maximize *LE* the station number must be minimized. *LE* is calculated as equation 3.9 that is described before.[28, 30]

Smoothness Index (*SI*): An important performance variable in a production line. *SI* indicates the total time when a station is idle (not working). It usually happens when an improper assignment had been done. The ideal value for *SI* is zero i.e. the best balancing. The minimum value of *SI* can be reached when the workload difference among workstations is decreased as much as possible. *SI* can be computed as equation 3.10. [29]

Variation (*V*): Another important performance variable on a production line by considering utilization of each station is variation. Same as *SI*, The minimum amount of *V* can be accomplished by decreasing the workload difference among workstations. *V* can be processed as mathematical statement which was described in equation 3.11.

4.2 Description of Improving the Initial Solution by Proposed Method

In Table 4.1, the number of workstations in proposed method in most tests had been reached to the optimum number of workstations that they are considered by Hwang et al. [8].

Table 4.1: Number of Workstations

Problem	No of task	Cycle time	Optimum No of work station	RPW	Revised-RPW	GES with Revised-RPW
Mitchell	21	14	8	9	8	8
		15	8	9	8	8
		21	5	6	6	5
Heskia	28	138	8	10	8	8
		205	5	6	6	5
		324	4	4	4	4
Sawyer	30	27	13	15	13	13
		33	10	13	11	10
		54	6	7	6	6
Tonge	70	176	21	23	21	21
		364	10	11	10	10
		468	8	8	8	8
Arcus	83	5853	13	15	14	13
		6842	12	13	12	12
		8412	10	10	10	10
		10816	8	8	8	8
Arcus	111	5755	27	29	27	27
		10027	16	17	16	16
		10743	15	17	15	15
		17067	9	10	9	9

In Table 4.2, six different problems with three or four different cycle times are considered. For each RPW, Revised-RPW and GES methods, three indexes such as line efficiency, variation and smoothness index are compared.

As our aim is improving the initial solution, with Table 4.2 we will find out Line efficiency, Smoothness index and variation of workload toward the Initial solution are improved.

As mentioned before, it is possible that in some cases, Initial solution is the optimal solution; in this case, we engage improving the Smoothness index in the following steps of proposed method.

Table 4.2: Results of Line Efficiency, Smoothness Index and Variation

Problem	No of task	Cycle time	RPW			Proposed					
						Revised-RPW			GES with Revised-RPW		
			Line Efficiency	Variation	Smoothness Index	Line Efficiency	Variation	Smoothness Index	Line Efficiency	Variation	Smoothness Index
Mitchell	21	14	83.33%	0.0662	9.6437	93.75%	0.0974	4.5826	93.75%	0.0236	2.8229
		15	83.33%	0.1579	9.6437	87.5%	0.1614	8.6603	87.5%	0.0428	3.09498
		21	83.33%	0.1711	1.2882	83.33%	0.3107	18.1384	100%	0	0
Heskia	28	138	76.42%	0.1869	127.5069	92.75%	0.1001	48.2494	92.75%	0.0055	13.57
		205	83.66%	0.1956	127.3656	83.66%	0.3658	202.0149	99.9%	0.002	1
		324	80.5%	0.2476	200.4146	80.5%	0.3635	272	79.01%	0.0218	37.66
Sawyer	30	27	80%	0.1553	26.4764	92.31%	0.0913	11.619	92.31%	0.0281	8.30
		33	75.52%	0.1855	36.5377	89.26%	0.1687	21.8861	98.18%	0.0219	3.4641
		54	85.71%	0.1611	30.7571	100%	0	0	100%	0	0
Tonge	70	176	76.7%	0.165	267.8059	94.97%	0.0926	85.0176	94.97%	0.0418	60.1258
		364	89.63%	0.0698	147.5873	96.43%	0.645	84.8764	96.43%	0.0058	22.5666
		468	96.22%	0.0416	72.5121	93.75%	0.1342	196	93.75%	0.0069	37.5807
Arcus	83	5853	86.51%	0.1917	5296.6	92.39%	0.106	2762.1	92.39%	0.0132	969.8034
		6842	86.35%	0.1424	4796.1	92.26%	0.1997	5074.1	92.26%	0.0186	1146.5863
		8412	90.22%	0.1552	4624.7	90.22%	0.2414	6909.9	90.22%	0.0201	1557.8687
		10816	87.55%	0.2011	7231.9	87.55%	0.2948	9782.2	87.55%	0.0153	1540.256
Arcus	111	5755	79.36%	0.1728	8881.2	96.79%	0.0674	2232	96.79%	0.0328	1545.9
		10027	88.6%	0.0794	5718.6	93.76%	0.211	8822.3	93.76%	0.0183	2230.8033
		10743	82.71%	0.2024	11743	93.38%	0.187	8247.7	93.38%	0.0172	2182.1166
		17067	90.01%	0.0882	7042.3	97.93%	0.0515	2841.3	97.93%	0.0041	643.8203

4.3 Comparing With an Available Method

After understanding that the proposed method cause improvement of Initial solution, we will be compared this technique toward method of the Hwang et al. [8]. Considering in the given study they did not mention Smoothness index (SI), we compared the number of Workstations, the Line efficiency and Variation in Table 4.3.

Table 4.3: Results of comparing the proposed method with Hwang et al. study [8]

Problem	No of task	Cycle time	Proposed			Multiple objective GA					
			GES with Revised-RPW			Fitness function (E)			Fitness function (EV)		
			No of work station	Line Efficiency	Variation	No of work station	Line Efficiency	Variation	No of work station	Line Efficiency	Variation
Mitchell	21	14	8	93.75%	**0.0236**	8	93.7%	0.055	8	93.7%	0.023
		15	8	87.5%	0.0428	8	87.5%	0.139	8	87.5%	0.023
		21	5	100%	**0**	5	100.0%	0.000	5	100.0%	0.000
Heskia	28	138	8	92.75%	**0.0055**	8	92.7%	0.112	8	92.7%	0.007
		205	5	99.9%	0.002	5	99.9%	0.001	5	99.9%	0.001
		324	4	79.01%	**0.0218**	4	79.0%	0.332	4	79.0%	0.062
Sawyer	30	27	13	92.31%	0.0281	13	92.3%	0.071	13	92.3%	0.023
		33	10	98.18%	0.0219	10	98.1%	0.024	10	98.1%	0.014
		54	6	100%	**0**	6	100.0%	0.000	6	100.0%	0.000
Tonge	70	176	21	94.97%	0.0418	21	95.0%	0.043	21	95.0%	0.029
		364	10	96.43%	**0.0058**	10	96.4%	0.023	10	96.4%	0.009
		468	8	93.75%	0.0069	8	93.8%	0.063	8	93.8%	0.004
Arcus	83	5853	13	92.39%	**0.0132**	13	92.4%	0.061	13	92.4%	0.014
		6842	12	92.26%	**0.0186**	12	92.2%	0.097	12	92.2%	0.019
		8412	10	90.22%	**0.0201**	10	90.0%	0.123	10	90.0%	0.038
		10816	8	87.55%	**0.0153**	8	87.5%	0.200	8	87.5%	0.089
Arcus	111	5755	27	96.79%	0.0328	27	96.8%	0.036	27	96.8%	0.019
		10027	16	93.76%	**0.0183**	16	93.7%	0.074	16	93.7%	0.029
		10743	15	93.38%	**0.0172**	15	93.3%	0.063	15	93.3%	0.031
		17067	9	97.93%	**0.0041**	9	97.9%	0.019	9	97.9%	0.004

According to the above table, we find out in 13 out of 20 cases we reached to the same or better result in Variation in comparison with the method that Hwang et al. introduced in their study [8]. Therefore, the higher degree of confident that can be achieved by applying this method. In addition, an enhanced assembly line is gained by having more possibilities of workload.

Chapter 5

CONCLUSION

In this research, the assembly line balancing problem according to the reducing number of workstation is considered. Primarily, a mathematical model and then two new state methods including an exact algorithm and hybrid grouping meta-heuristic algorithm were proposed. Former one was to find an initial solution and later one was to improve the initial solution and achieve the best solution, by using a method for selection operator to solve UALBP.

The proposed algorithm is rested on the Grouping Evolution Strategies method while the most useful meta-heuristic algorithm that already exist, are based on Genetic Algorithm. Moreover, to increase the performance of the proposed procedure, the COMSOAL method (Arcus, 1963) [25] is compounded. To put it in the nutshell the obtained outcomes demonstrate that the proposed algorithm in this thesis is more efficient and qualified than the meta-heuristic method that is used in Hwang et al. study [8] for solving UALBPs.

Finally, the possible future work topics based on our discussion in this thesis are: All the processes can be done in more than one station, which means that some parts of one activity can be done in one workstation and the rest in the others.

Minimization of cycle time of workstations can be considered as the simultaneous objective functions, this is the other objective of assembly balancing problem to decrease work hours. Using goal programming to optimize such a problem or using other meth-heuristic methods like Particle Swarm Optimization (PSO), neural network and etc. To compare them with our proposed method in this research, maybe we can reach to better solution in UALB problems.

REFERENCES

[1] Becker, C. & Scholl, A. (2006). "A survey on problems and methods in generalized assembly line balancing," *European Journal of Operational Research,* vol. 168(3), pp. 694-715.

[2] Monden, Y. (1998) "Toyota production system - An integrated approach to just-in-time," Kluwer, Dordrecht.

[3] Scholl, A. (1999) "Balancing and sequencing assembly lines, Heidelberg: Physica.

[4] Miltenburg, J. (2000) "The effect of breakdowns on U-shaped production lines," *International Journal of Production Research,* vol. 38(2), pp. 353-364.

[5] Miltenburg, J. (2001) "U-shaped production lines: A review of theory and practice," *International Journal of Production Economics,* vol. 70(3), pp. 201-214.

[6] Cheng, C., Miltenburg, J. & Motwani, J. (2000). "The effect of straight- and U-shaped lines on quality," *IEEE Transactions on Engineering Management* , vol. 47(3), pp. 321-334.

[7] Aase, G. R., Olson, J. R. & Schniederjans, M. J. (2004). "U-shaped assembly line layouts and their impact on labor productivity: An experimental study," *European Journal of Operational Research* , vol. 156(3), pp. 698-711.

[8] Hwang, R. K., Katayama, H. & Mitsuo, G. (2008). "U-shaped assembly line balancing problem with genetic algorithm," *International Journal of Production Research*, vol. 46, pp. 4637-4649.

[9] Erel, E., Sabuncuoglu, I. & Aksu, B.A. (2001). "Balancing of U-type assembly systems using simulated annealing," *International Journal of Production Research* , vol. 39, pp. 3003-3015.

[10] Kashan, A. H., Jenabi, M. & Kashan, M. H. (2009). "A new solution approach for grouping problems based on evolution strategies," in *IEEE International Conference of Soft Computing and Pattern Recognition*, SoCPaR.

[11] Falkenauer, E. (1994). "A new representation and operators for genetic algorithms applied to grouping problems," *Evolutionary Computation*, vol. 2, pp. 123-144.

[12] Kashan, A. H. & Kashan, M. H. (2013). "GPSO: a novel particle swarm optimizer for grouping problems," *Information Sciences*, vol. 252, pp. 81-95.

[13] Kashan, A. H. (2011). "An efficient algorithm for constrained global optimization and application to mechanical engineering design: league championship algorithm (LCA)," *Computer-Aided Design*, vol. 43, pp. 1769-1792.

[14] Rechenberg, I. (1973). "Evolutionsstrategie: Optimierung technischer Systeme nach den Prinzipien der biologischen Evolution".

[15] Beyer, H.G. & Schwefel, H.P. (2002). "Evolution strategies: a comprehensive introduction," *Natural Computing*, vol. 1, pp. 3-52.

[16] Wijngaard, J. (1994) "The U-line line balancing problem," *Management Science*, vol. 40, pp. 1378-1388.

[17] Miltenburg, J. & Sparling, D. (1995). *Optimal solution algorithms for the U-line balancing problem*, Hamilton: McMaster University.

[18] Scholl, A. & Klein, R. (1999). "ULINO: optimally balancing U-shaped JIT assembly lines," *International Journal of Production Research*, vol. 37(4), pp. 721-736.

[19] Gokcen, H., Agpak, K., Gencer, C. & Kizilkaya, E. (2005). "A shortest route formulation of simple U-type assembly line balancing problem," *Applied Mathematical Modelling* , vol. 29, no. 4, pp. 373-380.

[20] Gokcen, H. & Agpak, K. (2006). "A goal programming approach to simple U-line balancing problem," *European Journal of Operational Research* , vol. 171, no. 2, pp. 577-585.

[21] Toklu, B. & Ozcan, U. (2008). "A fuzzy goal programming model for the simple U-line balancing problem with multiple objectives," *Engineering Optimization* , vol. 40, no. 3, pp. 191-204.

[22] Jayaswal, S. & Agarwal, P. (2014). "Balancing U-shaped assembly lines with resource dependent task times: A Simulated Annealing approach," *Journal of Manufacturing Systems,* vol. 33, no. 4, pp. 522-534.

[23] Kashan, A. H., Rezaee, B. & Karimiyan, S. (2013) "An efficient approach for unsupervised fuzzy clustering based on grouping evolution strategies," *Pattern Recognition,* vol. 46, pp. 1240-1254.

[24] Fathi, M., Alvarez, M. J. & Rodríguez, V. (2011). "A New Heuristic Approach to Solving U-shape Assembly Line Balancing Problems Type-1," *World Academy of Science,* vol. 5, pp. 269-277.

[25] Arcus, A. L. (1966). "COMSOAL : a computer method of sequencing operations for assembly lines ; The problem in simple form," *Readings in production and operations management,* pp. 336-349.

[26] Helgeson, W. P. & Birnie, D. P. (1961). "Assembly Line Balancing Using the Ranked Positional Weight Technique," *Journal of Industrial Engineering,* vol. 6, pp. 394-398.

[27] Arnold, D. V. & Beyer, H. G. (2003). "A comparison of evolution strategies with other direct search," *Computational Optimization and Applications,* vol. 24, pp. 135-159.

[28] Ponnambalam, S. G., Aravindan, P. & Mogileeswar, G. (2000). "Multiobjective genetic algorithm for solving assembly line balancing problem," *Int J Adv Manuf Technol.,* vol. 16, pp. 341-352.

[29] Baykasoglu, A. (2006). "Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems," *Intell Manuf,* vol. 17, pp. 217-232.

[30] Ozcan, U. & Toklu, B. (2009). "A new hybrid improvement heuristic approach to simple straight and U-type assembly line balancing problems," *Intell Manuf,* vol. 20, pp. 123-136.

[31] Chiang, W. (1998). "The application of tabu search metaheuristic to the assembly line balancing problem, Annals of operation research," *European Journal of Operational Research*, vol. 77, pp. 209-227.

APPENDICES

Appendix A: GES Pseudocode

Algorithm (1+λ)-GES

Initial: $\beta, \lambda, 0 < a \leq 1, \alpha^0 > 0, \alpha^{min} > 0, G \geq 1, P_s$;

Begin

$t \leftarrow 0; G_s \leftarrow 0;\ \alpha \leftarrow \alpha^0$;

Generate an initial feasible solution $X^t and\ evaluate\ it$;

While stopping criteria are not true

For $i = 1\ to\ \lambda$

Given the parent solution X^t , apply New Solution Generator algorithm to obtain the offspring solution Y_i^t;

End for

Apply the comparison criteria between X^t and the λ generated offspring to select the best

Individual, which is known as X^{t+1} (ties are broken randomly);

if $f(X^{t+1}) < f(X^t)$

$G_s \leftarrow G_s + 1$;

End if

If $(t\ mod\ G) = 0$

$$\alpha \leftarrow \begin{cases} \alpha/a & if\ G_s/G \geq P_s \\ \max(\alpha_{min}, a \times \alpha & if G_s/G < P_s \end{cases}$$

$G_s \leftarrow 0$;

End if

$t \leftarrow t + 1$;

$\alpha^t \leftarrow \alpha$;

End While

End

Appendix B: Source Code

GES Code:

```
clc
clear
[CycleTime,TaskTime,Pi,lowerbound] = Solutions;
CycleTime
N = size(TaskTime,2);
maxnumofjob = 0;
time = cputime;

Max_iteration = 1000;
LAMBDA = 2;  %%%if theselection strategy is (miu,lambda)
then miu<lambda
beta_shape_parameter = 6;
alpha_shape_parameter  =  8;  %%%  should  be  equal  to
beta_shape_parameter
min_alpha = 0.1; %%%critical!
G_factor = 5;  %%%number of iteration to update alpha
a_factor = 0.98; %%%amount of vhange in alpha
prob_factor = 1/5; %%% the probability for dicrease or
increase
random_item_probability = 0.3; %%% critiacal!
selection_strategy  =  3;  %  1  is  (miu,lambda),  2  is
(miu+lambda)
%-----------------------------------------------------------
-----------------
new_offspring = zeros(1,N);
iteration = 0 ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   get   initial
solution
new_offspring = Revised_RPW(CycleTime, new_offspring);

smttemp = Smoothness_Index(CycleTime, new_offspring)
letemp = Line_Efficiency(CycleTime, new_offspring)
 new_offspringRPW        =        Revised_RPW(CycleTime,
new_offspring);
OBJECTIVE_VALUE = max(new_offspring);
OBJ_RPW = OBJECTIVE_VALUE;
ObjectiveValue_Of_GlobalBest = OBJECTIVE_VALUE;
Xi = [new_offspring OBJECTIVE_VALUE];
global_best = Xi;


if global_best(end) > lowerbound
    alpha_plot = [alpha_shape_parameter
        0];
    average_variation_plot = [];
```

```
    iteration = 0;
    Gs = 0;
    last_improvement = iteration;
    while iteration < Max_iteration
        offsprings = [];
        average_variation_in_population = 0;
        for ii = 1:LAMBDA   % create children
            X_id_t_1 = zeros(1,N);
            assigning_station_number = 0;
            average_variation_in_solution = 0;

            %%%%%%%%%%%%%%%%%%%%%%%%     to produce new
offspring by mutation
            max(Xi(:,1:N));
            for key = 1:max(Xi(:,1:N))
                X_id_t = find(Xi(:,1:N) == key);
                cardinality_X_id_t = size(X_id_t,2);
                beta_random_num                          =
betarnd(alpha_shape_parameter,beta_shape_parameter);    %
tolid yek adade random ba Beta_dist.
                X                =                floor((1-
beta_random_num)*cardinality_X_id_t);
                average_variation_in_solution            =
average_variation_in_solution+beta_random_num;

                %%%%%%%%%%%%%%%%%%%%%%%% choosing items
                if rand(1) <= random_item_probability
                    RAND_SEL                             =
randperm(cardinality_X_id_t);
                    X_id_t_1(X_id_t(1,RAND_SEL(1:X)))    =
key;
                else
                    sel = [X_id_t
                        TaskTime(X_id_t)];
                    sort_sel = sortrows(sel',2)'; %sort of
row no2 increasingly
                    X_id_t_1(sort_sel(1,end-X+1:end))    =
key;
                end
                %----------------------- choosing items
            end
            %-----------------------     to produce new
offspring by mutation

            if size(find(X_id_t_1 == 0),2) > 0
                maxnumofjob                              =
max(maxnumofjob,size(find(X_id_t_1 == 0),2));
                new_offspring = X_id_t_1;
                new_offspring     =     COMSOAL(CycleTime,
new_offspring);
```

```
                %%%%%%%%%%%%%% to arrange the batch number.
[1 2 2 5 7] changes to [1 2 2 3 4]
                randomsequence = [];
                jey = zeros(1,N);
                eff = 1;
                child = new_offspring;
                for g = 1:max(child)
                    if child == realmax
                        break
                    else
                        child(find(child == min(child)))
= eff;
                        jey(find(child == min(child))) =
eff;
                        child( :,find(child == eff)) =
realmax;
                        eff = eff + 1;
                    end
                end
                new_offspring = jey;
                for i=2:(size(Pi)-1)
                end
                end

                OBJECTIVE_VALUE = max(new_offspring);
            else
                new_offspring = X_id_t_1;
                OBJECTIVE_VALUE = max(new_offspring);
            end

            offsprings = [offsprings
                new_offspring  OBJECTIVE_VALUE];
            average_variation_in_population             =
average_variation_in_population+average_variation_in_solu
tion/key;
        end
sorted = sortrows([Xi
                offsprings],N+1);
            ran = find(sorted(:,end) == sorted(1,end));
            SmoothnessIndexMatrix = [];
            for k = 1:size(ran,1)
                new_offspring = sorted(k,1:N);
                SmoothnessIndex                         =
Smoothness_Index(CycleTime, new_offspring);
                SmoothnessIndexMatrix                   =
[SmoothnessIndexMatrix
                    SmoothnessIndex];
%%%%%%%%%%%%%%%%%%%% Success rule %%%%%%%%%%%%%%%%%%%%
        if min(offsprings(:,N+1)) < Xi(:,N+1)
```

```
            Gs = Gs + 1;
        end
        if mod(iteration,G_factor) == 0 && iteration>0
            if Gs/G_factor >= prob_factor
                alpha_shape_parameter                    =
alpha_shape_parameter/a_factor;
            else
                Gs/G_factor < prob_factor;
                alpha_shape_parameter                    =
max(min_alpha,alpha_shape_parameter*a_factor);
            end
            Gs = 0;
        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        iteration = iteration + 1
        alpha_plot = [alpha_plot [alpha_shape_parameter
            iteration]];
        average_variation_plot  =  [average_variation_plot
[average_variation_in_population/LAMBDA
            iteration]];
    end
end

iteration
global_best;
new_offspringRPW;
for i=1:size(new_offspring,2)
    new_offspring(i) = global_best(i);
end
OBJ_RPW

ObjectiveValue_Of_GlobalBest
LineEfficiency        =        Line_Efficiency(CycleTime,
new_offspring)
SmoothnessIndex       =       Smoothness_Index(CycleTime,
new_offspring)
```

Comsoal Code:

```
while (~isempty(find(new_offspring == 0)))
    new_offspring;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%     get  zero
columns
    n = 0;
    WithoutPredecessors = [];
    for i = 1:size(TempPi,2)
        if isempty(find(TempPi(:,i), 1))
```

```
            if new_offspring(i) == 0
                n = n + 1;
                WithoutPredecessors(n) = i;
            else
                TempPi(i,:) = 0;
                TempPi(:,i) = 0;
            end
        end
    end
    for i = (size(TempPi,2)):-1:1
        if isempty(find(TempPi(i,:), 1))
            if new_offspring(i) == 0
                n = n + 1;
                WithoutPredecessors(n) = i;
            else
                TempPi(i,:) = 0;
                TempPi(:,i) = 0;
            end
        end
    end

if RandomActivity == 0
        RandomAct = unidrnd(size(WithoutPredecessors,2));
        RandomActivity = WithoutPredecessors(RandomAct);
    end
TempPi;
    RandomActivity;

    if ~isempty(find(Pi(:,RandomActivity)))%    Just for
first column

        if ~isempty(find(Pi(RandomActivity,:)))%    Just
for last row

            Predecessors = find(Pi(:,RandomActivity));
            PredecessorsStations                        =
new_offspring(Predecessors);

            Successors = find(Pi(RandomActivity,:));
            SuccessorsStations                          =
new_offspring(Successors);
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
get Idle Time
            MSuccessorsStations                         =
max(SuccessorsStations);
            MPredecessorsStations                       =
max(PredecessorsStations);

            minSS = min(SuccessorsStations);
            minPS = max(PredecessorsStations);
```

```
            if MSuccessorsStations == 0
                Station = MPredecessorsStations;
            elseif minSS ==0
                Station = MPredecessorsStations;
            elseif MPredecessorsStations == 0
                Station = MSuccessorsStations;
            elseif minPS ==0
                Station = MSuccessorsStations;
            else
                Station                                    =
min(MPredecessorsStations,MSuccessorsStations);
            end

hile new_offspring(RandomActivity) == 0
                StationActs   =   find(new_offspring   ==
Station);
                StationTime = 0;
                for i=1:size(StationActs,2)
                    StationTime    =    StationTime    +
TaskTime(StationActs(i));
                end
                IdleTime = CycleTime - StationTime;
                if IdleTime - TaskTime(RandomActivity) >=
0
                    new_offspring(RandomActivity)       =
Station;
                else
                    Station = Station + 1;
                end

            end

        else
            Station = 1;
            while new_offspring(RandomActivity) == 0

                StationActs   =   find(new_offspring   ==
Station);
                StationTime = 0;
                for i=1:size(StationActs,2)
                    StationTime    =    StationTime    +
TaskTime(StationActs(i));
                end
                IdleTime = CycleTime - StationTime;
                if IdleTime - TaskTime(RandomActivity) >=
0
                    new_offspring(RandomActivity)       =
Station;
```

```
                else
                    Station = Station + 1;

                end
            end
        end
    else
        Station = 1;
        while new_offspring(RandomActivity) == 0
            StationActs = find(new_offspring == Station);
            StationTime = 0;
            for i=1:size(StationActs,2)
                StationTime     =     StationTime     +
TaskTime(StationActs(i));
            end
            IdleTime = CycleTime - StationTime;
            if IdleTime - TaskTime(RandomActivity) >= 0
                new_offspring(RandomActivity) =  Station;
            else
                Station = Station + 1;

            end
        end
    end
    Station;
    new_offspring;
    TempPi(RandomActivity,:) = 0;
    TempPi(:,RandomActivity) = 0;
    TempPi;
```

Revised RPW:

```
if max(W) < W_Max
Predecessors = find(Pi(:,Selected_Item));
        PredecessorsStations                                =
new_offspring(Predecessors);

        Successors = find(Pi(Selected_Item,:));
        SuccessorsStations = new_offspring(Successors);
        MSuccessorsStations = max(SuccessorsStations);
        MPredecessorsStations                               =
max(PredecessorsStations);

        minSS = min(SuccessorsStations);
        minPS = max(PredecessorsStations);

        if MSuccessorsStations == 0
            Station = MPredecessorsStations;
        elseif minSS ==0
```

```
            Station = MPredecessorsStations;
        elseif MPredecessorsStations == 0
            Station = MSuccessorsStations;
        elseif minPS ==0
            Station = MSuccessorsStations;
        else
            Station                                          =
min(MPredecessorsStations,MSuccessorsStations);
        end


        Station;
        new_offspring;
        while new_offspring(Selected_Item) == 0
            StationActs = find(new_offspring == Station);
            Station_Time = 0;
            for j=1:size(StationActs,2)
                Station_Time     =     Station_Time     +
TaskTime(StationActs(j));
            end
            Idle_Time = CycleTime - Station_Time;
            if Idle_Time - TaskTime(Selected_Item) >= 0
                new_offspring(Selected_Item) =  Station;
                TW(Station)=         TW(Station)         -
TaskTime(Selected_Item);
            else
                Station = Station + 1;
            end

        end
    else
        if TW(1)- TaskTime(Selected_Item)>= 0

             new_offspring(Selected_Item) = 1;
             TW(1) = TW(1) - TaskTime(Selected_Item);
        else
            new_offspring(Selected_Item) = 2;
            TW(2) = TW(2) - TaskTime(Selected_Item);
        end

    end
    W(Selected_Item) = 0;
    new_offspring;
    TW;
    End
```

Selection Operator (Smoothness index):

```
global TaskTime;
```

```
StationTimeMatrix = [];
MaxStationTimeMatrix = 0;
IdleTimesSquare = 0;
IdleTimesSquare2 = 0;
Ui = [] ;
Ut = 0 ;
aver = 0 ;
W = [];
VAR = 0 ;
for i = 1:max(new_offspring)
    StationActs = find(new_offspring == i);
    StationTime = 0;
    for j = 1:size(StationActs,2)
        StationTime        =        StationTime        +
TaskTime(StationActs(j));
    end
    StationTimeMatrix = [StationTimeMatrix StationTime];
    MaxStationTimeMatrix = max(StationTimeMatrix);

end
Ui = StationTimeMatrix;
for k = 1:size(StationTimeMatrix,2)
IdleTimesSquare = IdleTimesSquare + (MaxStationTimeMatrix
- StationTimeMatrix(k))^2;
end
for k = 1:size(StationTimeMatrix,2)
  Ui(k)=   Ui(k) / MaxStationTimeMatrix;
    end
Ut = sum(Ui);
aver = Ut/max(new_offspring);

for k = 1:size(StationTimeMatrix,2)

    W(k) = (Ui(k) - aver )^2;

end

SmoothnessIndex = (IdleTimesSquare)^0.5;
 VAR = (sum(W)/max(new_offspring))^0.5
```