universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Titel of The Master's Thesis

„Balancing Assembly Lines with Given Number of Machines per Workstation"

verfasst von / submitted by

Isil Basak Akgül

angestrebter akademischer Grad / in partial fulfillment of the requirements for the degree of

Master of Science (MSc)

Wien, 2017 / Vienna, 2017

Studienkennzahl lt. Studienblatt / degree programme as it appears on the student record sheet:	A 066 920
Studienrichtung lt. Studienblatt / Degree programme as it appears on the student record sheet:	Masterstudium Quantitative Economics, Management and Finance
Betreuet von / Supervisor:	O. Univ. Prof. Dipl.-Ing. Dr. Richard F. Hartl
Mitbetreut von / Co-Supervisor:	Dr. Margaretha Gansterer

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Die Arbeit wurde bisher weder in gleicher oder ähnlicher Form verfasst, noch einer anderen Prüfungsbehörde vorgelegt.

Wien, April 2017

Isil Basak Akgül

Acknowledgement

I would like to express my sincere thanks to my supervisor O. Univ. Prof. Dipl. - Ing. Dr. Richard F. Hartl for giving me a chance to write my thesis at his department and leading my research, Dr. Margaretha Gansterer for her valuable advices, reviews and patience through all these months.

I would also like to thank to my dearest mother for her endless support, my teacher Erinc for his inspiring will to teach, my boyfriend for his continuous motivation during my study and all of my teachers, professors who dedicated their lives for education and shaping the future.

Table of Contents

List of Tables

List of Figures

Abbreviations

ALB	Assembly Line Balancing
ALBP	Assembly Line Balancing Problem
GALBP	Generalized Assembly Line Balancing Problem
SALBP	Simple Assembly Line Balancing Problem
RPW	Ranked Positional Weight Rule
MA	Maximum Time Rule
MI	Minimum Time Rule
AllSuc	Descending Number of All Successors Rule
ImmSuc	Decreasing Number of Immediate Successor Rule
GA	Genetic Algorithm
DE	Differential Evolution Algorithm
DE-C	Modified Differential Evolution Algorithm
RALBP	Robotic Assembly Line Balancing Problem
COMSOAL	Computer Method of Sequencing Operations for Assembly Lines
GGA	Grouping Genetic Algorithm
SALBP-1	Simple Assembly Line Balancing Problem Type-1
SALBP-2	Simple Assembly Line Balancing Problem Type-2
SALBP-E	Simple Assembly Line Balancing Problem Type-E
SALBP-F	Simple Assembly Line Balancing Problem Type-F
SALBP-1M	Simple Assembly Line Balancing Problem Type-1 with Given Number of Machine Limit
ACO	Ant Colony Optimization

Zusammenfassung

Fertigungsbänder spielen eine wichtige Rolle in den Produktionsplanungssystemen und es werden vieleLösungsmethoden erzeugt, um die Effizienz und Leistung zu steigern.Das Ziel dieser Arbeit ist es, Fließbandfertigungssysteme zu verstehen und Methoden zu finden, umBandabgleich Probleme optimal lösen zu können. Es werden detaillierte Informationen über Bandabgleichskonzepte angeführt und es wird umfassende Literatur über diverse Lösungsmethoden aufgearbeitet. In der Fallstudie werden spezielle Bandabgleich Probleme, bei denen die Anzahl der Maschinen pro Arbeitsstation beschränkt ist, analysiert. Verschiedene Single Pass heuristische Regeln und vorgeschlagene genetische Algorithmen werden auf 2 verschiedenen Instanzarten, angewendet. Optimale Problemlösungen werden mithilfe eines GAMS-Model gefunden und die vorgeschlagenen Algorithmen mit den Resultaten von Pitaaso und Sethananverglichen (2015). Die rechnerischen Untersuchungen zeigen, dass die vorgeschlagenen genetischen Algorithmen die optimale Lösung für 19 Fälle von 20 Fällen für SALBP-1M erreichen und die Resultate besser sind als die von Pitakaso und Sethanan (2015) vorgeschlagenen Algorithmen."

Schlüsselwörter: einfaches Fertigungsband Problem, Maschinen Begrenzung, Priorität basierte heuristische Regeln, genetischer Algorithmus

Abstract

Assembly lines play a significant role in production planning systems and many solution methods are-generated in order to improve their efficiency and performance.The aim of this study is to understand assembly line systems and solution methods to balance assembly line problems. Detailed information is given about concepts of assembly line balancing and a comprehensive literature review is provided for various solution methods. In a case study, the simple assembly line balancing problem with a given number of machine limits is analyzed (SALBP- 1M). Various single pass heuristic rules and a genetic algorithm are applied to two different test instances. Optimal results of the problems are solved by a GAMS model and the results of the proposed algorithms are compared with results presented by Pita-kaso and Sethanan (2015). The computational study proves that the proposed genetic algorithm reaches optimal solutions for 19 instances out of 20 instances for SALBP-1M and produces better results than the algorithm of Pitakaso and Sethanan (2015).

Keywords: simple assembly line problem, machine limitation, priority based heuristic rules, genetic algorithm

1. Introduction

Assembly lines are fundamental for mass production and manufacturing systems in all areas. ALB is essential to improve efficient usage of resources such as raw materials, machines, labor and to increase production capacity by maximizing the productivity and efficiency.

ALBPs are complex optimization problems. In ALB, a group of tasks is orderly assigned to workstations considering the precedence constraints, cycle time restrictions and predetermined optimization measure in order to find optimal task assignments according to the chosen objective such as cost reduction, minimization of workstations, increasing line efficiency.

Due to globalization and increased competition, companies have to adapt to the fast change at production systems by developing productive methods and using resources efficiently. GAs have been developed to speed up optimization processes by inspiring from biology mechanisms. Nowadays, GA is a metaheuristic optimization method that can be optimized via computer-based applications.

There are many studies for ALBP due to technological improvements at production systems. Various exact, heuristic and metaheuristic methods are presented to find an optimal solution for ALBPs and these methods are also combined with resource constraints to reach better solutions. Agcak and Gokcen (2005) present a method to solve resource constrained ALBP which minimizes amount of used resources with maximum amount of workstations and given cycle time, they consider the minimization workforce and number of machines; Bautista and Pereira (2011) propose a bounded dynamic programming based method for time and space constrained ALBP with predetermined cycle time considering space availability for machines in order to decrease the number of workers; Chica, Cordon and Damas (2011) improve GA with multiple objectives for 1/3 variant of the time and space ALBP for minimization of amount and space of the workstations with given cycle time by inspiring from non-dominated GA on coding structure, genetic operators, diversification system with pareto fronts.

In this study, single pass heuristic methods and GA are proposed to balance simple assembly line problem with given number of machine limits (SALBP-1M) by objecting the minimization of number of workstations. Different single pass heuristic methods and the proposed GA are applied to two different problems as Pitakaso-1 and Pitakaso-2 that are obtained from research of Pitakaso and Sethanan (2015). Pitakaso-1 problem consists of 36 jobs and 6 different machines and Pitakaso-2 comprises 52 jobs and 5 different machines; proposed optimization methods are used to solve both problems with two different machine limit types and five different cycle times for each machine limit; so, 20 instances are analyzed to find an optimal solution for SALBP-1M.

The study is organized as following: in Chapter 2, general concepts of ALB is mentioned including the main objectives and classifications of ALB; Chapter 3 gives detailed information for solution methods of ALBP and a comprehensive literature review for heuristic solution methods and GA; Chapter 4 de-

scribes the mathematical formulation of SALBP-1M, the summary of related research and the results of the proposed model. Finally, Chapter 5 explains the conclusion drawn from this study.

2. The Concepts of Assembly Lines and Assembly Line Balancing

2.1 Assembly

Assembly is a manufacturing process for creating a finished product by putting together components or subassemblies. There are many different operations at the assembly process; these operations can be collected under the four main categories: Feeding, handling, insertion and check (Acar & Estas, 1991).

Feeding is operation that material or semi-finished product is taken away from the storage (station etc.) for assembly process.

Handling is operation which material is transferred to assembly station.

Insertion operation is installing various parts together.

Check is examination of the components and controlling of the tasks during the assembly process.

Manufacturing system and assembly work evolved according to needs and demands in the production over the years. During the industrialization period, the manufacturing system started to change in order to produce in high volumes, faster and lower cost. In 1913, Henry Ford revolutionized the production system by inventing moving assembly lines. Assembly line system was structured such that while some conveyors moved the assemblies along and the other conveyors constantly moved parts into position so that labours could insert the parts without carrying them (Ford Motor Company website, 2015). Each worker had defined tasks and assigned workstation.

Assembly line is a flow-oriented production system which consists of orderly located stations, connected by material handling system, performing different operations in work pieces which are moved from one station to another. At the end of this process, unfinished and semi-finished parts release from the system as finished products (Becker & Scholl, 2006a).

2.2 Assembly Line and Assembly Line Balancing

Products consist of combination of multiple parts that are put together in specific orders during assembly process. Assembly lines are production systems comprised of sequential stations, aligned along conveyor belt or other transportation system, to operate a set of tasks to convert materials to the finished products. Unfinished product moves along the line by material handling system from one station to another and launches out the line as finished products (Becker & Scholl, 2006a).

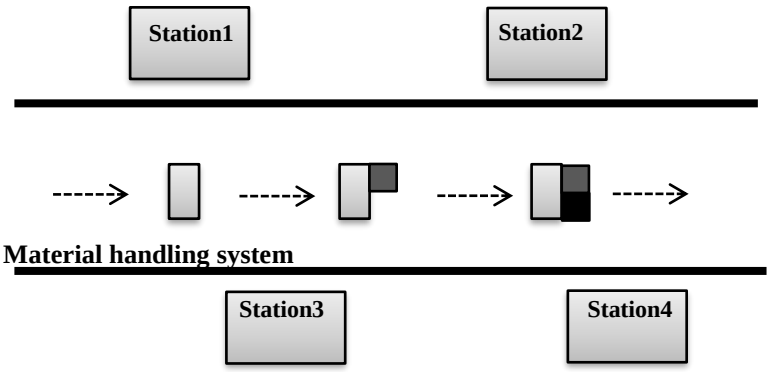


Figure 2.1 Concept of Assembly Line

ALB is a decision process of assigning tasks to the workstation to perform some objectives with respect to restrictions. Line balancing tries grouping the facilities or workers in an efficient in order to reach an optimum or most efficient balance of the capacities and flows of the production or assembly processes. Task allocation of each worker was succeeded by ALB to improve efficiency and productivity of the line (Kumar & Mahto, 2013).

The installation of an assembly line is a medium or long term decision and usually requires large capital investments, so balancing and designing of the assembly line is significant process to produce as efficiently as possible. In addition, balancing a new system, a running one has to be re-balanced at certain times or after any changes in the production system or in the production program have occurred. Due to the long term effect of balancing decisions, the objectives which are used have to be carefully determined by taking into account of all the strategic aims of the enterprise (Becker & Scholl, 2006a).

In the beginning, assembly lines were used for producing in high volumes and low variety of products. Production companies worked on the individualization of products because of which efficient flow line system for low volume of products have been improved and a modern terminology of mass customization has been presented (Hjálmarsson&Viktorsson). The fast qualitative and quantitative improvements in the market demands leaded manufacturers to look for new and more efficient methods for managing the assembly lines on behalf they can minimize the idle time and improve the flexibility of the production system. Thus, the designing of ALB system is still improving in order to satisfy market trends and changing needs of the customers.

From an economic point of view cost and profit related objectives should be also taken into account during the improvement of the ALB system. Assembly lines were developed for a cost efficient mass-production of standardized products, designed to take advantage of a high specialization of labour and the involved learning effects (Hjálmarsson&Viktorsson). Minimizing the cost of the line, long term investment cost and short term operating cost decrease the prices of produced goods, give competitive

advantages to manufacturer in the market by responding to the demands of the consumers with lower prices.

2.3 Basic Concepts of Assembly Line Balancing

Task is the smallest, indivisible work piece element of the total workload in an assembly line process. Tasks are practical and lowest segment of total work. Besides, task is the smallest rational work elements, which is not possible to be divided by two and more workers or stations (Baybars, 1986b). For example, we can define five different tasks for drilling five different holes; when it comes to ALB, the rational task is the drilling five different holes as grouped tasks.

Workstation is a location on the flow line where workers or robotic operators or machinery performs the tasks as assembling parts or processing operations. Stations can be categorized in two main types as *open station* and *closed station*. For closed stations, it is not possible for operators to cross the boundaries of station. On the other hand, open station is more flexible and boundaries can be crossed by adjacent operators as so not to interfere tasks of each workstation (Erel&Sarin, 1998).

Task processing time (task time) is the time required for performing an operation/task. The total is processing time is sum of the all tasks at assembly line process (Baybars, 1986b).

Workstation time (station time) is the total workload of a station. It is the sum of processing times of the tasks that are assigned to a particular station. Workstation time must not exceed cycle time of the assembly line (Boysen, Fliedner&Scholl, 2008)

Cycle time is the fixed time between the launches of two consecutive products from the assembly line. In other words, it is maximum amount of time which product can be processed in each station. It represents a particular time that all tasks are operated by workers in each station. The cycle time is predetermined considering the production rate (Scholl, 1999). Cycle time cannot be smaller than the largest task processing time. Although all tasks at the station are already processed, station time could not fill in cycle time.

Workstation idle time is the positive difference between the cycle time and the workstation time. The sum of idle times of all stations is called *total idle time* which related measure of the efficiency of assembly line design (Erel&Sarin, 1998). Tasks must be assigned to the stations in order to minimize the idle time of each workstation, hence the average assembly time for each product decreases to minimum.

Balance delay is the one of measurements for the efficiency of the line. Balance delay results from unbalanced assignments of the tasks to the workstations. If the tasks are assigned properly and all workstation time is equal to each other, the assembly line has effective work flow and no delay occurs

in the system. It is also referred as the ratio of average idle time at the workstations over cycle time (Kumar & Mahto, 2013).

Figure 2.2 Perfectly balanced n-station assembly line

Precedence relations represents in which order tasks must be performed in the task sequence (Becker & Scholl, 2006a).

Precedence graph (diagram) is a graphical representation of the precedence relations between the tasks. It contains nodes representing each task and arcs visualizing the precedence relations of the tasks; the processing time of each task is shown at right top of each node. Figure 2.3 shows precedence diagram with ten tasks; task 5 can only be operated after the completion of task 1 and task 3, which are the *immediate predecessors* of task 4, and the *indirect predecessor* task 2 (Becker & Scholl, 2006a).

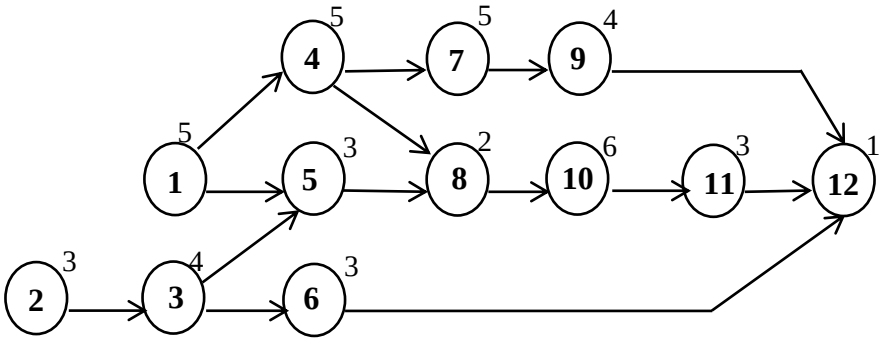


Figure 2.3 Precedence diagram

Precedence matrix is transformation of the precedence graph into a matrix. It shows the direct precedence relations between the tasks; if the task at row is immediate predecessor of the task at column of the matrix, the crossing point of these tasks are marked with one,0 is assigned the other points. Table 2.1 shows the precedence matrix of precedence diagram at figure 2.3 (Aydogdu, 2005).

Tasks	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	1	1	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	0	0	0	0	0	0
4	0	0	0	0	0	0	1	1	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	1	0	0	0
8	0	0	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	0	0

Table 2.1 Precedence matrix

2.4 Main Objectives of Assembly Line Balancing

Assembly lines are important part of the mass production system. Assembly lines can be classified in different types regarding their technical qualifications such as number of products, line control and layout, workflow, and etc. Although there is a huge diversity of assembly lines, the basic system of different assembly lines operates based on same procedure.

The main purposes of the ALB are as follows (Kumar & Mahto, 2013):

- To maintain balanced flow of material between the stations
- To use minimum amount of material for processing of the tasks
- To equally divide the workloads among workers to the assembly line
- To use optimum mix form of automation and manual assembly
- To utilize from the machinery and equipment capacities at maximum level
- To minimize the total amount of idle time and flow time
- To maximize the line efficiency
- To optimize the number of workstations
- To minimize the balance delay time and equally distribute the balance delay among all stations
- To minimize the total cost of ALB and improve productivity

It is not possible to satisfy the all objectives of the ALB at the same time and so the main aim is to reach optimum solution considering the constraints of assembly lines. During the solution optimization and improving productivity, the factors which influence the total cost of the line must be taken into consideration. The building of assembly line is long-term decision which involves large capital investments as long-term investment cost and short-term operating cost that are depending on the cycle time and the number of workstations. The minimization of the total cost depends on the decreasing the inventory costs, labour costs, material cost, setup costs, machinery costs (Scholl, 1999).

Besides performance and cost related purposes, social goals should be considered to have a better work environment by providing job enlargement, increasing responsibilities of an operator, allocating less monotonous tasks to an operator, ensuring sufficiently equipped and safe work plant.

2.4.1 Constraints Affecting Assembly Line Balancing

2.4.1.1 Basic Constraints

Cycle time is determined by given net production objective, gross operation percent and tolerance percent. Workstation time cannot exceed the cycle time.

Precedence relations are the task sequence shows in which order tasks must be operated. To operate the task, direct and indirect predecessors of this task must be processed before. Task assignments to the workstations based on precedence relations between the tasks.

2.4.1.2 Side Constrains

Position related constraints; in some cases, tasks can require a certain position of the work pieces thus it might not be possible to move the work pieces due to economic or physical reasons (e.g., heavy items such as a ship, drilling machine, etc.) (Wang&Wilson, 1986).

Workstation related constraints; in some cases, special machines or equipment needing the implementation of the certain tasks are only possible in one or a few workstations, cannot be transferred to another place (Becker & Scholl, 2006a).

Station workload related constraints; in some cases, some of the workstation times are required to be smaller than cycle time. This constraint is applied especially for first station to decrease the effects of the delay which may occur at the beginning process of the assembly line (Tanyas&Baskak, 2003).

Operator related constraints; in some cases, specific operators with different professional skills and education must do certain tasks depending on their complexity (Agrawal, 1985).

Task related constraints; in some cases, some tasks must be operated in the same or different workstations, these constraints are referred as positive or negative zoning constraints. Positive zoning constraints are in respect to use of common equipment, machines, tool or common processing conditions

for operating tasks, so that these tasks must be assigned to same workstation or successive connected workstations. Some tasks must not be performed at the same workstation regarding negative zoning constraints. For example, milling and measuring operations must not be performed at same workstation (Tanyas&Baskak, 2003).

2.5 Classification of Assembly Line Balancing

2.5.1 Automation Level Characteristics

Level of automation of assembly line can be categorized under three main types: Manual, automatic/ semiautomatic and robotic assembly lines. These types can be operated separately or combined to perform together. Firstly, the operation must be analysed considering the economy and conformity of the assembly line (Baskak, 2005).

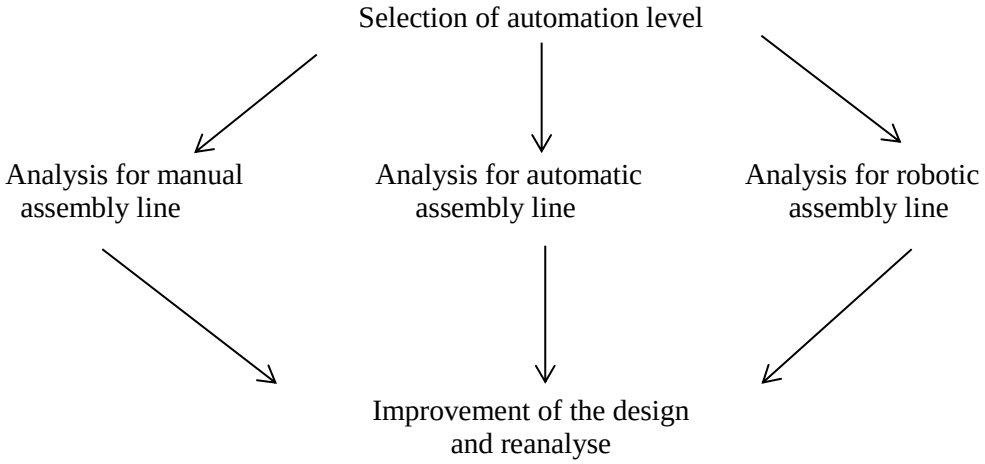


Figure 2.4 Designing steps for assembly analysis

2.5.1.1 Manual Assembly System (Baskak, 2005)

Basic tools and equipment are used in manual system for assembling small lots. Operator must have specific and important qualifications at manual system; operator can move work pieces from one place to another and install the work pieces to original place efficiently. On the other hand, there are some cases which it is hard and time-consuming for operators to replace the work pieces; such as very tiny pieces, very huge and heavy pieces, pieces which are hard to hold due to their sharp surface, brittleness.

The manual assembly system is commonly used in production of automotive, electric motor, camera, furniture. They can be designed as parallel or serial lines with single or multiple stations.

Figure 2.5 Manual assembly line *(www.pcstats.com)*

2.5.1.2 Automatic or Semiautomatic Assembly System

Specific transportation mechanisms are used at automatic assembly system. Automatic assembly systems operate on parallel lines and they have qualifications which make assembly process more efficient. The most important qualification is that instead of operators, automatic machines perform the mechanical assembly process. Thus, labour can be used on more specific operations which are not uneconomic and easy to automatize. Other most important advantages are as shown as below (Baskak, 2005):

- Assembly cost decreases.
- Efficiency of assembly line increases.
- Better qualified and consistent products can be produced.
- Safer working environment is provided by assigning operators to less dangerous operations.

Those must be considered at automatic assembly systems are as follows (Tanyas&Baskak, 2003):

i. The over loading and routing of work pieces must be minimized.
ii. Automatic systems should consist of basic motions of machine parts.
iii. The mobility of the fasteners must be limited.
iv. The fasteners like screw, bolt, and rivet must not be used.
v. Assembly process must comprise of breakout, assembly, labelling and drilling operations.
vi. Fasteners must have standard type and size

Figure 2.6 Automatic assembly line *(www.globalmarket.com)*

Automatic assembly systems consist of transportation system which transfers work pieces between stations, automatic assembly tools, control stations which check vacant stations used by operators to perform complex assembly operations and performance of operations. Automatic assembly machines are designed considering the transportation system used in the production process. For instance; the work pieces move along on straight line at straight (serial) assembly line machine and transportation follows rotational line at rotary assembly machines. For both type of machine, transportation could be discrete or continuous. The main problem of automatic assembly system is loss of production due to idleness of machines. At manual systems, operators can remove defective work pieces easily and less loss of production occurs. Automatic machines wait until the defective pieces are found and fixed, so it creates idle time at the production process. Therefore, high qualified pieces must be chosen for automatic assembly system (Tanyas&Baskak, 2003):

Automatic assembly systems are useful and profitable for production in high volume, so economic efficiency of system should be considered as important effect.

2.5.1.3 Robotic Assembly System

One or a pair of robot performs on a station or multiple stations; stations can be on parallel lines and serial lines. A robot that positions, fits, and assembles components or parts and aligns the finished product to perform. Robots are useful for the most demanding and complicated production processes.

Robots for simple assembly and material handling work via simple mechanical handle under control of computer based system and they are most common type of robots in the industries. Robotic assembly

system has various qualifications for capacity, ability and size. Simple assembly robots operate by different features and tools, such as x-y-z linear moving, electro-optical position centring system, screw driver. Specific equipment is with signal feedback system, servo-driven and computer controlled for torque and power sensitivities; these robots are used in small assembly applications, such as adding a small component of the electrical circuit board (Tanyas&Baskak, 2003).

Robotic assembly system has four main parameters:

- The weight of carried work piece
- Repeatability for assembly process
- Processor speed
- Complexity of operation

A robotic assembly system is more flexible and cost saving than an automatic system. Robotic assembly system ensures a variety of benefits including decreased labour, production time, ergonomic issues, as well as increased quality, efficiency and throughput (Motion Controls Robotics website, 2015).

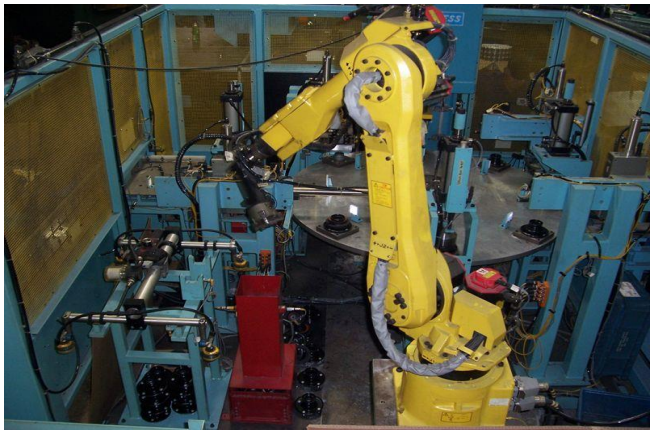


Figure 2.7 Robotic assembly line *(www.caeweb.com)*

2.5.2 Product Characteristics

2.5.2.1 Single Model Assembly Line

Assembly lines are used for large volume production of only a single product and workers work on same product. The station workload and task processing times are constant. The design of single model assembly line is simple; the task processing time and precedence relations are shown by a single precedence graph. Nowadays, the single model assembly line is not as useful as before due to customer demands and competitive business environment (Ullah et al., 2014).

2.5.2.2 Mixed Model Assembly Line (Thomopoulos, 1970)

Slightly differentiated models from same product parent are assembled in the mixed model assembly line. Each model nearly has same production process based on basic operations, some models can have different process and due to this, these different models can have larger or smaller workstation time than the other models. So, work load distribution may not be equal in each station and this can cause to non-smooth production.

The precedence relation is shown by consisting of precedence diagram of all models in the production. The same tasks are assigned to same stations to reduce set up time of different tasks of different models based on precedence relation of mixed model assembly line. Hence, task assignment to the station is important procedure at mixed model ALB to allocate the tasks of different models in a way that precedence relations and objectives are satisfied. Model sequencing is also important procedure; in each station, the different models of product are properly sequence for assembling.

2.5.2.3 Multi Model Assembly Line

In multi model assembly line, different products are produced and assembling of the different models are in batches. The different products are produced in batches without mixing with each other. After production of the batch of one model, the organization of station is changed and arranged according to the need of the next batch. Assembling in batched form can reduce of the time and set up cost but inventory cost may increase too. The lot-sizing problem arises for different batches of the models in multi model assembly line (Boysen, Fliedner&Scholl, 2008).

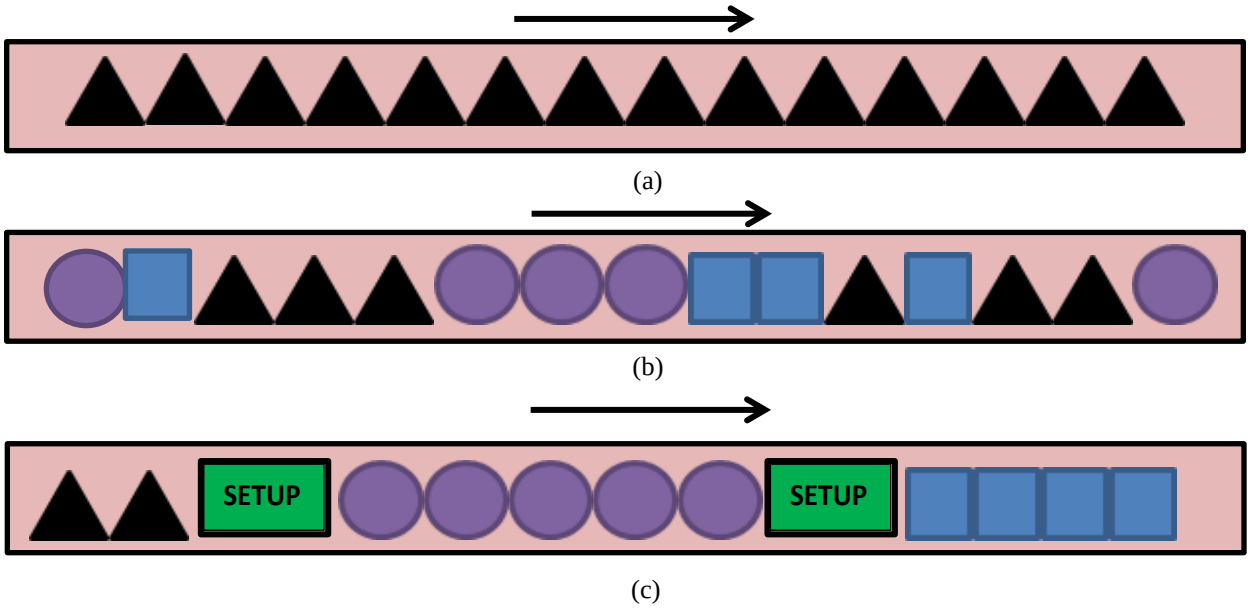


Figure 2.8 Assembly lines for (a) single model, (b) mixed model, (c) multi model (Kumar&Mahto, 2013, pp.30)

2.5.3 Workflow Characteristics

2.5.3.1 Paced Assembly Line

In paced assembly lines, all stations are supposed to have same cycle time and the work pieces are transferred to one station to another when tasks of station finish. All stations never pass over the cycle time; although the tasks at the station finish earlier than cycle time, they wait to move to next station until cycle time is over. In such case, station has an idle time which affects the efficiency of assembly line. In paced assembly line, finished work pieces move from one station to another by a conveyer belt. The minimum value of cycle time is determined based on the maximum workload of the stations on the assembly line. The production rate of paced assembly lines is fixed and depends on cycle time of the line and finished work pieces which are transferred by conveyor from one to another station without buffers between stations (Ullah et al., 2014).

2.5.3.2 Un-paced Assembly Line

In un-paced assembly lines, the work pieces are transferred to one station to another when they are finished on the station without waiting for that cycle time is over. Un-paced assembly line can be categorized in two types according to the movement of the finished work pieces from the stations.

Synchronous un-paced assembly lines transfers finished work pieces from one station to another after a fixed time simultaneously, therefore no buffers arise between stations. In *asynchronous un-paced assembly lines*, cycle time is different between the stations and workload of the station is equal to cycle time of this station. If all tasks of the station have already processed, it must wait till the successor becomes eligible and idle time may occur. So, buffers can be used to decrease the idle time in such cases (Urban&Chiang, 2006).

2.5.4 Layout Characteristics

Layout and design of the assembly line has effect on productivity of the line. Layout of assembly line can be chosen based on the location of the line and product to be produced. Assembly lines can be categorized as serial (straight) lines, parallel lines, U- shaped lines and two-sided assembly lines.

2.5.4.1 Serial Assembly Lines

Stations are aligned in a serial route by the sides of the conveyor in serial assembly lines. Work pieces are entered to assembly line and moved to first station. They are transferred from one to another station on the assembly line, until their processing is over and run through last station. The cycle time is determined based on the station with maximum workload. In serial assembly lines, lead-time is taken into

account for deciding cycle time as well. Therefore, a workload and dead time of the stations are included while calculating the cycle time (Ullah et al., 2014).

Serial assembly lines are used frequently due to their basic systems, easy place-ability, service ability, adapting practically to conveyor systems and effects on decreasing the expenditure.

2.5.4.2 Parallel Assembly Lines

In parallel assembly lines, the workload is distributed between stations. In case that cycle time of assembly line is more than expected, the workload of station with maximum workload can be split by paralleling this station for decreasing the cycle time of assembly line. Same group of tasks are assigned to duplicated parallel stations. The implementation of these lines increase flexibility and decrease the waiting time in the line by moving work pieces to the duplicated parallel workstations (Becker & Scholl, 2006a).

2.5.4.3 U-shaped Assembly Lines

In U-shaped assembly lines, the workstations are aligned among a "U" form line and the tasks are assigned to stations by moving forward and backward through the precedence graph. So, work pieces can be operated at different positions on the assembly line during same cycle (Erel, Sabuncuoglu &Aksu, 2001).

The U-shaped assembly lines are used commonly in assembly production system to operate tasks on different positions more efficiently. It provides a better balance of workstation loads by assigning workers flexibly along the line and improving possibilities on assigning tasks to workstations. In some situations of competitive market, U-shaped assembly lines easily adapt to changing conditions and improve the performance measures (Miltenburg&Wijngaard, 1994). Hence, U- shaped lines are chosen more than traditional serial lines.

2.5.4.4 Two -sided Assembly Lines

Two- sided assembly lines are designed to produce large sized and heavy, standardized products such as automobiles, buses and trucks. Tasks can be operated at both side of the line and more than a machine and worker can work together on a work pieces at the station simultaneously. Thus, there are less number of stations required for assembly process at two-sided lines (Lee, Kim &Kim, 2001).

Bartholdi (1993) proposed two-sided assembly line in which workstations are placed opposite to each other as the left and right side of the line. In some cases, idle time can occur due to precedence constraints between two opposite stations. There are some advantages of two-sided assembly lines (Bartholdi, 1993):

- The length of assembly line may be shorter than one-sided assembly line

- It may decrease the workers' movement, material handling cost, setup time and throughput time
- It can reduce the cost of equipment, tools and fixtures as well.

2.5.5 Task Time Characteristics

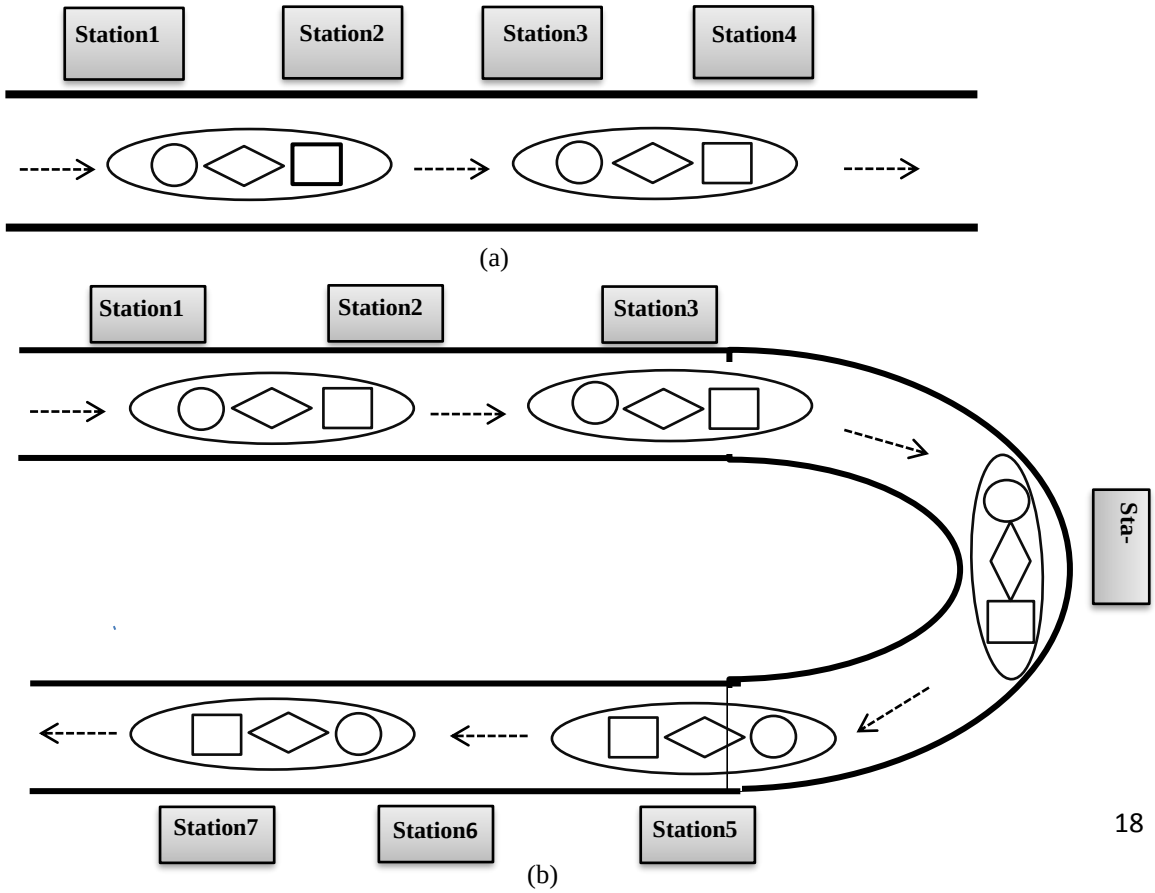
Task time can be categorized as deterministic task time, stochastic task time and dynamic task time. The characteristics of task processing times depend on the nature of tasks and operators.

2.5.5.1 Deterministic Task Time

When the expected variation of task time is enough small, e.g. in case of assembly lines with simple tasks or highly reliable automated stations, the task time is accepted as *deterministic*. Task time is supposed not to change during the assembly process and it is assumed as a fixed variable in deterministic approaches that simplify the solution for the assembly lines (Johnson, 1983).

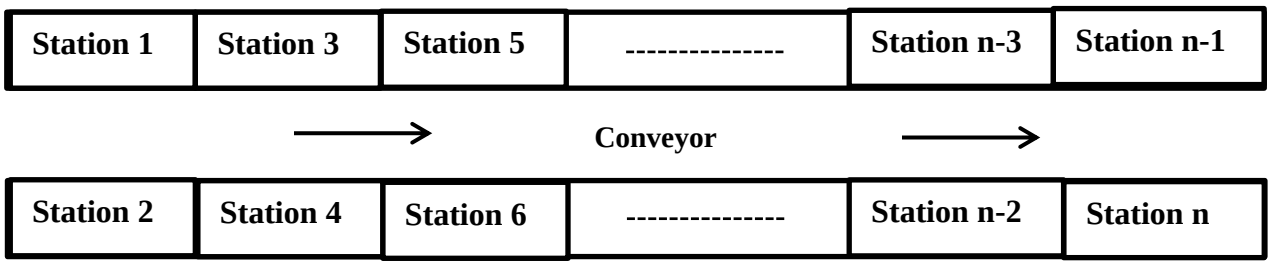
2.5.5.2 Stochastic Task Time

There are various uncertainties, which are caused by machine breakdowns, weak maintained equipment, the instability of worker`s skills (i.e., work rate, failure sensitivity) and motivation, defected raw materials and complex processes, require considering stochastic task times (Buzacott, 1990; Robinson, McClain& Thomas, 1990).



(a)

(b)



(c)

Figure 2.9 (a) Serial assembly line, (b) U-shaped assembly line, (c) Two-sided assembly line (Saif et al.,2014,pp-100-101)

2.5.5.3 Dynamic Task Time

In production systems, task time of assembly lines is varied due to labours, systematic reductions or successive improvements which caused by learning effects of production process. For example, when the worker first operates the given tasks, he can perform the tasks in longer time than his next operations of this task; because he gains experience at first time and he may perform the task in less time (Boucher, 1978; Chakravarty, 1988).

2.5.6 Objectives Characteristics

Assembly lines can be categorized based on the objective functions used during optimization process. Some assembly lines are optimized according to one objective function; on the other hand, some of them may have more than one objective functions to achieve during the optimization. Many researches have done to balance assembly lines by applying single or multi-objectives optimization methods.

2.5.6.1 Single Objective Optimization

Single objective optimization of assembly line is commonly used and there are several articles considering different types of objectives. The main objectives are listed as below (Boysen, Fliedner &Scholl, 2007):

- *Minimization of the number of workstations m* with a given cycle time or production rate; an objective to reduce the investment cost during designing a new line.
- *Minimization of the cycle time c* with a given number of workstations; can be referred as maximization of the production rate

- *Maximization of line efficiency E is* related with productivity of the line capacity
- *Minimization of cost for a given output target;* based on the optimization model, cost types can be allocated to different parts of assembly system, such as workstations, tasks, processing alternatives or resources
- *Maximization of profit*
- *Score* refers minimization or maximization of composed score which is based on one or more features explaining efficiency measurement or bottleneck aspects
- *Finding feasible solution* without objective function

Mixed model assembly lines have various workloads in each workstation because of different demand of models; so incomplete units may occur at certain models. To have a smoothed workstation times, two ways of smoothing has introduced:

- *Horizontal balancing* by Merengo,Nava and Pozetti(1999) to balance the workload on each station and smooth to workload in workstations
- *Vertical balancing* by Rachamadugu and Talbot (1991) to balance the workload among different workstations and smooth workload between workstations

2.5.6.2 Multi- Objective Optimization

In production industry, assembly lines have several objectives to satisfy simultaneously. In multi-objective optimization, combination of at least two objectives from single optimization problem can be applied. In some cases of assembly line optimization problem, chosen objectives may conflict with each other; for example, while optimization of one objective gives good results, the other one may not produce as good as the other at the same time. Therefore, it is not easy to satisfy all objectives at the same time by producing the best results in real environment (Ullah et al., 2014).

2.5.7 Problem Structure: Simple versus Generalized Assembly Line Balancing

Different classifications on problem structure of ALB have been represented in the literature since Salveson`s (1955) the first mathematical formalization of ALB based on the assignment of the tasks to the workstations. The most well-known classification on problem structure is proposed by Baybars (1986b) who categorized assembly line problem as SALBP and GALBP.

All problem types which generalize or eliminate some assumptions of SALP are called *GALPBs*. GALBP covers extensive variety of models including practice relevant view, such as balancing of mixed model, U-shaped lines, parallel stations or processing alternatives; thereby more realistic assembly line problems can be applied under GALBP (Becker&Scholl, 2006a).

SALBPs have lots of simplified assumptions, constraints and relations that make assembly line problem easy to solve and several researches have been done about SALBP in literature. Main characteristics of the classical single-model ALBPs are as follows (Scholl, 1999; Baybars, 1986b):

- Mass production of one homogeneous product performing by *n* operations of a given production process
- Paced line with fixed cycle time
- Deterministic and integral operation times
- No assignment restrictions besides the precedence constraints
- Serial line layout, one sided stations
- All stations are equally equipped with respect to machines and workers
- Fixed rate launching, launch interval equals to cycle time

SALBP has four different versions depending on objective functions: the feasibility problem (SALBP-F), the minimization number of workstations for a given cycle time (SALBP-1), the minimization of cycle time for a given number of workstations (SALBP-2), the maximization of the line efficiency (SALBP-E). Since all SALBP decreases partition problems which determine whether or not a set of positive integral weights may have two subsets with same sum of weights, they are NP-hard problems (Karp, 1972).

The notation of SALBP is presented below (Scholl, 1999):

n	number of tasks
V	set of all tasks (= {1,.....,n})
j	index for the tasks (j = 1,.....,n)
c	cycle time
p	production rate (= 1/c)
m	number of stations
k	index for the stations (k = 1,.....,m)
t_j	processing time (task time) of task j
p_j	workstation requirement of task j (= t_j/c = t_j. p)
t_{max}	maximal task time (=max { $t_j \mid j = 1,\ldots..,n\}$)
t_{min}	minimal task time (=min { $t_j \mid j = 1,\ldots..,n\}$)
t_{sum}	sum of task times (=$\Sigma j\ t_j$)
P_j	set of immediate predecessors of task j
F_j	set of immediate successors of task j
A	set of direct precedence relations (= {(i,j) $\mid i \in V$ and $j \in F_i$ })
S_k	station load, set of tasks assigned to station k
$t(S_k)$	station time of workstation k (= $\Sigma_{j \in S_k} t_j$)
E_j	earliest workstation of task j
L_j	latest workstation of task j

SI_j workstation interval
B_k set of potentially assignable tasks
T planning period/time of production
q_{max} maximum production amount
q_{min} minimum production amount
E line efficiency

2.5.7.1 The Feasibility Problem: SALBP-F

SALBP-F objects to find the feasible solution with a given number *m* of stations and a given the cycle time *c* of assembly line. In SALBP-F, assembly line is balanced with a given (*m, c*)- combination in order to have a feasible assignment.

Patterson and Allbracht (1975) presented the mathematical formulation of SALBF-F as below:

$$x_{jk} \text{ (binary variable)} = \left\{ \begin{array}{l} 1 \text{ if task j is assigned to station k} \\ \\ 0 \text{ otherwise} \end{array} \right\} \quad \text{for } j = 1,\ldots..,n \text{ and } k \in SI_j \qquad (2.1)$$

$$\sum_{k \in SI_j} x_{jk} = 1 \qquad \text{for } j = 1,\ldots..,n \qquad (2.2)$$

$$\sum_{k \in Bk} t_j.x_{jk} \leq c \qquad \text{for } k = 1,\ldots..,m \qquad (2.3)$$

$$\sum_{k \in SIh} k.x_{hk} \leq \sum_{k \in SIj} k.x_{jk} \qquad \text{for } (h, j) \in A \quad \text{and } L_h \geq E_j \qquad (2.4)$$

$$x_{jk} \in \{0,1\} \qquad \text{for } j = 1,\ldots..,n \text{ and } k \in SI_j \qquad (2.5)$$

x_{jk} is a binary variable presenting whether task *j* is assigned to workstation *k*. *Occurrence constraint* (2.2) and *integrality constraint* (2.5) refer that each task is assigned to only one workstation to find the line balance. The *cycle time constraint* (2.3) ensures that each workstation time is not greater than cycle time; workstation time must be smaller or equal to the cycle time of assembly line. The *precedence constraint* (2.4) guarantee that tasks has to be assigned to workstation according to their precedence relation; each task is assigned to workstation if all predecessors of the tasks have been already assigned.

The feasibility problem SALBP-F is related to other three types of optimization problems, so these constraints are applied by all types of SALBP to find an optimal solution.

2.5.7.2 The Minimization of Number of Workstations: SALBP-1

SALBP-1 aims to assign tasks to workstations in order to minimize the number of workstations *m* for a given cycle time *c*. SALBP-1 results from SALBP-F by presenting the amount *m* of workstations as a decision variable and improving the *m* minimization goal (Scholl, 1999).

Due to fixed cycle time and production rate, SALBP-1 is used for establishing a new assembly line in order to estimate demand of production system. In SALBP-1, cycle time is fixed when management decides the production ratio or production planning requests effect the upper bound (Erel&Sarin, 1998).

Bounds are useful to improve the solution process by decreasing time for solution period and size of the problem formulation. The most common used *lower bound formula* is based on the inequality $m.c \geq t_{sum}$ showing that the total task time is smaller than the total available time on the assembly line (Baybars, 1986b):

Lower bound for *m* number of workstations $m_{min} := \lceil t_{sum}/ c \rceil = \lceil \sum_j^n p_j \rceil$ (2.6)

The basic *upper bound* equals to number of tasks *n* considering each task can be assigned *n* number of different workstations (Scholl, 1999). The more extensive upper bound formula was proposed by Hackman, Magazine and Wee (1989), which gives at least one feasible solution with *m* workstations the load of the first *m-1* work stations in maximum level.

Maximality condition of the first *m-1* workstations $t(S_k) + t_{max} > c$ for k = ,…..,m-1 (2.7)

or

$t(S_k) \geq c+1- t_{max}$

Scholl (1999) summarized the inequalities in 2.7 and transformed the upper bound formula to as below:

Upper bound for *m* number of workstations $m_{max} := \lfloor (t_{sum} -1) / (c+1- t_{max} \rfloor + 1$ (2.8)

Different versions of objective functions are created in the literature. Patterson and Allbracht (1975) proposed a simple objective function, which is minimizing the number of workstation by assigning tasks to the latest used workstation considering precedence constraints, as follows:

Objective function Minimize $F(x) = \sum_{k \in Sln} k.x_{nk}$ (2.9)

Satisfying this objective may decrease the space requirements for building a new assembly line and reduce the labour cost while minimizing the number of workstations with a fixed cycle time.

SALBP-1 is related to some sequencing problems considering the objective and constraints. For example, the *bin packing problem* which aims to pack fixed amounts of items into minimum number of same sized bins and one type of *capacitated vehicle routing problem* which minimized the number of fixed

capacitated identical vehicles for distribution of goods from a warehouse to a number of customers (Hackman, Magazine& Wee, 1989; Labbe, Laporte &Mercure,1991).

2.5.7.3 The Minimization of Cycle Time: SALBP-2

SALBP-2 aims to assign tasks to workstations in order to minimize the cycle time *c* for a given number of workstations. SALBP-2 adjusts SALBP-F by presenting the cycle time *c* as a decision variable and improving the *c* minimization goal (Scholl, 1999).

While minimization of the cycle time for a given number of workstations, SALBP-2 also minimizes the sum of idle times and maximizes the production rate of an existing assembly line (Boysen, Fliedner &Scholl, 2007).

In order to improve the solution process, lower and upper bounds on cycle time *c* are proposed in the literature. The simplest *lower bound formula* for cycle time depends on the necessary feasibility condition $m.c \geq t_{sum}$ and $c \geq t_{max}$ is applied by tasks for indivisibility condition (Scholl, 1999):

Lower bound for *c* cycle time $c_{min} := \max\{ t_{max}, \lceil t_{sum}/ m \rceil \}$ (2.10)

Due to maximization of the production rate, when production amount *q* is considered with a fixed time planning period *T* (Hartl, 2014):

Lower bound inequality for *c* cycle time $c \geq \lceil T / q_{max} \rceil$ (2.11)

By summing up both lower bound formulas, we have a general expression for a lower bound formula:

Lower bound for *c* cycle time $c_{min} := \max\{ t_{max}, \lceil T / q_{max} \rceil, \lceil t_{sum}/ c \rceil \}$ (2.12)

The basic *upper bound form*ula that does not satisfy the precedence relations between the tasks is given (Scholl, 1999):

Upper bound for *c* cycle time $c_{max} := \max\{ t_{max}, 2.\lceil t_{sum}/ m \rceil \}$ (2.13)

Upper bound for cycle time can be found also from minimum production amount in time horizon *T* (Hartl, 2014):

Upper bound inequality for *c* cycle time $c \leq \lfloor T / q_{min} \rfloor$ (2.14)

In objective function of SALBP-2, cycle time *c* is introduced as a variable, $c \geq 0$ in order to minimization of *c*:

Objective function Minimize F(x, c) = c (2.15)

SALBP-2 is related to some assignment problems considering the objective and constraints. For example, the problem of scheduling jobs for *m identical parallel machines* in order to minimized the makespan and bottleneck transportation problems which aim to find a feasible distribution for minimization of maximum transportation time between supplier and buyer (Garfinkel&Rao, 1971;Coffman, Grey&Johnson, 1984).

2.5.7.4 The Maximization of Line Efficiency: SALBP-E

SALBP-E searches for a feasible combination *(m, c)* of the number *m* of workstations and cycle time *c* as well as maximizes of assembly line efficiency, $E = t_{sum} / (m.c)$ or equally minimizes *m.c*. SALBP-E covers the generalization of SALBP-1 and SALBP-2 and creates opportunities to improve production process (Scholl&Becker, 2006b). When assembly line has no idle times, line efficiency is equal to one with *(m, c)* combination of $m = 1$ and $c = t_{sum}$; so assembly line operates with 100% efficiency (Hartl, 2014).

As maximization of line efficiency is based on constant t_{sum} and minimizing non-linear term *m.c*; thus listed capacity oriented objectives are equal to maximizing line efficiency *E* (Scholl, 1999):

- Minimization of flow time : = *m.c*
- Minimization of balance delay time : = $m.c - t_{sum}$
- Minimization of balance delay ration : = *1-E*

For optimization of SALBP-E, a *lower bound* m_{min} for number of workstations must be found. Since an *upper bound* c_{max} for cycle time is predetermined, it is possible to find a *lower bound* m_{min} for the number of workstations by using feasibility condition $m.c \geq t_{sum}$ (Scholl, 1999):

Lower bound m_{min} for number of workstations $m_{min} := \lceil t_{sum} / c_{max} \rceil$ (2.16)

Restriction on number of workstations is also needed to reach the optimal solution in SALBP-E. A lower bound cmin for cycle time is reached by using the lower bound formula (2.12) in SALBP-2; so modified version of upper bound formula for number of workstations (2.8) is applied to find an *upper bound* m_{max} for number of workstations:

Upper bound m_{max} for number of workstations $m_{max} := \lfloor (t_{sum} - 1) / (c_{min} + 1 - t_{max} \rfloor + 1$ (2.17)

In order to summarized the bound conditions, instances for SALBP-E are restricted by $[m_{min}, m_{max}]$ and $[c_{min}, c_{max}]$.

The objective function of SALBP-E is created by taking objective function of SALBP-1, where the cycle time is accepted as an additional variable (Hartl, 2014).

Objective function $\quad$ Minimize $F(x) = c.\sum_{k \epsilon Sln} k.x_{nk}$ $\quad$ (2.18)

3. Solution Methods for Assembly Line Balancing and Literature Review

Due to rapid change of technological improvements and global competitions, companies pay attention for improving their production and assembly line systems by decreasing costs, production time and increasing machinery usage, productivity of the line. Since the first formulation on ALB problem was developed by Salveson (1955); many exact, heuristic and metaheuristic algorithms are improved by using computational tools and techniques in order to provide more optimal solutions (Scholl&Becker, 2006b).

3.1 Exact Solution Methods and Literature Review for Exact Solution Methods

3.1.1 Integer Programming

Salveson (1955) develops SALBP-1 as a linear programming problem including all possible combinations of task assignment to workstations. Due to split tasks, his methods can often produce infeasible solutions. Bowman (1960), as later modified by White (1961), solves this problem by creating *indivisibility constraint* and transformed linear programming formulation to an integer programming problem describing symbolizing task assignments to workstations with 0-1 binary variables. Klein (1963) proposes a simple integer programming to find an optimal combination for assigning tasks to workstations when the order of operations is specified. However, his approach was not practical for the real- sized ALBPs due to amount of the required enumeration.

Balas (1965) presents a tree search algorithm which uses information in the search to exclude portions of tree by solving linear programs with 0-1 variables. Then, Geoffrion (1967) modifies the additive algorithm of Balas (1965); he also improves a flexible and economical version of the “back-track” methods of integer programming.

Thangavelu and Shetty (1971) apply the additive algorithm of Balas (1965) and introduced their integer programming methods with two subroutines; one is for increasing the partial solution that may result a better feasible solution than incumbent solution, the other one is for backtracking and record keeping where a better feasible solution may occur or no solution may be reached. Thangavelu and Shetty (1971) also apply conditional feasibility test to speed up the implicit enumeration process.

Patterson and Allbracht (1975) introduce an integer programming method to analyse sequences of zero-one variables for feasible solution. They defined new parameters as early and late workstations for each task based one precedence relations, accordingly decreasing the number of variables. They apply a binary infeasibility test for the cycle time constraints and improved enumeration process for fathoming a partial solution.

Talbot and Patterson (1984) present an integer programming by applying integer variables instead of binary variables. They used the implicit enumeration of Balas (1965) for their formulation in which

precedence relations are sustained by immediate predecessor test and the evaluated idle time vector fulfills the cycle time constraints. Talbot and Patterson (1984) also introduce early and late workstations for each task; they also improved the problem form by enabling the fathoming and branching process.

3.1.2 Branch and Bound Algorithms

The branch and bound algorithms are acknowledged solution for combinatorial optimization with two main components which are *branching (enumeration)* and *bounding*.

Branching (enumeration) is the process that the initial problem is divided into continuously developing sub-problems (nodes) which generate a multi-level enumeration tree. The first stage of this tree composed of the *root node* represents *initial problem*; at next stages, *descending nodes* are constructed by developing an *ancestor node* at each previous stage of the problem and sub-problems of the found optimal solution are called as *leaf nodes*. *Branch* is a path starting from the root to any other node of the tree (Scholl, 1999).

Branch and bound algorithm is divided according to nodes sequence of the branched enumeration tree (Scholl&Becker, 2006b):

- At ***depth-first-search***, a single branch of the tree is improved since a leaf node is arrived. The search tracks the first potential alternative branch while it is moving back to the root. Each node is developed then their ancestor nodes are visited again. Depth-first-search is work as a laser search with respect to priority rules of descending node at the current branch. Firstly, the node with the highest priority is branched, then rests wait at candidate list during revisiting process of each node until they have priority.
- A ***minimal lower bound strategy*** always selects undeveloped node with the minimum lower bound from a candidate list. The selected node is branched by building all descending nodes. Until the current node is dropped, the remaining nodes wait in the candidate list which is aligned based on non-decremental bound values.

Branching strategies can be categorized as *task oriented* and *station oriented* procedures. At task oriented procedure, an available task is selected to assign to earliest available workstation; only one workstation is taken into account for assignment of a single task. On the other hand, at station oriented procedure, a complete load is constructed for one workstation at each step (Scholl, 1999).

Bounding is to drop the size of enumeration trees by calculating lower bounds on the number of workstations for each node. The lower bounds are drove from *relaxations* that are results of the omitting or relaxing constraints. The root node bound is called as *initial* or *global lower bound* and the other nodes of residual problems are *local lower bounds*. At the bounding procedure, logical tests such as *dominance* and *reduction rules* can be applied to decrease the size of enumeration trees (Scholl, 1999).

To find an optimal solution for the minimum number of workstations is an *upper bound* for the initial problem. When upper bound is equal to the global lower bound, the feasible solution is reached; so the procedure stops. All nodes which their local lower bounds are greater than upper bound are *fathomed* and left out of the solution because they produce solutions with number of workstations is more than upper bound.

The feasible solution of the SALBPs can be shown by a tree which each path represents a feasible solution with each arc as a workstation. Jackson (1956) introduces the first algorithm for SALBP-1 by representing the assembly lines with a tree. At algorithm of Jackson (1956), if any task can`t be assigned to a workstation considering the cycle time constraints and precedence relations, a workstation is specified as *maximal*. According to Jackson (1956), the arcs of the feasible solution tree presents only maximal workstations. The procedure is not exact enumeration; however, it goes through all possible assignments to the first *m-1* workstations before deciding any assignment to the *m*th workstations.

Hu (1961) contributes branch and bound approach to solve SALBP with parallel workstations. He introduced the ordering restrictions of ALBP as form in tree to find minimum number of workers needed to process the tasks with a given time by applying lower bounds on shortest time. At algorithm of Hu (1961) is based on finishing all tasks at the earliest time with a given number of workers by setting the tasks in several sequences. Hu (1961) defines his algorithm as a mechanical analogy in which metal rings represent nodes and a tree diagram that consists of tied rings by pieces of string of unit length. By holding a final node, the all other rings are released. "Then the algorithm is to cut off at most number of end-rings at a time, with preference given to bottom end-rings if there is more than number of end-rings available for cutting" Hu (1961 ,p.846). The researcher describes his algorithm as *cutting longest queue* Hu (1961):

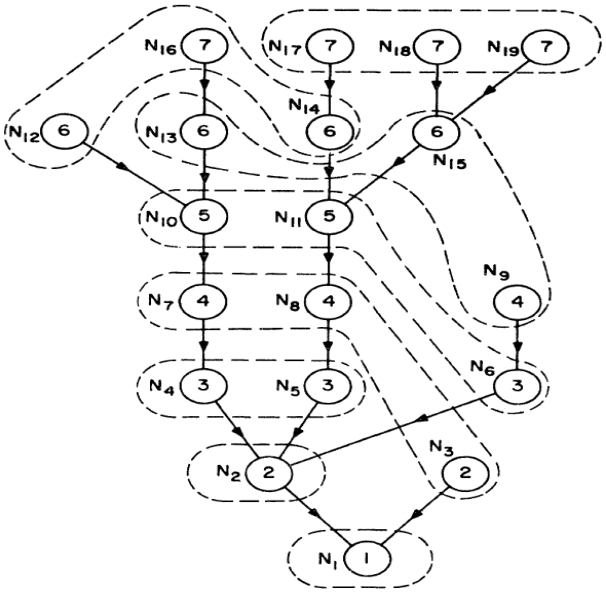


Figure 3.1 Illustration for mechanical analogy of Hu (1961)´s algorithm, pp. 846

Jaeschke (1964) introduces the first simple model strategy of a workstation oriented branching and the bound as a lower bound on *m* number of workstations by applying the minimal lower bound. The nodes with minimum total idle time are selected for branching and the nodes with the largest number of loaded workstations are chosen.

Mertens (1967) proposes a workstation oriented depth-first-search algorithm depending renumbering of tasks with priority rules and he used simple bound as a lower bound on *m* number of workstations.

Van Assche and Herroelen (1979) present a workstation oriented procedure by applying minimal lower bound strategy. They use Jackson (1956) ´s enumeration tree type and proposed dominance rules, bounding methods and branching heuristics with a node showing the tasks assignment to a single workstation. The process begins with an empty workstation and goes through the unordered list considering that each node is one of assigning the remaining tasks to the remaining workstations by searching for an immediate solution. When there is not any immediate solution, the node branches into a number of descendant node related to the optimal assignments of non-dominated next workstation and calculates a lower bound for the remaining number of workstations. Van Assche and Herroelen (1979) also introduce some penalty and tie breaking rules to discriminate the nodes with the lower bound and penalty.

Johnson (1973) describes a new workstation oriented approach for a branch and bound algorithm in which workstations are represented by arcs by applying "*maximal workstation assignment*" algorithm of Jackson (1956). Then, Johnson (1981) develops a new version of his algorithm (1973) by improving his bounding procedure as combining depth-first-search and minimal lower bound strategies. At Johnson (1981), the set of recently created nodes generate new arcs and a feasible solution is found by minimizing idle time for selected each arc at the workstation. If a feasible solution is found, the solution tree comprises from a complete path with a number of end nodes conjoint to the nodes of the path. Johnson (1988) introduces a new branch and bound algorithm called as FABLE which is task oriented procedure depth-first search organized as laser search for SALBP-1. FABLE reaches feasibility quickly by applying several logical tests such as maximum load rule, Jackson dominance rule. From the first workstation, sub-problems are built by assigning the candidate tasks to the current workstation. If there isn´t any assignable tasks, a new workstation will be opened. At each creating of new workstation, the task with the smallest index in the assignable tasks set is assigned firstly. Due to assigning the candidate tasks to the earliest workstation, the workstations are filled according to forward planning from the first to last workstation.

Gunther, Johnson and Peterson (1983) introduce the goal programming for a multiple objective formulation to the GALB and presented a branch and bound algorithm to find the optimal solution.

Talbot and Patterson (1984) propose feasible solutions depending on integer programming codes using branch and bound procedures with task oriented branching and depth-first-search. They use two different ways for backtracking by using *chains* and *network cuts*. A *chain* is a set of orderly numbered tasks which are immediate predecessors or successors of next members of the set. If there are chains, searching takes faster due to directly backtracking to the task with the highest number not existing in a chain.

A *network cut* is a parameter defines the application of the rules and presents a workstation number for describing if the fathoming and backtracking rules applied or not. Talbot and Patterson (1984) define two fathoming rules and two additional expediting rules depending on lower bound for idle time presented in each workstation to solve an imbedded knapsack problem.

Saltzman and Baybars (1987) present branch and bound algorithm which is task oriented and depth-first-search. Tasks are enumerated non-decremental order of operation times. An initial value for the upper bound is decided according to a priority-based heuristic of task assignment depending on non-increasing order of task times.

Betts and Mahmoud (1989) use same procedure of Johnson (1981) ´s algorithm except the method of building workstation loads by applying workstation-oriented procedure undirectionally.

Hackman, Magazine and Wee (1989) propose an algorithm as a station oriented branching with minimal lower bound and applied several dominance rules. They implement priority-based heuristics to decide an initial upper bound and to find partial solutions. They also use fathoming for selection of nodes.

Nourie and Venta (1991) extend FABLE by applying heuristics to find an initial upper bound and a memory saving method for storing all feasible subsets of tasks required by workstations in order to implement the feasible set dominance rule. Their algorithm is task oriented branching with depth-first-search. Due to huge storage requirement of the addressing of feasible task subsets, Nourie and Venta (1991) introduce a tree structure to improve the performance of the storage for all feasible subsets.

Berger, Bourjolly and Laporte (1992) change the algorithm of Hackman, Magazine and Wee (1989) by applying depth-first-search instead of minimal lower bound for restricted version of SALBP-1 with at most one predecessor for each tasks based on *out-tree* precedence graph.

Hoffman (1992) introduces another depth-first-search algorithm called as EUREKA to solve SALBP-1. EUREKA is a workstation based laser search method with lower bound procedure for unassigned tasks by implementing complete enumeration. The algorithm performs *forward* and *backward* planning. Workstation loads are formed based on increasing numbers of tasks at forward procedure, and vice versa at backward procedure by applying same enumeration method at both procedures in order adapting the precedence graph. After finishing the assignment of a workstation, the cumulative sum for slack time is computed. If the sum of slack time is greater than minimum slack time, all emanating branches are enumerated. If not, algorithm backtracks to the preceding workstation and creates a new tasks assignment before performing same procedures. When no feasible solution is reached in a defined computation time, the heuristic technique at Hoffman (1963) is implemented to find a feasible solution.

Scholl and Klein (1996) create SALOME-2 for SALBP-2 by customizing components of SALOME-1. SALOME-2 is a flexible bidirectional branching strategy with using local lower bound method and applying some dominance and reduction rules to ensure restrictions of SALBP-2.

Scholl and Klein (1997) present a new branch and bound algorithm for SALBP-1 called as SALOME-1; which combines and improves the most promising components of FABLE and EUREKA. Scholl and Klein (1997) improve a bi-directional branching by implementing local lower bound method for each sub-problem of the enumeration tree to reach preferable planning direction. The potential loads of the current workstation have two different classes while applying branching at each sub-problem. At first class, all loads have minimal idle time and do not need to increase the number of workstations and remaining loads are placed at second class. The loads of the first class are branched firstly; if no solution has been reached in related sub-trees, they increase by *1* the lower bound for the number of workstations, then the loads at second class start to be branched. Scholl and Klein (1999) expand the SALOME-1 algorithm by implementing new dynamic renumbering and dominance rules.

Bock and Rosenberg (1998) and Bock (2000) improve distributed version of SALOME-1 with a workstation oriented and local lower bound method.

Sprecher (1999) introduces a task oriented branch and bound algorithm called as the Adapted General Sequencing Algorithm which is performing a wide class of resource constrained project scheduling problems with only renewable resource considering precedence related enumeration process. Sprecher (2003) improves distributed and parallelized version of Sprecher (1999).

Scholl, Fliedner and Boysen (2010) improve an algorithm called as ABSALOM (SALOME for Assignment Bounded Problems) depending on basic procedure of SALOME with intensively usage of bounding processes, dominance rules and reduction procedures to associate processes from combinatorial optimization and constraint programming. They present ABSALOM to solve the ALBP with assignment restrictions.

3.1.3 Dynamic Programming

Dynamic programming is an optimization method for multi-stage decision processes. Dynamic programming problem consists of *stages* (sub-problems) that are solved orderly until the first stage (problem) reaches a solution (Erel&Sarin, 1998).

Each stage has a number of *states*, which define all potential situations of the decision process, must be taken into account. The set of all states of the decision process is described as a *state space*. States at a certain stage are *transformed to* states at the following stage by a *decisio*n. The states production is defined by *transformation (transition) functions* that are based on present stage and the decision. Each decision indicates values for specific problem variables. A sequence of the decisions on states between different stages is called as *policy* (Erel&Sarin, 1998).

Dynamic programming depends on the optimality principle which initial problem, linking the first state and terminal state of the decision process, must apply optimal policies for all sub-problems. The dynamic programming problem must be formulated as a multistage decision process. There are two different methods to solve the decision procesd (Erel&Sarin, 1998):

- *Forward Recursion:* From the initial state, the stages are organized orderly increasing number until the terminal state. At each stage, all potential states are created before the following stage is checked. The optimal solution of the problem is generated according to optimal policy which transforms the initial state to terminal state.
- *Backward Recursion:* This procedure is applied stage by stage starting from the terminal state to initial state. Each stage includes all state that can be possibly turned into the terminal state based on the optimal policy. At the preceding stage, all states that end up with at least one state at the succeeding stage are created. In this procedure, the optimal policy establishes a link between initial and terminal state of the optimal solution.

Dynamic programming can be classified in two groups according to decision which is used at the solution procedure (Erel&Sarin, 1998):

Workstation oriented procedure is that the stages connected to workstations and the states of each stage are determined by feasible subsets of tasks which can be assigned to workstations. The workloads that can be formed for the workstation according to a state at the preceding workstation generate decision according to optimal policy. The initial stage of the first state is defined by empty set and terminal state by the set of all tasks. Since the terminal state is known, the corresponding stage is unknown due to its corresponding to number of requested workstations.

Task oriented procedure comprises of decision process that are sub-classified to stages corresponding to the numbers of assigned tasks whose sequences form the policies. The states of each stage are determined by feasible subsets of tasks. "In each state at a stage *k*, *k* tasks are already assigned and the decisions are to assign one of the currently available tasks to position *k+1* of the sequence." (Erel&Sarin, 1998, p.130). Initial stage of the first state is defined by empty set and terminal state by the set of all tasks. The task sequences are connected to workstations according to maximum load rule. *Pulling* and *reaching* are the basic strategies applied to enumerate all feasible subsets at the task oriented procedure.

Jackson (1956) introduces a workstation oriented dynamic programming procedure for solving SALBP-1 and applied forward recursion to reach optimal solution. His algorithm firstly assigns all feasible tasks to the first workstation, then produces all feasible assignments to the second workstation, known the assignments of first workstation to generate a first-second workstation combination. After creating a first-second workstation combination, all feasible assignments are built for the third workstation. This process goes on until an optimal number for workstation is found, so all tasks cam be assigned to target number of workstations. Jackson (1956) applies several dominance rules to improve his solution. Application of the optimal policy, procedure can reach an optimal solution. However, Jackson (1956) does not use independency of related policies for the states of the procedure.

Held and Karp (1962) propose task oriented procedure for dynamic programming which they applied general rules of sequencing regarding to precedence relations. Then, Held, Karp and Shareshian (1963) present this dynamic programming procedure with pulling strategy in more details and introduced *fea-*

sible subsets and *feasible sequences* into the method. At their task assignment procedure, the number of workstation is minimized while tasks are performed based on their feasible sequences. Due to the optimal policy of dynamic programming, they introduced a *compact addressing function* to save each of all produced states in their memory position with their workstation requirement. This addressing function guarantees that memory positions are under monotonous order and it depends on recursively partitioning the precedence graph. At each memory position, related state is decided by calculating the inverse of the addressing function according to counting procedure that determines the cardinality for the space of each state. The addresses of all previous states must be calculated based on the requirement of pulling strategy.

Kao (1976) introduces a dynamic programming approach to assign tasks to the minimum amount of workstations considering the precedence relations between the tasks and the lower bound constraint for that the probability of the assigned workloads to each workstation not greater than cycle time while minimizing the labour cost of assembly line. The approach is suitable for the limited size of the problems due to storage space requirements.

Schrage and Baker (1978) improve a task oriented dynamic programming procedure with an easier addressing function which depends on task labels valued by the labelling dominance rule to store the updated minimum number of workstations. Each subset of the tasks requires a specified addressing in a memory to store workstation necessities of all feasible subsets. The address of each feasible subset is calculated by summing up labels of the inclusive tasks. They try to improve the compactness of addressing by applying diversified labelling methods

Lawler (1979) proposes a task oriented dynamic programming procedure with an enumeration technique operating the states from one stage to another stage. The procedure has forward recursion approach by applying a reaching strategy to improve the efficiency on the storage requirements and processing times of the assembly line.

Kao and Queyranne (1982) improve the enumeration process of the task oriented dynamic programming of Schrage and Baker (1978) by changing the algorithm to solve larger size of problems without requirements of a huge storage space and creating the states based on increasing order of the addressing for SALBP.

Bard (1989) presents a dynamic programming procedure to minimize the number of workstations by while balancing the cost of setting additional facilities such as operating costs, equipment costs for assembly lines with parallel workstations. Bard (1989) applies an enumeration procedure with utilizing the given lower bound.

3.2 Heuristic Solution Methods and Comprehensive Literature Review for Heuristic Solution Methods

3.2.1 Heuristic Solution Methods

Many heuristic solution methods have been developed for balancing assembly lines. Heuristic methods are logical procedure; even though they do not guarantee the optimal solution, feasible solutions with good results can be found. Heuristic procedures are mainly priority rules methods, enumerative heuristic methods, shortest path problems and an adaption of the heuristics for the cutting and packing problems. At this part, I would explain the milestone in heuristic solution methods for ALB.

Priority Rule Based Procedure is a constructive method to find the feasible solution for balancing assembly line by applying task ranking according the chosen priority rule. Many different priority rules have been proposed in the literature. Application procedure of priority rules is that each task is assigned to workstation based on their computed priority of the heuristic decision rule by not violating the precedence relations between the task and cycle time. Firstly, a task with highest numerical priority according to choosen heuristic priority rules is assigned to a workstation. Then, predecessors of the first assigned task are placed to available list checking the priority rule, remaining time at the workstation and precedence relation. The task with the highest priority from available (candidate) list is assigned to the workstation. At the assignment of each task, available list and the remaining time for workstation must be updated. This process goes on until there is no possibility to assign another task to workstation due to remaining time check. Then, the next workstation can be opened to assign to tasks at the updated available list. The assignment procedure ends when there are no tasks to assign at the assembly line and a feasible solution is reached in the end. The assignment procedure can be workstation oriented or task oriented.

The notation for priority rules is shown below (Erel&Sarin, 1998; Scholl&Voß, 1997):

c	cycle time
t_j	task time
P_j	set of tasks which must precede task j immediately (predecessors)
F_j	set of tasks which must follow task j immediately (successors, followers)
P_j^*	set of all tasks which must precede task j
F_j^*	set of all tasks which must follow task j
$\bar{m}$	upper bound in the number of workstations
pw_j	positional weight of task j ($= t_j + \sum_{h \in Fj*} t_h$)
pw_j^*	positional weight of task j ($= t_j + \sum_{h \in Fj} pw_h^*$)
r_j	number of arcs in the paths having j as its root ($= \lvert F_j \rvert + \sum_{h \in Fj*} \lvert F_h \rvert$)
r_j^*	recursive cumulated number of arcs in the paths with root j ($= \sum_{h \in Fj} (r_h^* + 1)$)

E_j earliest workstation of task j (= $\lceil (t_j / \sum_{h \in P_j^*} t_h) / c \rceil$)

L_j latest workstation of task j to improve (= $\bar{m}$ - $\lceil (pw_j / c \rceil$)

s_j slack of task j (= $E_j - L_j + \varepsilon$)

The *positional weight* is calculated by adding up the operation times of the task and all successors of this task at the precedence diagram. The *reverse positional weight* is found by adding the operation times of the task and all predecessors of this task at the precedence graph. The priority r_j^* presents each arc related to the number of successors that are depended to task j by this arc. The slack shows the freedom level for assigning a task to verified workstations. Small number "ε" is added to prevent splitting by zero in some of priority decision rules.

These priority rules are static and are divided into several subgroups during the solution procedure. The extensive collection of well-known and new priority rules is listed below (Erel&Sarin, 1998) :

MaxTime	decreasing task time t_j
MaxPW^{sm}	decreasing positional weight pw_j
MaxF^{sm}	decreasing number of followers $\lvert F_j^* \rvert$
MAXIF	decreasing number of immediate followers $\lvert F_j \rvert$
MinE^{m}	increasing earliest workstation E_j
MinL^{m}	increasing latest workstation L_j
MinSlack	increasing slack s_j
MaxAvgPW	decreasing average positional weight $pw_j / \lvert F_j^* + 1 \rvert$
MinAvgL^{sm}	increasing average latest workstation $L_j / \lvert F_j^* + 1 \rvert$
MaxTimeL	decreasing task time divided by latest workstation t_j / L_j
MaxFSlack	decreasing number of followers divided by slack $\lvert F_j^* \rvert / s_j$
MaxTimeSlack	decreasing task time divided by slack t_j / s_j
MaxA^{sm}	decreasing total number of following arcs r_j
MaxCumA^{sm}	decreasing cumulated number of following arcs r_j^*
MaxCumPW^{sm}	decreasing cumulated positional weight pw_j^*
MinTaskNo^{sm}	increasing task number j

The rules marked with "*m*" defines *monotonous* increasing or decreasing ranking values for a specific p_j ; for example, " $p_h \geq$ or $\leq p_j$ " is true for all arcs (h,j) at the precedence diagram. The rules with "s*m* " means strongly monotonous when " $p_h >$ or $< p_j$ " holds for all arcs (h,j).

"*MaxTime* " priority rule is presented by Moodie and Young (1965); at this priority rule, tasks are ranked according to their operation times from largest to smallest and are assigned based on decreasing operation times by choosing the task with assigned predeccessors. This rule can be applied as a reverse way which is assigning tasks according to increasing task times in order to precedence relations.

Helgeson and Birnie (1961) improve the most popular single pass heuristic priority rule called "*RPW Technique*". Positional weight for each task is calculated and tasks are listed in decreasing order, then assignment process is applied based on descending positional weight rule by considering precedence relations at the assembly line. During the assignment process, available list, consisting of candidate tasks to assign, must be updated after each assignment to find the feasible solution.

Tonge (1960) introduces the "*MAXIF* " which is assignment of the task according to decreasing number of immediate successors (followers). Tonge (1965) proposes "*Random Selection*" procedure that assigns tasks to workstations by randomly choosing the following task to place into present workstation.

Arcus (1963) proposes the "*MinTaskNo*" priority rule which is assigning tasks according to ascending task number. He applies a biased sampling procedure to reach a feasible tasks sorting for assigning to a workstation.

Priority rules like $MaxF^{sm}$, *MinSlack, MaxAvgPW* are proposed by Talbot, Peterson and Gehrlein (1986).

All station and task oriented procedure can be formed initially by prefixing tasks j with $E_j = L_j$ to workstation E_j. In this situation, all preceding tasks of j must be assigned to the workstation by starting first until E_j (Scholl, 1999). When a feasible solution can not be found by applying these static priority rules, we may use "*Upper Bound*" decision rules to find the minimum number of workstation in order to reach optimal solution. According the upper bound decision rules (Talbot,Peterson&Gehrlein,1986; Hartl, 2014):

- Tasks can be assigned monotonically *increasing upper bound* of each task j and their predecessors required number of workstations
- Tasks can be assigned monotonically *increasing upper bound* for latest possible workstation of each task j

The procedure for priority rules can be applied as forward, backward and bidirectional ranking. Forward procedure is improving solution by starting from the first workstation unidirectionally. On the other hand, backward procedure develops solution on reverse precedence diagram with reverse rule. Moreover, both procedure may be applied in order to assign task bidirectionally. Task ranking must be calculated based on forward and backward procedure. The bidirectional procedure begins with first workstation and $\overline{(m-1)}$ as present workstation by applying both directions. The Priority Based Pro-

cedure has static rules due to only considering the parameters, not the solution process. *Dynamic rules* are hold by using modified parameters through task assignments (Scholl, 1999).

Wee and Magazine (1982) develop "*Generalized- First – Fit* which is assigning all available tasks to the initial workstation by considering precedence relations and cycle time constraints. Secondly, available (candidate) task list must be updated after all assignment of the tasks whose immediate predecessors have been already assigned. Their methods relatively sequence the tasks based on the levels in the assembly line precedence diagram and task ranking score in each level of assignment procedure. Hackman, Magazine and Wee (1989) present "*Immediate- First – Fit*" which is heuristic task oriented procedure with assignment of tasks based on numerical score of each task; the task with the highest numerical score is assigned the first workstation. The assignment procedure is quite same as generalized method but in immediate first fit procedure, some restriction related with precedence relation. They also propose "*Rank and Assign*" which is less useful than other two procedures due to checking the constraints many time even it is not required.

Kilbridge and Wester (1961) introduce a heuristic assembly line procedure that is grouping tasks under the columns at the precedence diagram in which tasks are stated as left as possible based on their precedence relations according to their cumulative operation times of each task. This method is a good solution for assembly lines with large cycle time if a workstation passes through various columns. Tasks can be switched between each other in every column at the precedence diagram and they may be transferred to right sideward position from their columns according g precedence constraints to find the feasible balance for the assembly line. Kilbridge and Wester (1961) also analyse the effect of balance delay at different parameters of ALB.

Hoffman (1963) improves a unidirectional workstation oriented heuristic method in which assembly lines are balanced with a precedence matrix. This method starts with the first workstation to calculate the combination of tasks with resulting the minimum slack time at that workstation until all tasks are assigned. While finding the optimal solution with precedence matrix with iterations, each column of the matrix is totalled and these total values state another row attached to the bottom row of the precedence matrix.

Arcus (1966) proposes a computer based multi-pass heuristic solution approach as *COMSOAL* which is random generation of a large amount of feasible solutions and finds the best solution with minimizing the number of workstations. The procedure starts from first workstation for assigning tasks with assigned predecessors. The following task as candidate to be assigned is chosen randomly from the subset of the candidate tasks. When the subset becomes empty, new workstation is open.

Thomopoulos (1967) forms the method of Kilbridge and Wester (1961) to solve mixed model ALBPs. He assigned the tasks to workstations by calculating the penalty cost resulting from unproductiveness of ordering different models. The assignment procedure of Thomopoulos (1967) is a shift based and he introduces four different unproductiveness which are idleness, work deficiency, utility work and work congestion. If there is an idle waiting of workers, idleness happens; if the worker can operate the task

before the following task comes to the workstation, work deficiency happens; time shortage for the uncompleted work results with utility work and work congestion. Then, Thomopoulos (1970) improves a method in which assignment of tasks is done serially. The methods analyse a finite amount of feasible combination for tasks to minimize the deviation of the workstation tome from mean average workstation tıme.

Dar- El (1973) presents a method for SALBP-2 as "*MALB*" which is assigning tasks with a given number of workstation in order to minimize cycle time. So, MALB begins with minimum theoretical cycle time and processes by producing feasible assignments of task to the workstations. If there is no improvement at the feasible assignments, then the method proceeds a backtracking procedure by separating tasks in right way or giving a results as a rise of a time unit for cycle time.

Pinto, Dannenbring and Khumawala (1978) introduce a heuristic solution approach based on shortest path problem, only taking into account of subgraph of precedence graph. In the method, the nodes define a subset of tasks which can be executed in some sequence with no prior finishing of any task not placed in the subset. They also apply other heuristic procedure to create the nodes which is generated to be combined to develop a composite network.

Due to possibility of not finding the optimal solutions by applying heuristic methods, "*Worst Case Analysis*" is improved by Wee and Magazine (1982) to find the solution quality that is achieved on the average and in the worst case. During the ALB, related error bounds can be obtained from features of the optimal solutions. The worst case bound indicates that there is not more than twice the number of workstations as an optimal solution occurs in a heuristic solution (Wee&Magazine, 1982).

Agrawal (1985) improves a method with a decision rule as "*Largest set rule*" to divide the work to workstations. The method calculates the cumulative time for every task that is the sum of operation time of the task and operation time of all its predecessors. Afterwards, the task with the largest cumulative, less than cycle time, is chosen and related tasks are transferred to the worker. The method continues until all tasks are assigned and the work is shared by the workers.

Baybars (1986a) proposes a single-pass heuristic solution method with five different phases. Initial four phases decrease the size of ALBP by providing benefit from different features of the assembly line problem and the final phase is single-pass heuristic solution to be applied on the decreased problem. The procedure is backward process, so it starts with the last tasks of the precedence diagram.

3.2.2 Comprehensive Literature Review for Heuristic Solution Methods

Besides the well – known heuristic solution methods mentioned above, many researches are done and published from the 1960s till nowadays.

Mastor (1970) analyses the procedures of Helgeson and Birnie (1961), Kilbridge and Wester (1961), Hoffman (1963), Held, Karp and Shareshian (1963) and Arcus (1966) by applying on different types of ALBPs. The performance measures to compare the methods are the *output rate* to measure the effectiveness and cycle time of assembly line and the *cost of calculation*. The research of Mastor (1970) shows that there are particular differences on ordering strength, problem size and assembly line length between ALB methods. The best results achieved by methods of Held, Karp and Shareshian (1963).

Campbell, Dudek and Smith (1970) proposed a heuristic method to find approximate solutions via computer for very large sequence problems. They applied the method to *n* number of tasks, m number of machines with every task process with same order of machines. They aim to reach m machine sequence problems with minimization of total elapsed time not considering passing of tasks. As a result, the method is able to find up to m-1 sequences with a possible decreasing of expected error via usage of computer.

Macasskill (1972) introduces a computer applied heuristic methods for task assignment to balance mixed model assembly lines with deterministic task time.

Nevins (1972) present a general heuristic program as "*The Best Bud Search*" which has an upper bound for the number of workstations and is applied to minimize the number of workstations indirectly.

Reeve and Thomas (1973) research the single model assembly line with stochastic task processing time by comparing four solution methods. According their methods, the first balance is given and reassigned the tasks by minimizing the workstation time over cycle time. Their first method is based on "Trade and Transfer" which is trading the tasks one by one between workstations to decrease the probability of workstation times over cycle time. Their second method is based on bounding process that exploit the first balanced solution to set the probability of workstation times over cycle time the upper bound for the idle time. Their third method is a heuristic branch and bound technique by adding some heuristic rules to second method. The fourth method is called as *BABTAB* which is combination of their first and third methods. BABTAB performs well for short time periods and cannot be generally interpreted because small size of the ALBPs of the research. (Reeve&Thomas, 1973).

Kottas and Lau (1973) draw an attention on the including costs apart from labour cost of the assembly line and they propose a heuristic solution method for minimization of the product incompletion cost. The incompletion cost is related with incomplete tasks and labour cost at the assembly line. There is a connection between the cost of workstation idle time and the incompletion cost. In Kottas and Lau (1973) method, the unit comes down the assembly line to complete the remaining tasks as possible if there is an unfinished task; then all unfinished tasks are completed off the assembly line. The desirable tasks list is composed by specifying the candidate (available) tasks list with marginally desirable tasks for assigning to workstations. A task is marginally desirable if its predicted labour savings at the particular position is greater than its expected incompletion cost. The tasks with main certainty to be completed are initially assigned to workstation according to decreasing incompletion time until desirable list becomes empty, new workstation can be opened. Kottas and Lau (1976) propose a new method to

review the total expected incompletion cost of a design. Their method searches for all potential combination of incomplete tasks and computes the related costs. Kottas and Lau (1981) improve different heuristic solution method that produces likely assembly line designs and they develop their first work by adding new selection rules and possible combinations principle for the desirable tasks list.

Buxey (1974) presents a computer program for multiple parallel workstations, integral tasks and spatial zoning. Then, Buxey (1978) improve COMSOAL for parallel workstations to decrease the total idle time of assembly line.

Dar-El (1975) compares MALB with COMSOAL and 12 different single – pass heuristic rules for SALBP-2. He chooses the balance delay and computation time as a performance measure for efficiency of the solution methods. According to research of Dar – EL (1975) MALB better than the other methods and COMSOAL is better of ten of the single – pass rules. Dar-El and Rubinovitch (1979) introduce a backtracking method called as *Multiple Solutions Techniques -MUST*" which produces alternative solutions with same quality by applying exhaustive enumeration eight exponential improvement of number of subsets. Their algorithm performs better results or same result comparing with mixed model ALBPs in each case.

Schofield (1979) develops a computer based method called as "*Nottingham University Line Sequencing Program- NULISP*" that can be applied for SALBP-1 and SALBP-2 by working with different zoning restrictions and task processing times exceed cycle time.

Raouf and Tsui (1982) improve a method for single model assembly line with deterministic processing time by applying theorem in which when a new task enters into a tasks group, the coefficient variation of new workstation time is smaller than the previously built old workstation time based on lack of correlation between the tasks. The method accepts that the task processing times are not known and symmetrical distributed. They applied priority ranking heuristic solution procedure.

Sarker and Shanthihumar (1983) introduce a heuristic method for serial and parallel lines, quite similar to Moodie and Young (1965). During the ALB process, they also consider task with processing times is larger than cycle time.

Akagi, Osaki and Kikichi (1983) propose a multi – pass heuristic solution method with two phases in which more than one worker can be assigned to a workstation. The task assignment procedure is applied based on priority rules and is iterated for a verified number of workers at single workstation. After the assignment of workers to the workstations, tasks can be assigned to workers at each workstation.

Shtub (1984) develops a heuristic method for balancing assembly lines with stochastic task processing times and various assignment design of workstations. Determining of the available and desirable tasks list of Shtub's (1984) method is quite same to Kottas and Lau's (1973), but the desirability of task depends on the number of workers at each workstation. The average processing time of each task is ac-

cepted as a non – increasing discrete function of the number of workers at the workstation with the assigned tasks.

Silverman and Carter (1986) analyse relations between stochastic task processing times and the total operation cost of an assembly line where the incompletions are solved by switching off the all assembly line to finish the work. Tasks are chosen randomly for assignment and they introduce a cost function as below:

$$E\,(\,TC\,) = (\,C\,.\,K\,.\,L\,) + L_a \int_c^{\infty} \big(\,1 - G(\omega)\big).\,d\omega \qquad (3.1)$$

In this cost formula to find the lowest total cost as an optimal solution; K is the number of workstations at the assembly line, L represents the labour rate, L_a is the overtime labour rate, ω is the maximum time of workstation to finish their tasks in a certain cycle time and $G(\omega)$ is the cumulatve distribution. Their method performs better than the assembly line problems that have higher overtime rate (Silverman&Carter, 1986).

Chakravarty and Shtub (1986) improve two heuristic methods for mixed model assembly lines with stochastic time. Their methods combined the labour cost with in – process inventory holding cost and machine setup cost and it is possible to have in – process inventories between the workstations; main aim is minimization of total operating cost of the assembly line considering the constraints. Their first method is a single pass heuristic approach based on positional weight calculation and second method is a shortest – path heuristic approach based on the consecutive ordering of the tasks.

Gustavson (1986) proposes heuristic methods to solve the single and multiple product equipment balancing problem. He develops additional solution method to abstain from non-serial assembly line designs and the method is used for the fixed assembly sequence with inconveniences.

Bhattacharjee and Sahu (1988) introduce a single model ALB method which applies on different constraints such as fixed position, multiple parallel workstations, positive and negative zoning. Their assignment procedure based on a priority rule that is the sum of each task processing time and its total number of successors. The method is able to analyse large number of randomly generated ALBPs.

Chakravarty (1988) presents the learning effects of eack task on the starting design of workstation at assembly line in order to find solution without large idle times. He assumes the ALBP as a dynamic recursive optimization model that states the bottleneck workstations and decreases the idle time of rest of workstations according to these bottleneck workstations. Chakravarty (1988) proves the equality between the minimization of idle time and bottle time of the assembly line by applying the optimization model as a shortest path problem. He analyses idle times of the assembly line by considering the learning effects and not considering them.

Shtub and Dar-El (1990) propose a method with two main objectives to balance mixed model assembly lines for SALBP-1 and SALBP-2 by practicing a set of zoning restrictions in order to restrict the tasks

of chosen subassemblies to a single workstation. Their method has four models and two main objectives as minimization of the total idle time and minimization of the number of subassemblies processed at every workstation in order to enhance working technics and improve the jobs of workers as well. The first model aims to minimize the number of workstations and the weighted sum of the total amount of subassemblies at assembly line. The second model purposes to minimize the number of cycle time and the weighted sum of the total amount of subassemblies at assembly line. At the third model, the objective is the minimization of number of workstations with the constraint in which every workstation does not process above than predetermined amount of subassemblies. The fourt model minimize the cycle time with the constraint in which every workstation does not process above than predetermined amount of subassemblies. The efficiency of ALBP depends on how much effort is implied separately on two main objectives of the method.

Shin (1990) improves a cost related method to minimize the expected total cost which includes the total labour cost and cost of the unfinished tasks that are extracted from the assembly line to be completed later. The method is applicable for every deterministic heuristic algorithms or methods by processing with a large cycle time initially. The cycle time is decreased by a predetermined quantity until cycle time becomes equal to lower bound of the task processing time. The expected total cost related with the assembly line design is computed and the line balance related with the minimum expected total cost generates the solution of the ALBP under the method.

Rachamadugu and Talbot (1991) propose a method for solving manual assembly lines by analysing the scaling of workloads across workstations. The objective function is an average absolute deviation of workloads condition to balance the workload between workstations. Furthermore, unbalanced assignments are accepted as unfair and require some management related action like differential pay. These levelling task assignment method is applicable for SALBP-1 and SALBP-2 to find the optimal solution by balancing workload between workstations.

Ahmadi, Dansu and Tang (1992) improve a heuristic method with three different procedures for dynamic allocation problem with assigning models to assembly lines to find almost optimal solution by objecting to decrease penalty costs and changeover costs. The number of lines are fixed and identical. It may define as a scheduling problems for multiple types of tasks on parallel machines.

Rosenberg and Ziegler (1992) introduce two new heuristic methods to solve cost-oriented ALB and they are “The Wage Rate Method” and “The Wage Rate Smoothing Method”. They accept that the processing of workstation results in a wage rate for each time unit which is even to the maximum wage rate of entire tasks of the chosen workstation. The aim is to decrease and balance the wage rate at all workstations as much as minimizing the number of workstations. The elements of task are assigned to the workstation in decreasing sequence of the tasks element wage rates, changing on behalf of the maximal processing time.

Bartholdi (1993) analyses two-sided assembly lines in which couple of workstations are designed at the different sides as right and left sides of the lines and every couple of workstations operate on one prod-

uct at the same time. He proves that two-sided assembly lines need fewer workstations than a known type of one-sided assembly lines by satisfying precedence and cycle time constraints. Bartholdi (1993) applies an updated version of a priority rule based heuristic to a software program that give an opportunity for applicants to fix some tasks to chosen workstations.

Pannerselvan and Sankar (1993) search the single model ALBPs with an objective of minimizing the number of workstations. They analyse Dar-El 's (1975) six single pass heuristic rules and they finalize their research with introducing more developed version of three heuristic rules of Dar-El 's (1975) and new six heuristic rules which are the most efficient set of heuristics for solving SALBP-2.

Rubinovitz and Bukchin (1993) propose a heuristic method which is based upon the branch and bound algorithms and called as "*RALB*" with line design in which various robot kinds can be used, to analyse single model ALBPs. They assumed that every substitutive equipment has a specified purchase cost. They aim to reach an optimal solution by minimizing the number of workstations with chosen production rate.

Malakooti (1994) presents a heuristic approach to balance assembly lines with multiple decision criteria which includes different objectives (cost related, precedence relations, buffers etc.) and constraints. He applied the multiple decision criteria to single model ALBP with positioning and dimensioning buffers.

Miltenburg and Wijngaard (1994) introduce U-shaped assembly lines to improve flexibility of grouping tasks to the workstations. They improve three procedures to balance U-shaped, single model assembly lines: Their first procedure is dynamic programming method which is based upon Held, Karp and Shareshian (1963), second procedure used ranked position weight technique of Helgeson and Birnie (1961) and third procedure is based upon the enumeration method of Hoffman (1963).

Süer and Dagli (1994) improve a heuristic method to find out the number of assembly lines, the number of workstations allocated to every model and the allocation of models to assembly lines for every cycle term in specific time. The amount number of assembly lines must be fixed and the cycle time for the workstation must not be overcome. Their optimisation criteria are minimization of the mean system response time and the makespan when there is a request to be delivered. The production system can compose of any number of assembly line and there is no limitation on number of workstations; there is only limitation on sum of the number of eligible workstations. They apply six various scheduling rules and the main problem of the method is that the entire task can be split tasks in a sense which they can be placed to the necessary number of workstations.

Kim and Park (1995) propose a heuristic method in which task processing times can be used as motivation and the method does not consider the instability between workers and operation time of assembly line. They create a mathematical model and cutting plane algorithm to solve SALBP-1.

Boctor (1995) presents a composite heuristic method for SALBP-1 by applying four decision rules to assign tasks to workstations at prioritizing plan and the assignment procedure is same as at single pass heuristic methods. Four decision rules of Boctor (1995):

1. Choose the task with same duration to remaining time of workstation. If there is no eligible task, go to next rule. To break ties, assign the task which has the largest number of subsequent candidates.
2. Choose the severe task, which has a processing time larger than or equal to half og-f the cycyle time, with the largest number of subsequent candidates. Ifthere is no severe tasks, go to next rule. If there is a tie among tasks, select the task with the largest processing time.
3. Choose the combination of two task with a duration equal to remanining time of workstation. If there isn't any combination like that, go to next rule. If tie occurs, take the largest number of subsequent candidates.
4. Choose the task with the greatest number of subsequent candidates. To break ties, select the task with the largest number of severe immediate successors and if there is still a tie, assign the task having the largest processing time.

Boctor (1995) applies these four assignment rules on forward and reverse ALB procedures.

Nkasu and Leung (1995) improves a heuristic method which is quite similar to COMSOAL with regard to the best design is chosen among a couple of created by simulation. The processing times and cycle time can be taken from different probability distributions at the procedure and they take the minimization of cycle time, the number of workstations, balance delay as performance measures of their method.

Scholl and Voß (1996) analyse the forward, backward and bidirectional priority rules for the assignment of tasks for SALBP-1 and SALBP-2 by presenting forward and backward rankings. They applied the backward procedure by reversing the precedence relation graph of assembly line problem and calculating rankings as inverse positional weight of Helgeson and Birnie (1961). At the bidirectional procedure, they use forward and backward procedure simultaneously by creating ranking for each procedure. They introduce a task as *backward available* if all successors are already assigned and as *backward assignable* to workstation k if task is backward available and any of its successors are not assigned to previous workstations. The bidirectional procedure begins with workstations k_f and k_b as current workstations in which k_f is equal 1 and k_b is equal to a really large number. In every assigning iteration, the task with the highest priority is selected and assigned to the chosen workstation by checking if the task is forward assignable to k_f or backward assignable to k_b. The bidirectional procedure ends when there is no task to assign. Scholl and Voß (1996) find out that bidirectional procedure produces better results with comparison both forward and backward procedures for SALBP-1.

Malakooti and Kumar (1996) present different heuristic methods with multi-objective which are cycle time, the number of workstations, buffer size and cost oriented objectives for ALBP. They applied three steps interacting procedure for minimization of total cost of the assembly line. Firstly, they use "ranked positional weight" rule of Helgeson and Birnie (1961) and search the necessary buffer size via

an empirical formulation. In the end, they are able to estimate the total cost of assembly line. Malakooti and Kumar (1996) improve an additive utility function according to user's choices and split the all problem to three basic parts; minimization of the number of workstations with a given cycle time, minimization of the cycle time with given number of workstations and minimization of the total cost of assembly line with given cycle time and chosen number of workstations.

Park, Park and Kim (1997) propose a heuristic method which is developed by additional constraints as task incompatibilities and range constraints to provide flexibility for precedence relations in situations lack of enough precedence information and solve according to network theories. The method has two sub-problems which can be optimized by using neighbourhood search process. First sub-problem is a generalized bin packing assembly problem and second one is shortest path problem with polynomial time bound. The method is applicable for practical situations with an improvement at production rate.

Askin and Zhou (1997) introduce a nonlinear integer programming to balance mixed model assembly line with parallel workstations according to idle time and cost oriented constraints. At their method, every task is assigned to a stage at the serial production system and find out the amount of same parallel workstations at every stage. The aim of this heuristic method is to find optimal assignment by reducing the total cost of the assembly line by applying greedy approach.

Minzu and Henrioud (1997) develop a stochastic method called as kangaroo algorithm to solve assembly line with a given number of workstations. The method decreases the maximum workload of the workstations to reach a better balanced line.

Ugurdag, Rachamadugu and Papachristou (1997) propose a heuristic method with two stages to balance SALBP-2 by minimizing cycle time and smoothing the workload between workstations. At the first stage of the method, the initial solution is created according to processing times to comprise the simplex table for heuristic balancing. At the second stage called as ALMap- Line Mapping, the optimal solution is improved from initial solution by applying simplex process.

Süer (1998) presents a heuristic method with three phases for a single model assembly line to minimize total number of workers assigned to workstations. At the first phase, tasks are variously grouped according to number of the workstations; at the second phase, the parallel workstations are chosen and the number of the workers for each workstations is chosen based on integer programming model for improving the production rate. At the last phase, the solution with best production rate for every number of workers is chosen for parallel assembly lines.

Sparling (1998) introduces a heuristic method for balancing a Just-in-time production unit with U-shaped assembly line and multiline workstations for SALBP-1. At the method, the number of U-shaped assembly lines are determined, every line is assigned to a single product and three versions of workstations which are regular, crossover and multiline can be used. You may reach each U-lines through *start area* and workstations are not allowed to cross paths; in addition, workstations with multiline do

not have more than two U-lines. This assembly line problem is named as N U-line balancing in which locations of U-lines are undetermined.

Merengo, Nava and Pozzetti (1999) analyse a manual mixed model assembly lines by applying balancing and sequencing process to minimize the rate of incomplete tasks and decrease WIP (work in progress). At the balancing process, heuristic method is applied according to weighted differences at four different versions including vertical and horizontal balancing to smooth the workload of workstations and minimize the number of workstations. At the sequencing process, they highlight the uniform parts usage for minimization of incomplete units for just-in-time production systems.

Sarin, Erel and Dar-El (1999) develop a stochastic method for single model ALBP to minimize the total labour and expected incompletion cost with a given allocation and sequencing of tasks and given number of the workstations. At their method, ALBP is divided into sub-problems from which an initial solution is reached by applying dynamic programming; then, the initial solution is developed at the improvement procedure by applying branch and bound balancing method that forms an approximate solution. At the final solution of improvement procedure, less workstations and less cost amounts are found.

Sysoey and Dolgui (1999) propose an iterative pareto optimization approach for production systems by applying heuristic processes of the selection part of ALB with multiple decisions. Their method is equipment selection that is resource planning part of ALBP.

Bautista et al. (2000) present a *Greedy Randomized Adaptive Search Procedure* which is created by using some heuristic rules based on priority constraints and a GA that solve the problem for heuristic space. They analyse ALBP by taking into account incompatibilities between the tasks as their first aim, then their second aim is minimizing the cycle time with a given minimum number of workstations. The main characteristics of classical greedy heuristics is used for their method; they propose the random selection rule by applying probability distribution which based on an index resulting arises from priority based rules and they named their method as a *Greedy Randomize Weighted Adaptive Search Procedure* that uses the introduction of random selection rules by adapting greedy heuristics into their problem. The solution of problem can be based on order of priority rules and the tasks can be chosen randomly based on the fitness value depends on a chosen rule.

Gadidov and Wilhelm (2000) propose a new brand and cut method for single product assembly system design problem and their method is combination of heuristic, pre-processing and two cut-generating processes. The aim is the minimization of total costs at assembly line design and total cost includes fixed costs of operating workstations, machines used for operating and variable cost of operating during the planning period. All processing times of tasks, costs are accepted as deterministic and every task can be operated at one of a group of alternative machines. At their first variation of the problem, assembly line is parallel and has same machines to be placed at every workstation; it is possible to operate with the tasks that have larger processing time than cycle time and improve workstation availabil-

ity. At their second variation of the problem, they focus on the positional constraints resulted by non-ability of product assignment to same workstations front and back side of operations.

Amen (2000,2001) improves a workstation oriented priority rules focused method, which considers cost rate and duration of tasks, called as "*best change of idle costs*" priority rule; with application of this new rule, it is possible to check the idle time and wage rates differences between workstations. This rule differs from other priority rules due to consideration of production cost as outcome from cost wage rates and production time. Amen (2000,2001) also proposes a heuristic method as "*exact solution of sliding problem window*" that is a heuristic version of an exact solution method.

Lee et al. (2001) introduce a heuristic group assignment method for two-sided ALBP and aim of the method is maximization of work relatedness and slack time between tasks. The group assignment allocates the task groups to mated workstations in deterministic times. The computation results show that their method is able to improve the work relatedness and slackness with a small amount or no changes in cycle time and the number of workstations.

Matanachai and Yao (2001) develop a heuristic method for mixed model assembly line to find well-balanced workload between workstations and forming the daily sequence of tasks which supplies consistent workloads for assembly line. Their heuristic method is applied by using a filtered beam search algorithm which finds out applicable subsets at every workstation; the subsets with best objective values for every workstation are kept and the subset with the best objective value is divided to form eligible subsets for following workstation. After the feasible solution is reached, the tasks are transferred from one workstation to another to reach a better objective value.

Bukchin, Dar-El and Rubinovitz (2002) analyse the make to order environment for mixed model ALBP by applying three-stage heuristic method to minimize the number of workstations with a given cycle time. They group the tasks into two sets: the first tasks group is eligible to be assigned to a single workstation for all model kinds asking this task, the second tasks group is eligible to be assigned to various workstations for different model kinds. The heuristic solution method has three stages: SALBP-1 is solved to find number of workstations and place the tasks to workstation at the first stage; at the second stage, reassignment of the tasks of every model occurs and the tasks of every model are particular for current model to save the previous assignments and optimize assembly line based on the chosen goal; at the last stage, local search is applied to differ the assignment of the tasks by using bottleneck measure and the solutions is completed by assigning particular tasks.

Urban and Chiang (2002) present a hybrid heuristic method for U-shaped ALBP, their method is based priority rules processes and tasks times are accepted as stochastic.

Liu and Chen (2002) propose a two-stage heuristic method with two objectives as minimization of cycle time and minimization of total operation cost of assembly line for balancing multisection assembly line problem. At the first stage, a multiple objective mixed integer zero-one programming model and related interactive process are developed to minimize cycle time and numbe rof the workstations by

fulfilling the desired total operation cost. At the second stage, simulation is reviewed to evaluate potential operation variability, buffer size, quantity of pallets and capacity constraints. To satisfy the two objectives at the same time, they use goal programming at the objective function of the mathematical programming model.

Jina and Wu (2002) introduce a heuristic algorithm called as "*variance algorithm*" at Just-In-Time system for mixed model ALBP. They analyse the relationship between qualified parts and remaining sequence.

Fleszar and Hindi (2003) develop an enumerative heuristic method and reduction techniques with the aim as minimization of quantity of workstations for ALBP. Their heuristic method is improved from Hoffman's heuristic method and solves ALBP at both directions of precedence relations to reach optimal solution. The reduction techniques are used at rising processing time of tasks and connecting tasks.

Karabati and Sayin (2003) analyse the assembly line at mixed model sequencing environments with synchronous transfers and relation between task assignments and the sequencing decisions for synchronous transfers; their objective is minimization of total cycle time by incorporating the cyclic sequencing information. Karabati and Sayin (2003) build a mathematical model which compounds multiple model into a single model by summing up processing times generates a lower bound for the mathematical formulation and introduce a heuristic method to minimize the maximum sub-cycle time of ALBP. The result of study shows that proposed method find better solutions but with more computational cost.

Dolgui et al. (2004) develop a heuristic method for transfer line balancing to assign all tasks to given number of workstations or machines and blocks by keeping transfer line cost as low as possible, not exceeding chosen cycle time and considering precedence and compatibility constraints. At the assembly line, the tasks of every workstation are grouped into blocks and all tasks of each block are operated by a spindle head. They propose two heuristic algorithms, which is based on COMSOAL, as *RAP Algorithm-Recursive Assignment of Predecessors* and *FSIC Algorithm- First Satisfy Inclusion Constraints*. The recursive algorithm analyses the all constrains of collections of tasks sets only after satisfying the actual workstation. At the other algorithm, tasks placed according to the constraints at one workstation are operated firstly.

Lapierre and Ruiz (2004) work on a case study in which an assembly line with two sides and two different heights is balanced according to priority based heuristic rules and they adapt the heuristic procedure to industrial problem. They prove that efficient usage of randomness and logic is important to reach a good solution. Lapierre and Ruiz (2004) implement their algorithm into a software as MsAccess97.

Erel, Sabuncuoglu and Sekerci (2005) apply a beam search method for U-shaped, stochastic ALBPs at the first time. A beam search is a heuristic brand and bound procedure which analyses through a search tree. They aim to minimize total labor cost and total expected incompletion cost of assembly line with a

given cycle time. Their study performs better than well-known heuristic methods for straight lien problems.

Fonseca et al. (2005) propose a fuzzy logic method for stochastic ALBPs. They use a fuzzy set theory that allows for review uncertainty at assignment of task times and cycle times. They modified COMSOAL and RPW by using fuzzy representation for the time variables.

Liu, Ong and Huang (2005) present a bidirectional heuristic method to solve the probabilistic assembly line problems with a given quantity of workstations and fixed assembly reliability that is the ratio of line workload not over than cycle time of assembly line. Forward task assignment procedure is used at the first step; then the tasks are changed between workstations till the cycle time is decreased.

Gokcen, Agpak and Benzer (2006) introduce a heuristic method with a mathematical programming model for balancing the multiple or parallel ALBPs by aiming minimization of the number of workstations while balancing at least one line at the same time. At their solution method, each line can be assigned to one product, two parallel lines process same production operation and cross-trained operators and two parallel lines are able to share specific workstations.

Gamberini, Grassi and Rimini (2006) analyse assembly re-balancing problem with stochastic time by improving new heuristic method called as "*Technique for order preference by similarity ideal solution*". Their approach is based on a well-known cost-oriented heuristic approach of Kottas and Lau (1973) and objects to minimize unit labor and expected unit completion cost, and tasks reassignments. Then, they also aim to avoid costs related equipment movement, tasks changement, worker trainings at new balancing procedure.

Jiano, Kumar and Martin (2006) create a web-based interactive advisor to balance assembly lines with the aim of minimization of amount of workstations, maximization of the output for each week and mean workstation utilization. The web-based advisor is able to apply primitive heuristic methods and includes a schedule for different types of heuristic methods in its memory.

Bukchin and Rabinowitch (2006) develop a branch and bound method with backtracking algorithm and a branch and bound based heuristic method for mixed model ALBPs by objecting minimization of total cost of the workstations and task duplication; enabling a mutual task to be assigned to different workstations for different models. Their branch and bound based heuristic method aims to decrease the search to find optimal solution at large- scale problems and the main process of method is cutting branches by small possibility of reaching optimal solutions. They analyse three heuristic rules as "greedy rule, gap=1 rule, reduced candidate list rule"; combine "gap=1 rule" and "reduced candidate list rule" for their heuristic algorithm.

Becker and Scholl (2006a) present a survey on problems and methods in generalized ALB to describe the developments and classify according to their layout, task time type, equipment selection, objective function.

Scholl and Becker (2006b) make comprehensive research on exact and heuristic solution methods for SALBP up to now.

Dimitriadis (2006) analyses paced assembly line with multi-manned workstations, in which group of workers can operate on various assembly tasks on the same product and workstation. Every worker starts operating task as soon as assembly line is feasible without considering the product is ready. Dimitriadis (2006) improves a heuristic method from ALBP's two limiting cases which are SALBP that one worker can work in every workstation and the single- stage sequencing problem with several similar machines. The heuristic method includes two- level process and is based upon modified version of Hoffman's method. The upper level finds all feasible and eligible to assign subsets of work elements to group of workers operating on the same product and workstation; the lower level assigns the tasks to every worker. The objective of heuristic method is decreasing the length of assembly line according to cycle time and precedence constraints while total effectiveness of assembly line remains stable.

Van Hop (2006) introduces a heuristic method for mixed model ALB with fuzzy task times as first time and the heuristic method is built by applying flexible exchange sequence process to assign tasks into workstations. The general system of heuristic method is to gather fuzzy time and the precedence graphic in fuzzy approach for product models converting into a fuzzy single model ALBP. Then tasks are assigned to workstations according to gathered fuzzy times considering technological constraints and cycle time until the best solutions is reached.

Boysen, Fliedner and Scholl (2007) categorize ALBP according to the objectives, workstation station and line characteristics, precedence graph characteristics; present a wide literature review based on solution methods of ALBP.

Agpak and Gokcen (2007) propose a chance constrained 0-1 integer programming methods for the stochastic traditional U-shaped ALBPs for minimization of quantity of workstations. A goal programming approach is explained to improve system reliability; their model can be used as validation tools for heuristic solution methods.

Boysen and Fliedner (2008) improve a versatile, two-stage graph algorithm to solve SALBP with different objective types such as SALBP-1, SALBLP-2, SALBLP-E, GALBP according to different constraints that are parallel workstations and jobs, zoning constraints, stochastic task times, resource and wage synergies, processing alternatives. At the first stage of *Avalanche-A Versatile Assembly Line Algorithm for Numerous Characteristic Extensions*, the precedence graph is used to construct a sequencing of tasks. When a feasible sequence is reached, the tasks are eligible for assignment to workstations by solving a shortest-path problem. Avalanche is a flexible algorithm which can be modified according to various constraints.

Boysen, Fliedner and Scholl (2008) define ALBP and classify the extensions of ALBP according to the objectives, precedence diagram characteristics, workstation and line characteristics. They categorize the ALB with respect to number of the model used, line control, frequency, automation level.

Andres, Miralles and Pastor (2008) analyse the balancing and sequencing of simple assembly lines by adding sequence-dependent setup time restrictions in a way that a task is placed next to other at the same workstation, a setup time is summed up to calculate the global workstation time. They explain their method on the mathematical model; eight different heuristic rules with task-oriented and workstation oriented objectives are designed and applied according to several task selection ordering measure. They also apply *greedy randomized adaptive search procedure* algorithm to find a solution for the ALBP.

Miralles et al. (2008) present a new ALBP called as "*Assembly Line Worker Assignment and Balancing Problem* " which is application for sheltered work centres for disabled; they apply a branch and bound procedure with three search strategies and different factors, they modified heuristic methods to depth-first-search algorithm for to solve large problems. The problem gives two results as solution which are the assignment of task to workstations and the assignment of eligible workers to workstations and aims to assign more tasks for disable workers.

Grzecha (2008) analyses time and cost oriented single ALBP in which only one product can be operated; time oriented assembly line problem decreases the idle times of workstations and a cost oriented assembly line problem reduces the manufacturing cost of final good. Author applies eleven different priority based heuristic rules for both problem and additional measurement that affects the line efficiency, smoothness index and the time of line.

Xiaofeng, Erfei and Ye (2008) develop a workstation-oriented enumerative assignment method for two-sided ALBP and combine their method with Hoffman`s heuristic method to optimize two-sided assembly lines. They introduce the time transfer function and integrate with precedence relation constraint to calculate the earliest and the latest starting time of tasks. The workstation-oriented method based upon the start time is planned for task assignment considering direction and cycle time restrictions by beginning from the left workstation to the right workstation of the place. They apply their algorithm on the benchmark set of instances and their algorithm gives efficient results.

Bautista and Pereira (2009) introduce a dynamic programming based heuristic method called as Bounded Dynamic Programming with graph search exploration to solve SALBPs by minimizing the number of workstations. Their deterministic method applies heuristic procedures to decrease amount of all states and uses bounds to decrease the search space.

Becker and Scholl (2009) consider ALBPs with variable parallel workplaces and improve a branch and bound algorithm which is also applicable as heuristic method. Becker and Scholl (2009) work on lower bounds and reduction rules for their method and they modify the branch and bound algorithm by adding two more time restrictions. After reaching the first feasible assignment, the heuristic solution method heuristically examines all nodes on the present branching path and goes back to the root node of network after specific time. The heuristic version of branch and bound algorithm performs better at large sized problem with larger cycle time.

Ege, Azizoglu and Ozdemirel (2009) develop an exact and heuristic based branch and bound algorithms for ALB with workstation paralleling by objecting minimization of total workstation opening cost and total tooling cost. Their method assigns random quantity of parallel workstations to very stage and performs better at medium sized optimization problems. They prove that their heuristic based branch and bound algorithm performs better at large problems with low tooling cost, finds nearly optimal solutions.

Guschinskaya and Dolgui (2009) compare well-known exact and heuristic solution methods for transfer line balancing problem in order to minimize total quantity of the necessary equipment by aiming to group the tasks into blocks and to assign the blocks to machines. They analyse four different mixed integer programming, the shortest path method, FISC heuristic algorithm, deterministic decomposition based on precedence diagram, heuristic multi-start decomposition and the aggregate solving which considers previously solved sub-problems by changing some assignment processes. They find out that exact methods should be applied for small sized problems, for medium sized problems the shortest graph method should be used and the heuristic multi-start decomposition with aggregate solving of sub-problems should be used for large sized problems.

Yeh and Kao (2009) create a new bidirectional heuristic method by combining with *the critical path method* to examine the task assignment process of ALBP. They initially apply the critical path method to get the critical tasks which ranks in higher precedence for assigning of tasks. After critical method, they apply bidirectional method for random task assignment as forward and backward directions. Computational results show that their heuristic method performs effectively.

L. Capacho et al. (2009) work on Alternative Subgraphs ALBP that considers versions for different parts of production. ASALBP includes two sub-problem: the decision problem for deciding one assembly sub-diagram for every sub-assembly with the usage of alternatives and the balancing problem for assignment of tasks. At the problem, there are group of tasks for that several choices of assembly variants are eligible and the tasks must be allocated to the set of workstations. Every variant of every sub-assembly is shown by separated sub-graphs and chooses tasks for assembling and their precedence order; total task operation time can be different between assemblies. They make comparative analysis by applying 39 single-pass and 17 multi-pass heuristic methods based on random selection and priority rule as decision criteria. According to their experiments, multi-pass heuristics performs better than single-pass heuristics while applying random selection for sub-graphs.

Toksari et al. (2010) propose effects of learning and deterioration task in ALB with an objective of minimization of workstation quantity by improving mixed integer programming model and modifying COMSOAL method for such SALBPs. They apply same learning and deterioration rates at workstations, but learning and deterioration for every workstation reinitializes due to difference of workers or machines at workstations. COMSOAL method is adapted to solve large sized SALBPs with learning effects and deterioration tasks. They find out that when problem size or cycle time goes up, learning effect is more dominant in case of raising amount of tasks at every workstation. Based on computational experiments at large scale problems, the learning affects more than the deterioration.

Kilincci (2010) introduces a Petri-net based heuristic method for solving SALBP-2 by applying reachability analysis and the token movement at assignment process. He iteratively solves the problem by using different trial cycle times; if the cycle time is not convenient for a given quantity of workstations, the heuristic methods raises the cycle time until reaching an optimal solution; a binary search approach is applied between the first feasible and the last feasible solution to enhance the solution. Three versions of the heuristic methods are introduced by implementing forward, backward and bidirectional assignment procedures. According to computational results and comparisons, his heuristic method performs well for SALBP-2, it is perfectly suitable for large scale problems.

Martino and Pastor (2010) analyse the GALBP with setup times where every time a task is allocated to following to another at the same workstation, a setup time must be added to calculate the total workstation time during arranging the task order for every workstation. They improve heuristic methods based on priority rules for GALBPS with aim of minimizing the quantity of workstations; the heuristic methods are a workstation-oriented method based on not-weighted priority rules, a task oriented method with priority rules, a workstation-oriented method based on weighted priority rules and improved task assignment schemes by checking of all positions which an applicant task can be allocated, processing a local optimisation after a workstation is assumed as closed, processing a local optimisation at each time a new task placed to a chosen workstation. The experimental results prove that the scheme of local optimization of the tasks assigned to a workstation after each assignment outperforms.

Lee (2010) presents a modified heuristic mixed model assembly line method to ensure stable workstation assignments on a model by mode as well as on a workstation based. His modified heuristic method is developed from Thomopoulos method (1970) and applies Hoffman's precedence matrix (1963) and Arcus's method (1966) for reaching quickly to large amount of feasible assignments. The experimental results show that modified algorithm decreases the fluctuations in processing times between the models as well as workstations and balance delays.

Jonnalagedda and Dabade (2010) define seven priority rule with a workstation oriented heuristic method for mixed model ALBP to minimize cycle time with a given number of workstations. They implement cycle time, model variability, workstation variability and combination of these objectives to direct the heuristic method. Bottleneck measure that engages in variability is used for modelling the priority rule. The computational results show that according to the performance based on cycle time, the priority rules as positional weight or positional weight integrated with model variability and/or workstation variability outperforms other approaches in the research. The application of the priority rule based on mean positional weight or combination with model variability and/or workstation variability performs better than used priority rules.

Yegul, Agpak and Yavuz (2010) create a new hybrid design which is combination of two-sided and U-shaped assembly lines and apply a multi-pass random allocation process to minimize the quantity of workstations. One part of assembly line is organized as U-shaped line allowed workstation with crossover and the other part of assembly line is balanced as a traditional straight line flow; they name the new design of assembly line as *Two-sided ALB with One Side in U-Shape.* It is possible to find two

different balances (quantity of workstations) according to left or right side of the line having the U-shaped line design. Based on computational results, it can be solved in low computational time for small scale problems and provides a solution with less workstations.

Kilincci (2011) proposes new heuristic algorithm which applies an order of firing sequence of transition based on Petri-net approach of precedence graph to solve SALBP by minimizing the quantity of workstation with a given cycle time. The proposed firing sequence backward algorithm-FSb with single pass heuristic rule and Petri-net approach has two stages. At the first stage, the firing sequence backward algorithm finds firing sequence of transitions considering token movement in the network; at the second stage, the proposed algorithm allocates the last task of the sequence list to the last workstation applying backward method. Kilincci (2011) compare new algorithm with single pass, multi pass and iterative backtracking heuristics by implementing on Talbot's and Hoffman's benchmarking problem sets and categorizing analyses into problem size, data sets and order strength. The experimental results prove that firing sequence algorithm performs well at Talbot's sets for small and large scale problems and the algorithm also provides better solution than single pass heuristic methods at the literature.

Bautista and Pereira (2011) develop a bounded dynamic programming based method for time and space constrained ALBP with a given cycle time and space availability for minimization of number of workers. The time and space constrained ALBP considers the space requirements of machine and assembly equipment and their algorithm can be explained in two parts. At the first part, the process for enumerate states is based upon the Hoffman heuristic (Hoffmann,1963) which consequentially reaches each task assignment to workstation to result with an optimal one; they add new search limit for the number of task assignment amount to be reach and apply four different heuristic evaluation techniques to find the feasible task allocation. At the second part, their method solves the traditional ALBPs caused by dynamic programming with an exponential amount of positions applying heuristic rules to decrease the state space and their aim is limitation memory and time requirements connected to final application of dynamic programming formulas. Their bounding method is applicable for two different fields that are fast lower bounds for enumeration process for heuristic and exact methods, and tight bounds based upon on a mathematical formulation to compare solutions quality. According to computational experiments, their method performs better than other solution methods for time and space constrained ALBP with minimization of number of workstations. Additionally, their method also can be applied for GALBPs.

Fazlollahtabar et al. (2011) introduce a heuristic method based on RPW Algorithm for solving stochastic ALBPs with stochastic activity time parameters by minimizing number of workstations; they also apply another approximated approach which calculates the integral curve without capability and Monte Carlo simulation. At the proposed heuristic method, the time for every activity uses normal distribution with the mean and standard deviation value and stochastic times are converted to probabilities by applying standard normal distribution table. According to experimental results, Monte Carlo simulations proves that proposed heuristic performs efficiently by finding same number of workstations as solution.

Hu (2011) presents a heuristic algorithm to solve two-sided ALBP with multi-objectives as minimization line length and smoothness index. The heuristic algorithm includes two stages: at the first stage, a branch and bound algorithm is used to find feasible solution with the minimum line length and the algorithm allocates jobs to workstations to decrease the amount of positions and at the second stage, the weight of workstations is rebalanced for minimization of smoothness index according to precedence and line length constraints. An example is applied to present the process of heuristic algorithm and the better result is found.

Yin, Su and Wu (2011) analyse a heuristic method to solve the two-sided with multi-parallel workstations ALBPs with positional constraints. The aim of their method is minimization of number of opened positions and the workstations. At the multi-parallel workstations, two workstations operate randomly at opposite sides of same task and they define as the workplace occurred by workstations as position. At the heuristic algorithm, each task must be allocated to only one position and workstation. The proposed heuristic method is effective to apply for bus and track assembly factories.

Pastor (2011) creates a new ALBP called as "*Lexicographic bottleneck ALBP"* that reduces the workload of the most loaded workstation pursued by the workstation with second most workload and pursued by the workstation with third most workload and goes on. Pastor (2011) designs two mixed integer linear programming method and three heuristic method based upon mixed integer linear programming. The mixed integer linear programming methods are *the global hierarchical model* that minimizes a weighted sum of function with enough variable weights to protect the hierarchy between presented objectives and *the successive hierarchical model* that performs a pre-emptive goal programming to minimize the objectives when the optimal values of the objectives with the highest priority have been reached. The three heuristic methods comprise of running these two mixed integer programming methods for limited computation time. According to experimental results, the heuristic method based on successive hierarchical model gives better results than other methods.

Pastor, Chueca and Villoria (2012) improve a new algorithm which compounds a heuristic method for getting an initial solution and a few local search processes for solving lexicographic ALBP. The improved heuristic method is based upon dividing iteratively sub-problems into smaller sub-problems and balancing every problem as a SALBP-2 (minimization of cycle time with a given number of workstations); the found solution of every sub-problem is utilized to renew the global solution of the real problem and a local search is performed at every iteration of heuristic method according to exploration of the actual workstation. The main characteristic of heuristic method is the usage of local search process based on trade and transfer of tasks between a pair of workstations at every iteration of method.

Moreira et al. (2012) consider the assembly line worker assignment and balancing problems by applying heuristic methods based on task and worker priority rules. They implement three worker and sixteen task priority rules and the main idea of their method is usage of task and worker priority rules to identify which operator and which group of tasks will be allocated to every workstation. The heuristic methods perform workstation oriented assignment process according to selected priority rule to minimize the cycle time with given number of workstation. According to experimental results, tasks priority

rules perform better than other heuristic rules for every instance. In addition, they also use the presented heuristic method as a solution decoder for a hybrid GA which optimizes significant priorities for every task-operator set.

Avikal et al. (2013) compare the labour productivity in U-shaped assembly lines and traditional straight assembly lines by implementing bidirectional assignments with heuristic based critical path method (CPM). The introduced heuristic method is modified version of the method of Yeh and Kao (2009) but at the introduced method, tasks can be allocated to same workstation from either end of the assembly line or precedence network at U-shaped assembly line system. The heuristic method has four steps for task assignment process: determining the critical path of precedence diagram of the problem at the first step, design of new virtual workstations at the second step, the assignment of suitable tasks to virtual workstations at third step and the last step is converting virtual workstations to actual workstations. Computational results prove that their heuristic method works well and improves labour productivity with smaller number of workstations.

Gao et al. (2013) work on a mathematical model and developed algorithm based upon heuristic rules that apply cumulative RPWas process for task allocation and plan the orders of each branch. They aim to minimize number of workstation and length of assembly line by implementing heuristic rules to decide priority of branch nodes and allocate them considering precedence relations. The applied heuristic rules are as early-stating-time rule to decrease the waiting time, selection tasks according to operational direction constraints when there are multiple tasks and task assignment according to decreasing cumulative RPW when there are multiple tasks. They solve ALBP with sixteen tasks and their algorithm produces good result.

Jaturanondo et al. (2013) introduce a heuristic method for finding a task-workstation assignment solution that decrease balance day and improve the smooth postural load among operator at the assembly line by implementing the algorithm of Kilbridge and Wester (1961) for getting an initial solution. A composite index of variation is determined as measurement for the effectiveness of the solution. A task reassignment algorithm is implemented for reaching the initial solution by reallocating tasks to new workstations in order to have a minimum variation. They also consider that the method may allocate tasks to workstation equally. They define two quantitative measures based upon the workstation processing time and the workstation grand score that is total of the grand scores from whole tasks, for tasks-worker assignment solution. The composed index of variation permits common valuation of the workstation processing time and the workstation grand score. After trying their method on an example of clothes assembling, the computational results show that the heuristic method gives good results for large scale ALBPs with smooth postural load.

Great and Offiong (2013) propose a heuristic approach as the Longest Operation Time method for increasing productivity in breweries companies. The Longest Operation Time method is heuristic assignment rules that allocates tasks according to decreasing operation time; the objective is minimization of number of workstation and labour and idle times of assembly line as well. The case study shows that application of heuristic method improves line efficiency and decreases the idle time.

Gurevsky et al. (2013) analyse a stability measure for feasible and optimal solutions of GALBP with parallel tasks according to potential alternatives of the operation time of tasks by applying heuristic method to reach a compromise between objective function and the required stability measure. The initial optimization problem has two aims as improving total cost of the line and its robustness; Pareto optimality is applied to reach a solution to satisfy both objectives. At first, feasible solution has just one workstation including one empty workplace, then the heuristic method allocates tasks to this workplace until there is no task to allocate with regard to problem constraints; then, a new empty workplace is opened and assignment process starts. The experimental results suggest a probability to contain a robustness measure during the design process and utilize line configurations in terms of line stability by trying small variations of tasks operational times.

Scholl et al. (2013) improve Setup ALB and Scheduling Problem by using more realistic setups, defining more compact mathematical model formulation and applying efficient heuristic method. This problem creates a minimum amount of ordered workstations loads such that every task is allocated to only one workstation, the precedence restriction is considered and workstation times are not greater than cycle time with regard to cyclic task with sequence-dependent setup times. They consider this type as a mixed binary linear model that has the procedure of simple assembly line traditional model and form of traveling salesman problem; they develop a new joint balancing and sequencing problem that combines setup times of assembly system through diversification between backward and forward setups. Scholl et al. (2013) present a toolbox composed of different heuristic processes that are applicable for real life instances to find solution with high quality. In their heuristic method, they apply modified version of greedy randomized adaptive search procedure with random selection of priority rules; they use different procedure, that firstly calculates priorities for entire tasks with eligibility for the instant position of workstation sequence at re-optimization process; they accept an upper bound for minimum productive time at fathoming; they make developments at shortest path calculation and use lower bound and data reduction. According to computational results, new heuristic method outperforms other method on quality of solution and computational times.

Su et al. (2014) consider the mixed model ALBP in order to maximize the efficiency of assembly line by implementing Petri-net based heuristic method and creating a mathematical model. The heuristic method includes two stages: at the first stage, Petri-net model with a P-invariant algorithm is applied to reduce the quantity of workstations and at the second stage, P-invariant algorithm with binary search algorithm is used for minimization of cycle time with a given quantity of workstation reached at the first stage. Based on example and experimental results, the proposed heuristic minimizes the idle time of models and maximizes the line efficiency; performs well for solution accuracy and large scaled problems.

Otto and Otto (2014a) define general design policies on combination of priority rule-based methods to build good performing assembly line systems, ensure a cross-validation of the computational results and sample how to apply the formulised design policies. They submit that there are some policies to have solution with better quality and they categorize these policies as principles of aggregation, combination, structural specificity and principle of random influences.

Otto and Otto (2014b) form a new problem type as "*ALB with Learning Effects"* by introducing exact methods for small-medium scale problems and heuristic method for large scale problems. They focus on the initial process of ALB as a beginning of production step and modify ALBP by integration the learning stage and permitting individual learning curves for every task. Their methods have two objectives as minimization of quantity of workstation after learning ensues and minimization of length of learning stage. The introduces solution methods are exact methods of learning ALBP, branch and bound algorithm based methods and heuristic method as *Priority Rules Based Method for ALB with Learning Effects* that allocates a priority value in decreasing order to every task regarding construction scheme as workstation oriented or task oriented. The priority values are based upon task times, the quantity of predecessors or successors and their cumulative time and learning stage times. Experimental results show that their heuristic method produces good solution for large scale problems in a short time.

Borba and Ritt (2014) develop a mixed integer programming model, a novel heuristic method based upon beam search and a task oriented branch and bound method in order to minimize cycle time and maximize the production rate for assembly line worker assignment problem with given number of workstations. Their heuristic method is based on probabilistic beam search procedure and operates for verified candidate cycle times from an interval concluding at the present upper bound. The probabilistic beam search tries to reach a feasible assignment for every candidate cycle time and divides the workstation oriented assignment process into two perspectives: at the beginning, the proposed method selects one of the eligible tasks with a probability proportional according to chosen priority during allocation of tasks to actual workstation; then it implements beam search to reach the best allocation of workers and their tasks. According to computational tests, their heuristic methods perform better than well-known heuristic methods in comparison to computational time and quality of results.

Manavizadeh et al. (2015) propose a new heuristic method for U-shaped mixed model ALBP in order to find a good combinational of tasks balancing and models sequencing as multi-objective by applying three aims as minimization of cycle time, minimization of wastages at every workstation and minimizing the work overload of assembly line. They implement their heuristic method with an initial solution which may be found by an exact or metaheuristic method, so every single objective model is solved by an initial balancing method. Then, they use these results at other objective models, reach the objective solution. After that, they have a multi-objective function from the initial method, they search for a feasible solution until the stopping criteria is reached and their heuristic method is used to improve the solutions. They also compare the effects of straight assembly line and U-shaped assembly line for small size and large size problems. The experimental results show that the heuristic method can produce better results than initial methods; it is possible to enhance the solutions at a minimum for one model of three models and the decision maker can determine the design of assembly line.

Moreira et al. (2015) present a new ALBP coming about conventional and disable workers of the assembly line, called as "*Assembly Line Worker Integration and Balancing Problem*" that increase the productivity of assembly line by decreasing the quantity of workstation during integration of quantity of disabled workers into assembly line. They enhance an integer linear model for minimization of the

quantity of workstation while considering occurrence of disabled workers, then one version, which decreases the idle time of workstation with disabled operators, is presented. They apply a heuristic method named as *Constructive Insertion Heuristic* which begins with SALBP process and adds the eligible disable workers in order to decrease the quantity of workstations. The heuristic algorithm work in way that firstly, without consideration of heterogeneous workers and application of SALBP-1 process; splitting remaining assembly line into segments; checking entire eligible heterogeneous workers in every workstation of the first line segment; choosing the best allocation and arranging the solutions on workstations prior to the chosen one; these steps continue until there are no workers to allocate to workstations. In addition, two post-optimization processes for the presented heuristic method are implemented according to neighbourhood search of mixed integer programming.

3.3 Metaheuristic Solution Methods and Comprehensive Literature Review for Genetic Algorithm

The concept of metaheuristic implies to a kind of optimization methods for reaching near-optimal solutions. The metaheuristic solution methods are iterative generation processes operates as algorithm and modify heuristic methods to find the search space efficiently.

The efficiency of metaheuristic method depends on the balance between diversification ad intensification. *Diversification* explores various search areas in the search space, while *intensification* utilizes from the explored search areas to find hard-charging solutions. Blum and Roli (2003) summarizes the main characteristics of metaheuristic method as follows:

- Metaheuristics are guide for search processes.
- The aim is efficient exploration of search space to obtain (near-) optimal solution.
- Metaheuristic algorithms are applicable for verified processes, from local search procedures to complex learning processes.
- Metaheuristic algorithms produce approximate results and generally non-deterministic.
- Metaheuristics avoid be kept in confined regions of the search space.
- The basic conceptions of metaheuristic methods allow a summary as description.
- Metaheuristic methods are not specified according to problem types.
- Metaheuristics apply domain-specific knowledge in the heuristic methods which are under control of upper level strategy.
- Nowadays, more improved metaheuristic methods apply search experience to guide the procedure.

There are several categorizations for the metaheuristic methods according to the focus of objective function and characteristics, such as population-based or trajectory-based (single solution-based), nature-inspired or non-nature-inspired, static or dynamic objective function, local or global search capa-

bility, memory-based algorithm or memoryless, implicit or explicit or direct metaheuristics etc. (Blum&Roli,2003). Figure 3.2 summarizes the classification of metaheuristic methods as graphic.

At the next sections, the well-known metaheuristic methods are shortly described; the comprehensive description of GA and literature review for GA is analysed.

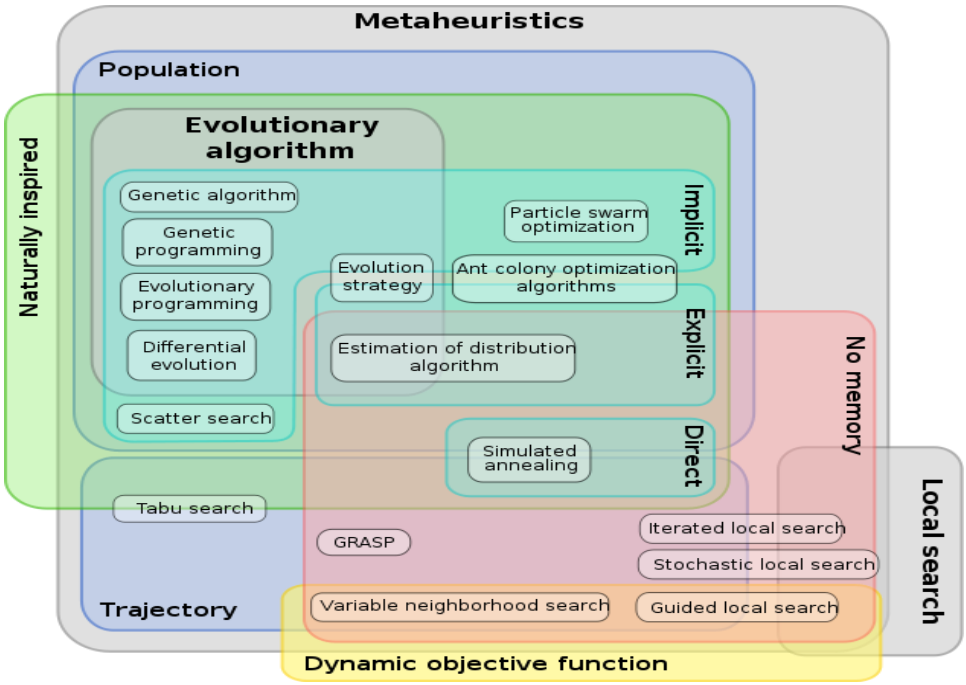


Figure 3.2 Graphical Classification of Metaheuristic Methods (*www.metah. nohjan.net*)

3.3.1 Simulated Annealing

Simulated Annealing is one of the oldest and most popular metaheuristic method based on single-solution process and proposed independently by Kirkpatrick et al. (1983) and Cerny (1985) with inspiration from cooling process of liquids. The main idea is permitting moves resulted in solutions with worse performance than present solution to avoid being trapped in local minima.

The aim of cooling process is collocation of atoms in the most ordinate way at crystalline procedure and the quality of the procedure at final arrangement depends on cooling rate. If the cooling process is applied really fast or initial temperature is not chosen high enough, the annealing process produces inefficient results. On the other hand, if the cooling process is enough slow, a proper atomic collocation can be reached and the annealing process obtains high quality crystals. One of the most important feature of cooling process in nature is the possibility of high energy generation. The true probability which a state with energy E happens at temperature T is determined based upon the Boltzmann probability distribution. When this probability increases for greater values of T, then it permits the independent moves of atoms and states of high energy (Zapfer, Braune&Beogl, 2010).

The formulation of Simulated Annealing depends on the *Metropolis Algorithm* from the statistical mechanics. It illustrates a thermodynamical procedure by generating an alignment of states or configurations at chosen temperature. A new configuration may be gained from an available one by randomly dislocating of an atom. ΔE represents the energy difference between two configurations. If dislocation yields a fall in energy level such in case of $\Delta E \leq 0$, the new configuration becomes acceptable. When $\Delta E > 0$, it is possible to accept the new configuration according to probability named as the *Metropolis acceptance criterion* that is written below (Zapfel, Braune&Beogl, 2010):

$$P(\Delta E) = \exp(-\Delta E / bT) \tag{3.2}$$

Zäpfel, Braune and Beogl (2010) define the Simulated Annealing as the repeated implementation of the Metropolis Algorithm for non-increasing alignment of temperature values *T* according to assumptions explained below:

- Every configuration accounts for a solution for a chosen optimization problem.
- Dislocating a single atom is even for implementation a modified to a solution.
- The energy of configuration has an indirect effect on quality of the solution.
- Simulated Annealing needs a cost function *C* for a description of general formulation. Based upon the cost function, the *Metropolis acceptance criterion* can be modified for applications of SA, as following:

$$P(\Delta C) = \exp(-\Delta C / T) = 1 / (\Delta C / T) \tag{3.3}$$

Simulated Annealing can be categorized as homogenous and inhomogeneous: *Homogenous SA* fixes the temperature value during every run of Metropolis algorithm and its aim is to permit the system in order to attain an equilibrium at every temperature level; *inhomogenous simulated annealing* instantly updates the temperature after every Metropolis trial and violates the attaining of any kind of equilibrium state at a chosen temperature value (Zapfel, Braune&Beogl, 2010).

3.3.2 Tabu Search

Glover (1986) presents the Tabu Search based upon earlier improvements of Glover (1977). Tabu Search is created according to special memory structure that states exact solutions set as *tabu* at every repetation; Tabu Search uses the search history to improve local search as basic components and applies a *short term memory* in order to avoid to be trapped in local minima and cycles (Blum&Roli, 2003). Zäpfel et al. illustrate the solution process of Tabu Search and use *s* and *s`* to symbolize the present and the following solution.at Figure 3.3 and summarize the general concept of Tabu Search below (Zäpfel, Braune&Beogl (2010):

- When Tabu Search goes by a new solution, it selects the *best available* solution.

- If it is required, Tabu Search permits temporal deteriorations for quality of the solution.
- Tabu Search is based on a memory structure to escape to be kept in cycles.
- Tabu Search generates entire solutions accessible by implementing the chosen modification steps.
- Tabu Search selects the best solution from this set that is not included by the memory.

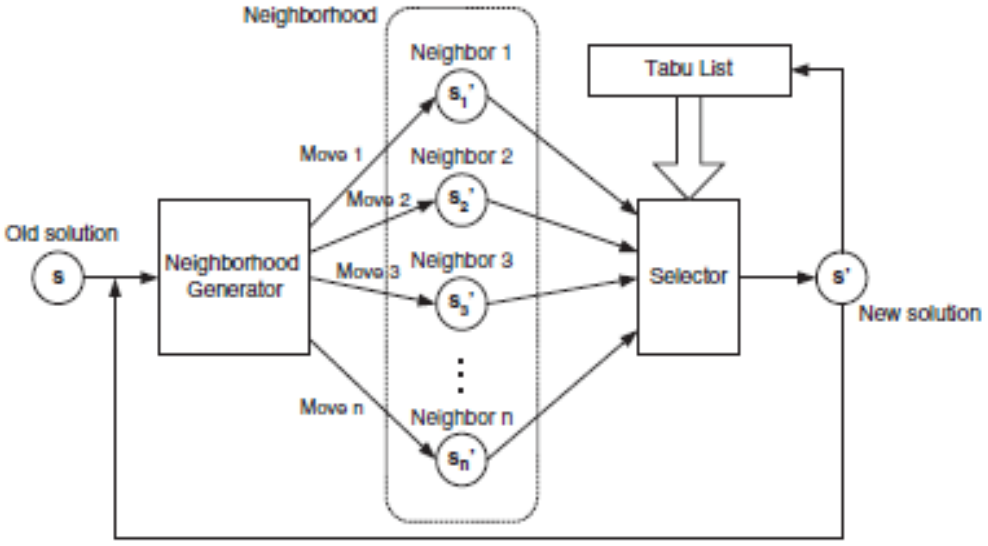


Figure 3.3 Tabu Search Detailed Solution Processing (Zäpfel, Braune&Beogl, 2010, pp.103)

The short term memory is applied as a *tabu list* which records the most lately visited solutions and prevents moving back to these solutions in order to getting caught to endless cycling. The neighborhood of present solution is thereby limited to the solutions which are not at the tabu list. At every iteration, the new present solution is selected from best solution of allowed set and put to tabu list, then a solution that were already placed in tabu list is excluded from the list according to determined order. The algorithm ends until a termination condition is reached. The length (capacity) of the tabu list decides the *tabu tenure* checks the memory of search procedure (Blum&Roli,2003). Tabu list with stable length cannot exactly escape the generation of cycles. Two main procedure have been improved to enhance cycle protection (Zäpfel, Braune&Beogl, 2010):

- Diversifying the list capacity over time
- Diversifying the tabu tenure of every tabu element

Attributes are used as key components of solutions in order to notice the tabu functionality. Chosen attributes of lately found solutions are stored in the list and whole new solutions including one of chosen attributes are accepted as *tabu*. The group of attributes and related tabu lists describe the *tabu conditions* that are filters to solution neighborhood and create the allowed set (Blum&Roli,2003).

3.3.3 Ant Colony Optimization

ACO is firstly proposed by Dorigo (1992) inspired by foraging behaviour of ants in the nature. The foraging behaviour makes easier for ants to get the shortest paths between their nest and food resources by following the trial of the most condense pheromone that is deposited at their metabolism. This behaviour is the fundamental for a collaborative interaction that results with occurrence of shortest path (Blum&Roli,2003).

ACO algorithm depends on a parametrized probabilistic model, *pheromone model*, which is used for formulation of chemical pheromone trials. To find the shortest path at optimization problem, an artificial *pheromone trial* is allocated to every edge between the start node and the target node. Every artificial pheromone trial is symbolized by a value that is created by the ants during their pass of the related edge. During passing an edge, an ant updates the pheromone amount by rising it applying a stable amount. While ants are trying to reach their aimed nodes, they also must make decisions beside saving pheromone. When a specific ant reached to a node that is linked to multiple follower nodes, the decision must be settled. The possibility of selecting significant path depends on the proportion of the pheromone concentration of the path. The moves of several ants starting from star node and leading to the target node will rise the pheromone condensation on the paths and will result with occurrence of the shortest path in the end (Zäpfel, Braune&Beogl, 2010).

Zäpfel, Braune and Beogl (2010) illustrate the detailed solution concept and components of ACO at Figure 3.4.

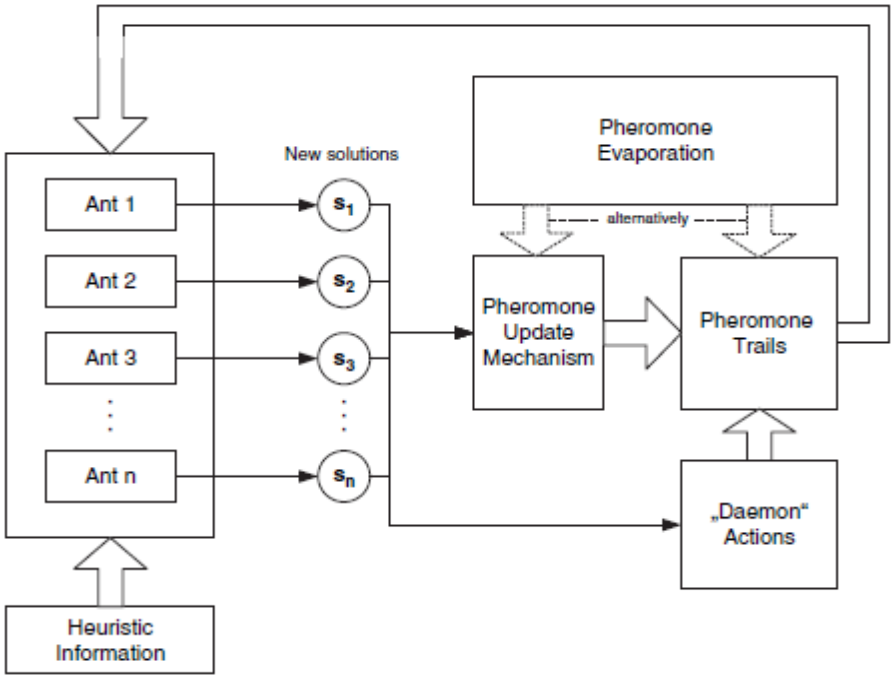


Figure 3.4 Detailed Graphics of ACO solution process (Zäpfel, Braune&Beogl, 2010, pp.88)

Heuristic information supplies a specific information for the optimization problem by identifying best potential component for following selection. As we know, the pheromone trails are updated directly an ant makes decision. When an ant has already get a solution, it can reuse the same path by saving the path at its memory; the pheromone trails of implemented component are updated with regard to reached solution quality. *Pheromone evaporation* is significant component of ACO by reason of hindering the search from being trapped without reaching a (near) optimal solution. Pheromone evaporation also has a diversification impact due to reducing the pheromone concentration. The autocatalytic impact leads to a fast increase at the pheromone concentration of trails that are part of at least one related good solutions; so that, the probability of choosing various components decreases at solution construction process. Pheromone evaporation prevents the search procedure from rapidly resulting with a sub-optimal solution by regularly decreasing the concentration on the trails. *Daemon actions* are methods that are implemented from global view and cannot be fulfilled by single ants. Based on the activities they carry out, they notice diversification or intensification or both. Intensification is successfully concluded by more improvements at the solution by ants or by offline pheromone updates based on the best solution got at the present iteration. In other respects, daemon actions can reduce pheromone trails to improve exploration (Zäpfel, Braune&Beogl, 2010).

3.3.4 Particle Swarm Optimization

Particle Swarm Optimization is one of the main swarm intelligence optimization methods introduced by Kennedy and Eberhart (1995). The Pparticle swarm optimization is created based on the inspiration from existing motion of a flock birds which is flying around for finding food source; the motion of every individual bird of the flock is affected by potential foraging areas which the bird has visited as yet and by the movements of other birds at the flock. Likewise, a swarm of individuals considered as particles tries for reaching the global minimum or maximum value according to objective function of this algorithm. The multidimensional scope of the objective function determines the search space for the particles and the function values related with the quantity of food at various positions (Merkle&Middendorf, 2005).

Every particle flies along the search space with regard to itself `s velocity vector which is arranged at every iteration of particle swarm algorithm. For the new velocity vector of a particle, its primary individual best position and global best position, which is reached as entire best position by particle until now, behave as attractors. The primary individual best position affects the cognitive sense and the global best position effects social sense of particles behaviour. One of the specific feature of this method is that it is possible to balance their global and local search skills by arranging related effect of local and global best solution while update of velocity (Merkle&Middendorf, 2005).

Many variations of particle swarm optimization is improved to increase the speed of reaching stopping criteria and quality of solution. The variation is affected by an amount of control parameters referred to the problem dimension, the amount of particles (swarm size), acceleration coefficients, neighbourhood magnitude, inertia weight that is a mechanism to check search and exploitation skills of the swarm and

to eliminate the requirements of velocity clamping, amount if the iteration and the random values that evaluate the effects of the cognitive and social senses (Rini, Shamsuddin&Yuhaniz, 2011).

Various kinds of neighbourhoods have been analysed; neighbourhoods of particles can be formed according to locations of the particles at the search area. At the fixed neighbourhood, the particles are linked by a graph, which can be rings, pyramids, meshes or linear arrays, and the particle neighbourhood consists of all its neighbours at the graph. At the dynamic neighbourhood, the group of particles at the neighbourhood might alter at each iteration, such as in form of k-nearest particles (Merkle&Middendorf, 2005).

Merkle and Middendorf (2005) summarize the advantages of particle swarm optimization algorithm as following:

- The algorithm does not need a specific analytical features.
- The algorithm implements only the functions value at the particle position to lead the exploration, so it is very favourable to apply for non-differentiable objective functions.
- The algorithm is population-based algorithm and the search procedure with random feature, therefore it cannot get stuck in local minimum.
- The algorithm can find balance between the global and local exploration of search area.
- The algorithm does not have complex processes, it can be easily and efficiently applied.

3.3.5 Differential Evolution

DE is stochastic, population-based evolutionary algorithm developed by Storn and Price (1997) in order to optimize the real valued functions. DE has major three advantages as reaching the true global minimum without considering the initial parameters, quick convergence and the requirement of small number of control parameters. The main steps of DE are initialization, mutation, recombination and selection; the main process depends on the differences of randomly chosen couples of solution at the population. Detailed formulation of DE is explained at Section 4.1.

The optimization task comprise of D parameter can be defined by a D-dimensional vector. Initially, the population of target vectors are generated randomly according to determined upper and lower bound for every parameter. Then, DE applied mutation process as exploration mechanism and a mutant vector is created by using randomly chosen vectors from the generation. By using the parts of the current population elements to generate trial vectors, the recombination process efficiently exchanges information about the good qualified combinations and explores the better solution area. The trial vector is created from elements of target vector and mutant vector; elements of the mutant vector are added the trial vector according to chosen probability. At the selection process, the trial vector (child) and the target vec-

tor (parent) are compared based on their performance; the best vector is selected for next generation. These steps continue until the stopping criteria is reached (Karaboga&Okdem, 2004).

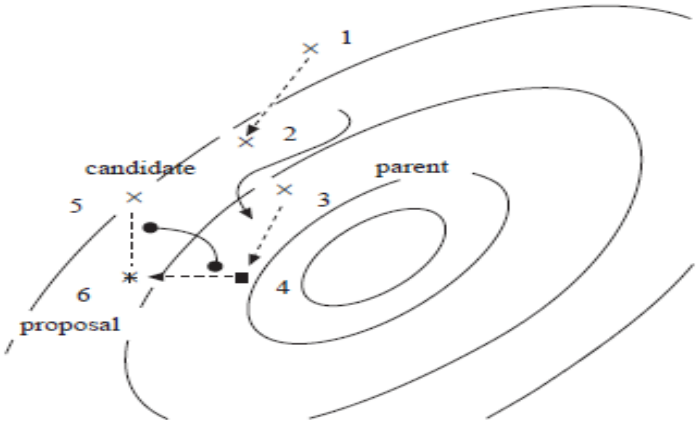


Figure 3.5 Development of new proposal at DE (Karaboga&Okdem, 2004, pp.55)

Karaboga and Okdem (2004) explain the detailed processes of DE in Figure 3.5. The difference between two members of the population (1,2) is added on a third member of population (3); the outcome (4) is liable to the mutation with the candidate for replacement (5) to get a proposal (6). If it performs better, the proposal is considered and substitute the candidate.

3.3.6 Genetic Algorithm

Evolutionary Algorithms are metaheuristic, population –based optimization methods inspired from biology mechanism and adapted from "survival of the fittest" theory, in which dominant chromosomes have higher probability to reproduce exponentially in the population along generations by reason of longer existence than the weaker individuals, to improve group of solutions iteratively. GA is subgroup of evolutionary algorithm and is computer-based optimization method adapted from genetic procedure of biological organisms. GA continuously develops by modifying the previous individuals and selects randomly two parents from current population at every iteration in order to create individuals for new population (Zekai, 2004).

GA start with a group of solutions represented by chromosomes, referred as initial population. Solutions of a population is obtained and used to create a new population that is supposed that new population will perform better than older population. Additionally, solutions are chosen based on their fitness value which is performance indicator. The main operators of GA are selection, crossover and mutation. The general procedure of the GA is graphed at Figure 3.6.

3.3.6.1 Basic Concepts of Genetic Algorithm (Michalewicz, 1996)

The structure of GA should be generated to find a solution and concepts of the GA should be determined to specify the parameters. The basic concepts of GA are defined in details below:

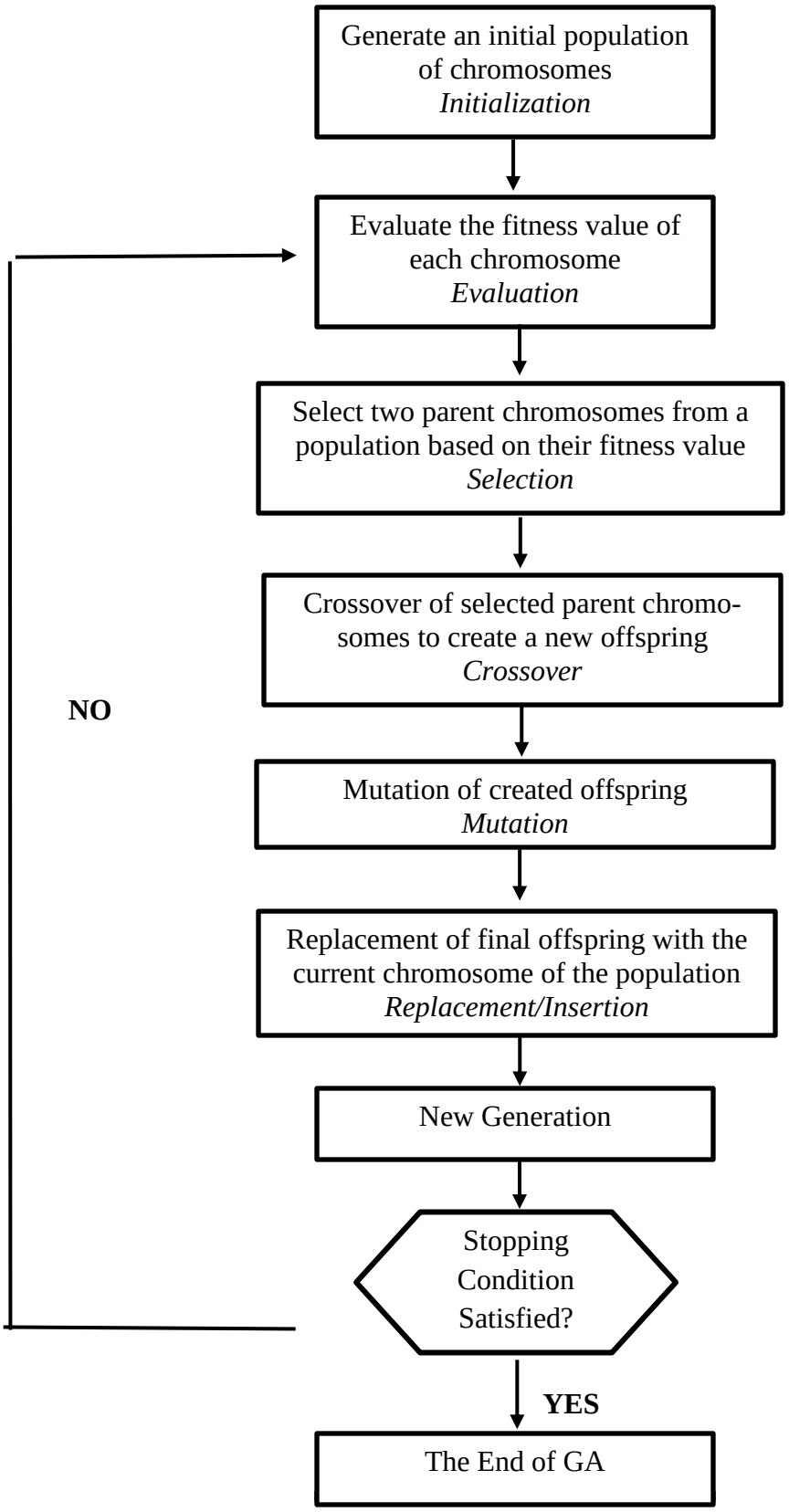


Figure 3.6 The General Procedure of GA

Gene is the smallest constituent of chromosomes as formation of single bits at every specific position or successively joined design of bits, that keeps genetic information and acts as a decision variable at optimization model.

Coding is defined as forming the GA according to chosen objective of optimization problem. Coding is differentiated based on implementation type of optimization problem.

Chromosome is string as combination of either one gene or group genes which have all information about the solution of the optimization problem.

Individual is member in the population that comprises of chromosomes and each individual is solution candidate for the optimization problem.

Population is a colony that is consisted of definite amount of individuals which have relations with each other to improve the quality of the population.

Generation is the new population of chromosomes generated as an output of process in every iteration of the GA.

Generation Gap is the partition of new individual attending in a population at every generation.

Parent is an individual which is chosen for mating from current generation in order to generate new individuals named as child or offspring. ***Child*** or ***Offspring*** is chromosome regeneration that is output of mating of two or more parent chromosomes.

Fitness value is quality indicator for every chromosome after pre-established evaluations, that is benefited during the selection process to bias the algorithm in order to choose fitter individuals for crossover process and during the insertion process to bias the algorithm in order to eliminate weaker individuals from the population.

Diversity is the average difference between the individuals of the population. If the average difference has a high value, it is called as high diversity. If the average difference is low, it is called as low diversity. Diversity is significant component of GA due to its determination to bounds of the search space.

3.3.6.2 Fundamental Operators of Genetic Algorithm

The process of GA starts with chromosome coding based on given optimization problem, generating initial population and deciding the fitness function. The basic operators are selection, crossover and mutation; there are different types of chromosome coding, selection, crossover and mutation methods which are chosen according to type of optimization problem. On the other hand, determination of the fitness function and stopping criteria have an important impact on solution of GA. The success of the GA depends on these leading fundamentals listed below (Taskin&Emel, 2009):

- Chromosome representation and coding
- Generation of initial population
- Deciding the fitness function
- Describing the evolutionary period involving various types of genetic operators, parent selection methods, different crossover and mutation rules and replacement/insertion method
- Determination of GA specific parameters as population size, crossover rate, mutation rate, number of individuals in a generation (i.e. population), number of generations used in the evolution process etc.
- Handling with infeasibility

a. Chromosome Coding (Malhotram Singh&Singh, 2011)

Encoding methods of GA are based on optimization problem and these methods convert solution of problem to chromosomes. There are different kinds of chromosome coding methods explained below.

Binary Encoding is the most popular coding method in which the data value is transformed into binary strings. Binary encoding provides verified possible chromosomes with small amount of alleles which is either 0 or 1 in a bit string. Each bit carries a characteristic of the solution, find the simple example below:

Chromosome A	1	0	1	1	0	0	1	0
Chromosome B	1	1	0	0	1	0	0	1

Figure 3.7 Binary Encoding

Permutation Encoding is suitable for ordering and sequencing problems especially for travelling salesman optimization problem. Each chromosome is represented as string of numbers in an order.

Chromosome A	3	8	5	6	1	0	4	2
Chromosome B	0	2	3	8	7	1	6	5

Figure 3.8 Permutation Encoding

Value Encoding can be model number, real numbers on features to some complex values. Each chromosome is a string of some values and is implemented at requirement of more complex values.

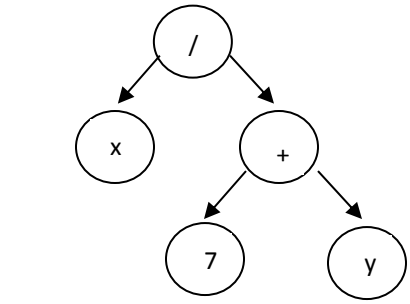
Chromosome A	3.6917	4.8997	6.1243	6.110	2.7863	1.8907	2.1093	7.1643

Chromosome B	West	North	East	North	South	West	South	North

Figure 3.9 Value Encoding

Tree Encoding is suitable method for evolving statements or program like genetic programming. Each chromosome is a tree of objects, process between objects, functions or commands at programming.

Chromosome A



y = (/x (+6y))

Figure 3.10 Tree Encoding

b. Determination of Fitness Function

GA has an objective function in order to evaluate proximities of individuals to the solution of problem. The objective function scales the performance of each individual or fitness of each individual in the search space. Fitness value is the quality measure of individual in GA; the fitness of the solution leads to improvement of search process (Cagatay&Emel, 2009).

c. Generation of Initial Population

In GA, an initial population is firstly created and the generation of the population is based on optimization problem type. Determination of the population size and generation method of initial population must be considered before operating the algorithm. When we increase the number of initial population, we can end up with high computational time; if the number of initial population is decreased, the diversity of the algorithm reduced and the algorithm may be stuck in the local optimum.

In GA, it is possible to implement random sampling and particularly modelled construction heuristics during initialization process. The last pace of initialization process is to evaluate the fitness value of the individual by applying fitness function (Zapfel, Braune&Beogl, 2010).

d. Selection

During each successive generation, a proportion of current population is selected to produce a new generation. Some chromosomes are chosen as a parent from current population to be assigned in the rebreeding process according to their good characteristics. The selection decision of an individual is based upon the fitness of the solutions and its fitness value in comparison to all or a proportion of the population. Thus, the fitter chromosomes have more chance to be selected for crossover process (Zapfel, Braune&Beogl, 2010). There are different rules for selection of parent for mating to breed an offspring in recombination process.

Elitism migrates chosen number of the fittest individuals from current generation based on their fitness values in order to save them from to be destroyed by operators along generations (Mitchell, 1999).

Roulette Wheel Selection is the most popular method for applying fitness proportionate selection. This method allocates all individuals to imaginary roulette wheel and the magnitude of slice proportional to fitness value of each individuals in the population. Fittest individual has the maximum slice magnitude on the wheel, so it has more chance to be selected. Firstly, the fitness values of all individual is summed up, then choose a random number from interval of the population; finally, it goes over the whole population and fitness sum, if the sum is greater than fitness criteria, it stops and go to the individual (Malhotra, Singh&Singh, 2011).

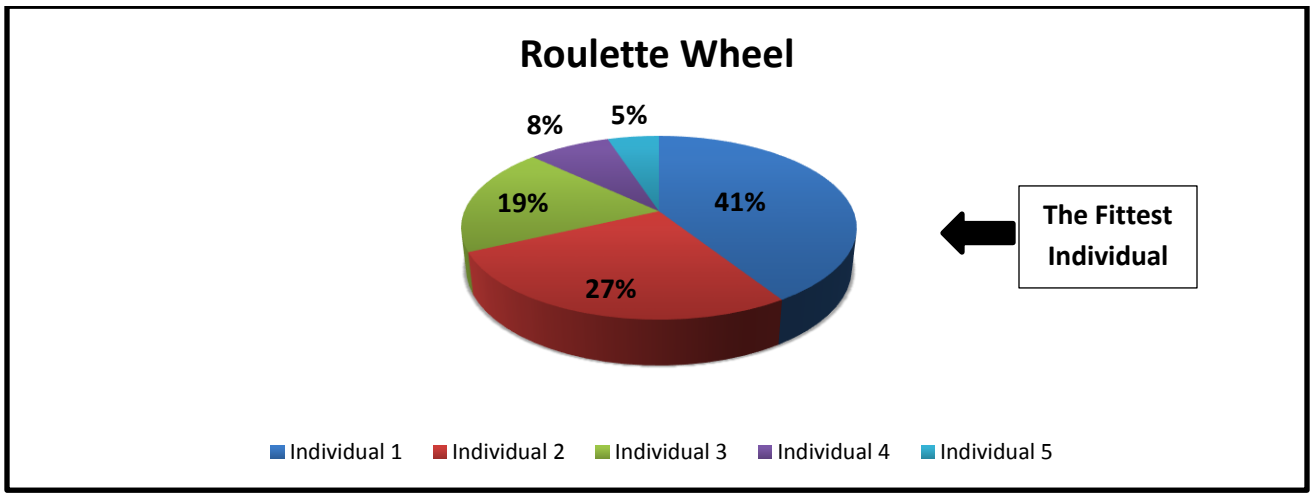


Figure 3.11 Roulette Wheel Selection

Ranking Selection ranks individuals in the population according to predetermined objective and each individual gets a fitness value assigned by this ranking. This method hinders the premature convergence, the individuals are graded based on their fitness value and the assigned value of every individual is based on its rank not its real fitness value (Malhotra, Singh&Singh, 2011).

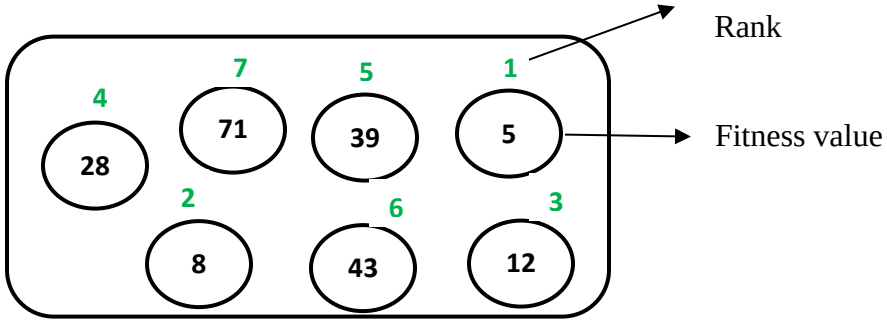


Figure 3.12 Ranking Selection – Higher fitness value, fitter individual

Tournament Selection is a method that the fittest individual overcome the remains at the end of the tournament competitions. The competition goes on until the amount of winners attain to predetermined amount of parents. The selection pressure and variety at the population can be arranged by switching the amount of individuals who join to tournament competition for selection (Mitchell, 1999).

Steady-State Selection has a major idea that more part of chromosome may migrate to next population. Only little number of individuals in every generation are replaced and few newly generated offsprings are placed instead of weakest individual (Malhotra, Singh&Singh, 2011).

Random Selection is a method that individuals are selected as parents according to uniformly random numbers, so to be selected as parents is same for whole individuals.

e. Crossover

After the selection process for mating, crossover operators generate new individuals by recombining genes of selected parents in order to produce more promising child/offsprings. The crossover rate decides how many times crossover of chromosomes will be occurred in a generation and crossover rate can be chosen between 0% and 100%. When the crossover rate is really small, the diversity is low since offsprings will have same characteristics of parent chromosomes. There are various crossover methods, the well-known ones are explained below (Chapter 2- Literature Review GA, www.prr.hec.gov.pk).

Single-point crossover is the easiest version of crossover methods. A single crossover point is select randomly; the genes positioned until this point is taken from first parent and the rest of the genes are taken from second parent. The order of exchanging process is different to create two non-identical offsprings (Zekai, 2004). Figure 3.13 shows the single-point crossover:

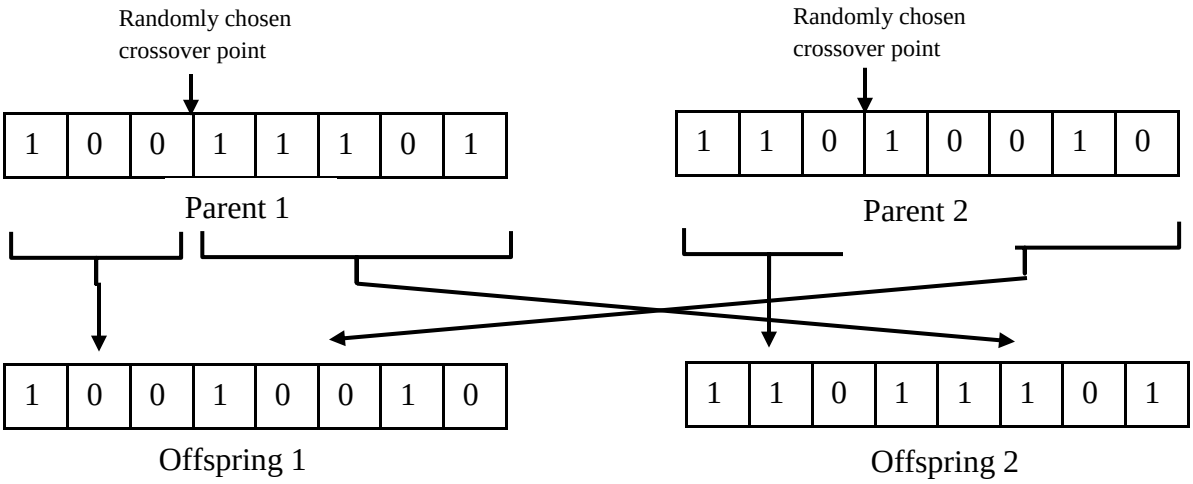


Figure 3.13 Single-point crossover

Two-point crossover is the method in which two positions are randomly chosen and the genes between these two positions are exchanged as described in Figure 3.14 (Zekai, 2004):

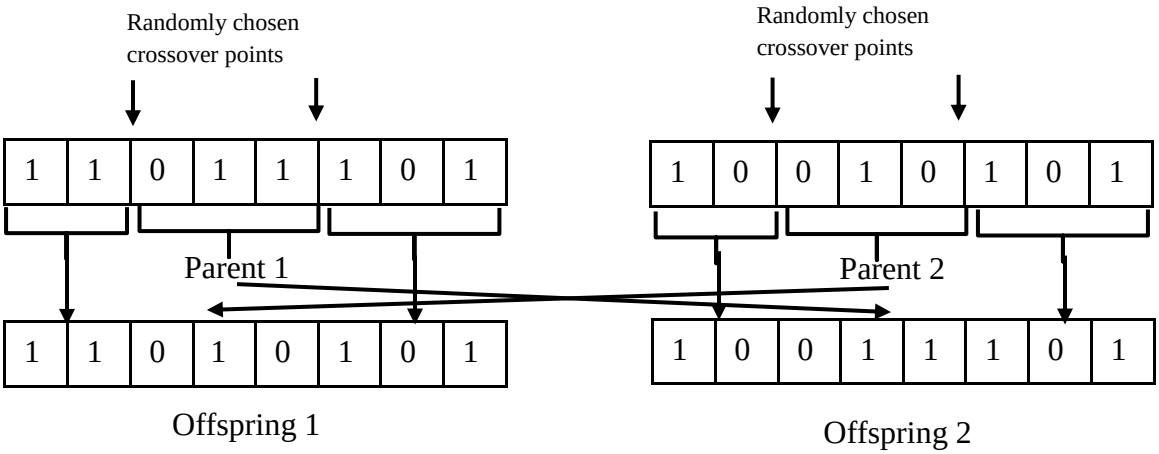


Figure 3.14 Two-point crossover

Multi-point crossover is method in which at least two points are chosen randomly from parents, then each gene bounded by the crossover points are reunited to create a new offspring (Zekai, 2004).

Uniform crossover applies fixed mixing ratio as 0.5 for two parents in order to enable every gene of offspring to be inherited every allele from both parents. Uniform crossover does not have positional bias since given opportunity for every gene (Mitchell, 1999).

Fusion operator is fitness-based crossover operator, which considers the structure and related fitnesses of the parents, presented by Beasley and Chu (1996). Only single offspring is created in consequence of crossover process of two parents by fusion operator. To give an example in which higher fitness means fitter individual, when fusion operator will combine two chromosomes according to their fitness values which are 4 and 6, the probability of every gene of the generated offspring is taken from first parent with 0.4 probability and from second parent with 0.6 probability. The fusion operator is more creative than other crossover methods since it has more diversification on recombination of the parents.

PMX crossover is partially matched crossover based on ordering procedure proposed by Goldberg and Lingle (1985) and used especially at travelling salesman problems. Only one offspring is generated from two parents and randomly chosen genes from one parent are directly copied to same position of offspring and the rest is created randomly as non-existing values.

OX crossover is order crossover introduced by Davis (1985). The selected two parent are cut at randomly chosen two points and the offspring gets the identical genes outside from chosen points at same positions as the parent and the genes between two points are intermixed based on the order which they are placed in other parent.

f. Mutation (Michalewicz, 1996)

The main goal of implementing mutation is to cause a particular level of diversity in the population, thus GA can avoid to get stuck in a local optimum. During the mutation process, a small changed is occurred at the genetic model of chromosome and the offspring result of crossover process, is randomly altered by a mutation operator. The mutation rate is probability that mutation occurs, the small mutation rate gives better results since it prevents GA to be trapped in local minima. There are several types of mutation method which are explained below.

Displacement mutation chooses randomly two points and removes these genes between two points from string; after that, these genes are inserted in their original order to randomly chosen positions.

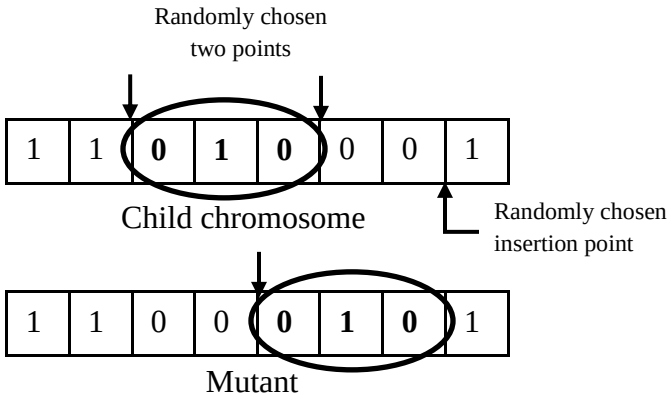


Figure 3.15 Displacement mutation

Exchange mutation chooses randomly two genes and swaps their positions on the chromosome.

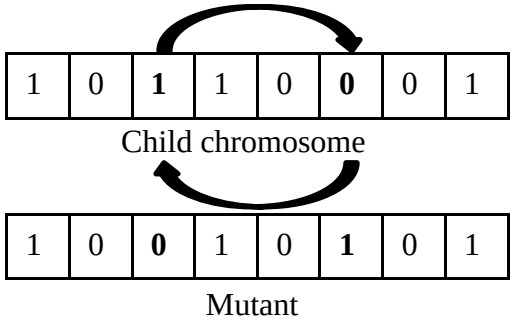


Figure 3.16 Exchange mutation

Insertion mutation randomly selects a points and removes the gene at this point; then inserts the gene randomly selected points.

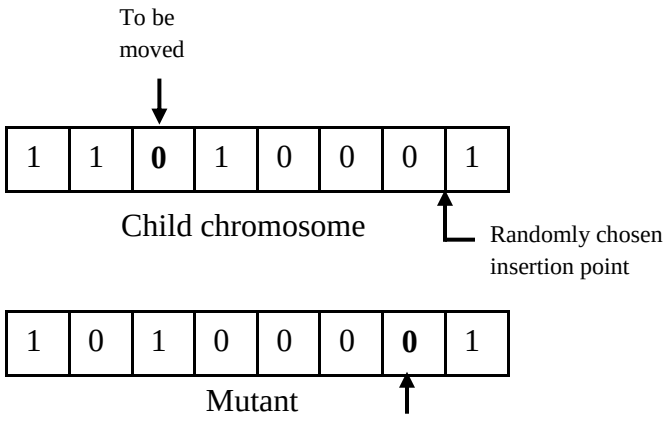


Figure 3.17 Insertion mutation

Simple inversion mutation randomly selects two cut-points and substring between two cut-points are reversed to produce mutant offspring.

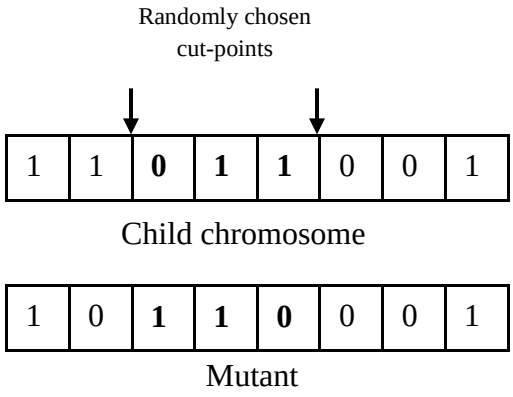


Figure 3.18 Simple inversion mutation

Inversion mutation randomly selects two positions and removes the genes from chromosome between two positions; then inserts these genes in reversed order into randomly chosen position.

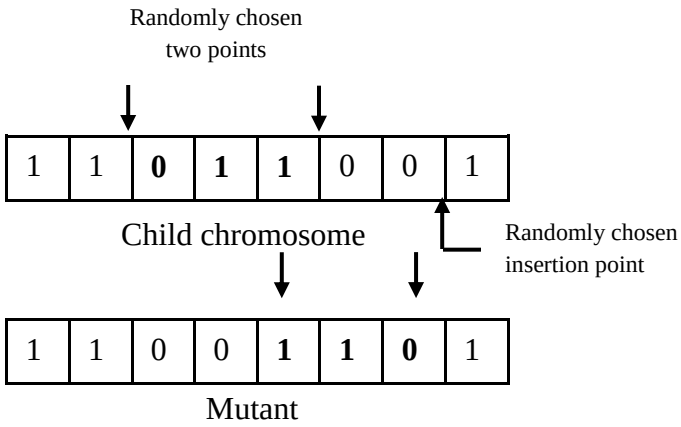


Figure 3.19 Inversion mutation

Scramble mutation randomly chooses two cut-points and the genes between these cut-points are randomly switches their positions.

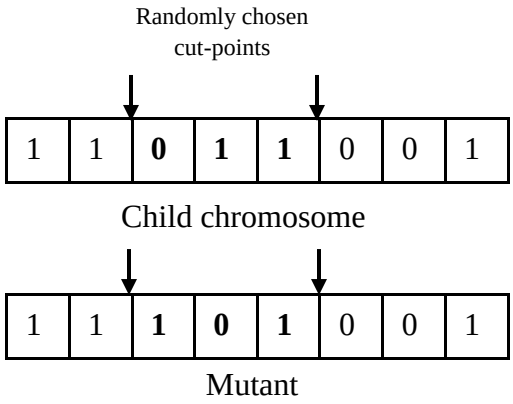


Figure 3.20 Scramble mutation

f. Replacement/Insertion

Replacement is step in which individuals of current generation will be exchanged with newly created offspring or will survive in the population; the amount of these quitting individuals are considered as generation gap. Fitter individuals are transported to the next generation to improve the performance of the algorithm (Zapfel, Braune&Beogl, 2010).

g. Handling with Infeasibility

The infeasible solution can occur as a results of design of crossover or mutation operators. Unfortunately, an infeasible solution does not satisfy the constraint of optimization problem. There are three ways to handle with infeasible solutions (Zapfel, Braune&Beogl, 2010) :

- Discarding infeasible solutions
- Penalizing infeasible solutions
- Repairing the infeasible solutions.

h. Termination of Genetic Algorithm

The algorithm may stop when one of the following criteria are reached:

- Conclusion with a single solution or a group of same solutions
- No feasible solution is found
- Getting pre-specified threshold value for fitness value
- Reaching the pre-specified iteration number
- Reaching the pre-specified time limit for running the algorithm
- Finding a feasible solution

3.3.6.3 Difference of Genetic Algorithm from Traditional Optimization Methods

GA differs from other traditional optimization methods in its fundamental procedures (Goldberg, 1989):

- GA operates with a coding of parameter group, not the parameters themselves.
- GA explores from a population of points, not only one point.
- GA applies payoff (objective function) information, not derivative or other subsidiary parameters
- GA implements probabilistic transition methods, not deterministic methods.

3.3.7 Comprehensive Literature Review for Genetic Algorithm

Metaheuristic methods are applied to solve optimization problem by resulting optimal solution through a finite or countable infinite amount of choices. Evolutionary algorithms are population based metaheuristic optimization methods which is inspired from biology and natural selection procedure to build iteratively number of solutions. GA is also a computer based, evolutionary algorithm that can reach near-optimal solution with an acceptable computation time. Up to now, many research and applications are done on usage of GA.

GA is firstly introduced by John Holland (1975) based on survival of the fittest theory and genetic science in order to find optimal solution by applying directed random search for large scale optimization problems. Every cycle of GA is named as generation that contains the evaluation of solutions, the implementation of selection and crossover, mutation as genetic operators.

Goldberg (1989) presents an intensive research on GA theory and analyses different structures on selection, crossover and mutation processes.

Falkenauer (1991) proposes the grouping GA (GGA) in order to apply in grouping optimization problems in which members of a set as a small amount of families to satisfy the objective function according the chosen constraints. There is a particular chromosome representation model and genetic operators.

Falkenauer and Delchambre (1992) implement GA as grouping type for SALBP with objective of minimization of number of workstations. In their groping representation, the workstations are shown by increasing the workstation according to chromosome with a group section and they firstly improve their representation for bin packing problem, then form it particularly for genetic operators. They randomly create the initial population and size of the population; use modified bin packing crossover method and

mutation for the group section of the chromosome in which the workstation based section of the chromosome does not change.

Anderson and Ferris (1994) analyse the productive application of GAs with description of classic serial implementation of GA for the ALBP with objective of minimization of cycle time and workstation based chromosome representation. They consider the effects of different GA operators on performance based and develop a parallel type of GA as an alternative in which every individual of the population are located at a processor. They use the heuristic solutions on creation of initial population and the chromosome length is represented by the quantity of tasks; they apply elitism as survival theory, one-point crossover and mutation methods and a penalty cost for the evaluation function. The comparison of serial and parallel GA proves that the performance of the best solutions is better at serial GA.

Leu et al. (1994) improve a GA for SALB with different techniques to analyse the feasibility problems at population initialization by aiming the minimization of number of workstations. For the creation of initial population, they apply both random and heuristic procedure and give more flexibility at the population by permitting to build up before survival theory is considered. The chromosomes are determined as task based precedence list and length of the chromosome is equal to the amount of tasks; order crossover and scramble mutation is chosen at GA. Their objectives are minimization of the idle mean idle time and the minimization of the mean squared idle time.

Rubinovitz and Levitin (1995) compound GA with a basic local optimization search method for SALBP-2, in which proposed GA is able to balance the assembly line when task processing times depend on workstation. They produce initial population randomly and use fragment reordering mutation in order to improve diversification at the population and fragment reordering crossover which preserves the heritage of positions and related sequence of members at the procedure, supplies variations within fragment without disrupting precedence relations. They compare proposed GA with Dar-El and Rubinovitz's MUST; the experimental results show that GA performs faster than MUST for large scale problems with high flexibility proportion.

Tsujimura et al. (1995) are the first on the application of GA to GALBPs. They use fuzzy numbers to symbolize the cycle time and the task processing times since these processing times are not stable because of worker and machine influences. They solve single-machine ALBP with GA in order to minimize the number of the workstations by symbolization of the fuzzy task times via triangular membership elements. They randomly generate the initial population, the chromosome is defined as task based; additionally, they use the adjusted PMX crossover and exchange mutation methods. According to the computer simulations, they establish that their GA method is suitable for solving fuzzy ALBPs.

Watanabe et al. (1995) develop a GA for ALBP in order to reach optimal task assignment by minimizing the cycle time with a chosen amount of workstations. Three alternative algorithms are enhanced to find optimal solutions via genetic coding, to create initial population randomly for evolution model and usage of the roulette wheel and ranking selection operators, single-point crossover and insertion mutation. They confirm that their GA is not suitable for large-size problems.

Gen et al. (1996) consider ALBP with fuzzy processing times in order to minimize overall processing times at every workstation. They practice two-point crossover and one-point mutation methods as genetic operators; as a result of these methods, infeasible chromosomes may occur. In order to deal with infeasibility of chromosome, they execute the precedence-based random consecutive allocation technique for assuring the feasibleness of the initial population and repairing method for forcing these chromosomes which are created by earlier populations to fulfil the precedence constraint for following generation.

Kim et al. (1996) introduce a GA to balance assembly line with different objectives which are minimization of number of workstations, minimization of cycle time, maximization of workload smoothness, maximization of work relatedness, multi-objective of maximizing the workload smoothness and work relatedness. They apply various crossover methods as partially mapped, enhanced edge recombination, order, uniform, cycle crossover and non-usual crossover operators as one-point, two-point and uniform; on the other hand, six various mutation methods are applied, which are reciprocal exchange, insertion, inversion, displacement, feasible insertion and scramble mutation operators. Repair mechanism is improved in order to convert an infeasible chromosome to feasible one by organizing tasks based on precedence and other relative constraints. According to experimental results, application of partially mapped crossover and reciprocal exchange mutation operators are better off the other combination of operators for different objectives; their GA performs better than compared heuristic methods by producing various Pareto optimal results.

Suresh et al. (1996) proposed an adjusted GA for SALB-1 with stochastic operation times to solve the irregular search space issue by using two populations -one takes into account infeasible solutions in order to not be caught in a trap at local minimum- and switching problem specifications at proper periods. They indicate that infeasible solution may be authorized at the population when the genetic operators may end with feasible solution starting from infeasible solutions. The computational results show that using two populations may accomplish better solutions than using only feasible population.

Falkenauer (1997) combines grouping GA with branch and bound algorithm in order to minimize the number of workstation at assembly line with resource dependent operation times. Firstly, grouping GA allocates tasks to workstations and later branch and bound algorithm decides the ideal source for every workstation. The task times are based on the chosen resources, thus, resources with various cost and speed are assigned to every task besides task allocation to workstations, so the overall cost of the assembly line can remain minimal.

Ajenblit and Wainwright (1998) present a GA for U-shaped SALBP-1 by aiming the objectives as the minimization of the total idle time or/and to balance the workload between workstations. They create six various assignments algorithms for representing a chromosome and task allocation to workstations and the fitness value of one chromosome is calculated by these six assignment algorithms. The first assignment allocates the alternatively non-allocated tasks at initial and the final point of tasks order; the second one applies same by prioritizing the tasks placed at the end of the order; at the third assignment procedure, all available non-allocated tasks at the initial point of the order are assigned and then all

available non-allocated tasks at the final point of the order are assigned; then fourth one allocates all available tasks at the end of the order and allocates all available task at the beginning of the order; the fifth assignment procedure prioritizes the tasks at the beginning of the order while the sixth one prioritizes the tasks at the end of the order. Order crossover is used while any mutation method is not used due to not getting possible improvements. They compare the performance of their GA with dynamic programming and different heuristic algorithms and the GA gives better results than compared algorithms at most of the case.

Chan et al. (1998) develop a GA for balancing assembly lines with minimization of number of workstations in the clothing industry. Their aim is to increase the efficiency of line by reducing the time used during assembly line planning. They consider the labour effect by adjusting different skill levels to the problem. The initial population is generated randomly; they change the elitism strategy in a sense that the parent chromosomes are displaced with child chromosomes instead of the worst performing individual at the population is displaced with child. According to experimental results, GA produces better solutions than greedy algorithm which is implemented for optimize the ALBP in many industries.

Kim, Kim and Cho (1998) work on a heuristic based GA to balance workload between the workstation at the ALBP. They modify the many parts of GA, such as evaluation function, genetic operators. Initial population is created randomly and chromosome representation is based on workstation. They improve heuristic based genetic operators as a heuristic structural crossover method which uses problem specific restrictions to decide various groups of workstations in order to reproduce from two parents to an offspring, a heuristic structural mutation method which chooses randomly tasks from every chromosome according to mutation rate then relocates chosen tasks. The computational results prove that their GA performs better than well-known heuristic methods and the standard GA.

Rekiek et al. (1999) introduce a grouping GA based on Equal Piles procedure for SALBP by assignment of tasks to a fixed amount of workstations in order to equalize the workload among workstations. The introduced GA is based on *boundary stones process* where the boundaries are considered as seeds to fill workstations. Tasks are grouped into a predetermined number of workstation, so that precedence constraints are not violated and operation times of workstations are almost same.

Bautista et al. (2000) analyse the SALBPs with incompatibilities between the tasks by objecting firstly the minimization of number of workstations and then minimization of cycle time with the smallest number of workstations. They propose *Greedy Randomized (Weighted) Adaptive Search Procedure* which is the application of classic heuristic priority based rules and GA with finding for a solution at heuristic search space. They apply the main qualifications of greedy heuristic rules and random selection process for generating solution with probability distribution; the applicant task list is restricted to increase the probability of the most eligible applicant tasks. At their methods, the probability distribution is based on an index resulting according the chosen priority rule. Six different greedy randomized heuristic rules are applied with a selection probability for task allocation commensurate to the parameter for chosen priority rules and these six priority rules allocate tasks based on longest processing time,

ranked positional weight, average ranked positional weight, processing time divided by upper bound, maximum number of successors divided by slack. Seven different forms of presented GA are used by varying crossover and regeneration methods, mutation and crossover probabilities. The experimental results show that GAs and presented GA with index resulting probability distribution produce better solutions with comparison of other chosen heuristics.

Kim et al. (2000) improve a GA for two-sided ALBP by examining the major parts of GA which contains encoding and decoding processes, generation of initial population, genetic operators. They affirm the usage of problem specific information and GA's self-adaptation ability. The group number encoding and a heuristic decoding method are modified; tournament selection method is chosen at proposed GA. The authors especially indicated that the proposed GA is applicable for ALBP with different objectives by making some minor changes.

Ponnambalam et al. (2000) create a multi-objective GA for ALBP in order to evaluate the performance according to the amount of workstations, the line efficiency, the smoothness index before/after trade and transfer. Twelve different heuristic priority rules and two created heuristic rules, whose task assignments are based on *maximum task time of follower task* and *maximum position weight of follower task*, are applied for chromosome representation. The performance of their GA is compared with six well-known heuristic optimization methods, which are ranked positional weight, Kilbridge and Wester (1961), maximum task time method of Moodie and Young (1965), precedence matrix of Hoffman (1963), immediate update first fit and rank and assign methods, by examining on twenty different assembly line networks with five cycle times. The comparative results prove that their multi-objective GA is better off than six heuristic methods; on the other hand, the execution time of GA is much more than others due to research of globally optimal solutions.

Sabuncuoglu et al. (2000) propose a GA for single- model ALBP by implementing a new technique for a particular chromosome design "*dynamic partitioning*" which alters the chromosome design by assigning tasks to workstations in order to comply chosen restrictions and then, go on with residual tasks. The chromosomes are presented based on precedence relations of tasks, the initial population is generated randomly and their fitness function has two objectives as minimization of quantity of workstations and balancing the workload between workstations. While order crossover, scramble mutation and roulette wheel selection methods are applied as genetic operators; new elitism method from Simulated Annealing is implemented to the model. They state that their dynamic chromosome design method improves the solution quality and save computation times besides resulting with less chromosome size. Comprehensive experimental results demonstrate that their GA performs better than popular heuristic methods in the literature.

Carnahan et al. (2001) consider the physical demand criteria for labours integrated with production objectives for ALB. Their aim is to establish methods in order to minimize the cycle time of assembly line and the maximum physical gripping demands needed of an operative allocated to a workstation in the assembly line. They work on three optimization methods which are ranking heuristics, a combinatorial GA and a problem space GA; every optimization method is structured for minimization of both

objectives. The comparative results indicate that the problem space GA shows better performance than the others.

Simaria and Vilarinho (2001) develop a two-staged iterative search method with application of GA for mixed model assembly line with parallel workstations in order to minimize the cycle time with a given number of workstations. The iterative search method is demonstrated on a case study with two assembly models and twenty-five tasks. The method initializes with a lower bound of cycle time; when the optimal solution is reached, GA operates in order to diminish the cycle time. During minimization of the cycle time, the method also aims to balance the workload between workstations as the second objective.

Chen et al. (2002) introduce a hybrid GA associated with self-tuning method for assembly line planning including several different goals which are minimization of cycle time, maximization of the workload smoothness, minimization of the tool change frequency, minimization of the quantity of machines and tools' usage, and minimization of the complexity of assembly sequences. The introduced tuning method is able to sustain useful schemata of chromosomes in order to avoid infeasible precedence relations in the assembly at GA process. Firstly, various popular heuristic methods are applied to find feasible solutions which are then added to randomly generated population of evolving pool; they aim to minimize the search space by adding these heuristic solutions, so that the search time may decrease too. The computational results prove that introduced method notably enhances the solution quality and decreases the computation time by integrating with heuristic solutions.

Goncalves and De Almedia (2002) propose a hybrid GA combined with a heuristic priority rule method and a local search process for SALBP-1. The proposed GA applies a random key heuristic based chromosome representation, elitism as selection operator and a parameterized uniform crossover. They implement their GA to a few ALBPs from the literature and the proposed GA produces efficient results.

Miltenburg (2002) develops a GA for balancing and sequencing mixed model U-shaped ALBPs in order to minimize the number of workstations. The initial population is randomly created and the chromosome representation is the combination of task and model sequence based. Miltenburg (2002) provides comprehensive information on performance of the developed GA by explaining the changes at the computation time according to altering the crossover operator from two-point to cycle crossover.

Valente, Lopes and Arruda (2002) improve a GA for balancing two-sided car assembly line as real-world implementation in order to minimize the cycle time in which every workstation's length is predetermined and stable. They use workstation based chromosome representation, one-point crossover and bit by bit mutation. The adopted results show that their GA decreased the total time of assembly line by 28.5%.

Wu, Liu and Wu (2002) consider a GA for master production schedule problem in a processing-assembly line with exact same machines in order to reduce total holding cost of the assembly line while satisfying the chosen numerous and complex restrictions. They apply roulette wheel selection, one-

point crossover and a uniform transpiring method as crossover method. The experimental results state that their GA outperform particularly problems with tight restrictions.

Abe, Yamada and Matsui (2004) analyse a design method for ALBP by creating a GA considering cycle time, line length, precedence relations and lead times. They implement traditional GA, Adam-Eve GA in which new offsprings generated by crossover operator do not employ their parent's space and are added into population as new individuals, a new operator as *death* represents the lifetime of every individual, and two-stage GA in order to optimize the assembly line design.

Brudaru and Valmar (2004) introduce a hybrid method combined GA with branch and bound for SALBP-1 in order to decide fitness function. They use *embryonic chromosome representation* which examines the subsets of solutions instead of individual solutions; they also develop a growing operator as genetic operator for the embryonic chromosome representation in order to make easier to evolution of chromosome along all length chromosome.

Martinez and Duff (2004) optimize the SALBP by practising 10 different heuristic priority rules such as MA, RPW, maximum total number of follower or successor tasks to minimize the number of workstations. Then, they adjust GA of Ponnambalam et al. (2000) by using these heuristic priority rules in order to generate initial population.

Simaria and Vilarinho (2004) propose a mathematical programming model and an iterative GA process to solve mixed model ALBP with parallel workstations in order to improve the production ratio of the assembly line with a given number of operators with consideration of issues on operating circumstances at real-world assembly systems. Their mathematical model is based on the usage of parallel workstations and zoning constraints: positive zoning, in which the pairs of tasks are pushed to be allocated to the same workstations, and negative zoning as group of pairs of incompatible tasks. The objective function has two aims as minimization of the cycle time with a given number of workstations and balancing the workload between workstations. Their proposed GA has three stages which are constructive heuristic solving for mixed model SALBP-1, GA procedure as GA-1 and finally GA procedure as GA-2. The *constructive heuristic solution* begins by reaching an initial solution for the assembly line problem from a lower bound for the cycle time. When a task has been selected for assignment from the group of eligible tasks, the heuristic randomly selects the priority rule to be applied and the sum of number of workstations comes from the solution of mixed assembly line problem Type 1 is bigger than the current number of workstation of the original problem mixed mode SALBP-2, the cycle time is grown by a unit and another mixed assembly line problem Type 1 is solved; the solution of the first stage activated to stage 1 of the heuristic that practises the GA-1 process. *GA-1 process* decreases the cycle time by one unit at the end of first stage and attempts to reach feasible solutions; at each time of reaching a feasible solution, the cycle time is reduced and GA-1 begins again until the stopping criteria is reached and then, the cycle time is grown by a unit in order to trigger last stage. *GA-2 process* aims to adjust the solution of GA-1 by finding the workload balance between workstations to guarantee that almost same amount of work is operated at every workstation for each model. GA-2 has approximately same procedure with GA-1 except the fitness function and proceeding with unfeasible solutions. The

proposed GA performs well for mixed model ALBP with aim of minimisation of cycle time and maximization of production ratio.

Brown and Sumichrast (2005) compare the solutions of the traditional GA over a wide filed of grouping problems with GGA of Falkenauer (1991) for balancing SALBP with minimization of number of workstations. They use two procedures as standard GA and GGA on group of problems in order to compare the performance regarding quality and computational duration of the solution. Their experimental study shows that both procedures are able to reach to the optimal solution but GGA comes to the solution faster.

Hui (2005) introduce a new solution method for ALBP by applying hybrid GA with different types of selection, crossover and mutation processes. Self- adaptive mutation rate is used in order to prevent generation of premature chromosomes. Hui (2005) works on 100 generations with 128 individuals; his experimental study indicates that linear ranking as selection operators performs better and derivative tree crossover produces better results out of examined ones.

Due to increase at the usage of robots at production process in assembly line systems, Levitin, Rubinovitz and Shnits (2006) analyse GA for large scale and RALBP in order to improve automation and flexibility in assembly line. The objective is minimization of cycle time with a given number of workstations while balancing the workload between workstations by assignment of the best fitted robot to every workstation; this algorithm gives a solution for the way of grouping the work activities operated at a certain number of workstation and the way to allocate a single robot of one of different types of number of robots to every workstation in order to satisfy the objective of the problem. Two distinct methods are presented for the adaptation of GA with basic evolution principle for RALBP by assignment of robots with various qualifications to workstations: a recursive assignment procedure and a consecutive assignment procedure. *Recursive assignment procedure* objects to allocate activities to workstations with consideration of vector sequence; *consecutive assignment procedure* splits the vector into parts as number of workstations, thus allocation of the activities pre-determined in a defined sequence through workstations and assignment of robots to workstations. The results of GA are reformed by a local optimization (hill climbing) work-piece exchange procedure. Experiments operated in randomly created problems prove that consecutive assignment procedure produces better qualified solutions and GA finds better results in a comparison with Branch and Bound Algorithm for RALBP.

Noorul Haq, Jayaprakash and Rengarajan (2006) develop a hybrid GA for mixed model ALBP with minimization of number of workstations by applying modified ranked position weight method into randomly creation of initial population of Ga in order to narrow the search space within global search space Tests establish that their GA with modified ranked position method gives better results than standard GA

Wong, Mok and Leung (2006) introduce a GA to optimize operator allocation in apparel assembly line by minimising the operator idle time. The applied method rearranged the operator allocation after each fixed time period based on the most updated production condition to eliminate the bottle necks in the

assembly line. Initial population is generated randomly; elitism rule and roulette wheel selection rule are used as selection operator and single-point crossover and random resetting mutation is implemented at introduced GA. The computational results prove that proposed GA is able to promote the assembly makespan because the optimised outputs of GA are less than the constant theoretical operator allocation and to shorten total production time by decreasing the idle time of each operator.

Yu, Yin and Chen (2006) present a multi-objective GA for mixed assembly lines and implement pareto ranking method to measure efficiency of algorithm's results. Pareto ranking method measures the performances of each of the workstations in assembly lines according to their task completion process and quality. Furthermore, pareto ranking method may be used for multi-objective based designs in order to determine the performances of the systems. To find optimal results, pareto ranking method transfers the objectives of vectors into fitness values and eliminates *"random moving*" variables during measuring the rankings. Researchers focus on completion time, reduction of costs and application of crossover operation between workstations at ALBP. Experimental results show that proposed multi-objective GA submits better and smoother level as compared to traditional and heuristic models regarding the scheduling problems in assembly lines.

Baykasoğlu and Ozbakir (2007) analyze stochastic U-shaped ALBP by their proposed model on multiple-rule-based GA in terms of task completion process and reduction of costs. The initial population is randomly created, roulette wheel selection, single and two-point crossover operators are used; the fitness function of GA takes into account of idle time of every workstations and the non-completion probabilities of every workstation while minimizing the amount of workstations. The output of examined tests finalize that their GA can optimize practical-sized problems with acceptable computational times.

Su and Lu (2007) consider the effectiveness of GA for mixed- model ALBPs by minimization of cycle time through application of combining GA for non-robotic assembly lines. Researchers improve a simulation of a mixed model assembly line to use it for proposed GA method in order to observe the productivity and problem solving capacity of improved model. Their GA with combination of simulation accelerates cycle time of assembly lines, therefore, this method provides better results as compared to previous methods for balancing assembly line problems.

Suwannarongsri, Limnarat and Puangdownreong (2007) develop a new hybrid intelligent methods that consists of the combination of Tabu Search and GA models and named as *TSGA-based method* to be used for solving ALBP. The proposed method is initially employed to the tasks that are assigned to each workstation. Then, the summation of all of the tasks is equalized to the total task of the problem in assembly lines. After that, the method arranges the sequences of tasks according to precedence restrictions. To summarize, tabu search leads the amount of the tasks allocated for every workstation while GA organize the tasks order based on precedence restrictions. The computation results address that their method gives better solution than conventional COMSOAL and four single-based assembly lines' balancing problems are completely resolved by applying this new technique.

Suwannarongsri and Puangdownreong (2008) apply combination of Tabu Search and GA models method on U-shaped assembly line in order to optimize the balancing conditions and augment the production quality of the line. The proposed multi-objective GA model with aim of balancing workload between workstations, decreasing idle time and increasing line efficiency gives better results for U-shaped assembly line in comparison with single-objective models.

Guo et al. (2008) investigate the scheduling problem in flexible assembly lines by presenting a GA in which a new chromosome is added. The presented method initially assigns the operations to workstations and determines the proportions of tasks within each of the different workstations. After that, to route the operations of each product, heuristic operation route is used. Experimental results outline that the proposed optimization algorithm submits superior outputs as compared to heuristic attempts within flexible assembly lines problems.

Hwang, Katayama and Gen (2008) work on multi-objective GA for balancing U-shaped ALBP with including performance criteria as: efficiency of assembly line and minimizing the amount of workstations as well, and balancing the workload between workstations. The comparison of results show that multi-objective based GA method produces better solution for line efficiency in U-shaped assembly lines as compared to single-objective methods and traditional heuristic techniques.

Kulak, Yilmaz and Günther (2008) examine the performance of GA based solution approach on balancing printed circuit board assembly lines which comprise of a number of various machined for mounting electronic parts on printed circuit board. In this study, component feeders are initially attributed to the placement machines with the tasks for ALB; then, specific machine optimization algorithms are integrated with a number of candidate solution approaches. The examined results prove that the operation times which are based on fine-tuned placement are reduced by the application of GA based solution and more importantly, printed circuit board assembly line production times are minimized by the candidate solution approaches.

Zang, Gen and Lin (2008) consider that multi-objective GA by a generalized pareto-based scales independent fitness function (gp-siffGA) in order to optimize ALBP with worker assignment and task allocation to workstation for minimization of cycle time, diversification of workload and reducing the total cost of the line**.** At first, a random key representation procedure is implemented for prioritized task vector readjusting the GA is proposed, during randomly created procedure is followed for worker assignment vector. After that, improved genetic operators are adjusted to special chromosome form. The computational results prove that their approach enhances the solution quality in comparison with other GA approaches.

Gao et al. (2009) develop hybridized GA with local search on balancing problems in type 2 robotic assembly where tasks are allocated to workstations and every workstation requires to choose one of the eligible robots to operate the allocated task considering the aim of minimization of cycle time. Also, the proposed model focuses on local search procedures and improve the ability of searching of GA in as-

sembly lines. GA implements the partial representation techniques in which only some portion of the determination information about an applicant answer is stated in the chromosome and the remaining is figured by a heuristic procedure. To extend search space, local search methods proceed based on GA scheme. Results of the tests indicate that developed method is applicable for small-scaled problems by reaching optimal solution in short computational time.

Hwang and Katayama (2009) work on measuring the efficiency and performance level of multi-objective GA for mixed model ALBPs with performance scale as the amount of workstations and diversification of workload. The priority based GA structures an amelioration form with a GA in order to enhance the balance of workload in mixed model ALBPs. The priority based GA leads for production of efficient chromosome with weight mapping crossover operator. The experimental results show that multi-objective GA methods gives better solutions for mixed model assembly balancing problems as compared to single based GA techniques.

Kim, Song and Kim (2009) combine a mathematical model and GA for the two-sided ALBP with a goal of minimization of cycle time with a given amount of matched workstations. This proposed model involves the strategy of localized evolution and steady-state reproduction in order to advance the diversification of population and search efficiency in two-sided assembly lines. While structuring their GA, they consider qualifications of two-sided ALBP. The examined results show that their GA is better off on quality of solution in comparison with the heuristics and the tested GAs.

Moon, Logendrand and Lee present GA integrated with mixed integer linear program for ALB with resource restrictions. The main aim of this research is to minimize the total workstation costs and reduce salaries of workers, which are assigned to them according to predetermined cycle time. The presented model finds more efficient results as compared to existing GA methods due to combination with integer linear program. Also, the resource restriction problem is completely resolved by the implementation of this new proposed model.

Akgunduz and Tunali (2010) introduce and adaptive GA by taking into account of variation in part consumption ratios, sum of the utility work and setup costs, for mixed model ALBPs. The introduced process harmonizes an adaptive parameter control to enhance the competences of algorithm and this parameter also determines the mutation ration and elitism proportion. Experimental results indicate that introduced adaptive GA perform better solutions considering amount and the quality, in comparison with non-adaptive algorithms.

Minghai and Huanmin (2010) examine the effects of hybridized GA on reconfigurable ALBPs. To solve the balancing problems, the researchers develop a hybridized GA model that is integrated with a mathematical model, and applied to reconfigurable assembly lines in order to reduce the total production time and increase the productiveness of the production quality. The results show that the validity and feasibility of the proposed algorithm for the balancing problems in reconfigurable assembly line.

Tang et al. (2010) work on field based research that measures the influence of a novel GA approach where chromosome representation based on task sequence are coded with satisfaction of precedence constraints and partitioned dynamically under the restrictions of unidirectional workstations ALBP in manual automobile assembly lines. This new method depends on producing the optimal/near task allocations along the assembly lines and assigning the tasks to appropriate workstations in the lines. Furthermore, it aims to decrease the total production time and reduce the costs of whole manufacturing operation of assembly The experimental results show that their method produces outperforming solutions in manual automobile assembly lines and proves its validity and feasibility of this model for the first time.

Yang and Gao (2010) consider the effectiveness of multi-objective based GA on rebalancing problems in mixed model assembly lines to minimize the total processing times of reassigned tasks for scaling cost of rebalancing and integrating amount and difficulty of the reassigned tasks. The additional objectives of proposed GA are minimization of the amount of workstations and descending the differences between workstation time and mean workstation time for evert model. The results prove the proposed method is able to solve at the rebalancing problems for mixed model assembly lines and works smoother as compared to traditional heuristic techniques.

Yu and Yin (2010) propose an adaptive GA with reconfigurable crossover and mutation rates which are dynamically determined based on the fitness value of each individual in order to minimize the number of workstations and balance the workload between workstations. Additionally, sequence-based coding solution guarantees that chromosome represented precedence constraints of tasks at assembly ALBP. The computational results illustrate that proposed adaptive GA performs better solution in comparison with traditional heuristic solution methods.

Akpinar and Bayhan (2011) present a hybridized GA for mixed model ALBP with parallel workstations and zoning restrictions in order to minimize the amount of workstations, increase the smoothness of workload among workstations by using three well-known heuristic methods as RPW, Kilbridge and Wester, Moodie and Young on generation of initial population. The examined results indicate that their GA produces better solutions with higher quality as compared with traditional GAs.

Kazemi et al. (2011) implement a novel two-stage GA method to minimize total cost related with amount of workstations and duplication costs of tasks for balancing U-shaped ALBP. The operation procedure of this new GA method is based on selecting population primarily, and detecting the best workstation for the assigned tasks. The tests show that their GA works well for small and medium-sized problems with short computational time; on the other hand, for large-sized problems, GA finds solution in acceptable computational times.

Ozcan, Kellegoz and Toklu (2011) focus on effectiveness of GA on both of the problems of sequencing and balancing in U-shaped assembly lines by minimizing the number of workstations for predetermined cycle time with stochastic processing times. Their model is capable to solve two interrelated problems simultaneously, which are line balancing and model sequencing. It primarily initializes the

population and operates a local search procedure in order to improve the best individual performance within the production process. Then, if the termination condition is not fitted into the location, the model iterates the following operations: It chooses two parent chromosomes from the current population and detect which solution component is going to be applied. If the random value of generated one is below the predetermined score, then, it implements the balance part for the solution. The results signify the validity and feasibility of their GA in balancing and sequencing problems for U-shaped assembly lines.

Taha et al. (2011) consider GA for two-sided assembly lines problems to minimize the amount of mated workstations for improving the efficiency of line. Their proposed GA improve generation of initial population by using new methods as forward, backward and combination backward and forward methods at creating the initial population while hybrid crossover and a modified scramble crossover genetic operators are used in in order to extend the search space efficiently for finding best accessible results. The experimental results prove that proposed GA is able to reach the optimum or nearly optimum results with restricted amount of iterations.

Zhang and Gen (2011) analyze effect of the demand ratio-based cycle time for solving balancing problems in order to reach better results. The researchers apply a new GA for multi-objective mixed model ALBP by objecting minimization of the cycle time, diversification of workload between workstations and total cost of line while pareto-based scale is also implemented to measure the efficiency of assembly line. The examined results indicate that new GA is able to perform well for mixed model ALBPs.

Chen et al. (2012) introduce a GGA for ALBP with various labour capabilities and to smooth the workload among workstations at garment industry. The aim is to minimize the mean absolute deviations in production line in order to enhance the efficiency of assembly line. Their GGA model provides shorter time cycle for production line and higher level of production quality. Therefore, their model can be important role for garment industry in current agenda.

Hamzadayi and Yildiz (2012) present a priority based GA for balancing mixed model U-shaped assembly line problems with parallel workstations and zoning constraints in order to minimize the amount of workstation and smooth the workload among workstations. The fitness function of presented GA is inspired from fitness evaluation of simulated annealing optimization method. The iteration number can be defined by user in addition to required minimum amount of iteration. The computation results indicate that fitness evaluation based on simulated annealing method for priority based GA improves the performance of GA for practical-scaled problems with acceptable computational time.

Mamun et al. (2012) examine GA to optimize mixed model ALBPs by minimizing the number of workstations and make user to decide number of iterations. Researchers modify traditional GA by using heuristic method at reallocation of tasks after crossover process which violates the restrictions; used heuristic methods are MA, RPW, AllSucc, ImmSucc and maximum average processing times. The tests prove that their GA is able to solve small and large sized problems in reasonable computational times.

Wang, Che and Chiang (2012) work on hybridized multi-objective based GA as on product plan selection problem in sequencing and balancing conditions of assembly lines. They combine guided modified GA and weighted pareto based multi-objective GA to find the most effective production plan by creating mathematical model with multiple objectives for choosing the product plan. Four different performance measures are taken into account, which are minimization of the cost, time and amount of workstations and reaching maximum amount of connector homogeneity. The experimental results show that pareto optimal process is effective for balancing multi-objective ALBP with large dimensions.

Yolmeh and Kianfar (2012) propose a hybridized GA via implementing dynamic programming method for setup balancing and scheduling SALBP. Every chromosome's scheme should include two properties as task allocation to workstation and sequence of operating tasks in a workstation. A basic permutation is applied for representation of a chromosome and initial population consisting of these chromosomes is randomly created; this basic permutation decides the sequence of tasks operated by worker in every workstation. Dynamic programming method is combined with GA to find an optimal tasks allocation to workstations considering sequencing too. The GA operators and parameters are rearranged by multifactor diversification analyse. The examined results signify that their GA produces better solutions comparing with other algorithm applied to solve same ALBP type.

Akpinar, Bayhan and Baykasoglu (2013) combine GA with ACO method for mixed model ALB with setups Type-1 problem in order to improve performance of ACO by inserting GA as local search plan while considering specific characteristics as parallel workstations, zoning restrictions and sequence based on setup times among workstations. At presented model, on one hand ACO enhances diversification of algorithm, on the other hand, GA improves the intensification since they aimed to speed up ACO and to extend the search space GA. The experimental results show that presented model performs better in comparison with pure GA and ACO, other hybrid GA methods.

Mutlu, Polat and Supciller (2013) implement an iterative GA for assembly line worker assignment and balancing problem with objective of minimization of cycle time in order to optimize the processing times difference caused from worker qualifications by considering task and worker allocation to workstations. Researchers use three search methods, which are GA, iterated local search and modified bisection search, to enhance diversification and efficiency at their model; the all parameters and operators are converted to chosen type of ALBP and heuristic methods are applied on computation of cycle time, organizing the task and worker allocation and performance sequence of tasks. The results indicate that proposed iterative GA gives efficient solution for assembly line worker assignment and balancing problem.

Purnomo, Wee and Rau (2013) introduce a mathematical model with application GA and iterative first fit rule to balance two-sided assembly lines with allocation restrictions by minimizing cycle time with predetermined amount of matched workstations. The allocation restrictions of two-sided ALBP are zoning, resource, workstation, distance and synchronous task constraints. GA uses the greedy search at genetic operator to control local and global search while iterative first fit rule implements the minimization of the gaps between hypothetical and existing cycle time according to predecessor best cycle

time. The tests prove that GA performs in less computational time than first fit rule which submits superior cycle time for medium sized problems.

Zacharia and Nearchou (2013) focus on a single model ALB with fuzzy task operation times in order to maximize efficiency of assembly line by taking into account of minimization of amount of workstation and cycle time. Their model is formulated based on triangular fuzzy membership functions and GA is used to solve the model. Researchers apply two-stage GA that GA randomly creates initial population and proceeds evaluation by constituting high qualified solutions at first stage, GA works one more times restarting from a new randomly generated initial population which is seeded by the chromosome with best solution. Their mode produces outperforming results for single model ALBP with fuzzy processing times.

Baykasoglu and Ozbakir (2015) optimize SALBP effectively by applying single pass heuristic task assignment rules which are found out via genetic programming. Their model is used in two combined segments: the fundamental algorithm is applied as genetic programming that reviews the balancing module in which single pass heuristic solution methods are carried out based on rules created by genetic programming. Every individual in genetic programming population is a task allocation rule to allocate tasks to workstation based on chosen objective and problem.

Kucukkoc and Zhang (2015) present a mathematical formulation and GA to solve parallel two-sided ALB in which at least two number of two-sided assembly line are structured to each other. They aim to examine the all system to various stages of parallel lines' cycle time and reach the best cycle time couple that submits the most efficient level of assembly line. The presented GA produces promising results for balancing parallel two-sided assembly lines.

Sikora et al. (2015) develop a GA for balancing assembly line problems with objective of minimizing cycle time with predetermined amount of workstations by considering precedence constraints and physical restrictions. The developed GA concentrates on reaching the efficient search space and gives nearly optimal results for instances in the literature.

Delice, Aydogan and Ozcan (2016) analyze GA and well-known priority rule based heuristic methods to optimize the stochastic two-sided, U-shaped ALBP by minimization of amount of positions and workstations. Researchers explain the solution process via examples and adapt new allocation and selection approaches to their model. The computational results indicate that proposed model gives outperforming solution for two-sided, U-shaped ALBP.

Tang et al. (2016) propose a hybrid GA with novel logic strings for mixed model ALBP with sequence-based task to minimize the cycle time and balance workload between workstations. Firstly, sequence-based connections and precedence network are inserted into the incorporated precedence diagram to convert the actual problem to single-model ALBP. Secondly, three heuristic methods as processing time, number of immediate successors and number of updated tasks are used to generate the initial population of GA. Thirdly, logic strings are formed to guarantee chromosome feasibility while two-

point crossover and insertion mutation process. The experimental results signify that proposed GA is able to find near optimal solution in reasonable computational time.

Zhao et al. (2016) create a mathematical model for balancing assembly line with multiple objectives respecting quality of product and efficiency at production by considering the mental workload which is defined as the consuming human internal resource to fulfil a task. After structuring model, GA is used to solve chosen problem by mental workload and task assignment to workstations regarding to cycle time constraint. The examined results prove that altering cycle time at every workstation is effective on solution quality and experience has an important effect for mental workload.

4. Experimental Study Review

4.1 Summary of The Paper of Pitakaso & Sethanan (2015) (Pitakaso&Sethanan, 2015)

Pitakaso and Sethanan (2015) introduce "*DE-C*" for SALBP, beside the classic DE for SALBP-1 with minimization of number of workstations with a given cycle time. Additionally, they also take account of the number of machine types operated at every workstation (*SALBP-1M*) in order to improve the space usage at the production line. In SALBP-1M, every task has to be practiced at specific machine with consideration of number of machine types for each workstation while satisfying precedence relation, cycle time constraints. Figure 4.1 demonstrates a simple model of precedence diagram for SALBP-1M with eight tasks and three different machine types in which every task is operated particularly:

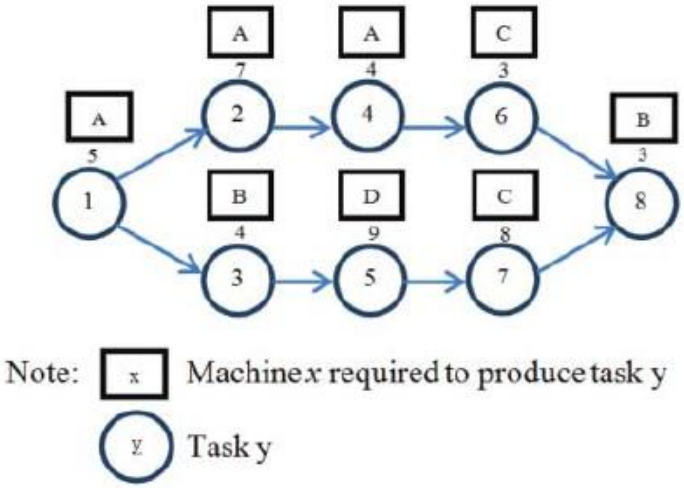


Figure 4.1 Precedence Diagram of Simple Example for SALBP-1M. (Pitakaso&Sethanan,2015, pp.2)

In general, *DE* algorithm consists of four steps as generation of initial solution, mutation, recombination and selection. The recombination process actually produces one trial vector; in the modified algorithm DE-C of Pitakaso and Sethanan, two trial vectors are created as one dominant trial vector (gene) and one recessive trial vector (gene) with an inspiration from human genes in order to perform better quality compared the traditional DE algorithm.

Generation of an initial set of target vectors is the first step in order to have a population from randomly generated target vectors. Every vector has dimension *D* which is the amount of tasks to be allocated to a particular number of workstations. For example, the precedence diagram at Figure 4.1 has eight different tasks, so a vector with eight dimensions is randomly created to symbolize one solution;

when the population size is fixed to six vector, it means that six vectors with eight dimensions are randomly created. At this first step of modified DE, they generate a random number for each task between 0 and 1 and the tasks are sorted based on these assigned random numbers in order to satisfy chosen objective function. During the assignment process of SALBP-1M, the proposed restrictions for number of machine used in each workstation must be considered beside general restrictions as cycle time, precedence relations.

We can explain the assignment process of SALBP-1M on an example. The precedence diagram at Figure 4.1 has eight tasks. They randomly generate values for each task/position at target vectors and decode one vector into solution at Figure 4.2:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0.92	0.08	0.65	0.05	0.99	0.02	0.68	0.44

Figure 4.2 Example for a target vector (Pitakaso&Sethanan,2015, pp.6)

The first row represents the values in a position of these tasks and second row represents the randomly generated number for each task; the assignment sequence of tasks can be shown as below with priority of decreasing values process:

5	**1**	**7**	**3**	**8**	**2**	**4**	**6**

Figure 4.3 Assignment sequence of the given target vector.

The cycle time is 12 and the limit for the number of machine is 1 at the given example; when we satisfy the precedence relations, cycle time and machine limit restrictions considering the assignment sequence for the assignment procedure of SALBP-1M, we should start with task 1 with value 0.92 and machine A due to lack of predecessors. Then, we should check the sequence, machine limit and cycle time on allocation of the second task; according to assignment sequence, it should be task 5, since we consider the other restrictions, we should allocate the task 2 with processing time 7 and machine A to the first workstation. We should open the second workstation since there is no remaining time at the first workstation; task 3 and task 4 are the candidate for assignment. When we take into account the assignment sequence, we should allocate task 3 with value 0.65 since it has greater value than task 4; after allocation of task 3, remaining time of workstation 2 is 8. If we check the all constraints especially machine

limitation, there is no eligible to task to assign to workstation 2, so we close workstation 2 and open new workstation 3. Task assignment procedure of given target vector continues based on SALBP-1M process at Table 4.1.

Workstation No	Candidate tasks	Assigned tasks	Remaining Time	Machine Type
1	**1** **2,3**	**1** **2**	**7** **0**	**A** **A**
2	**3,4**	**3**	**8**	**B**
3	**4,5**	**5**	**3**	**B**
4	**4,7**	**7**	**4**	**D**
5	**4**	**4**	**8**	**C**
6	**6**	**6**	**9**	**C**
7	**8**	**8**	**9**	**B**

Table 4.1 Task assignment procedure of target vector for solving SALPB-1M

The result of task assignment procedure is shown in Table 4.2:

Workstation	Workstation 1	Workstation 2	Workstation 3	Workstation 4	Workstation 5	Workstation 6	Workstation 7
Task	1,2	3	5	7	4	6	8
Machine type	A	B	D	C	A	C	B

Table 4.2 Results of task assignment procedure of target vector for solving SALPB-1M

Mutation process is performed as second step; the *mutant vector*, $V_{i,j,G}$ is created using a set of randomly selected target vectors, $X_{i,j,G}$ from current generation and a vector which produces better solution so far at the proposed algorithm by applying the formulation of Storn (2008) for creating mutant solution :

$$V_{i,j,G} = X_{best,j,G} + F(X_{r1,j,G} - X_{r2,j,G}) + F(X_{r3,j,G} - X_{r4,j,G}) \quad (4.1)$$

r1, *r2*, *r3* and *r4* are the randomly chosen vectors/individuals from individual index, *i*; *j* is the position index and *G* is the generation index, best refers to best vector reached so far by introduced algorithm. The scale factor, *F* is determined as *2* at DE-C.

Recombination process are applied as third step and two different kind of trial vectors are generated: *dominant trial vector* and *recessive trial vector* which is firstly presented in DE-C.

The dominant trial vector is created under traditional recombination process by using the formulation of Pitakaso (2014) in which a random number is generated between 1 and j, number of the positions/tasks and values of the position between these numbers are taken from the chosen target vector and the rest are copied from the chosen mutant vector:

$$U^{1}_{i,j,G} = \begin{cases} X_{i,j,G} & \text{when} \quad rand_{i,j,1} < j < rand_{i,j,2} \\ V_{i,j,G} & \text{otherwise} \end{cases} \tag{4.2}$$

The recessive trial vector is created under modified recombination process by applying three types of process:

1. *Vector transition process* is applied as following: randomly generation of number for transition points, randomly selection of transition points at the vector and randomly replacement of the values at the selected positions with randomly generated numbers between 0 and 1.
2. *Vector exchange process* is randomly selection of two positions and exchange their values.
3. *Vector insertion process* progresses like that: randomly selection of one insertion and one moving point, insertion of the value at the moving point to insertion point and one position movement of the values between these two points.

Randomly chosen transition points

Original/Dominant vector

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0.92	0.08	0.65	0.05	0.99	0.02	0.68	0.44

Accompany/Recessive vector

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0.45	0.15	0.65	0.05	0.73	0.02	0.68	0.44

Figure 4.4 Vector transition example (Pitakaso&Sethanan,2015, pp.9)

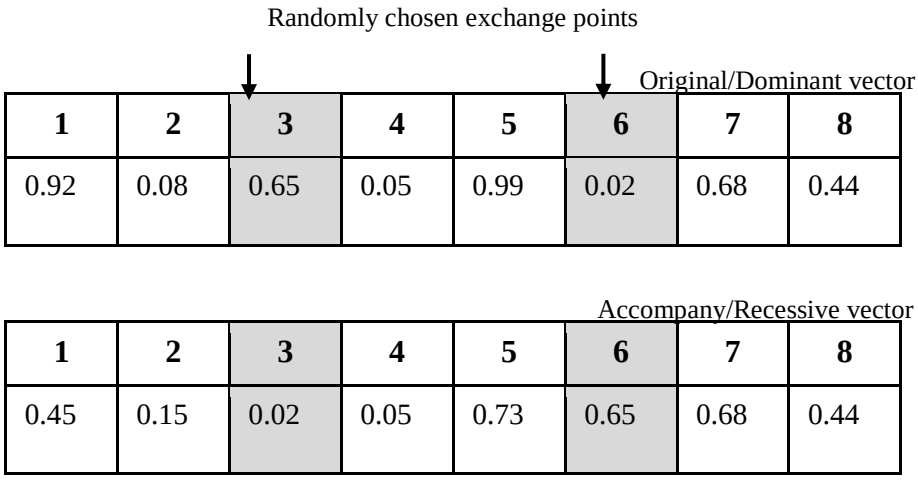


Figure 4.5 Vector exchange example (Pitakaso&Sethanan,2015, pp.9)

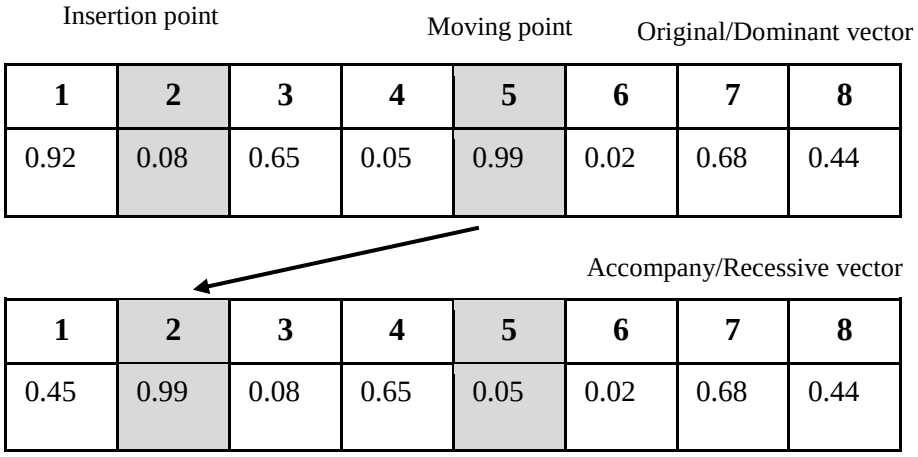


Figure 4.6 Insertion vector example (Pitakaso&Sethanan,2015, pp.9)

The recessive trial vector, $U^{2}_{i,j,G}$ is modified by adding three more positions to original trial vector; these three new positions represent the number of creation of vector processes: *T* is value in the position of vector transition process, *E* is value in the position of vector exchange process, *I* is value in the position of vector insertion process. The highest value between these positions of T, E, I is chosen to create the recessive trial vector:

1	2	3	4	5	6	7	8	T	E	I
0.92	0.08	0.65	0.05	0.99	0.02	0.68	0.44	0.15	0.61	0.50

Figure 4.7 Extended vector used to choose the creation process (Pitakaso&Sethanan,2015, pp.10)

The performance of dominant trial vector, $U^1_{i,j,G}$ is compared with the performance of the recessive trial, $U^2_{i,j,G}$ t and the trial vector is determined according to formulation below:

$$U_{i,j,G} = \begin{cases} U^{1}_{i,j,G} & \text{if} \quad f\left(U^{1}_{i,j,G}\right) \le f\left(U^{2}_{i,j,G}\right) \\ U^{2}_{i,j,G} & \text{otherwise} \end{cases} \tag{4.3}$$

In DE-C, just one recessive vector is created from every dominant vector to have shorter computational time.

Selection process is the final step, whose result is chosen as a *target vector* to be in following generation and it will be an initial point of the mutation process at next generation. At the selection process, the best vector, either the trial vector or the target vector, is chosen for next generation according to formulation below:

$$X_{i,j,G+1} = \begin{cases} U_{i,j,G} & \text{if} \quad f\left(U_{i,j,G}\right) \le f\left(X_{i,j,G}\right) \\ X_{i,j,G} & \text{otherwise} \end{cases} \tag{4.4}$$

They compare performance of DE-C with several benchmark problems, such as GAs, tabu search algorithms, DE particle swarm optimization to solve SALBP-1; their DE-C performs better than some of compared algorithm or produces same results. They analyse the output of traditional DE and modified DE-C, both have quite same solution quality but DE-C reaches to optimal solution in less computational time.

They also apply DE-C method to two case studies as Pitakaso-1 with 36 jobs-6 machine types and Pitakaso-2 with 52 jobs-5 machine types for SALBP-1M considering the machine limits; DE-C performs much better than classical heuristic priority methods. The results of DE-C are explained in details at the following sections.

4.2 GAMS Results

Two different case studies of SALBP-1M taken from Pitakaso and Sethanan (2015) are solved by GAMS (General Algebraic Modelling System) in order to find optimal solutions of SALBP. Pitakaso-1 problem comprises of 36 jobs and 6 machine types with various cycle times 1.23, 1.80, 2.00, 2.50 and 3.00 seconds; precedence diagram of Pitakso-1 is shown in Figure 4.8. Pitakaso-2 problem consists of 52 jobs and 5 machine types with different cycle times 1.89, 2.00, 2.20, 2.50 and 2.95 seconds; precedence graph of Pitakaso-2 is illustrated in Figure 4.9.

These two cases of SALBP-1M objects to minimize the number of workstations with a given cycle time by considering the machine limitation for every workstation. As is seen from precedence diagrams of Pitakaso-1 and Pitakaso-2, each job is operated on specific machines under particular processing times. These two problems are optimized based on two different machine limits as single machine and two machines with 5 different cycle time for each problem, so we have 20 instances to optimize.

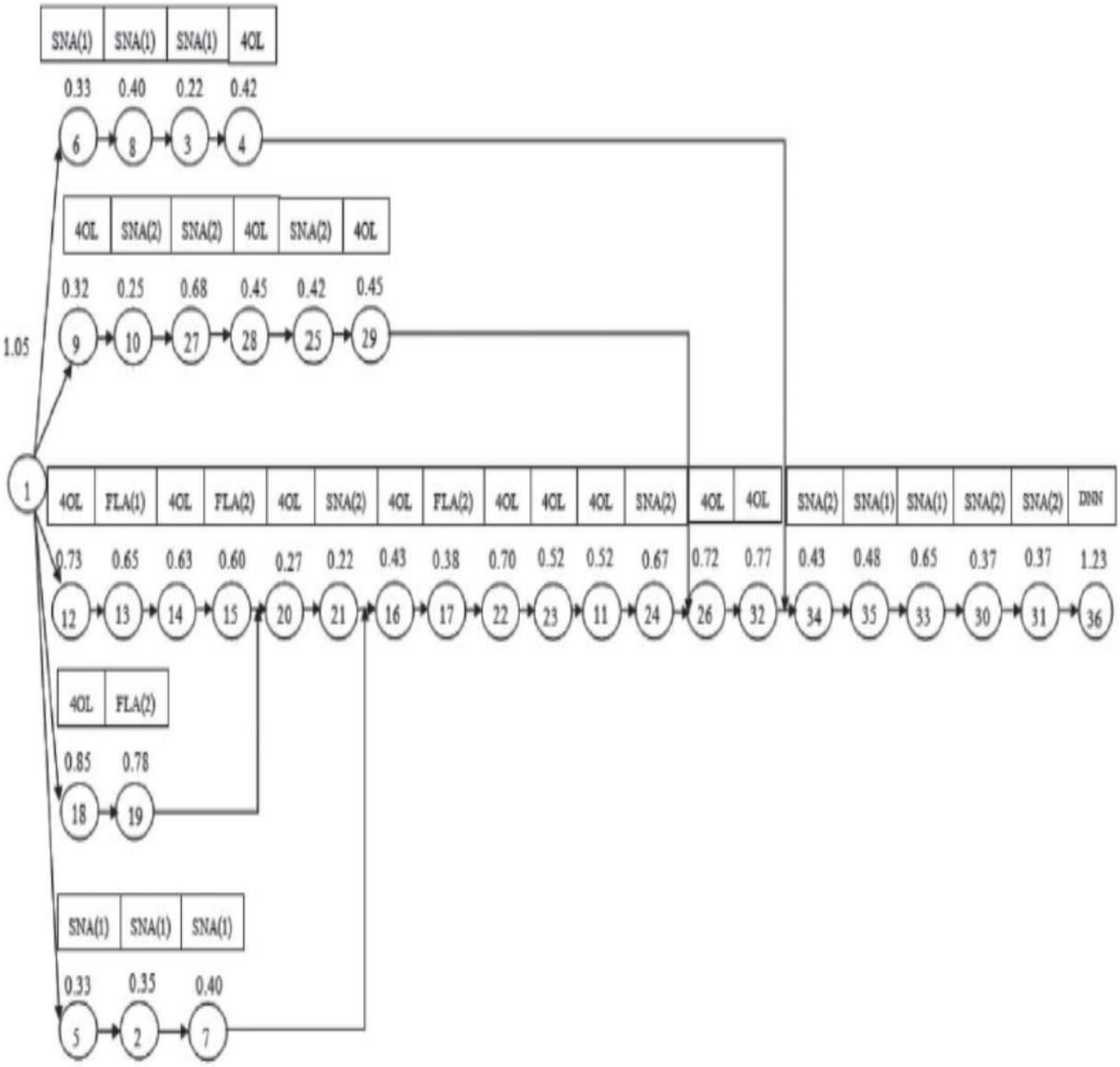


Figure 4.8 Precedence diagram of Pitakaso-1 problem (Pitakaso&Sethanan,2015, pp.15)

The mathematical formulation of SALBP-1M proposed by Pitakaso and Sethanan (2015) is analysed considering all parameters, indices, decision variables and constraints in order to formulate the problemwith GAMS. The indices, parameters and decision variables are applied in the mathematical model of SALBP- 1M outlined below (Pitakaso&Sethanan, 2015):

Indices:

n index of tasks n while $n=1,.....,N$

k index of workstation k while $k=1,.....,M$

N total number of tasks

M total number of workstations

G total number of machines

BM big number suck as 10.000

LG highest number of machines in each workstation

Parameters:

P_n processing time of task n

CT cycle time of workstation

$F_{nj}=1$ if task n is predecessor of task j; otherwise 0

$W_{ng}=1$ if task n uses machine g to produce; otherwise 0

Decision variables:

$X_{nk}=1$ if task n assign to workstation j; otherwise 0

$Y_k=1$ if workstation k is opened; otherwise 0

$H_{kg}=1$ if workstation k is operated by machine g (at least one per machine); otherwise 0

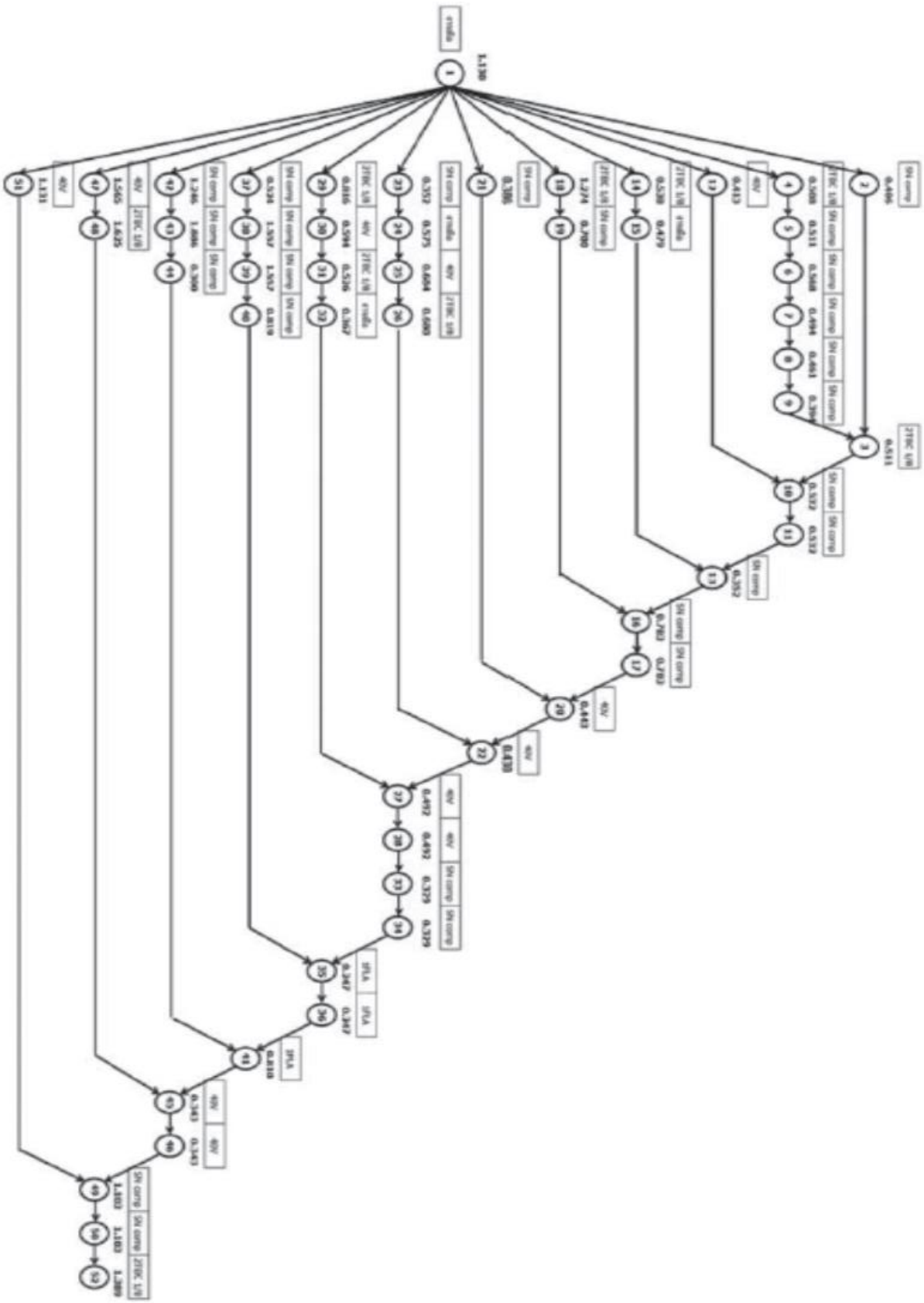


Figure 4.9 Precedence diagram of Pitakaso-2 problem (Pitakaso&Sethanan,2015, pp.16)

Objective function:

$$\text{Min Z} = \sum_{k=1}^{M} Y_k \tag{4.5}$$

subject to:

$$\sum_{k=1}^{M} \text{X}_{\text{nk}}=1 \quad \forall \text{n} = 1,..,N \tag{4.6}$$

$$\sum (k \,.\, X_{jk}) - (k \,.\, X_{\text{nk}}) \geq 0 \; \forall n = 1,\ldots,N,\; j = 1,\ldots,M,\; F_{nj} = 1 \tag{4.7}$$

$$\sum X_{nk} \,.\, P_n \leq CT \,.\, Y_k \; \forall k = 1,\ldots,M \tag{4.8}$$

$$Y_k \leq Y_{k-1} \; \forall k = 2,\ldots,M \tag{4.9}$$

$$\sum X_{nk} \,.\, W_{ng} \leq H_{kg} \times BM \; \forall k = 1,\ldots,M,\; \forall g = 1,\ldots,G \tag{4.10}$$

$$\sum H_{kg} \leq LG \; \forall k = 1,\ldots,M \tag{4.11}$$

The first equation (4.5) is objective function of minimization of the number of workstations. Equation (4.6) assures that each task has to be allocated to just one workstation; equation (4.7) represents the precedence relations between tasks; equation (4.8) checks that processing time of each workstation cannot exceed the predetermined cycle time. Equation (4.9) controls that the workstations are in increasing under during task assignment procedure. Equation (4.10) guarantees that when each task is assigned to workstation, machine which task is operated must be allocated to same workstation; equation (4.11) represents that maximum machine type limit cannot be surpassed by assigned machine types to each workstation.

These mathematical formulation is used to create our GAMS model in which indices are *n* task, *k* workstation and *m* machine, parameters are outlined to assign values for each element of every set; decision variables are defined algebraically with their particular indices. GAMS produces every instance of the variable based on specified type. Objective function and equations of related constraints are stated with specific names and their mathematical formulation are defined. The GAMS model is named as *ALB* and all equations are used in the model. Finally, GAMS solves the problem by CPLEX solver while minimizing the objective function value.

The optimal results of Pitakaso-1 and Pitakaso-2 with two different machine type limits, obtained from GAMS are shown below:

PITAKASO-1			
Single Machine		**Two Machines**	
Cycle Time	**GAMS Optimal Results**	**Cycle Time**	**GAMS Optimal Results**
1,23	22	1,23	17
1,8	18	1,8	12
2	17	2	11
2,5	17	2,5	9
3	17	3	8

Table 4.3 GAMS results of Pitakaso-1 for solving SALPB-1M

PITAKASO-2			
Single Machine		**Two Machines**	
Cycle Time	**GAMS Optimal Results**	**Cycle Time**	**GAMS Optimal Results**
1,89	25	1,89	22
2	23	2	20
2,2	22	2,2	19
2,5	18	2,5	16
2,95	16	2,95	13

Table 4.4 GAMS results of Pitakaso-2 for solving SALPB-1M

4.3 Proposed Models for Solving SALBP-1M

Single pass heuristic solution methods based on priority rules and GA are introduced in order to solve 20 different instances of Pitakaso-1 and Pitakaso-2 for SALBP-1M.

4.3.1 Proposed Single Pass Heuristic Methods

Different single pass heuristic methods are applied to find an optimal solution for 20 instances of SALBP-1M; some of the applied rules are same with priority rules used by Pitakaso&Sethanan (2015) in order to compare the results of each solution.

MA priority rule is used for sorting the jobs with respect to decreasing order of their processing times and *MI* priority rules is implemented for sorting the jobs based on increasing processing times from smallest to largest. *RPW* is tested to allocate the jobs with respect to decreasing positional weight value. *Random Selection* rule is applied in order to assign jobs to workstation by randomly selecting the successor job to put into current workstation. *ImmSuc* rule is used for allocation of jobs regarding descending number of direct successors and *AllSuc* rule is tested for sorting jobs with respect to decreasing number of all successors. These six priority based heuristic methods are applied according to *greedy approach* by selecting the job at the top of the sorted list to balance SALBP-1M.

In order to extend the solution space of algorithm, *Randomized RPW* technique is implemented by randomly selecting *n* number of jobs instead of top ranked job. There is a cumulative lookup table *LT* containing the cumulative selection probabilities for each eligible job to assign and sum of the probabilities has to be equal to 1. In this procedure, a number between 0 and 1 is randomly chosen, then it is checked at which interval this randomly chosen number is located in cumulative probabilities of eligible jobs in order to decide the job for assigning to workstation. *Randomized RPW* can be explained with a small example: We may assume 3 jobs are randomly selected from eligible jobs, the first selected job has 70% probabilities to be assigned, the second has 20% and the third has %10; in this case by considering the cumulative probabilities of each job, LT (1) is 0,7; LT (2) is 0,2 and LT (3) is "1". If randomly chosen number is 0,8; so the randomly selected second job is firstly assigned to workstation.

Task assignment procedure with respect to "*first fit*" jobs to be placed to workstation is explained following: There must be eligible jobs which do not have immediate predecessors to initiate the proposed algorithm which works through five steps. At the first step, the *selected job* is determined by using *priority rule* which is chosen from applied priority rules at proposed model. At the second step, the first station no is assumed as "*0"*, the *first available workstation* is searched in order to assign the *selected job* by looking for where all immediate predecessors of selected job are allocated; then workstation with the minimum number is chosen.

Third step is the assignment of *selected job* to *first available workstation* considering cycle time and machine type limit. If the *processing time of selected job* is less than remaining time of the workstation and the *machine type limit* is examined by checking the *machine of selected job* is already placed in this

workstation, the selected job can be assigned to this workstation and remaining time is updated by subtracting the processing time of selected job. If the remaining time constraint is satisfied but the machine of selected job is not already placed in this workstation, the number of machine type must be taken into account by controlling the remaining places for the selected job's machine; supposing that there is enough place for the selected job's machine, selected job and the machine of selected job can be assigned to this workstation, thus the remaining time of this workstation is updated by subtracting the processing time of selected job and the machine type limit is also amended by adding the machine of selected job to this workstation.

If *selected job* is still not assigned, the algorithm moves to the fourth step in which *a new workstation is opened*; selected job and the machine of selected job is assigned to newly opened workstation, machine type limit and remaining time of the workstation must be adjusted as well.

At final/fifth step, the eligible jobs list must be amended by eliminating the assigned selected job from eligible jobs and accepting as already placed job. In order to update the eligible job list, immediate predecessors of selected job's immediate successors are checked if they are already assigned to workstations; if not, they are considered as eligible job to assign. The proposed algorithm goes through all steps until there is no eligible job to assign. The steps of proposed algorithm are shown at Figure 4.10 below:

```
While (|E| > 0)
1       Set selectedJob using the seleceted "Priority Rules"
2       Find the firstStationId that selectedJob can be assigned
3       Assign the selectedJob to the "first available" workstation
4       Given the selectedJob is not assigned, then open a new workstation and assign the job
5       Update the eligible jobs and already placed jobs lists
End While
```

Figure 4.10 Pseudo code for task assignment algorithm with first fit rule

4.3.2 Proposed Genetic Algorithm

GA is very promising metaheuristic approach for solving difficult optimization problems due to its ability on moving from one solution set to another one and its flexibility on incorporation of particular characteristics of the problem.

The effectiveness of GA is based on the chosen operators which are the key components for solution structure. GA and its concepts are explained in section 3.3.6; the proposed GA for solving SALBP-1M is explained in this section.

Encoding used in proposed GA is that individuals are encoded using the number of tasks to be assigned. Each individual encoding is composed of a task sequence, which indicated the priority during the task assignment procedure. At generation of ***initial population***, individuals are generated randomly or applying single pass heuristic methods: special individuals are generated by implementing *RPW, MA* and *MI* priority based heuristic task assignment rules, so assignment sequence of each special individual is based on one of these rules; remaining number of individuals are created randomly, in which a random number between 0 and 1 is generated for each task and tasks are sorted with respect to assigned number from greater to smaller.

Once the assignment sequence is decided, *task assignment procedure* can be applied in order to solve SALBP-1M. Two different approaches are implemented for task assignment procedure: *first fit approach* is task placement with respect to allocation of each task to first available workstation (explained in previous section) considering the machine and cycle time constraints and *best fit approach* is task placement with respect to allocation of each task to the best suitable workstation considering nearly matching remaining time of workstation and checking all open workstations, task processing time, other constraints as well in order to minimize the remaining idle time at workstations.

Fitness function is determined based on the objective of our SALBP-1M which is to minimize the number of workstations with machine type limit constraints; by taking inspiration from double-barrelled objective function of Sabuncuoglu et al. (2000) following formulation is used in proposed GA to find *better balanced* solution.

$$Fitness = \sqrt{\frac{\sum_{s=1}^{S} (W_{max} - W_s)^2}{S}} + 2\,\frac{\sum_{s=1}^{S} (W_{max} - W_s)}{S} \qquad (4.12)$$

In the equation 4.12, W_s is the workload of workstation s, W_{max} is the workload of the workstation with the highest workload and S is the number of workstations in the solution. The first part of the fitness function objects to reach the best balance between the solution with same number of workstations while the second part aims minimization of number of workstations; the second part is multiplied with '*2*' since the second part has more importance for our optimization problem.

Selection process is executed in proposed GA as following: *Elitism* is to migrate the proportion of best individuals based on their fitness values from current generation to next generation in order to keep the high quality individuals of current generation, *migration ratio* for elitism is determined as *0,20*; the remaining number of individuals are selected with respect to *selective random/uniform process* in which an individual is chosen randomly from current population during a pure random individual is generated, then the fitter individual is selected between them. Roulette wheel and ranking selection parameters are also defined in the algorithm as an option which can be provided by user using an external *.xml file*.

Crossover process aims to carry characteristic information from genes of parents to offsprings to the next generation as a genetic operator. Two parents are needed for mating to produce an offspring and therefore, the first parent is chosen via *selective random/uniform process* and second parent is chosen via *roulette wheel by fitness* with regard to minimization objective of our problem; several crossover methods are applied in proposed GA.

a. *Modified uniform crossover* takes into account the fitter of the parents by creating threshold probability with respect to fitness values of parents', so that the generated offspring inherits more characteristic from fitter parents. Since our objective function is based on minimization, inverse proportion is considered for calculating threshold probability. This method can be explained on simple example: 2 parents, one with fitness value as 40 and the other with fitness value as 80, are selected by going through the processes above for mating; threshold probability can be found as 80/ (40+80). Then, random number is generated for each gene between *0* and *1*; if the randomly created number for each gene is smaller than the threshold probability 0.66, this gene is copied from fitter parent with fitness value 40 to offspring's gene; otherwise, this gene is taken from the other parent with 80 fitness value to offspring's gene.
b. *Uniform crossover* has a fixed mixing ratio to copy genes from parents, in which each gene is tested to copy it from one of the parent; 0.5 is used as mixing ration at proposed GA. A random number is generated between *0* and *1*; if randomly generated number is smaller than fixed mixing ratio 0.5, the gene is taken from first parent which is chosen under selection process, otherwise the gene is taken from second parent which is selected based on roulette wheel method.
c. *Bias to fitter uniform crossover* uses the bias probability defined by user in order to take genes from fitter parent by testing each bit for creation of offspring. To create an offspring, a random number is generated between 0 and 1; if random number is smaller than selection probability, this gene is copied from fitter parent, otherwise it is duplicated from less fit parent.
d. *One-point crossover* randomly selects single crossover point, then the genes placed until this point are replicated from first parent and the remaining of genes are copied from second parent to produce first offspring; the genes positioned until this point are taken from second parent ad rests are taken from first parent at creation of second offspring. So, all data prior to this point is swapped between two parents and the fitter offspring is chosen.
e. *Two-point crossover* randomly decides two crossover point and the genes between these two points are exchanged between the parents in order to generate two offsprings; then, a fitter offspring is selected.

Some applied crossover methods' examples at proposed GA are illustrated below:

0.92	**0.08**	**0.65**	**0.05**	**0.99**	**0.02**	**0.68**	**0.44**

Parent 1

0.13	**0.41**	**0.91**	**0.15**	**0.16**	**0.34**	**0.61**	**0.21**

Parent 2

0.61	0.74	0.31	0.04	0.11	0.96	0.07	0.23

Randomly generated number

0.13	0.41	0.65	0.05	0.99	0.34	0.68	0.44

Offspring

Figure 4.11 Uniform crossover example with mixing ratio 0.5

0.92	0.08	0.65	0.05	0.99	0.02	0.68	0.44

Fitter Parent 1

0.13	0.41	0.91	0.15	0.16	0.34	0.61	0.21

Parent 2

0.61	0.74	0.31	0.04	0.11	0.56	0.07	0.23

Randomly generated number

0.13	0.41	0.65	0.05	0.99	0.02	0.68	0.44

Offspring

Figure 4.12 Bias to fitter uniform crossover example with selection probability 0.6

Crossover process is initiated by generation a random number, then comparison of this number with crossover probability; if randomly generated number is smaller than crossover probability, crossover process can be implemented. Crossover methods is chosen randomly and crossover probability is selected user to determine how often crossover of chromosomes is processed; crossover probability is chosen *0.8* after examining different values. After crossover process, the fitness values of newly created offsprings are calculated at proposed GA.

Mutation process is used to improve genetic diversity from one generation of a population to the next to avoid to be trapped in a local optimum by changing a little of genetic model of chromosome. Mutation process is initiated by creation of a random number as well and comparing this random number with mutation probability for application of process on one individual; mutation process occurs based on a user-definable mutation probability which is determined as 0.05, two different mutation methods are used in proposed GA.

a. *Bit by bit randomly mutation* has a fixed threshold for determination of whether to mutate gene or not. A random number is generated between 0 and 1; if random number is smaller than

threshold for mutation probability of chromosome, the gene is mutated by regenerated this gene randomly until all genes on chromosomes are tested.

b. *Two-point mutation* selects two points and the genes positioned between these points are replaced with a randomly created numbers.

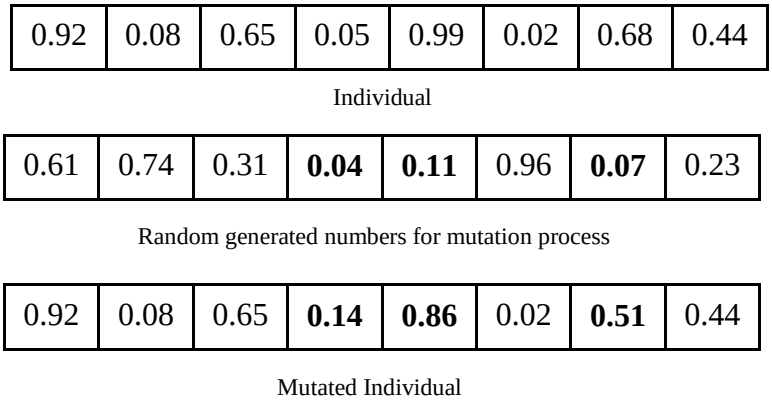


Figure 4.13 Bit by bit randomly mutation example with fixed threshold 0.15

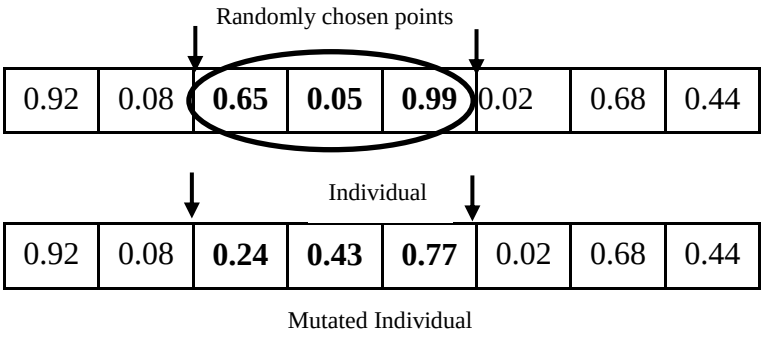


Figure 4.14 Two-point mutation example

Insertion process continues until population limit is reached; fitness values of individuals are calculated and these individuals with fitter values are replaced with current individuals of the population.

These migration, selection, crossover, mutation and insertion processes proceed until the ***stopping condition*** is reached; proposed GA stops when the specific numbers of generations have evolved. Since survival has a great importance in GA which outputs individual with a low fitness values and sustains the population to better solutions. Proposed GA finally reports the individual with the highest fitness value considering minimization problem in the final population. When GA reaches the stopping criteria, proposed GA reports the best performing individual as solution.

4.4 Comparison of Pitakaso&Sethanan (2015)'s and Proposed Model's Results

We have applied different single pass heuristic methods and proposed GA in order to find optimal solutions for 20 instances with a goal of minimization of number of workstations considering the machine limit for each workstation.

Pitakaso-1 problem comprises of 36 jobs and 6 various machine types as *SNA-1, SNA-2, 4OL,* FLA-1, FLA-2 and *DNN*; on the other hand, Pitakaso-2 problem consists of 52 jobs and 5 different types of machine as *SN0-comp, 2TBC-1/8, 4OV,1FLA.* There is two different machine limit at each workstation for two problems of Pitakaso; so, single machine and two machines restriction are considered for solving both problems. In addition, there are 5 different cycle time for each problem; Pitakaso-1 problem is solved with various cycle times as *1.23, 1.8, 2.0, 2.5* and *3.0* seconds and Pitakaso-2 problem has diversified cycle times as *1.89, 2.0, 2.2, 2.5* and *2.95* seconds.

Firstly, single pass heuristics results for both problem, 20 instances are analysed by comparing proposed model and Pitakaso and Sethanan (2015)'s model. The researchers implement RPW, MA and MI heuristic assignment rules to optimize instances under constraints of SALBP-1M. We apply same heuristic rules as well; in addition, we test ImmSuc, AllSuc, Pure Random, Randomized RPW single pass heuristic rules; except random assignment rules, greedy approach is used for task assignment procedure. Randomized RPW technique is explained in Section 4.3.1; pseudo code of Randomized RPW is summarized belo

```
If |E| < n
        Normalize LT(j) for all j∈ E such that ∑j LT(j)=1
End If
//Create r random number (0,1)
Find job j∈ E such that LT(j)≤r≤LT(j+1)
return E(j)
```

Figure 4.15 Pseudo code for Randomized RPW

In Figure 4.14, *j* represents job; *E(j)* represents eligible job list, *n* is the window length for randomly selecting job, *LT* is A cumulative lookup table containing the cumulative selection probabilities and r represents the randomly selected number between *0* and *1*.

Randomized RPW is applied with 3 and 4 number of jobs with two different selection probabilities for each to improve solution space at proposed single pass heuristic methods:

Heuristic Rule	Randomly Chosen Number of Jobs *n*	LT(1)	LT(2)	LT(3)	LT(4)
Randomized RPW1	3	0.70	0.90	1	
Randomized RPW2	3	0.60	0.90	1	
Randomized RPW3	4	0.60	0.85	0.95	1
Randomized RPW4	4	0.50	0.75	0.90	1

Table 4.5 The values used for Randomized RPW in proposed model

20 instances of Pitakaso's problems are solved by these single pass heuristic methods at our proposed algorithm based on task assignment procedure by assigning jobs to first available workstation. The results of our proposed single pass heuristic methods are shown for Pitakaso-1 problem with 10 instances at Table 4.6.

PITAKASO-1								
		Proposed Single Pass Heuristic Methods Results						
Machine Type Limit	Cycle Time	Pure Random	MA	MI	RPW	Randomized RPW1	ImmSucc	AllSucc
Single/One machine	1,23	22	24	26	22	26	26	23
Single/One machine	1,8	18	20	21	18	22	21	18
Single/One machine	2	17	20	21	17	22	21	17
Single/One machine	2,5	17	20	21	18	23	21	17
Single/One machine	3	17	21	21	18	23	21	18
Two machines	1,23	18	18	20	18	20	20	18
Two machines	1,8	13	13	14	12	14	14	13
Two machines	2,0	13	12	12	12	13	13	11
Two machines	2,5	10	10	10	9	11	10	9
Two machines	3	9	9	10	9	10	10	9

Table 4.6 Results of proposed single pass heuristic methods for Pitakaso-1

The algorithm runs 10 times for finding a solution of methods above and the outperforming results are marked with green. *Pure random* method produces better results than the other implemented methods for Pitakaso-1 with single machine limit. At Pitakaso-1 with single machine limit problem, *RPW* also gives same results with *pure random* method for cycle time 1.23, 1.8, 2.0 seconds and *AllSucc* method finds same solution with *pure random* method for cycle time 2.0 and 2.5. On the other hand, *Pure random*'s results are not the best for chosen each cycle time for Pitakaso-1 with two machines limit problem; but this method produces best result for cycle time 1.23 as same as *MA, RPW* and *AllSucc* algo-

rithms. For cycle time 1.8, only *RPW* outperforms for finding minimum number of workstation, *12* while *AllSucc* method gives the best results for cycle time 2.0 as 11 workstations for Pitakaso-1 with two machines limit problem. *RPW* and *AllSucc* methods finds the best results for cycle time 2.5, as 9 workstations; for cycle time 3.0, addition to these two methods, *pure random* and *MA* methods submit the best solution as 9 workstations for Pitakaso-1 with two machine limits.

In order to improve the solution quality, random algorithms run 1000 times and randomized RPW are diversified by increasing chosen random number of jobs and changing selection probabilities; the results of 1000 times run random algorithm are shown below Table 4.7.

PITAKASO-1						
		Proposed Single Pass Heuristic Methods Results				
Machine Type Limit	**Cycle Time**	**Pure Random (1000)**	**Randomized RPW1 (1000)**	**Randomized RPW2 (1000)**	**Randomized RPW3 (1000)**	**Randomized RPW4 (1000)**
Single/One machine	1,23	22	22	22	23	23
Single/One machine	1,8	18	18	18	18	18
Single/One machine	2	17	17	17	18	17
Single/One machine	2,5	17	17	17	18	17
Single/One machine	3	17	17	17	18	17
Two machines	1,23	18	18	18	18	18
Two machines	1,8	12	12	12	12	12
Two machines	2,0	11	11	11	11	11
Two machines	2,5	9	9	9	9	9
Two machines	3	8	8	8	9	8

Table 4.7 Results of proposed single pass heuristic randomized methods for Pitakaso-1

The algorithm runs 1000 times for finding a solution of each method above and the outperforming results are marked with green. *Pure random* method gives same results as before for Pitakaso-1 with single machine limit. On the other hand, results of *RPW1* are improved for each cycle time in comparison with the previous results of this method. According to Table 4.7, *pure random, RPW1* and *RPW2* produce the best results at every cycle time for Pitakaso-1 with single machine limit problem while *RPW4* also gives the best results all cycle times except cycle time 1.23. For Pitakaso-1 problem with two machines limit, all implemented methods find the best results for all cycle times, except *RPW3* only for cycle time 3.0 seconds. Pure random and RPW1 are developed for each cycle time comparing with their previous results for Pitakaso-1 problem with two machines limit.

The results of proposed single pass heuristics are compared with same heuristic methods implemented by Pitakaso&Sethanan (2015):

PITAKASO-1							
		Single Pass Heuristic Methods Compared Results					
Machine Type Limit	Cycle Time	MA	MA-Pitakaso	MI	MI-Pitakaso	RPW	RPW-Pitakaso
Single/One machine	1,23	24	24	26	27	22	22
Single/One machine	1,8	20	21	21	25	18	18
Single/One machine	2	20	21	21	21	17	18
Single/One machine	2,5	20	20	21	23	18	18
Single/One machine	3	21	19	21	23	18	18
Two machines	1,23	18	18	20	21	18	20
Two machines	1,8	13	13	14	13	12	13
Two machines	2,0	12	12	12	13	12	12
Two machines	2,5	10	10	10	10	9	10
Two machines	3	9	10	10	10	9	10

Table 4.8 Proposed single pass heuristic methods results comparison for Pitakaso-1

The best results are marked with green, the same results produced by both model are marked with colour orange at Table 4.8. For Pitakaso-1 problem with single machine limit, results of Pitakaso outperforms for *RPW* method with 2.0 cycle time and *MI* with 3.0 cycle time as 1 workstation difference for both; the results are same with both proposed methods which are *MA* with 1.23 and 2.5 cycle time, *MI* with 2.0 cycle time. *RPW* with each cycle time except 2.0 seconds. The proposed model outperforms at *MA* with 1.8 cycle time, *RPW* with 2.0 cycle time as 1 workstation difference for both; on the other hand, *MI* produces much better results in comparison with Pitakaso at every cycle time except 2.0 seconds for Pitakaso -1 problem with single machine limit. For Pitakaso-1 problem with two machines, there are 7 tie results between two proposed models, especially at *MA* method and our proposed model gives better results especially at *RPW* method. To sum up, our proposed single pass heuristic model's results are better of single pass heuristic model of Pitakaso&Sethanan (2015) at almost every instance of Pitakaso-1 problem, after improvement of our random methods.

The results of our proposed single pass heuristic methods are shown in Table 4.9. for Pitakaso-2 problem set which is composed of 10 different instaces. The algorithm runs 10 times for finding a solution of each method below and the outperforming results are marked with green.

For Pitakaso-2 problem with two machines limit, *pure random*, *MI* and *ImmSucc* methods are not able to produce the best results for all cycle times. Randomized *RPW1* outperforms for cycle time 1.89, 2.5

and 2.95 with 27, 21 and 20 number of workstations while *RPW* gives the best results for cycle time 2 and 2.2 with 25 workstations per each cycle time as same as *AllSuc* method for cycle time 2.2 at Pitakaso-2 problem with single machine limit. For Pitakaso-2 problem with two machines limit, *ImmSuc* and *AllSuc*c methods are not able to submit the best results at each cycle time. *MA, RPW* and *Randomized* RPW1 produce outperforming results as 23 workstations for cycle time 1,89 while *Pura Random, MA* and *Randomized RPW1* give the best outcomes as 21 workstations for cycle time 2 seconds at Pitakaso-2 problem with two machine limits. *RPW* produces the best solutions as 19 workstations for cycle time 2.2 and as 14 workstations for cycle time 2.95; *MA* is able to find the best result as 16 workstations for cycle time 2.5 seconds in Pitakaso-2 problem with two machines limit.

PITAKASO-2								
		Proposed Single Pass Heuristic Methods Results						
Machine Type Limit	**Cycle Time**	**Pure Random**	**MA**	**MI**	**RPW**	**Randomized RPW1**	**ImmSucc**	**AllSucc**
Single/One machine	1,89	30	30	30	28	27	31	29
Single/One machine	2	29	28	29	25	27	28	28
Single/One machine	2,2	28	27	27	25	27	26	25
Single/One machine	2,5	25	23	24	23	21	24	23
Single/One machine	2,95	23	23	21	21	20	22	22
Two machines	1,89	24	23	25	23	23	24	24
Two machines	2	21	21	24	22	21	24	23
Two machines	2,2	20	20	22	19	20	21	21
Two machines	2,5	18	16	19	17	17	18	17
Two machines	2,95	15	15	16	14	16	15	15

Table 4.9 Results of proposed single pass heuristic methods for Pitakaso-2

In order to improve the solution quality and search space, random algorithms run 1000 times and randomized RPW are verified by ascending selected random number of jobs and altering selection probabilities; the results of 1000 times run random algorithm are illustrated below Table 4.10.

PITAKASO-2						
		Proposed Single Pass Heuristic Methods Results				
Machine Type Limit	**Cycle Time**	**Pure Random (1000)**	**Randomized RPW1 (1000)**	**Randomized RPW2 (1000)**	**Randomized RPW3 (1000)**	**Randomized RPW4 (1000)**
Single/One machine	1,89	25	25	25	25	25
Single/One machine	2	25	25	24	25	24
Single/One machine	2,2	23	24	24	23	23
Single/One machine	2,5	19	20	20	19	20
Single/One machine	2,95	19	19	19	18	18
Two machines	1,89	22	22	22	22	22
Two machines	2	21	21	21	21	21
Two machines	2,2	19	19	19	19	19
Two machines	2,5	16	16	16	16	16
Two machines	2,95	14	14	14	14	14

Table 4.10 Results of proposed single pass heuristic randomized methods for Pitakaso-2

The algorithm runs 10000 times for finding a solution of each method above and the outperforming results are marked with green. For Pitakaso-2 problem with single machine limit, all methods give the best output as 25 workstations for 1.89 cycle time. *RPW2* and *RPW3* submit the best results as 24 workstations for 2 seconds of cycle time while *pure random* and *RPW4* find the outperforming result as 19 workstations for 2.5 seconds. Pure random, *RPW3* and *RPW4* reach the best solution for 2.2 cycle time; furthermore, *RPW3* and *RPW4* also gives the best solution as 18 workstations for 2.95 cycle time too. For Pitakaso-2 problem with two machines limit, all proposed method find the same outperforming solutions. Performance of *pure random* and *RPW1* methods are improved for each cycle time of Pitakaso-2 problem after running algorithm 1000 times.

PITAKASO-2							
		Single Pass Heuristic Methods Compared Results					
Machine Type Limit	Cycle Time	MA	MA-Pitakaso	MI	MI-Pitakaso	RPW	RPW-Pitakaso
Single/One machine	1,89	30	34	30	28	28	33
Single/One machine	2	28	34	29	26	25	33
Single/One machine	2,2	27	32	27	24	25	31
Single/One machine	2,5	23	28	24	22	23	30
Single/One machine	2,95	23	29	21	20	21	28
Two machines	1,89	23	25	25	23	23	26
Two machines	2	21	25	24	21	22	25
Two machines	2,2	20	22	22	17	19	21
Two machines	2,5	16	20	19	18	17	18
Two machines	2,95	15	15	16	14	14	16

Table 4.11 Proposed single pass heuristic methods result comparison for Pitakaso-2

The best results are marked with green, the same results produced by both model are marked with colour orange at Table 4.11. For Pitakaso-2 problem with single machine limit, *MI of Pitakaso* outperforms as 24, 22 and 20 number workstations for cycle 2.2, 2.5 and 2.95 seconds while our *RPW* method produces the best results as 25 workstations at cycle time 2 and at cycle time 1.89, as 28 workstations which is same output with results of *MI-Pitakaso*. For Pitakaso-2 problem with two machines limit, our *MA* and *RPW* and MI-Pitakaso submit the best solution as 23 workstations for 1,89 cycle time while *MA* and *MI-Pitakaso* give the best results as 21 works for cycle time of 2 seconds, *RPW* and *MI-Pitakaso* produced the outperforming solution as 14 workstations for cycle time 2,95. *MI method of Pitakaso* finds the best solution for cycle time 2.2; on the other hand, our *MA* gives the best solution for cycle time 2.5 for Pitakaso-2 problem with two machines limit. To sum up, our proposed single pass heuristic model's results are better of single pass heuristic model of Pitakaso&Sethanan (2015) at almost every instance of Pitakaso-2 problem, after improvement of our random methods.

GA model is also proposed in order to find optimal solutions for 20 instances presented by Pitakaso&Sethanan (2015) by minimizing number of workstations considering the machine type limit for each workstation. As it is mentioned, initial population of proposed GA is created randomly and implementing single pass heuristic methods which are *RPW, MA* and *MI* priority based heuristic task assignment rules; the task assignment procedure is applied with two approach as first fit approach assigning jobs to first available workstations or *best fit approach* by checking remaining time of all open workstations in order to allocate jobs to workstations. Population size is decided as 1000 number of individuals and generation size is determined as 1000; the fitness function is chosen based on minimi-

zation of the number of workstations considering task allocation balance between workstation as proposed GA. Migration ratio is implemented as 0.2, crossover probability is accepted as 0.85 and mutation probability is accepted as 0.05. Selective random method is used at selection process; if the selected individual perform crossover process, mate is chosen via roulette wheel method and randomly chosen crossover is applied; crossover method bias to fitter uniform's mixing ratio is determined as 0.6 and fixed mixing ration for uniform crossover is 0.5. Mutation probability is decided as 0.05, mutation rule to implement is randomly selected; bit by bit randomly mutation method has fixed threshold as 0.15. The stopping condition of proposed algorithm is reaching 1000 number of generations. The summary for the steps of proposed GA is below:

1. Generate initial population (of size *N=1000* number of individual) randomly and using single pass heuristic rules, evaluate the fitness of each individual
2. *Repeat* until stopping condition (L = 1000 generations) is reached

   2.1. *Migration:* (Select the most fit individuals based on migrate ratio 0.2 from the most recent generation)

   2.2. *Repeat* until population limit is reached.

      2.2.1. *Select* individual via selective random process

      2.2.2. *Crossover*: Generate a random number. If the number is less than the crossover probability *(0.85),* apply crossover

      Select mate by *roulette wheel*; select crossover rule randomly and apply crossover

      2.2.3. *Mutation*: Generate a random number. If the number is less than the mutation probability (0.05), apply mutation by selection a mutation rule randomly

      2.2.4. *Insertion*: Calculate fitness and insert the individual into the current population
3. Report the individual with the highest fitness value in the final population.

The steps of proposed GA are illustrated at Figure 4.14 below:

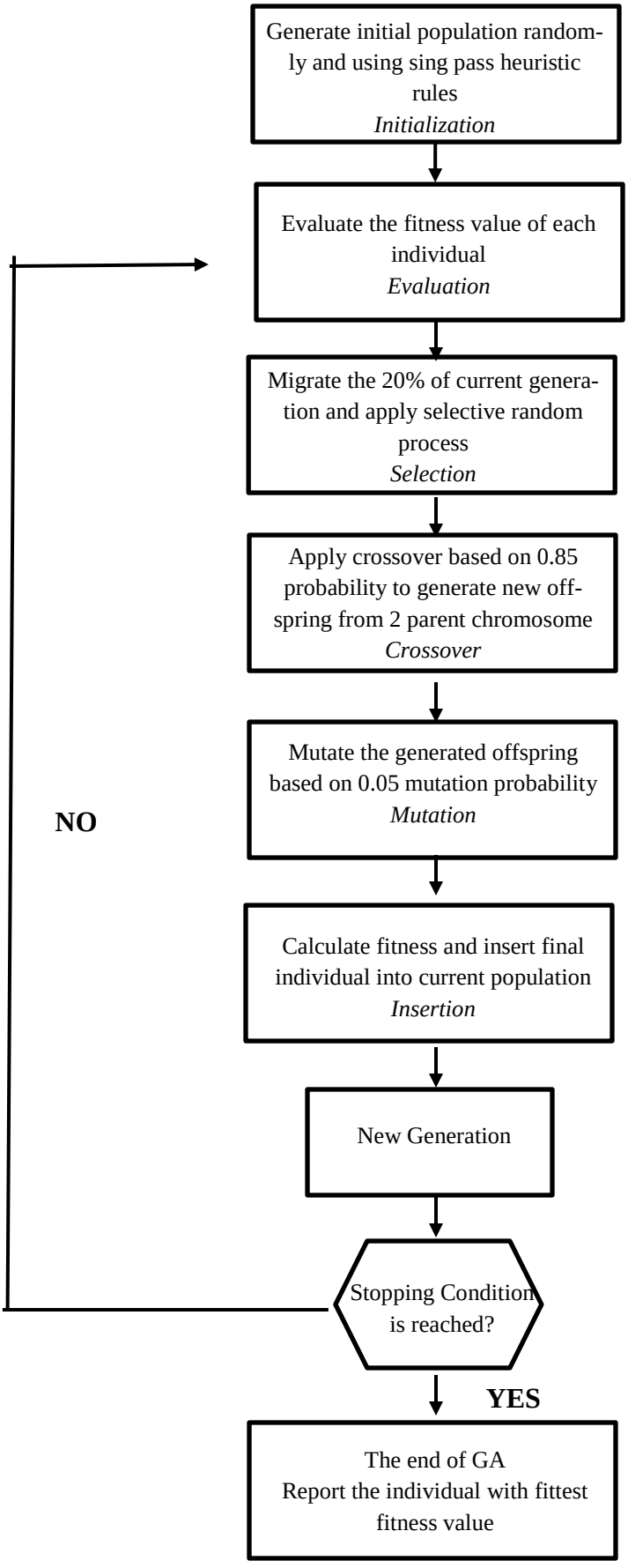


Figure 4.16 Steps of Proposed GA

For SALBP-1M, 20 instances of Pitakaso's problems are solved by proposed GA based on both task assignment procedures which produces same results. The best results of proposed GA are explained for Pitakaso-1 and Pitakaso-2 problems with 20 instances at Table 4.12.

PITAKASO-1		
Machine Type Limit	**Cycle Time**	**GA**
Single/One machine	1,23	22
Single/One machine	1,8	18
Single/One machine	2	17
Single/One machine	2,5	17
Single/One machine	3	17
Two machines	1,23	17
Two machines	1,8	12
Two machines	2	11
Two machines	2,5	9
Two machines	3	8

PITAKASO-2		
Machine Type Limit	**Cycle Time**	**GA**
Single/One machine	1,89	25
Single/One machine	2	23
Single/One machine	2,2	22
Single/One machine	2,5	19
Single/One machine	2,95	16
Two machines	1,89	22
Two machines	2	20
Two machines	2,2	19
Two machines	2,5	16
Two machines	2,95	13

Table 4.12 The results of Proposed GA

Proposed GA is tested for different size of population and generation in order to improve the solution quality; GA with 1000 generations with 1000 individuals gives better solution according to main aim of minimization number of workstations. Regarding to computational times of proposed GA; GA with 500 individuals and 200 generations solves one instance around 110 seconds, GA with 500 individuals and 500 generations submits a solution for one instance around 400 seconds, GA with 1000 individuals and 500 generations gives an output for one instance in 750 seconds. The proposed GA algorithm produces the best solution with 1000 individual and 1000 generation around 1150 and 1200 seconds according to instance to solve.

The outperforming results of our proposed model as single pass heuristic methods and GA are compared with GAMS optimal results and the presented model of Pitakaso and Sethanan (2015) for Pitakaso-1 and Pitakaso-2 problems.

PITAKASO-1							
Machine Type Limit	**Cycle Time**	**Proposed Best Single Pass Heu.**	**Proposed Best Single Pass Heu. Pitakaso**	**Proposed GA**	**DE_ Pitakaso**	**DE_C Pitakaso**	**GAMS Optimal**
Single/One machine	1,23	22	22	22	21*	20*	**22**
Single/One machine	1,8	18	18	18	18	18	**18**
Single/One machine	2	17	18	17	17	17	**17**
Single/One machine	2,5	17	18	17	16*	16*	**17**
Single/One machine	3	17	18	17	16*	16*	**17**
Two Machines	1,23	18	18	17	17	17	**17**
Two Machines	1,8	12	13	12	12	12	**12**
Two Machines	2	11	12	11	11	11	**11**
Two Machines	2,5	9	10	9	9	9	**9**
Two Machines	3	8	9	8	8	8	**8**

Table 4.13 The results of Pitakaso-1 problem

** Numbers marked with a star indicate that there is a discrepancy between the number reported by Pitakaso and Sethanan (2015) and the optimal result.*

For Pitakaso-1 problem with single machine limit, *pure random, RPW1 and RPW2* with 1000 runs from proposed single pass heuristics and proposed GA produce the optimal results same as the results of GAMS model; on the other hand, single pass heuristic methods of Pitakaso&Sethanan (2015) only find optimal solution for cycle time 1.23 and 1.8 seconds and DE and DE-C of Pitakaso&Sethanan (2015) reach optimal results for only cycle time 1.8 and 2.0 seconds. Proposed GA, DE and DE- C submits optimal results for every cycle time at Pitakaso-1 problem with two machines limit. Furthermore, *pure random, RPW1, RPW2, RPW3, RPW4* with 1000 times running from proposed single pass heuristic methods submit optimal results except at cycle time 1.23seconds.

PITAKASO- 2							
Machine Type Limit	Cycle Time	Proposed Best Single Pass Heu.	Proposed Best Single Pass Heu. Pitakaso	Proposed GA	DE_Pitakaso	DE_C Pitakaso	GAMS Optimal
Single/One machine	1,89	25	28	25	23*	23*	**25**
Single/One machine	2	24	26	23	22*	22*	**23**
Single/One machine	2,2	23	24	22	21*	21*	**22**
Single/One machine	2,5	19	22	19	18	18	**18**
Single/One machine	2,95	18	20	16	16	16	**16**
Two Machines	1,89	22	23	22	22	22	**22**
Two Machines	2	21	21	20	20	20	**20**
Two Machines	2,2	19	17*	19	19	19	**19**
Two Machines	2,5	16	18	16	16	16	**16**
Two Machines	2,95	14	14	13	13	13	**13**

Table 4.14. The results of Pitakaso-2 problem

** Numbers marked with a star indicate that there is a discrepancy between the number reported by Pitakaso and Sethanan (2015) and the optimal result.*

For Pitakaso-2 problem with single machine limit, proposed GA outperforms at used each cycle time and produces same solutions same as optimal solution of GAMS model besides for the problem with single machine and 2,5 cycle time and for this problem,Pitakaso and Sethanan (2015) find the optimum results with 18 workstation; unfortunately, proposed GA reaches 19 workstaion which is 1 more work-station more then optimum results. Some of the proposed single pass heuristic methods, which are *pure random, RPW1, RPW2, RPW3, RPW4* with 1000 times runs, give the optimal result as 25 workstations for cycle time 1.89 seconds. Pitakaso&Sethanan (2015)'s DE and DE-C submits optimal results as 18 and 16 workstations for cycle time 2.5 and 2.95 at Pitakaso-2 with single machine limit. Proposed GA, DE and DE-C find optimal solutions for Pitakaso-2 problem with two machines limit; on the other hand, *pure random, RPW1, RPW2, RPW3, RPW4* with 1000 times runs from applied single pass heuris-tic methods give optimal results as 22 workstations, 19 workstations and 16 workstations for cycle time 1.89, 2.2 and 2.5 seconds.

5. Conclusion

ALB is an extensively studied area. SALBP-1 is one of the best-known ALBPs, which aims to minimize the number of workstation at the line. We considered SALBP-1M in which there is a machine limit type for each workstation while minimizing the amount of workstations and balancing workload between workstations.

In this study, different single pass heuristic optimization methods and a GA were proposed to optimize SALBP-1M. Random, MA, MI, RPW and randomized RPW, AllSuc and ImmSucc were used as heuristic priority rules in order to assign jobs to workstations. This is done based on created task assignment procedure which can be based on first fit rule and best fit rule approach. The proposed GA was implemented with various genetic operators to extend the search space considering improvements at diversification and to avoid to be trapped in local minima.

The proposed optimization methods were applied to solve two problems which are introduced by Pitakaso and Sethanan (2015). These problems were previously optimized by single pass heuristic methods and DE-C. Pitakaso-1 includes 36 jobs and 6 different kinds of machines and Pitakaso-2 contains 52 jobs and 5 machine types; these two problems are solved with two distinct machine type limits. Each problem is examined with five different cycle times for every machine type limit.

Optimal results of both problems were found by a GAMS model in order to check if the proposed algorithms are able to reach the optimal solutions. The results of the proposed single pass heuristic methods and GA were compared with results of heuristic methods, DE and DE-C of Pitakaso and Sethanan (2015). For the single pass heuristics methods, the proposed heuristic rules produced efficient output and found the optimal results for one problem type with. In addition, the proposed GA gave optimal results for both problems except Pitakaso-2 problem with one machine limit per workstation and 2,5 cycle time while the modified evolutionary algorithm was able to reach optimal results only for problems with two machine type limits.

Different priority based heuristic rules could be implemented to generate initial populations. Future research may be developing metaheuristic methods for SALBP-1M to optimize Pitakaso-1 and Pitakaso-2 problems in order to compare the results with this stud

References

A.A. Mamun, A.A. Khaled, S.M. Ali, M.M. Chowdhury (2012). "A heuristic approach for balancing mixed-model assembly line of type I using genetic algorithm". *International Journal of Production Research*, 50, pp. 5106-5116.

Abdolmajid Yolmeh, Farhad Kianfar (2012). "An efficient hybrid genetic algorithm to solve assembly line balancing problem with sequence-dependent setup times." *Journal Computers and Industrial Engineerin*g, Vol. 62, Issue 4, pp. 936-945.

Abe, K., Yamada, T. and Matsui, M., (2004). "A Design Problem of Assembly Line Systems using Genetic Algorithm under the BTO Environment". *IEEJ Transactions on Electronics, Information and Systems*, Vol. 124, Issue 10, pp. 2006-2013.

Acar, N., Eştaş, S, (1991). "Kesikli Seri Üretim Sistemlerinde Planlama ve Kontrol Çalışmaları". *Milli Prodüktivite Merkezi Yayınları* No: 309, Ankara

Agpak K., Gokcen H., (2007). "A chance-constrained approach to stochastic line balancing problem *European Journal of Operational Research* 180, pp.1098 1115.

Agrawal, P.K., (1985)." The related activity concept in assembly line balancing". *International Journal of Production Research,* Vol.23, No.2, pp. 403-421.

Ahmadi, R.H., Dasu, S., Tang, C.S. (1992). "The dynamic line allocation problem". *Management Science* 38, pp. 1341-1353.

Ajenblit, D. A., Wainwright, R. L. (1998). "Applying genetic algorithms to the U-shaped assembly line balancing problem". *In The proceeding of the IEE international conference on evolutionary computation,* Anchorage, Alaska, USA, pp. 96–101.

Akagi, F., Osaki, H., and Kikichi, S., (1983). "A method for assembly line balancing with more than one worker in each station". *International Journal of Production Research,* 21, pp.755-770.

Akgündüz, O. S., Tunalı, S., (2010). "An adaptive genetic algorithm approach for the mixed-model assembly line sequencing problem*". International Journal of Production Research,* 48(17), pp. 5157-5179.

Akpinar, S., Bayhan, G. M., Baykasoglu, A., (2013). "Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks". *Applied Soft Computing,* 13(1), pp. 574-589.

Alena Otto, Christian Otto (2014a). "How to design effective priority rules: Example of simple assembly line balancing". *Computers and Industrial Engineering*, 69, pp.43-52.

Alexander Jensen Hjálmarsson, Viktor Ari Viktorsson. "Assembly Line Balancing". Faculty of Industrial-Mechanical Engineering and Computer Science University Iesland.

Amen, M. (2000). "Heuristic methods for cost-oriented assembly line balancing: a survey". *International Journal of Production Economics* 68, pp. 1-14.

Amen, M., (2001). "Heuristic methods for cost-oriented assembly line balancing: A comparison on solution quality and computing time". *International Journal of Production Economics* 69, pp. 255–264.

Anderson, E. J., & Ferris, M. C. (1994). "Genetic algorithms for combinatorial optimization: The assembly line balancing problem". *ORSA Journal on Computing*, 6, pp. 161–173.

Arcus, A. (1963). "An Analysis of a Computer Method of Sequencing Line Operations," Ph.D. Thesis, *University of California*, Berkeley.

Arcus, A. L., (1966). "COMSOAL: A computer method of sequencing operations for assembly lines". *International Journal of Production Research*, 4, pp. 259-277.

Armin Scholl, (1999). "*Balancing and Sequencing of Assembly Lines*". 1st Edition, Heildelberg, Physica-Verlag.

Armin Scholl, Christian Becker, (2006b). "State-of-the-art exact and heuristic solution procedures for simple assembly line balancing". *European Journal of Operational Research*, Vol.168, Issue 3, pp. 666-693.

Askin, R.G., Zhou, M., (1997). "A parallel station heuristic for the mixed-model production line balancing problem". International Journal of Production Research 35, pp. 3095–3105.

Aydoğdu, A., (2005). "Takım Esaslı Montaj Hattı Dengeleme" *Bitirme Tezi İ.T.Ü.Endüstri Mühendisliği Bölümü*, İstanbul

Balas, E., (1965). "An additive algorithm for solving linear program with zero- one variables". *Operations Research,* Vol 13, issue 4, pp. 517-546.

Bard, J. F. (1989). "Assembly line balancing with parallel workstations and dead time". *International Journal of Production Research*, 27(6), pp. 1005-1018.

Bartholdi J.J., (1993). "Balancing two-sided assembly lines: A case study". *International Journal of Production Research*, Vol.31, No.10, pp. 2447–2461

Baskak, M., (2005). "Üretim Hatlarının Modellenmesi" *Ders Notları, İ.T.Ü. Endüstri Mühendisliği Bölümü,* İstanbul.

Bautista, J., Suarez, R., Mateo, M., Companys, R., (2000). "Local search heuristics for the assembly line balancing problem with incompatibilities between tasks". *In: Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp. 2404– 2409.

Baybars, I., (1986a). "An efficient heuristic method for the simple assembly line balancing problem". *International Journal of Production Research* 24(1), pp.149-166.

Baybars, I., (1986b). "A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem". *Management Science* Vol.32, No. 8, pp. 909-932.

Baykasoğlu, A., Özbakır, L., (2007). "Stochastic U-line balancing using genetic algorithms". *The International Journal of Advanced Manufacturing Technology*, 32(1-2), pp. 139-147.

Baykasoğlu, A., Özbakır, L., (2015). "Discovering task assignment rules for assembly line balancing via genetic programming". *The International Journal of Advanced Manufacturing Technology*, Vol. 76, Issue 1, pp 417–434.

Beasley, J.E., Chu, P.C., (1996). "A Genetic Algorithm for the Set Covering Problem". European Journal of Operational Research, 94, pp. 392-404.

Berger, I., Bourjolly, J.-M., Laporte, G., (1992). "Branch-and bound algorithms for the multi-product assembly line balancing problem". *European Journal of Operational Research*, Vol. 58, pp. 215–222.

Betts, J., Mahmoud, K.I., (1989). "A method for assembly line balancing". *Engineering Costs and Production* Economics, Vol.18, pp. 55–64.

Blum, C., Roli, A., (2003). "Metaheuristics in combinatorial optimization: Overview and conceptual comparison". *ACM Computing Surveys (CSUR)*, 35(3), pp. 268-308.

Bock, S., (2000). "Modelle und verteilte Algorithmen zur Planunggetakteter Fließlinien: Ansatze zur Unterstützung eines effizienten Mass Customization, *Gabler edition Wissenschaft.*

Bock, S., Rosenberg, O., (1998). "A new distributed fault-tolerantalgorithm for the simple assembly line balancing problem". *In: Kischka, P. et al. (Eds.), Operations Research Proceedings. Springer, Berlin,* pp. 474–480.

Boctor, F.F., (1995). "A multiple-rule heuristic for assembly line balancing". Journal of the Operational Research Society 46, pp. 62–69.

Borba, L., Ritt, M. (2014). "A heuristic and a branch-and-bound algorithm for the Assembly Line Worker Assignment and Balancing Problem". *Computers & Operations Research* 45, pp.87–96.

Boucher T O., (1978).Choice of assembly line design under task learning. *International Journal of Production Research*, Vol.25, pp. 513–524.

Bowman, E. H., (1960). "Assembly line balancing by linear programming". *Operations Research*, Vol. 8, pp.385-389.

Brown, E. C., Sumichrast, R. T., (2005). "Evaluating performance advantages of grouping genetic algorithms". *Engineering Applications of Artificial Intelligence*, 18, pp. 1–12.

Brudaru, O., Valmar, B., (2004). "Genetic algorithm with embryonic chromosomes for assembly line balancing with fuzzy processing times". *The 8th international research/expert conference trends in the development of machinery and associated technology*, TMT, Neum, Bosnia and Herzegovina.

Bukchin, J., Rubinovitz, J., (2002). "A weighted approach for assembly line design with station paralleling and equipment selection". *IIE Transactions* 35, pp.573–585.

Buxey, G. M., (1974). "Assembly line balancing with multiple stations". *Management Science,* 20, pp. 1010-1021.

Buxey, G., (1978). "Incompletion costs versus labour efficiency on the fixed-item moving belt flow-line". *International Journal of Production Research*, 16(3), pp. 233–247.

Buzacott J.A., (1990). "Abandoning the moving assembly line: Models of human operators and job sequencing". *International Journal of Production Research*, Vol.28, No.5, pp.821–839.

Campbell, H. G., Dudek, R. A. and Smith, M. L., (1970). "A Heuristic Algorithm for the *n* Job *m* Machine Sequencing Problem". *Management Science* 16, pp. B630-637.

Carlos Andres, Cristobal Miralles, Rafael Pastor (2008). "Balancing and scheduling tasks in assembly lines with sequence-dependent setup times". *European Journal of Operational Research* Vol.187, pp.1212–1223.

Carnahan, B. J., Norman, B. A., Redfern, M. S., (2001). "Incorporating physical demand criteria into assembly line balancing". *IIE Transactions*,33, pp. 875–887.

Cerny, V. (1985). "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm". *Journal of Optimization Theory and Applications*, 45(1), pp. 41-51.

Chakravarty A K., (1988). Line balancing with task learning effects. *IIE Transactions*, Vol.20, pp.186–193.

Chakravarty, A.K., and Shtub, A., (1986). "A cost minimization procedure for mixed model production lines with normally distributed task times". *European Journal of Operational Research*, 23, pp. 25-36.

Chan, C. C. K., Hui, P. C. L., Yeung, K.W., Ng, F. S. F., (1998). "Handling the assembly line balancing problem in the clothing industry using a genetic algorithm". *International Journal of Clothing Science and Technology*, 10(1), pp. 21–37.

Chapter 2, Literature Review-Genetic Algorithms, pp. 27-33, downloaded on 23 June 2015 *from* < www.prr.hec.gov.pk/Chapters/487S-2.pdf >

Chen, J. C., Chen, C. C., Su, L. H., Wu, H. B., Sun, C. J., (2012). "Assembly line balancing in garment industry. *Expert Systems with Applications*, 39(11), 10073-10081.

Chen, R. S., Lu, K. Y., Yu, S. C., (2002). "A hybrid genetic algorithm approach on multi-objective of assembly planning problem". *Engineering Applications of Artificial Intelligence*, 15, pp. 447–457.

Chiang, W.-C., Urban, T.L., (2002). "A hybrid heuristic for the stochastic U-line balancing problem". *Working Paper, University of Tulsa*, Oklahoma, USA.

Chica, M., Cordón, Ó., Damas, S., (2011). "An advanced multi-objective genetic algorithm design for the time and space assembly line balancing problem".*Computers & Industrial Engineering*, 61(1), pp.103-117

Christian Becker, Armin Scholl (2009) "Balancing assembly lines with variable parallel workplaces: Problem definition and effective solution procedure". *European Journal of Operational Research Vol*.199, pp.359–374.

Christian Becker, Armin Scholl, (2006a). "A survey on problems and methods in generalized assembly line balancing". *European Journal of Operational Research* Vol.168, pp.694–715.

Christian Otto, Alena Otto (2014b). "Extending assembly line balancing problem by incorporating learning effects". *International Journal of Production Research*, 52 pp. 7193-7208.

Coffman, E. G., Jr., Garey, M. R., and Johnson, D. S., (1984) ``Approximation Algorithms for Bin-Packing," in *Algorithm Design for Computer Systems Design, Ausiello, G., Lucertini, M., and Serafini, P. (eds.), Springer-Verlag*, pp.49-106.

Computer Age Engineering Inc. website. Downloaded on May6, 2015 *from* <http://caeweb.thomasnet.com/item/products/automated-assembly-equipment/item-1032?>

Cristobal Miralles, Jose P. Garcia-Sabater, Carlos Andres, Manuel Cardos (2008). "Branch and bound procedures for solving the Assembly Line Worker Assignment and Balancing Problem: Application to Sheltered Work centres for Disabled". *Discrete Applied Mathematics* Vol.156, pp.352–367

D. E. Goldberg (1989). "Genetic Algorithms in Search, Optimization & Machine Learning". Addison-Wesley, Reading MA.

D.E. Goldberg and R. Lingle (1985). "Alleles, loci and the traveling salesman problem". In J.J. Grefenstette (ed.), Proceedings of an International Conference on Genetic Algorithms and Their Applications. Lawrence Erlbaum Associates, Hillsdale, New Jersey, pp. 154–159.

D.J. Fonseca, C.L. Guest, M. Elam, and C.L. Karr (2005) "A Fuzzy Logic Approach to Assembly Line Balancing". *Mathware & Soft Computing*, Vol.12, pp. 57-74.

D.P. Rini, S.M. Shamsuddin, S.S. Yuhaniz (2011). "Particle swarm optimization: technique, system and challenges". International Journal of Computer Applications 14 (1), 19-27.

Dar–El, E.M. (1975). "Solving large single model assembly line balancing problems-A comparative study". *AIIE Transactions*, 7(3), pp. 302-310.

Dar-El, E.M. and Rubinovitch, Y., (1979). "MUST- A multiple solutions technique for balancing single model assembly lines". *Management Science*, 25, pp. 1105-1111.

Dar-El, E.M., (1973). “MALBĐ- A heuristic technique for balancing large single-model assembly lines”. *AIIE Transactions*, 5, pp. 343- 356

Davis, L. (1985). “Applying algorithms to epistatic domains”. Proceedings International Joint Conference on Artificial Intelligence, pp.162-164.

Delice, Y., Kızılkaya Aydoğan, E., Özcan, U., (2016). “Stochastic two-sided U-type assembly line balancing: a genetic algorithm approach”. *International Journal of Production Research*, 54(11), pp. 3429-3451.

Dervis Karaboga, Selcuk Okdem (2004). “A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm”. Turkish Journal of Electrical Engineering and Computer Science, 12, pp. 53-60,

Dolgui, A., Finel, B., Guschinsky, N., Levin, G., Vernadat, F., (2004). “A heuristic approach for transfer lines balancing”. *Working Paper, University of Technology of Troyes*, France.

Dorigo, M. (1992). “Optimization, learning and natural algorithms”. *Ph.D. thesis, DEI, Politecnico di Milano,* Italy. pp. 140.

Erel, E., S.C. Sarin, (1998). “A Survey of the Assembly Line Balancing Procedures”. *Production Planning and Control* Vol.39, No.13, pp.3003.3015.

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

Erel, E., Sabuncuoglu, I., Sederci, H., (2005). “Stochastic assembly line balancing using beam search”. *International Journal of Production Research* 43, pp. 1411–1426.

Falkenauer, E. (1991). “A genetic algorithm for grouping”. In The proceedings of the fifth international symposium on applied stochastic models and data analysis. Granada, Spain.

Falkenauer, E. (1997). “A grouping genetic algorithm for line balancing with resource dependent task times”. *In The proceedings of the fourth international conference on neural information processing*, New Zealand, pp. 464–468.

Falkenauer, E., & Delchambre, A. (1992). “A genetic algorithm for bin packing and line balancing”. *In The proceedings of the1992 IEEE international conference on robotics and automation*). Nice, France, pp. 1189– 1192.

Fazlollahtabar, H., Hajmohammadi, H., and Eshaghzadeh, A., (2011). “A heuristic methodology for assembly line balancing considering stochastic time and validity testing”. *International Journal of Advanced Manufacturing Technology*, 52(1-4), pp. 311-320.

Ford Motor Company website, Downloaded on April 17, 2015 *from* <http://fordmotorhistory.com/history/assembly_line.php >

Fred M. Tonge, (1960). “Summary of a Heuristic Line Balancing Procedure”. *Mathematics Divisio*n, the RAND Corporation.

Fred M. Tonge, (1965). “Assembly Line Balancing Using Probabilistic Combinations of Heuristics”. Management Science, Vol. 11, No. 7, Series A, pp. 727-735.

Fu-peng Yin, Jia-kun Sun, Ai-hua Wu, (2011). “Two-sided with multi-parallel stations assembly line balancing based on heuristic algorithm”. *Industrial Engineering and Engineering Management (IEEM)*, IEEE 18Th International Conference Changchun, pp. 996 – 998.

Gadidov, R., Wilhelm, W., (2000). “A cutting plane approach for the single-product assembly system design problem”. *International Journal of Production Research* 38, pp. 1731– 1754

Gamberini R, Grassi A, Rimini B., (2006). “A new multi-objective heuristic algorithm for solving the stochastic line rebalancing problem”. *International Journal of Production Economics* 102, pp. 226–243

Gao J., Sun L., Wang L., Gen M., (2009). “An efficient approach for type II robotic assembly line balancing problems”. *Computers & Industrial Engineering* 56 (3), pp. 1065–1080.

Gen Mitsuo, Tsujimura Yashuhiro, Li Yinxiu (1996). “Fuzzy Assembly Line Balancing Using Genetic Algorithms”. *Computers & Industrial Engineering*, Vol. 31, Issue 3-4, pp. 631-634.

Geoffrion, A.M., (1967). "Integer Programming by Implicit Enumeration and Balas' Method" SIAM Review, Vol. 9:2, pp.178-190.

Global market website, (2011). "The professional energy-saving light of SHENDU”. Downloaded on May 6, 2015 *from* <http://www.globalmarket.com/sourcingtips/lighting/-the-professional-energy-saving-light-of-shendu-5.html >

Glover, F. (1977). “Heuristics for integer programming using surrogate constraints”. *Decision Sciences* 8, pp. 156-166.

Glover, F. (1986). “Future paths for integer programming and links to artificial intelligence”. *Computers and Operations* Research 13, pp. 533–549.

Gökçen, H., Agpak, K., Benzer, R. (2006). “Balancing of parallel assembly lines”. *International Journal of Production Economics* 103, pp. 600-609.

Goncalves, J. F., De Almedia, J. R., (2002). “A hybrid genetic algorithm for assembly line balancing”. *Journal of Heuristic*, 8, pp. 629–642.

Grzechca, W., (2008). “Estimation of time and cost oriented assembly line balancing problem”. *19th International Conference of System Engineering, IEEE*

Gunther, R. E., Johnson, G. D., and Peterson, R. S., (1983). "Currently practiced formulations for the assembly line balance problem". *Journal of Operations Management*, 1, pp. 209-221.

Guo, Z. X., Wong, W. K., Leung, S. Y. S., Fan, J. T., Chan, S. F., (2008). "A genetic-algorithm-based optimization model for scheduling flexible assembly lines". *The International Journal of Advanced Manufacturing Technology*, 36(1-2), pp. 156-168.

Gurevsky Evgeny, Battaïa Olga, Dolgui, Alexandre (2013). "Stability measure for a generalized assembly line balancing problem". *Discrete Applied Mathematics*, Vol.161(3), pp.377-394

Gustavson, R. E. (1986). "Design of cost effective assembly systems". *C.S. Draper Laboratory Report*, N°P-2661, Cambridge.

Hackman, S.T., Magazine, M.J.,, Wee, T.S., (1989). "Fast, effective algorithms for simple assembly line balancing problems". *Operations Research* Vol.37, pp. 916–924.

Hamzadayi, A., Yildiz, G., (2012). "A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model U-lines with parallel workstations and zoning constraints". *Computers & Industrial Engineering*, 62(1), pp. 206-215.

Held,M., Karp,R.M, and Shareshian,R., (1963). "Assemblyline balancingĐ Dynamic programming with precedence constraints". *Operations Research*, 11, pp. 442- 459.

Helgeson,W. B, and Birnie, D. P., (1961). "Assembly line balancing using the ranked positional weight technique". *Journal of Industrial Engineering*, 12, pp. 394- 398.

Hoffmann, T. R. (1963). "Assembly line balancing with a precedence matrix". *Management Science*, Vol. 9, No.4, pp. 551-562.

Hoffmann, T. R., (1992). "Eureka: a hybrid system for assembly line balancing". *Management Science*, Vol.38, No.1, pp. 39-47.

Holland, J. H. (1975). "Adaptation in natural and artificial systems". Ann Arbor, Michigan: The University of Michigan Press.

Hu Xiaofeng, Wu Erfei, Jin Ye (2008) "A station oriented enumerative algorithm for two-sided assembly line balancing*". European Journal of Operational Research Vol*.186, pp.435–440.

Hu, T. C., (1961). "Parallel sequencing and assembly line problems". *Operations Research*, Vol.9, No.6, pp. 841-848.

Hui, S. (2005). "Performance evaluation of hybrid genetic algorithm for assembly line scheduling". *In 17th IEEE International Conference on Tools with Artificial Intelligence*, pp. 224-231.

Hwang, R. K., Katayama, H., Gen, M., (2008). "U-shaped assembly line balancing problem with genetic algorithm". *International Journal of Production Research*, 46(16), pp. 4637-4649.

Hwang, R., Katayama, H., (2009). "A multi-decision genetic approach for workload balancing of mixed-model U-shaped assembly line systems". *International Journal of Production Research*, 47(14), pp. 3797-3822.

Ibrahim Kucukkoc, David Z. Zhang (2015). "A mathematical model and genetic algorithm-based approach for parallel two-sided assembly line balancing problem". *Production Planning & Control*, 26:11, pp. 874-894.

Jackson, J. R., (1956). "A computing procedure for a line balancing problem". *Management Science*, Vol.2, No.3, pp. 261-271.

Jaeschke, G., (1964). “Branching and Bounding: Eine allgemeine Methode zur Lösung kombinatorischer Probleme”. A*blaufundPlanungsforschung* 5, pp.133–155.

Jaturanonda, Chorkaew, Nanthavanij, Suebsak , Das, Sanchoy K (2013). “Heuristic Procedure forAssembly Line Balancing Problem with Postural Load Smoothness”. *International Journal of Occupational Safety and Ergonomics (JOSE),* Vol.19, No.4, pp. 531– 541.

Jiano J., Kumar A., Martin W., (2006) “A web-based interactive advisor for assembly line balancing”. *International Journal of Advanced Manufacturing Technology* 27, pp. 1192–1201

Joaquin Bautista, Jordi Pereira (2009) “A dynamic programming based heuristic for the assembly line balancing problem*”. European Journal of Operational Research* Vol.194, pp.787–794.

Joaquin Bautista, Jordi Pereira (2011). “Procedures for the Time and Space Constrained Assembly Line Balancing Problem”. European Journal of Operational Research Vol.212, pp. 473-481.

Johnson, R. V., (1973). "Branch and Bound Algorithms for Assembly Line Balancing and Job-Shop Scheduling", *Unpublished Ph.D. Thesis, University of California*, Los Angeles.

Johnson, R. V., (1981). "Assembly Line Balancing Algorithms: Computation Comparisons". *International Journal of Production Research,* Vol.19, pp. 277-287.

Johnson R.V., (1983). “A branch and bound algorithm for assembly line balancing problems with formulation irregularities”. *Management Science,* Vol.29, No.11, pp.1309–1324.

Johnson, R.V., (1988). “Optimally balancing large assembly lines with ‘‘FABLE”. *Management Science*, Vol.34, pp. 240–253.

K. Agpak, H. Gokcen, (2005). “Assembly line balancing: Two resource constrained cases”. *International Journal of Production Economics* 96 (1), pp. 129-140.

Kao, E. P. C, and Queyranne, M., (1982). “On dynamic programming methods for assembly line balancing”. *Operations Research,* 30, pp. 375-390.

Kao, E. P. C., (1976). “A preference order dynamic program for stochastic assembly line balancing”. *Management Science,* 22, pp. 1097-1104.

Karabati, S., Sayin, S., (2003). “Assembly line balancing in a mixed model sequencing environment with synchronous transfers”. *European Journal of Operational Research* 149, pp. 417–429.

Karp, R. M., (1972). “Reducibility among combinatorial problems,” in Complexity of Computer Computations: Proceedings of a Symposium on the Complexity of Computer Computations, *R. E. Miller and J. W. Thatcher, Eds., The IBM Research Symposia Series, New York, NY: Plenum Press*, pp. 85-103.

Kazemi, S. M., Ghodsi, R., Rabbani, M., Tavakkoli-Moghaddam, R., (2011). “A novel two-stage genetic algorithm for a mixed-model U-line balancing problem with duplicated tasks”. *The International Journal of Advanced Manufacturing Technology*, 55(9-12), pp. 1111-1122.

Kennedy, J., & Eberhart, R. C., (1995). “Particle swarm optimization”. *In Proceedings of IEEE International Conference on Neural Networks, New Jersey*: Piscataway, pp. 1942-1948.

Kilbridge, M.D., and Wester,L., (1961). “A heuristic method of assembly line balancing”. *Journal of Industrial Engineering*, 12, 292- 298.

Kim, H., and Park, S., (1995). “A strong cutting plane algorithm for the robotic assembly line balancing problem”. International Journal of Production Research, 33, pp. 2311-2323.

Kim, Y. J., Kim, Y. K., & Cho, Y., (1998). “A heuristic-based genetic algorithms for workload smoothing in assembly lines”. *Computers & Operations Research,* 25(2), pp. 99–111.

Kim, Y. K., Kim, Y. J., Kim, Y. H. (1996). “Genetic algorithms for assembly line balancing with various objectives”. *Computers&Industrial Engineering,* Vol.30, Issue 3, pp. 397–409.

Kim, Y. K., Kim, Y., Kim, Y. J., (2000). “Two-sided assembly line balancing: A genetic algorithm approach”. *Production Planning and Control,* 11(1), pp. 44-53.

Kim, Y. K., Song, W. S., Kim, J. H., (2009). “A mathematical model and a genetic algorithm for two-sided assembly line balancing”. *Computers & Operations Research,* 36(3), pp. 853-865.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., (1983). “Optimization by simulated annealing”. *Science,* 220, pp. 671-680.

Klein, M., (1963). “On assembly line balancing”. Operations Research, Vol. 11, Issue 2, pp. 274-281.

Klein, R., Scholl, A., (1996). “Maximizing the production rate in simple assembly line balancing—A branch and bound procedure”. *European Journal of Operational Research,* Vol. 91, pp. 367–385.

Kottas, J. F., and Lau, H. S., (1973). “A cost oriented approach to stochastic line balancing”. *AIIE Transactions,* 5, pp. 164-171.#Kottas, J. F., and Lau, H. S., (1976). “A total operating cost model for paced lines with stochastic task times”. *AIIE Transactions,* 8, pp. 234- 240.

Kottas, J.F., and Lau, H.S., (1981). “A stochastic line balancing procedure”. *International Journal of Production Research,* 19, pp. 177-193.

Krzysztof Fleszar, Khalil S. Hindi (2003) “An enumerative heuristic and reduction methods for the assembly line balancing problem”. *European Journal of Operational Research Vol.* 145, pp. 606–620.

Kulak, O., Yilmaz, I. O., Günther, H. O., (2008). “A GA-based solution approach for balancing printed circuit board assembly lines”. *OR Spectrum,* 30(3), pp. 469-491.

L. Capacho, R. Pastor, A. Dolgui, O. Guschinskaya, (2009). “An evaluation of constructive heuristic methods to solve the alternative subgraphs assembly line balancing problem”. *Journal of Heuristics,*15, pp. 109 132 .

L. Schrage, K.R. Baker, (1978). “Dynamic programming solution of sequencing problems with precedence constraints”. *Operations Research*, 26, pp. 444–449.

Labbe, M., Laporte, G., Mercure, H., (1991). “Capacitated vehicle routing on trees “, *Operations Research*, Vol. 39, pp. 616-632.

Lapierre, S.D., Ruiz, A.B., (2004). “Balancing assembly lines: An industrial case study”. *Journal of the Operational Research Society* 55, pp. 589–597

Lawler, E. L., (1979). "Efficient Implementation of Dynamic Programming Algorithms for Sequencing Problems". *Report BW 106/79*, Stichting Mathematisch Centrum, Amsterdam.

Lee T.O., Kim Y., Kim Y.K., (2001). “Two-sided assembly line balancing to maximize work relatedness and slackness”. *Computers & Industrial Engineering*, Vol.40,No.3, pp. 273–292

Lee, T.O., Kim, Y., Kim, Y.K., (2001). “Two-sided assembly line balancing to maximize work relatedness and slackness”. *Computers and Industrial Engineering* 40, pp. 273–292.

Leu, Y. Y., Matheson, L. A., Rees, L. P. (1994). “Assembly line balancing using genetic algorithms with heuristic generated initial populations and multiple criteria”. *Decision Sciences*, 15, pp. 581–606.

Levitin, G., Rubinovitz, J., Shnits, B., (2006). “A genetic algorithm for robotic assembly line balancing”. *European Journal of Operational Research*, 168, pp. 811–825

Liu SB, Ong HL, Huang H.C., (2005). “A bidirectional heuristic for stochastic assembly line balancing type-II problem”. *International Journal of Advanced Manufacturing Technology* 25, pp. 71–77.

Liu, C.-M., Chen, C.-H., (2002). “ Multi-section electronic assembly line balancing problems: A case study”. *Production Planning and Control* 13, pp. 451–461

Luigi Martino, Rafael Pastor (2010). “Heuristic procedures for solving the general assembly line balancing problem with setups”. *International Journal of Production Research*, 48:6, pp. 1787-1804.

M. Duran Toksari, Selcuk K. Isleyen, Ertan Guner, Omer Faruk Baykoc (2010). “Assembly line balancing problem with deterioration tasks and learning effect”. *Expert Systems with Applications* Vol.37, pp.1223–1228.

Macaskill, J. L., (1972). “Production line balances for mixed-model lines”. *Management Science*, 19,4.

Malakooti, B., (1994). “Assembly line balancing with buffers by multiple criteria optimization”. *International Journal of Production Research*, Vol. 32, pp.2159-2178.

Malakooti, B., Kumar, A., (1996). “A knowledge-based system for solving multi-objective assembly line balancing problems”. International Journal of Production Research 34, pp. 2533– 2552.

Manavizadeh, Neda; Rabbani, Masoud; Radmehr, Farzad (2015). “A new multi-objective approach in order to balancing and sequencing U-shaped mixed model assembly line problem: a proposed heuristic algorithm”. *The International Journal of Advanced Manufacturing Technology*, Vol.79 (1), pp.415-425.

Martinez, U., Duff, W. S. (2004). “Heuristic approaches to solve the U-shaped line balancing problem augmented by genetic algorithms”. *In The proceedings of systems and information engineering design symposium*, pp. 287–293.

Mastor, A. A., (1970). “An experimental investigation and comparative evaluation of production line balancing techniques”. *Management Science*, 16, pp. 728- 746

Matanachai, S., Yano, C.A., (2001). “Balancing mixed-model assembly lines to reduce work overload”. *IIE Transactions* 33, pp. 29–42.

Merengo, C., Nava, F., Pozetti, A., (1999). “Balancing and sequencing manual mixed-model assembly lines”. *International Journal of Production Research* 37, pp. 2835–2860.

Merkle, D., & Middendorf, M. (2005). “Swarm intelligence. In E. K. Burke, & G. Kendall, (Eds.), Search Methodologies - Introductory tutorials in optimization and decision support techniques”. Springer Science Business Media Inc., pp.401-435.

Mertens, P., (1967). “Fließbandabstimmung mit dem Verfahren der begrenzten Enumeration nach Müller-Merbach”. *Ablaufund Planungsforschung* 8,pp. 429–433.

Metha Nojhan website, Classificacation of metahueristics, Downloaded on June 2, 2015 *from* < http://metah.nojhan.net/?q=different+classification+of+metaheuristics >

Michalewicz, Z., (1996). “Genetic Algorithms + Data Structures = Evolution Programs”. (3rd Ed.). Springer-Verlag, London, UK.

Miltenburg, G.J., Wijngaard, J., (1994). “The U-line line balancing problem”. *Management Science*, Vol.40, No.10, pp.1378–1388.

Miltenburg, J. (2002). “Balancing and sequencing mixed-model U-shaped production lines”. *International Journal of Flexible Manufacturing Systems*, 14, pp. 119–151.

Minghai, Y., Huanmin, X., (2010). “Reconfigurable assembly line balancing with the hybrid genetic algorithm”. *IEEE, In The 2nd International Conference on Information Science and Engineering*, pp. 1398-1401.

Mingzhou Jina, S. David Wub (2002) “A new heuristic method for mixed model assembly line balancing problem”. *Computers & Industrial Engineering*, Vol. 44, pp. 159-169.

Minzu, V. and J-M. Henrioud (1997). “Assignment stochastic algorithm in multi-product assembly lines”. In: Proceedings oflSATP'97, pp. 109-114, IEEE Press.

Mitchell, M. 1999. “An Introduction to Genetic Algorithms”. 5th ed. A Bradford Book. The MIT Press. Cambridge, Massachusetts- London

Moodie,C.L., and Young,H.H., (1965). “A heuristic method of assembly line balancing for assumptions of constant or variable work element times”. *Journal of Industrial Engineering*, 16, pp. 23- 29.

Moon I., Logendran R., Lee J., (2009). “Integrated assembly line balancing with resource restrictions”. *International Journal of Production Research* 47(19), pp. 5525–5541.

Moreira Mayron, Ritt Marcus, Costa, Alysson, Chaves Antonio (2012). “Simple heuristics for the assembly line worker assignment and balancing problem”. *Journal of Heuristics*, Vol.18(3), pp.505-524.

Moreira, Mayron César O.; Miralles, Cristóbal; Costa, Alysson M. (2015). “Model and heuristics for the Assembly Line Worker Integration and Balancing Problem”. *Computers and Operations Research*, Vol.54, pp.64-73.

Motion Controls Robotics website. Downloaded on May 6, 2015 *from* <http://motioncontrolsrobotics.com/robotic-applications/robotic-assembly/ >

Mutlu, O., Polat, O., Supciller, A. A., (2013). "An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-II". *Computers & Operations Research*, 40(1), pp. 418-426.

Naveen Kumar, Dalgobind Mahto, (2013). "Assembly Line Balancing: A Review of Developments and Trends in Approach to Industrial Application". *Global Journals Inc*. Vol.13, Issue 2 pp. 29-50

Nevins, A. J., (1972). "Assembly line balancing using best budsearch". *Management Science*, 18, pp. 529- 539

Nils Boysen, Malte Fliedner (2008). "A versatile algorithm for assembly line balancing". *European Journal of Operational Research* Vol.184, pp.39–56

Nils Boysen, Malte Fliedner, Armin Scholl, (2007). "A classification of assembly line balancing problems". *European Journal of Operational Research,* Vol. 183(2), pp.674-693.

Nils Boysen, Malte Fliedner, Armin Scholl, (2008). "Assembly line balancing: Which model to use when?". *Int. J. Production Economics* Vol.111, pp.509–528

Nkasu, M.M., and Leung, K.H., (1995). "A stochastic approach to assembly line balancing". International Journal of Production Research, 33, pp. 975-991.

Noorul Haq, A., Rengarajan, K., Jayaprakash, J., (2006). "A hybrid genetic algorithm approach to mixed-model assembly line balancing". *The International Journal of Advanced Manufacturing Technology*, 28, pp. 337-341.

Nourie, F.J., Venta, E.R., (1991). "Finding optimal line balances with OptPack". *Operations Research Letters*, Vol. 10, pp. 165–171.

O. Kilincci, (2010). “A Petri net-based heuristic for simply assembly line balancing problem of type 2”. *International Journal of Advanced Manufacturing Technology*, 46, pp.329–338.

O.Univ.Prof. Dipl.-Ing. Dr. Richard F. Hartl, “Management Science Lecture Notes”. University of Vienna. Downloaded on 06.03.2014 *from* < http://prolog.univie.ac.at >

Olga Guschinskaya, Alexandre Dolgui (2009) “Comparison of exact and heuristic methods for a transfer line balancing problem”. *International Journal of Production Economics* Vol.120, pp.276–286.

Osman, I.H., Laporte, G., (1996). “Metaheuristics: A bibliography”. *Annals of Operations Research* 63, pp. 513–623.

Osuya Emeke Great, Dr. Aniekan Offiong (2013). “Productivity improvement in breweries through line balancing using heuristic method”. *International Journal of Engineering Science and Technology*, Vol.5 (3), pp.475-486

Ozcan Kilincci (2011). “Firing sequences backward algorithm for simple assembly line balancing problem of type 1”. Computers & Industrial Engineering, Vol.60, pp.830-839.

Özcan, U., Kellegöz, T., Toklu, B., (2011). “A genetic algorithm for the stochastic mixed-model U-line balancing and sequencing problem”. *International Journal of Production Research*, 49(6), pp. 1605-1626.

P. Th. Zacharia, Andreas C. Nearchou. (2013). “A meta-heuristic algorithm for the fuzzy assembly line balancing type-E problem”. *Computers and Operations Research*, Vol.40, Issue 12, pp. 3033-3044.

Panneerselvarn, R. and Oudaya Sankar, C. (1993), "New heuristics for assembly line balancing problems", *International Journal of Management and Systems*, 9, pp. 25 - 36.

Park, K., Park, S., Kim, W., (1997). “A heuristic for an assembly line balancing problem with incompatibility, range, and partial precedence constraints”. Computers and Industrial Engineering 32, pp. 321–332.

Pastor, R., (2011). “LB-ALBP: the lexicographic bottleneck assembly line balancing problem”. *International Journal of Production Research,* 49 (8), pp. 2425–2442.

Patterson, J.H.,Albracht, J.J., (1975). “Assembly line balancing: 0-1 Programming with Fibonacci Search”. *Operations Research,* Vol.23, pp. 166-174.

Pcstats website, (2008). “*How motherboards are made: A gigabyte factory tour”.* Downloaded on May 6, 2015 *from* <http://www.pcstats.com/articleview.cffm?articleid=1722&page=12>

Ping Su, NaiQi Wu & ZhaoQin Yu (2014). “A Petri net-based heuristic for mixed-model assembly line balancing problem of Type-E”. *International Journal of Production Research*, 52:5, pp. 1542-1556.

Pinto, P. A., Dannenbring, D. G., and Khumawala, B.M., (1978). “A heuristic network procedure for the assembly line balancing problem”. *Naval Research Logistics Review*, 25, pp. 299 – 307.

Pitakaso, R. (2014). “Differential Evolution Algorithm for Simple Assembly Line Balancing (SALBP-1)”. Department of Industrial Engineering, Faculty of Engineering Technical Report.

Ponnambalam, S. G., Aravindan, P., Naidu, G., Mogileeswar, G., (2000). “Multi-objective genetic algorithm for solving assembly line balancing problem”. *International Journal of Advanced Manufacturing Technology,* 16(5), pp. 341–352.

Purnomo, H. D., Wee, H. M., Rau, H. (2013). “Two-sided assembly lines balancing with assignment restrictions”. *Mathematical and Computer Modelling,* 57(1), pp. 189-199.

R. S. Garfinkel and M. R. Rao, (1971).”The bottleneck transportation problem”. *Naval research Logistics Quarterly*, Volume 18, Issue 4, pages 465–472.

Rachamadugu, R., Talbot, B., (1991). “Improving the equality of workload assignments in assembly lines”. *International Journal of Production Research,* Vol. 29, pp. 619–633

Rachamadugu, R.and Talbot, B., (1991). “Improving the equality of workload assignments in assembly lines”. *International Journal of Production* Research 29, pp. 619–633.

Rafael Pastor, Ignacio Chueca, Alberto García-Villoria (2012). “A heuristic procedure for solving the Lexicographic Bottleneck Assembly Line Balancing Problem (LB-ALBP)”. *International Journal of Production Research,* 50:7, pp. 1862-1876.

Rahul Malhotra, Narinder Singh, Yaduvir Singh (2011). “Genetic Algorithms: Concepts, Design for Optimization of Process Controllers”. Computer and Information Science Vol. 4, No. 2, pp. 39-54.

Raouf, A., El-Sayed, E, A., and Tsui, C. L., (1980). “A new heuristic approach to assembly line balancing”. *Computers and Industrial Engineering,* 4, pp. 223- 234.

Rapeepan Pitakaso, Kanchana Sethanan (2015). “Modified differential evolution algorithm for simple assembly line balancing with a limit on the number of machine types”. *Engineering Optimization* 48(2), pp. 1-19.

Reeve, N.R., and Thomas, W. H., (1973). “Balancing stochastic assembly lines”. *AIIE Transactions,* 5, pp. 223-229.

Rekiek, B., de Lit, P., Pellichero, F., Falkenauer, E., Delchambre, A., (1999). “Applying the equal piles problem to balance assembly lines*”. In The proceedings of the ISATP 1999*, Porto, Portugal, pp. 399–404.

Robinson L.W., McClain J.O., Thomas L J., (1990). “The good, the bad and the ugly: Quality on an assembly line”. *International Journal of Production Research*, Vol. 28, No.5, pp.963–980.

Rosenberg, O., Ziegler, H. (1992). “A comparison of heuristic algorithms for cost-oriented assembly line balancing”. *Zeitschrift fur Operations Research*, 36, pp. 477-495.

Rubinovitz, J., Bukchin, J. and Lenz, E., (1993). “RALB – A Heuristic Algorithm for Design and Balancing of Robotic Assembly Lines”. *Manufacturing Technology* 42(1), pp. 497-500.

Rubinovitz, J., Levitin, G. (1995). “Genetic algorithm for assembly line balancing”. *International Journal of Production Economics*, 41, pp. 343–354.

Sabuncuoglu, I., Erel, E., Tanyer, M. (2000). “Assembly line balancing using genetic algorithms”. *Journal of Intelligent Manufacturing*, 11(3), pp. 295–310.

Saltzman, M.J., I. Baybars, (1987). “A two-process implicit enumeration algorithm for the simple assembly line balancing problem”. *European Journal of Operational Research*, Vol.32, pp. 118-129.

Salveson, M. E., (1955). “The assembly line balancing problem”. *Journal of Industrial Engineering*, Vol.6, pp.18–25.

Sarin, S. C., Erel, E., & Dar-El, E. M. (1999). “A methodology for solving single-model, stochastic assembly line balancing problem”. *Omega*, 27(5), pp. 525-535.

Sarker, B.R., and Shanthikumar, J.G., (1983). “A generalized approach for serial or parallel line balancing”. *International Journal of Production Research*, 21, pp. 109-133.

Schofield, N.A., (1979). “Assembly line balancing and the application of computer techniques”. *Computers and Industrial Engineering*, 3, pp. 53- 59

Scholl A, Klein R., (1997). “SALOME: A bidirectional branch and bound procedure for assembly line balancing”. *INFORMS Journal on Computing* 9, pp. 319–334.

Scholl A, Klein R., (1999). “Computing lower bounds by destructive improvement—An application to resource-constrained project scheduling”. *European Journal of Operational Research* 112, pp. 322–346

Scholl, A., and Voss, S., (1996). “Simple assembly line balancing heuristic approaches”. Journal of Heuristics, 2, pp. 217-244.

Scholl, A., Fliedner, M., Boysen, N., (2010). “Absalom: Balancing assembly lines with assignment restrictions”. *European Journal of Operational Research* 200/3, pp. 688-701.

Scholl, Armin, Voß, Stefan, (1997). “Simple assembly line balancing - Heuristic approaches”. *Journal of Heuristics*, Vol 2(3), pp. 217-244.

Scholl, Armin; Boysen, Nils; Fliedner, Malte (2013). “The assembly line balancing and scheduling problem with sequence-dependent setup times: problem extension, model formulation and efficient heuristics”. *OR Spectrum*, Vol.35(1), pp.291-320.

Sen Zekai. (2004).“Genetik Algoritmalar ve En Iyileme Yöntemleri“. Su Vakfi, Istanbul.

Sener Akpinar,.Mirac Bayhan (2011). “A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints”. *Engineering Applications of Artificial Intelligence* 24, pp. 449–457.

Shin, D., (1990). “An efficient heuristic for solving stochastic assembly line balancing problems”. *Computers and Industrial Engineering*, 18, pp. 285-295.

Shtub, A., (1984). “The effect of incompletion cost on the line balancing with multiple manning of workstations”. *International Journal of Production Research*, 22, pp. 235-245.

Shtub, A., and Dar-El, E. M., (1990). “An assembly chart oriented assembly line balancing approach”. *International Journal of Production Research*, 28, pp. 1137-1151.

Shwetank Avikal, Rajeev Jain, P.K. Mishra, H.C. Yadav (2013). “A Heuristic Approach for U-Shaped Assembly Line Balancing to Improve Labor Productivity”. *Computers & Industrial Engineering* Vol.64, Issue 4, pp. 895–901

Sikora, C. G. S., Lopes, T. C., Silv, H., Magat, L. (2015). “Genetic algorithm for type-2 assembly line balancing”. *IEEE, In 2015 Latin America Congress on Computational Intelligence* (LA-CCI), pp. 1-6.

Silverman, F.N., and Carter, J.C., (1986|). “A cost-based methodology for stochastic line balancing with intermittent line stoppages”. Management Science, 32, pp. 455-463.

Simaria, A. S., Vilarinho, P.M., (2001). “A genetic algorithm approach for balancing mixed model assembly lines with parallel workstations”. *In The proceedings of the 6th annual international conference on industrial engineering theory, applications and practice*, San Francisco, USA.

Simaria, A. S., Vilarinho, P. M., (2004). “A genetic algorithm based approach to mixed model assembly line balancing problem of type II”. *Computers and Industrial Engineering*, 47, pp. 391–407.

Sotirios G. Dimitriadis (2006) “Assembly line balancing and group working: A heuristic procedure for workers’ groups operating on the same product and workstation”. *Computers & Operations Research* Vol.33, pp.2757–2774.

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

Sprecher, A., (1999). “A competitive branch and bound algorithm for the simple assembly line balancing problem”. *International Journal of Production Research*, Vol. 37(8), pp.1787–1816.

Sprecher, A., (2003). “Dynamic search tree decomposition for balancing assembly lines by parallel search”. *International Journal of Production Research* Vol.41, pp.1413–1430.

Storn, R. (2008). “Differential Evolution Research: Trends and Open Questions.” In Advances in Differential Evolution, edited by U. K. Chakraborty, 1–32. Berlin: Springer.

Storn, R., and K. Price. (1997). “Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces.” Journal of Global Optimization 11, pp. 341-359.

Su, P., Lu, Y., (2007). “Combining genetic algorithm and simulation for the mixed-model assembly line balancing problem”. *In Third International Conference on Natural Computation, IEEE*, Vol. 4, pp. 314-318.

Süer, G.A. (1998). “Designing parallel assembly lines”. *Computers & Industrial Engineering* 35, pp. 467-470.

Süer, G.A., Dagli, C.H., (1994). “A knowledge-based system for selection of resource allocation rules and algorithms”, *in Handbook of expert system applications in manufacturing:* structures and rules, (eds. A. Mital and S. Anand), Chapman and Hall, 109-129.

Sungyoul Lee (2010). “A Modified Heuristic Algorithm for the Mixed Model Assembly Line Balancing”. *DBPia Journal*, Vol.15(3), pp. 59-65.

Suresh, G., Vinod, V. V., Sahu, S. (1996). “A genetic algorithm for assembly line balancing”. *Production Planning and Control,* 7(1), pp. 38–46.

Suwannarongsri S., Limnararat S., Puangdownreong D., (2007). “A new hybrid intelligent method for assembly line balancing”. *In: IEEE international conference on industrial engineering and engineering managemen*t, pp. 1115– 1119.

Suwannarongsri, S., Puangdownreong, D., (2008). “Optimal balancing of multi-objective U-shaped assembly lines using the TSGA method”. *In 2008 IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 307-311.

Sysoev, V. and A. Dolgui (1999). A Pareto optimization approach for manufacturing system design. In: Proceedings of the International Conference on Industrial Engineering and Production Management (IEPM'99), Glasgow, Book 2, pp. 116-125. FUCAM

T. Watanabe, Y. Hashimoto, I. Nishikawa, H. Tokumaru (1995). “Line Balancing using Genetic Evolution Model”. *Control Engineering Practice* Vol. 3, Issue 1, pp. 69-76.

T.K. Bhattacharjee and S. Sahu, (1988) "A Heuristic Approach to General Assembly Line Balancing". *International Journal of Operations & Production Management*, Vol. 8 Iss: 6, pp. 67-77.

Taha, R. B., El-Kharbotly, A. K., Sadek, Y. M., Afia, N. H., (2011). “A genetic algorithm for solving two-sided assembly line balancing problems”. *Ain Shams Engineering Journal*, 2(3), pp. 227-240.

Talbot, F.B. , Patterson, J.H., Gehrlein, W.V., (1986). ” A comparative evaluation of heuristic line balancing technique”. *Management Science* 32, pp. 430–454.

Talbot, F.B., Patterson, J.H., (1984). “An integer programming algorithm with network cuts for solving the assembly line balancing problem”. *Management Science*, Vol. 30, No.1, pp. 85-99.

Tang, Q., Liang, Y., Zhang, L., Floudas, C. A., Cao, X. (2016). “Balancing mixed-model assembly lines with sequence-dependent tasks via hybrid genetic algorithm”. *Journal of Global Optimization*, 65(1), pp. 83-107.

Tang, Q., Xiao, Z., Liang, Y., Deng, M., Xi, Z. (2010). "Novel approach for balancing manual automobile assembly based on genetic algorithm". *In Industrial Engineering and Engineering Management (IEEM),* IEEE International Conference pp. 2028-2032.

Tanyaş, M., Baskak, M., (2003). "Üretim Planlama ve Kontrol". *İrfan Yayıncılı*k, İstanbul.

Taskın Çagatay, Gül G. Emel, (2009). "Sayısal Yöntemlerde Genetik Algoritmalar". Alfa Aktüel

Thangavelu, S.R., Shetty, C.M., (1971). "Assembly line balancing by 0-1 integer programming". *AIIE Trans*., 3, pp.61-68.

Thomopoulos, N. T., (1967). "Line balancing-sequencing for mixed-model assembly". *Management Science,* 14, pp. B59-B75.

Thomopoulos N T, (1970). "Mixed model line balancing with smoothed station assignments". *Management Science,* Vol.16, No.9, pp. 593–603

Tsujimura, Y., Gen, M., Kubota, E. (1995). "Solving fuzzy assembly line balancing using genetic algorithms". *Computers & Industrial Engineering*, 29, pp. 543-547.

Ugurdag, H.F., Rachamadugu, R., Papachristou, C.A., (1997). "Designing paced assembly lines with fixed number of stations". *European Journal of Operational Research* 102, pp. 488–501

Ullah Saif, Zailin Guan, Baoxi Wang, Jahanzeb Mirza, Shiyang Huang (2014). "A survey assembly lines and its types". Frontiers of Mechanical Engineering, Vol.9, Issue 2, pp. 95-105.

Urban T.L., Chiang W.C., (2006). “An optimal piecewise-linear program for the U-line balancing problem with stochastic task times”. *European Journal of Operational Research*, Vol. 168(3), pp. 109–120

V. L. V. Jonnalagedda, B. M. Dabade, (2010). “Heuristic procedure for Mixed Model Assembly Line Balancing Problem”. IEEE International Conference on Industrial Engineering and Engineering Management, pp. 552-556.

Valente, S. A., Lopes, H. S., Arruda, L. V. R., (2002). “Genetic algorithms for the assembly line balancing problem: A real-world automotive application”. *In R. Roy, M. Köppen, S. Ovaska, T. Fukuhashi, &F.Hoffman (Eds.), Soft computing in industry - recent applications* Berlin: Springer-Verlag, pp. 319-328.

Van Assche, F., W. S. Herroelen, (1979). "An Optimal Procedure for the Single-Model Deterministic Assembly Line Balancing Problem". *European Journal of Operational Research*, Vol. 3, pp. 142-149.

Van Hop, N., (2006). “A heuristic solution for fuzzy mixed-model line balancing problem”. *European Journal of Operational Research* 168, pp. 789–810.

Wang, F., Wilson, R.C., (1986). “Comparative analyzes of fixed and removable item mixed model assembly lines”. *IIE Transactions*, Vol.18 No.3, pp. 313-317.

Wang, H. S., Che, Z. H., Chiang, C. J., (2012). “A hybrid genetic algorithm for multi-objective product plan selection problem with ASP and ALB”. *Expert Systems with applications*, 39(5), pp. 5440-5450.

Wee, T.S., Magazine, M.J., (1982). “Assembly line balancing as generalized bin packing”. *Operations Research Letters* 1/2, pp. 56–58.

White, W. W., (1961). “Comments on a paper by Bowman”. *Operations Research*, 9 (March-April), pp. 274-276.

Wong, W. K., Mok, P. Y., Leung, S. Y. S. (2006). “Developing a genetic optimisation approach to balance an apparel assembly line”. *The International Journal of Advanced Manufacturing Technology*, 28(3-4), pp. 87-394.

Wu Yi, Liu Min,Wu Cheng (2002). “A genetic algorithm for optimizing the MPS of a processing-assembly production line with identical machines”. *Machine Learning and Cybernetics*, Proceedings.International Conference, Beijing.

Xiaofeng Hu (2011). “Heuristic Algorithm for Two-sided Assembly Line Balancing Problem with Multi-objectives”. *Industrial Engineering and Engineering Management (IEEM)*, IEEE International Conference, pp. 1407 – 1410.

Xiaosong Zhao, Chia-YuHsu, Pei-Chann, Chang, LiLi. (2016). “A genetic algorithm for the multi-objective optimization of mixed-model assembly line based on the mental workload”. *Engineering Applications of Artificial Intelligence*, 47, pp.140–146.

Yan Gao, Tianxiang Chang, Yangyang Wang,Yao Liu, HongYan Quan,Jianzhong Xu (2013). “An Improved Two-sided Assembly Line Balancing Algorithm based on Heuristic Rules”. Applied Mechanics and Materials, Vol. 404, pp 758-763.

Yang, C., Gao, J., (2010). “A multi-objective genetic-algorithm for mixed-model assembly line re-balancing problems”. *IEEE In Computers and Industrial Engineering (CIE), 2010 40th International Conference on*, pp. 1-6.

Yegul M.F., Agpak K, Yavuz M., (2010). “A new algorithm for U-shaped two-sided assembly line balancing”. *Transactions of the Canadian Society for Mechanical Engineering* 34(2), pp. 225–241.

Yeh DH., Kao HH., (2009). “A new bidirectional heuristic for the assembly line balancing problem”. *Computers and Industrial Engineering* 57, pp. 1155–1160.

Yossi Bukchin, Ithai Rabinowitch (2006) “Production, Manufacturing and Logistics A branch-and-bound based solution approach for the mixed-model assembly line-balancing problem for minimizing stations and task duplication costs”. *European Journal of Operational Research* Vol.174, pp. 492–508.

Yu, J., Yin, Y., (2010). “Assembly line balancing based on an adaptive genetic algorithm”. *The International Journal of Advanced Manufacturing Technology*, Vol. 48, Issue 1, pp 347–354.

Yu, J., Yin, Y., Chen, Z., (2006). “Scheduling of an assembly line with a multi-objective genetic algorithm”. *The International Journal of Advanced Manufacturing Technology*, 28(5-6), pp. 551-555.

Yunus Ege, Meral Azizoglu, Nur E. Ozdemirel (2009). “Assembly line balancing with station paralleling”. *Computers & Industrial Engineering* Vol.57, pp.1218–1225.

Zäpfel, G., Braune, R., Bèogl, M. (2010). “Metaheuristic Search Concepts: A Tutorial with Applications to Production and Logistics”. *Springer-Verlag Berlin Heidelberg*.

Zhang, W., Gen, M., (2011). “An efficient multi objective genetic algorithm for mixed-model assembly line balancing problem considering demand ratio-based cycle time”. *Journal of Intelligent Manufacturing*, 22(3), pp. 367-378.

Zhang, W., Gen, M., Lin, L. (2008). “A multi-objective genetic algorithm for assembly line balancing problem with worker allocation”. *In Systems, Man and Cybernetics, IEEE International Conference on*, pp. 3026-3033.