A MULTI-OBJECTIVE MEMETIC ALGORITHM FOR MIXED-MODEL TWO-SIDED DISASSEMBLY LINE BALANCING PROBLEM

Serkan MUTLU

Master's Thesis

Department of Industrial Engineering

Supervisor: Asst. Prof. Dr. Banu GÜNER

Eskişehir

Eskişehir Technical University

Institute of Graduate Programs

September 2021

ABSTRACT

A MULTI-OBJECTIVE MEMETIC ALGORITHM FOR MIXED-MODEL TWO-SIDED DISASSEMBLY LINE BALANCING PROBLEM

Serkan MUTLU

Department of Industrial Engineering

Eskişehir Technical University, Institute of Graduate Programs, September 2021

Supervisor: Asst. Prof. Dr. Banu GÜNER

Environmental impact is increasing day by day with the increasing population, affluence level and technological developments. Today, as a result of the increasing environmental impact with the increasing population, affluence level and technological developments, the level of Carbon Dioxide has exceeded 500 parts per million (ppm). In order to overcome environmental impacts such as the increased level of Carbon Dioxide, it is necessary to change the classical production/consumption approach. For this, Life Cycle Engineering (LCE) is recommended as a new approach. LCE is an approach that aims to minimize the environmental impact caused by the product, production or consumption by being involved in every step of the product life cycle. An important step of LCE is to design an easily disassemble, recyclable product at the design stage of product life cycle in order to minimize the environmental impact it will create when the products are at the end of their life as End-of-Life (EOL) product. Considering that the designed products will become EOL products after a while, the design of efficient disassembly systems is of critical importance in order to minimize the environmental impact. The Disassembly Lines that emerged in this context allow the efficient disassembly of the EOL products in bulk. Maximizing the efficiency of installed disassembly lines is critical to minimizing environmental impact. For this reason, the Disassembly Line Balancing (DLB) problem emerged in 1999 in order to optimize various performance criteria. The DLB problem has found wide coverage in the literature, and a lot of research has been done. However, the number of studies for the Two-Sided Disassembly Line Balancing (TDLB) problem, which is a line design that minimizes costs such as the cost of rotating on the line for disassembly of large-size products, is very few. In particular, the Mixed Model Two-Sided Disassembly Line Balancing (MTDLB) problem, in which the number of models is more than one, was introduced for the first time within the scope of this thesis.

In this thesis, the MTDLB problem has been investigated in order to minimize the sum of the design cost, the number of stations and sum of the selected disassembly task times of the disassembly line, which allows more than one similar or dissimilar products to be disassembled on the two-sided disassembly line. The Transformed AND/OR Graph (TAOG) precedence relations diagram is used for the MTDLB problem. The TAOG precedence relations diagram is a precedence diagram that takes into account all possible disassembly sequences in the event that only the physical conditions are considered due to the nature of the disassembly, and the functionality is not important. A Mixed Integer Linear Programming (MILP) based mathematical model has been developed to optimize the performance measures determined for the MTDLB problem. When the objective function of the developed mathematical model is hierarchical, it gives optimum results for small and medium sized cases with Gurobi solver. However, for the solution of large-size cases, the solution time is growing considerably. Therefore, a Multi-Objective Memetic Algorithm (MOMA) has been proposed for the solution of the MTDLB problem. The proposed approach is tested on a series of test problems and the results are compared with the highly preferred algorithms for multi-objective optimization in the literature. The results obtained show that the results of the MOMA algorithm, investigated for the MTDLB problem, give better results than the other algorithms.

Keywords: Life cycle engineering, Disassembly line balancing, Two-sided layout, Mixed-model, Multi objective memetic algorithm.

ÖZET

KARMA MODELLİ ÇİFT TARAFLI DEMONTAJ HATTI DENGELEME PROBLEMİ İÇİN BİR ÇOK AMAÇLI MEMETİK ALGORİTMA

Serkan MUTLU

Endüstri Mühendisliği Anabilim Dalı

Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Eylül 2021

Danışman: Öğr. Gör. Dr. Banu GÜNER

Her geçen gün artan teknolojik gelişmeler, refah seviyesi ve nüfus ile birlikte çevresel etki artmaktadır. Nitekim günümüzde Carbon Dioxide seviyesi 500 milyonda bir (parts per million – ppm)'i aşmıştır. Bu durumun üstesinden gelmek için klasik üretim/tüketim anlayışını değiştirmek ve yeni bir yaklaşım olarak Yaşam Zinciri Mühendisliği'ni benimsemek gerekmektedir. YZM, ürün yaşam döngüsünün her adımına dahil olarak üründen, üretimden veya tüketimden kaynaklı çevresel etkiyi en aza indirmeyi amaçlayan bir yaklaşımdır. YZM'nin önemli bir adımı, ürünlerin ömrünü tamamladığında yani Ömrü Tamamlanmış ürün konumuna geldiğinde, yaratacağı çevresel etkiyi en aza indirmek için tasarım aşamasında kolay ayrıştırılabilir, dönüştürülebilir bir ürün tasarlamaktır. Tasarlanan ürünlerin bir süre sonunda Ömrü Tamamlanmış ürün konumuna geleceği düşünüldüğünde yine çevresel etkiyi en aza indirebilmek için verimli demontaj sistemlerinin tasarımı kritik önem taşımaktadır. Bu kapsamda ortaya çıkan Demontaj Hatları yığın halindeki Ömrü Tamamlanmış ürünlerin verimli bir şekilde demontajına olanak sunmaktadır. Kurulan demontaj hatlarının verimliliğinin makisimize edilmesi çevresel etkinin en aza indirilmesi için kritiktir. Bu sebeple 1999 yılında çeşitli performans kriterlerini en iyilemek amacıyla Demontaj Hattı Dengeleme (DHD) problemi ortaya çıkmıştır. DHD problemi literatürde geniş yer bulmuş çokça araştırma yapılmıştır. Fakat, büyük ebatlı ürünlerin demontajı için hat üzerinde döndürme maliyeti gibi maliyetleri en aza indiren hat tasarımı olan Çift Taraflı Demontaj Hattı Dengeleme (ÇDHD) problemi için çalışma sayısı oldukça azdır. Özellikle model sayısının birden fazla olduğu Karışık Modelli Çift Taraflı Demontaj Hattı Dengeleme (KÇDHD) problemi bu tez kapsamında ilk kez tanıtılmıştır.

Bu tez kapsamında, birden fazla birbirine benzeyen ya da benzemeyen ürünlerin çift taraflı olarak tasarlanan demontaj hattı üzerinde demonte edilmesine olanak sağlayan

demontaj hattının tasarım maliyeti, istasyon sayısı ve seçilen demontaj görev süreleri toplamını en aza indirmek amacıyla KÇDHD problemi araştırılmıştır. KÇDHD problemi için Dönüştürülmüş VE/VEYA Grafiği (DVVG) öncelik ilişkisi kullanılmıştır. DVVG öncelik ilişkisi, demontajın yapısı gereği yanlızca fiziksel koşulların göz önüne alındığı fonksiyonelliğin önemli olmadığı durumda olası tüm söküm sıralarını dikkate alan öncelik diyagramıdır. KÇDHD problemi için belirlenen performans göstergelerini en iyileyecek bir Karışık Tam Saylı Doğrusal Programlama (KTDP) tabanlı matematiksel model geliştirilmiştir. Geliştirilen matematiksel modelin amaç fonkisyonu hiyerarşik yapıldığında Gurobi çözücüsü ile küçük ve orta boyutlu vakalar için optimum sonuç vermektedir. Fakat büyük boyulu vakaların çözümü için çözüm süresi oldukça büyümektedir. Bu yüzden KÇDHD probleminin çözümü için bir Çok Amaçlı Memetik Algoritma (ÇAMA) önerilmiştir. Önerilen yaklaşım bir dizi test problemi üzerinde test edilmiş ve sonuçlar literatürde çok amaçlı optimizasyon için oldukça tercih edilen algoritmalar ile karşılaştırılmıştır. Elde edilen sonuçlar, KÇDHD problemi için oluşturulan ÇAMA algoritması sonuçlarının diğer algoritmalara oranla daha iyi sonuçlar verdiğini göstermektedir.

Anahtar Sözcükler: Çevresel etki, Demontaj hattı dengeleme, Çift taraflı yerleşim, Karışık model, Çok amaçlı memetik algoritma

ACKNOWLEDGEMENTS

Yüksek lisans sürecinde bana desteklerini bir an olsun esirgemeyene sayın hocam Dr. Banu Güner'e sonsuz teşekkür ederim. Bu tezi her anımda yanımda olan annem Fatma Mutlu'ya, babam Saffet Mutlu'ya ve abim Yusuf Mutlu'ya armağan ediyorum.

"*Hayatta en hakiki mürşit ilimdir.*" Mustafa Kemal Atatürk

I would like to thank my professor, Dr Banu Güner, for his unwavering support during my master's degree. I dedicate this thesis to my mother Fatma Mutlu, my father Saffet Mutlu and my brother Yusuf Mutlu, who are always with me.

"*Our true mentor in life is science.*" Mustafa Kemal Atatürk

Serkan MUTLU

06/09/2021

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with "scientific plagiarism detection program" used by Eskişehir Technical University, and that "it does not have any plagiarism" whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

Serkan MUTLU

CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

GLOSSARY OF SYMBOLS AND ABBREVIATIONS

ALB	:	Assembly Line Balancing
Algo-ADL	:	Algorithm of Assign Disassembly Line
Algo-CDS	:	Algorithm of Choose Disassembly Sequence
AOG	:	AND/OR Graph
DLB	:	Disassembly Line Balancing
DWP	:	Disappearing Work-pieces Phenomena
EOL	:	End-of-Life
EWP	:	Exploding Work-pieces Phenomena
ILP	:	Integer Linear Programming
LCE	:	Life Cycle Engineering
LCM	:	Life Cycle Management
LP	:	Linear Programming
MA	:	Memetic Algorithm
MILP		Mixed Integer Linear Programming
MOMA		Multi-Objective Memetic Algorithm
MSDLB		Mixed-Model Straight Disassembly Line Balancing
MTALB		Mixed-Model Two-Sided Assembly Line Balancing
MTDLB		Mixed-Model Two-Sided Disassembly Line Balancing
NLP		Non Linear Programming
NP-Hard		Non Polynomial Solution Time
NSGA-II		Non-Dominated Sorting Genetic Algorithm – II
SSDLB		Single-Model Straight Disassembly Line Balancing
STDLB		Single-Model Two-Sided Disassembly Line Balancing
TAOG		Transformed AND/OR Graph
TDBL		Two-Sided Disassembly Line Balancing
TPD		Traditional Precedence Diagram

1. INTRODUCTION

Human beings are in danger due to excessive consumption today, and the increase in consumption day by day poses a threat to the future. If the excessive consumption of products and services continues at the ongoing rate, the Earth's resources will eventually run out, thousands of species will be threatened with extinction, and the global climate will change dramatically. As a result of the excessive consumption experienced today, atmospheric Carbon Dioxide levels are higher than the last 800,000 years. The atmospheric Carbon Dioxide level change over the last 800 thousand years is shown in Figure 1.1. For 800 thousand years, atmospheric Carbon Dioxide level has varied between 150 parts per million (ppm) and 300 ppm, but today it has increased to 409.8 ppm due to the increasing consumption rate with the industrial revolution (Lindsey, 2020). Atmospheric Carbon Dioxide level, which is increasing day by day, continues to threaten life on earth and the environment.

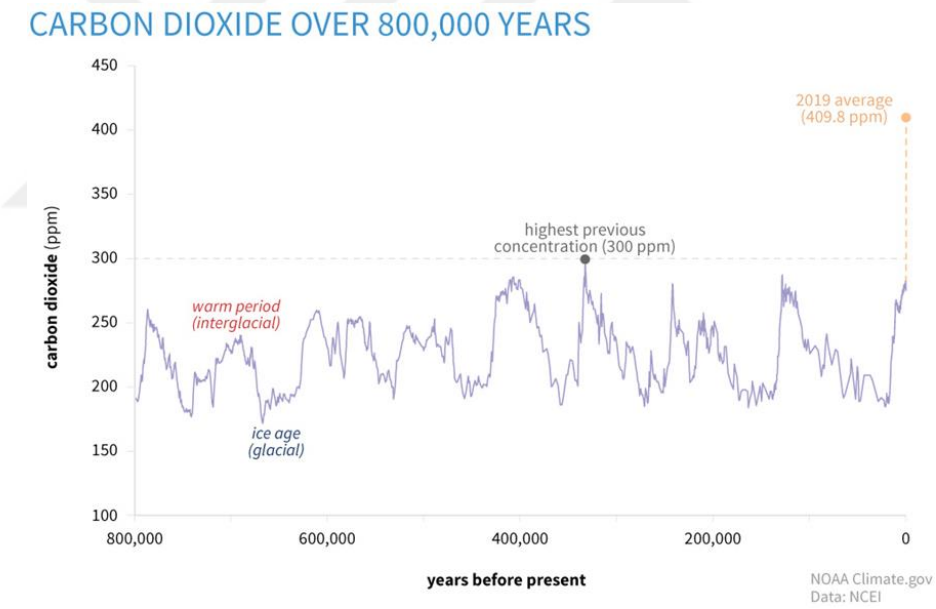


Figure 1.1. *Level of Carbon Dioxide over 800000 years* (Lindsey, 2020)

To see the atmospheric level of Carbon Dioxide, the European Space Agency SCIAMACHY instrument used to measure sunlight on the ENVISAT (ENVIronmental SATellite) satellite, and the TANSO-FTS with high optical efficiency, fine spectral resolution and wide spectral coverage on the GOSAT (Greenhouse gases Observing SAtellite) satellite. It measures atmospheric Carbon Dioxide levels on Earth with the TANSO-CAI, a radiometer of the ultraviolet (UV), visible and SWIR spectral ranges to correct for FTS and cloud and aerosol interference. Figure 1.2 shows two atmospheric

Carbon Dioxide levels recorded in August 2002 and April 2012. Carbon Dioxide level, which was 370 ppm on average in 2002, reached 400 ppm levels in 2012 in line with factors such as developing technological developments and increasing affluence level. There has been an increase of 30 ppm in a period of approximately 10 years. According to the report published by the European Space Agency, an increase of 5ppm of atmospheric Carbon Dioxide occurs every year. The increase in technological developments, consumption rate and population day by day causes an alarming increase in the level of Carbon Dioxide in the world and increases the possibility of scenarios that may be very bad for the future.

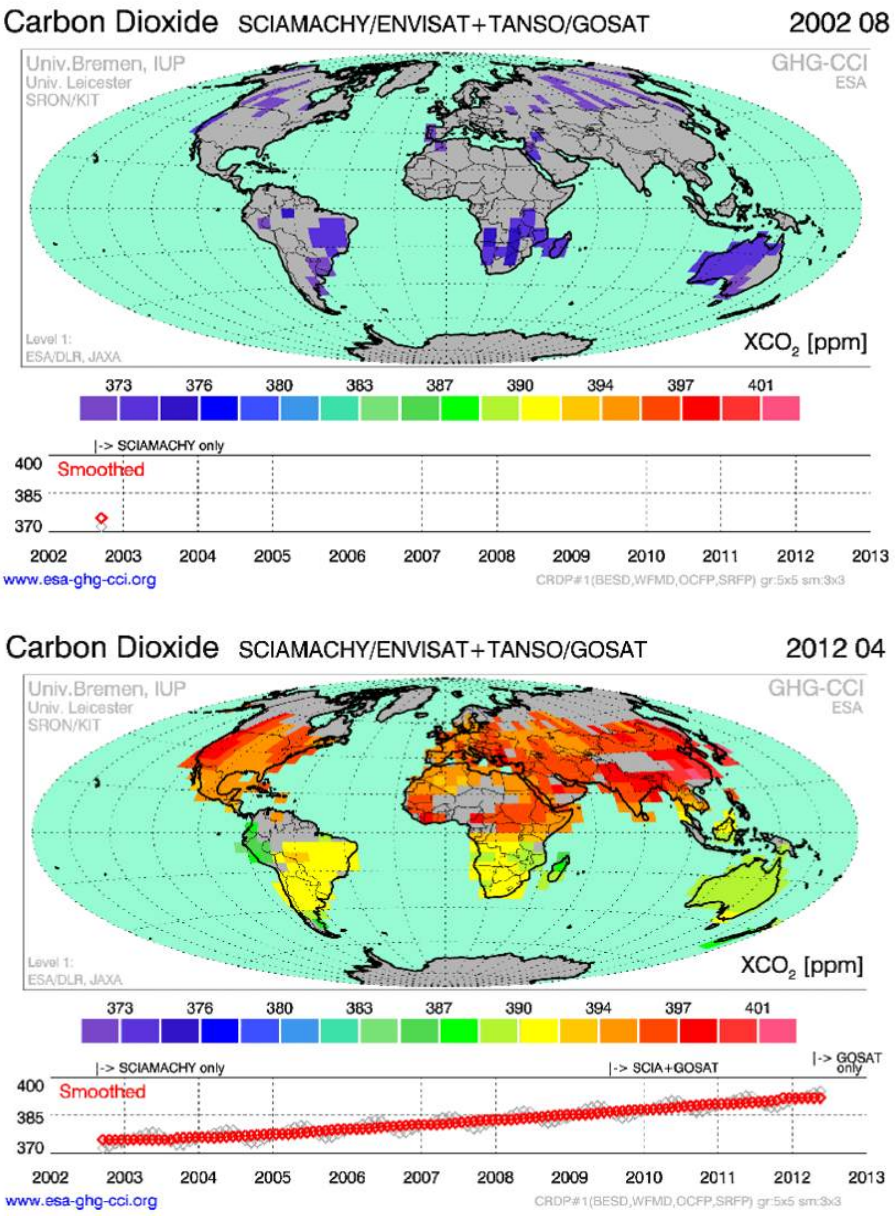


Figure 1.2. *Satellite image of Carbon Dioxide level between 2002 and 2012*

When the political/commercial decisions taken to reduce the Carbon Dioxide level in the atmosphere are strictly followed, it is 550 ppm in 2100, and 900 ppm if not (NCA, 2009). In both cases, the level reached is critical. Especially if the decisions taken are not followed, that is, if we continue to live according to today's living conditions, our world will become uninhabitable in 2100. The predicted increase is as in Figure 1.3.

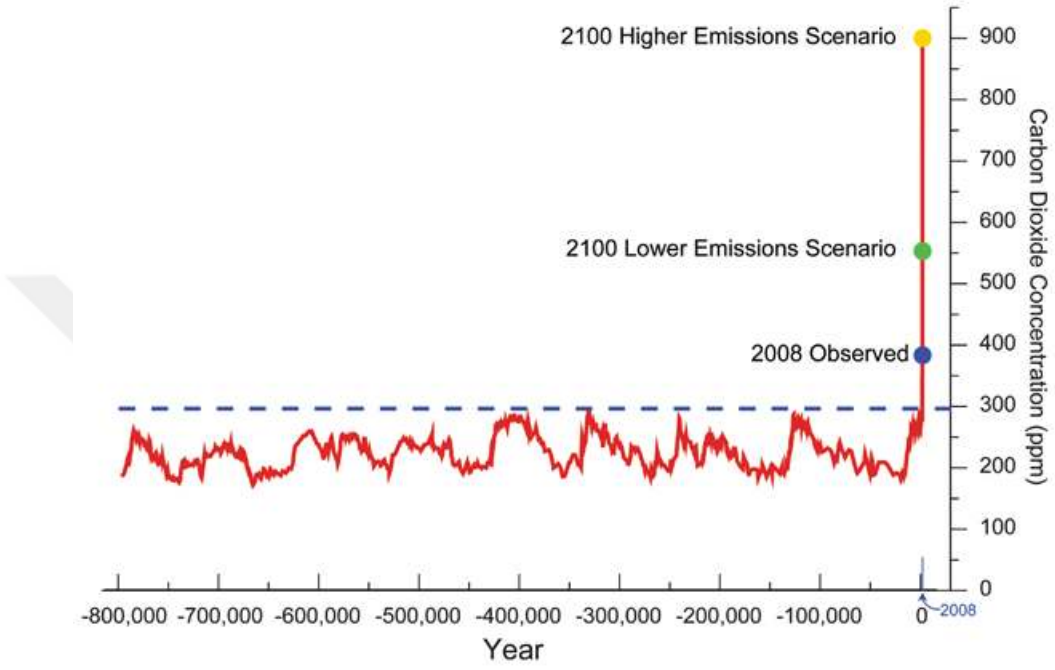


Figure 1.3. *Carbon Dioxide level scenarios* (NCA, 2009)

The Carbon Dioxide level is a result of environmental impact. Environmental impact increases in direct proportion with population, affluence and technology. In the field of sustainability this is known as the IPAT equation. The IPAT equation is as given by Equation 1.1. The IPAT equation was developed in 1970 as a result of a discussion between Barry Commoner, Paul R. Ehrlich and John Holdren (Holdren, 1993).

$$Environmental\ Impact = Population \times Affluence \times Technology$$
$$I = P \times A \times T \tag{1.1}$$

Increases in the human population in the last millennium (see. Fig. 1.4) bring along some environmental problems. Population growth; As a result of the increase in land use, habitat loss for other species, changes in vegetation as a result of increased resource use, the emergence of bacteria and viruses that may cause disease as a result of increased pollution and damage to the ecosystem lead to increased climate change and loss of biodiversity.

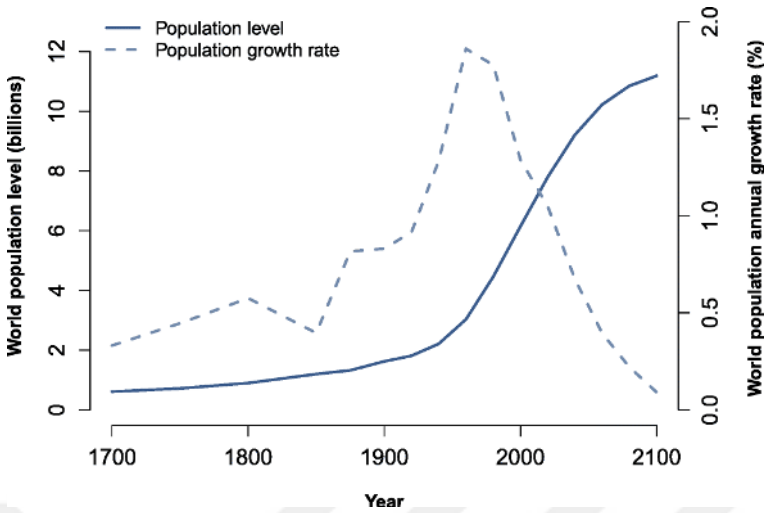


Figure 1.4. *Population level and Population growth rate by years*

Affluence is considered by many to be insignificant for environmental impact. But with the increase in affluence, consumption also increases. Therefore, as the consumption of each person increases, the environmental impact also increases. Gross Domestic Product (GDP) per capita is usually used to see consumption. GDP has been growing steadily over the last millennium. Along with this, per capita consumption is also increasing at the same rate. The higher the per capita consumption, the greater the environmental impact. Depending on consumption, it causes great effects on the environment with direct or indirect resources.

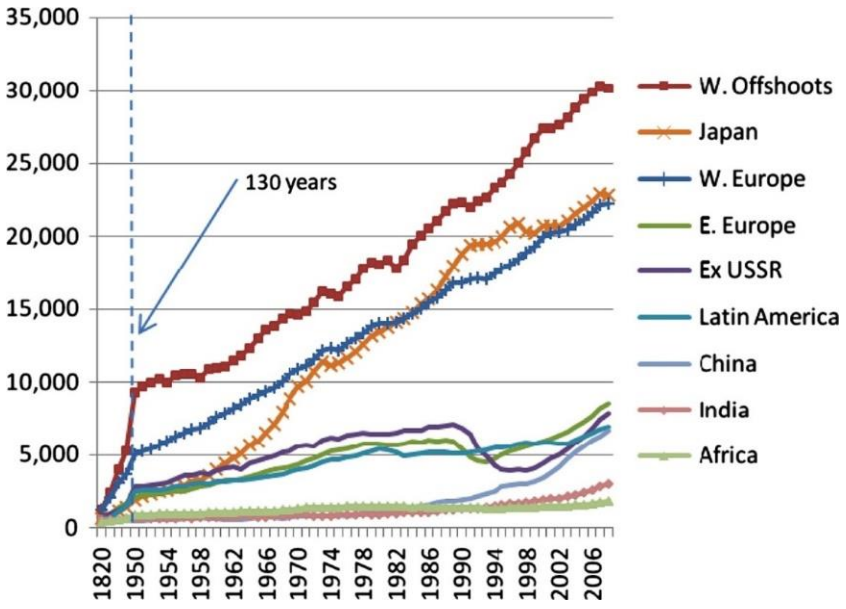


Figure 1.5. *Affluence level of regions by years*

With the development of technology, improvements in processes such as transportation, storage, production and transportation are possible. In this way, producers can produce more and consumers can consume more. Improvements in efficiency can reduce the amount of resources used, reducing the T factor in the IPAT equation, ie reducing environmental impact. However, since this situation will increase affluence for some people and businesses, it increases the environmental impact more than the resource advantage it provides.

Eco-efficiency is a management strategy of producing more products/services with less resources. Developed eco-efficient technologies cause us to use more products and larger physical sizes. This shows that affluence and technology combine well. For example, LCD televisions with LED light, developed for the production of televisions that consume less energy, have increased the dimensions of televisions up to 100 inches. This is one of the best examples that explains the increase in environmental impact due to affluence and technology.

Life Cycle Engineering (LCE) has emerged as a new engineering paradigm to reduce environmental impact. LCE is a product development activity within the scope of sustainability in the life cycle of one or more products. The aim of these activities is to create a sustainable production environment in order to meet the needs of both present and future generations.

A product's life cycle is about everything from its design to the manufacturing process, aftermarket use to the end of its life. A typical product life cycle is as given in Figure 1.6. The process begins with the extraction of the raw materials required for the product from nature. The raw materials obtained are directly or semi-product into the structure of the product with the production stage and the product emerges. Produced products are packaged and set out for customers using various modes of transport (road, sea, rail, air or pipeline). Customers buy and use products produced in line with their needs. As a result of consumption, the non-functional End-of-Life (EOL) product is left to nature as inert. EOL products are transformed into raw materials/semi-finished products, and the process returns to the beginning. While the processes in the Life Cycle are carried out, there are some consumptions that harm the nature but are not visible in the process. Some of these are Carbon Dioxide emissions, water use, plant/facility use and energy use. LCE is an approach that aims to minimize these idle consumptions.

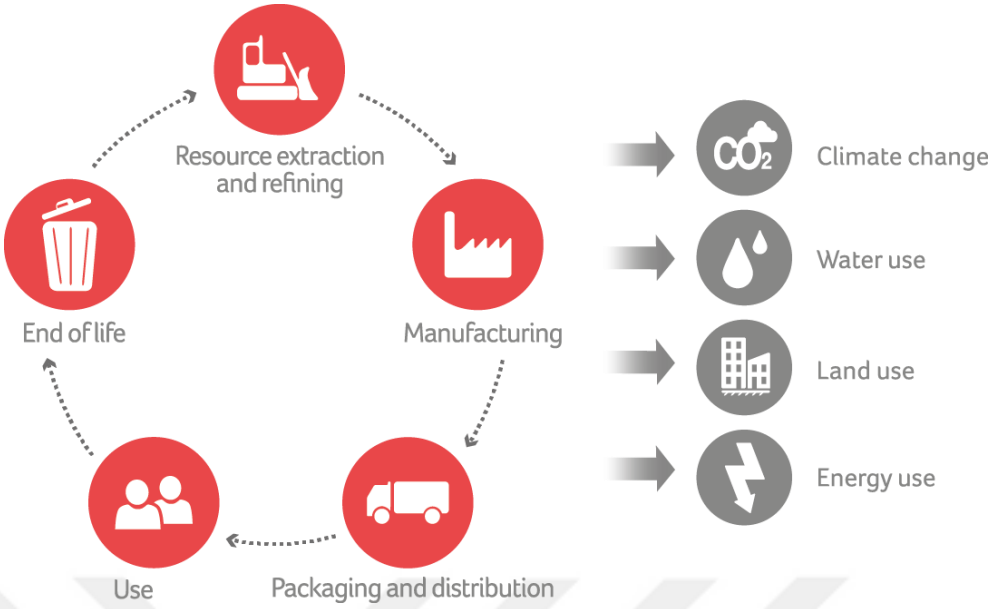


Figure 1.6. *A typical product life cycle and invisible output*

In order to position LCE with sustainability and other disciplines, a framework with the scope of environmental concern on the y-axis and the scope of temporal concern on the x-axis and given in Figure 1.7 has been introduced (Hauschild et al., 2017). Sustainability encompasses all approaches included in the framework. Sustainable Development is the most inclusive approach after Sustainable, focusing on societies on an environmental scale and human generations on a temporal scale. Industrial Ecology, a discipline, is based in this graph on economy on an environmental scale and human generations on a temporal scale. Concepts such as Circular Economy and Industrial Symbiosis are also represented at this level. All of these approaches are Top-Down concepts that are not part of the activities of industrial organizations. Just below the Industrial Ecology concept is Life Cycle Management (LCM), a Bottom-Up approach that is part of the activities of industrial organizations. LCM is a business management approach used to improve the products and services of any business, namely the sustainability performance of the company and its associated value chains. The LCE just below is also a Bottom-Up approach and is located in the middle of the frame. LCE enables the development of products and services, taking into account the growth of production volume and technological changes to systematically minimize environmental impacts at all stages of the life cycle. In this way, when there is a problem with environmental sustainability, it is prevented from shifting from the life cycle stage in

which it is located to the other life cycle stage. This is especially critical in the product design process, where 80% of the environmental footprint is determined.

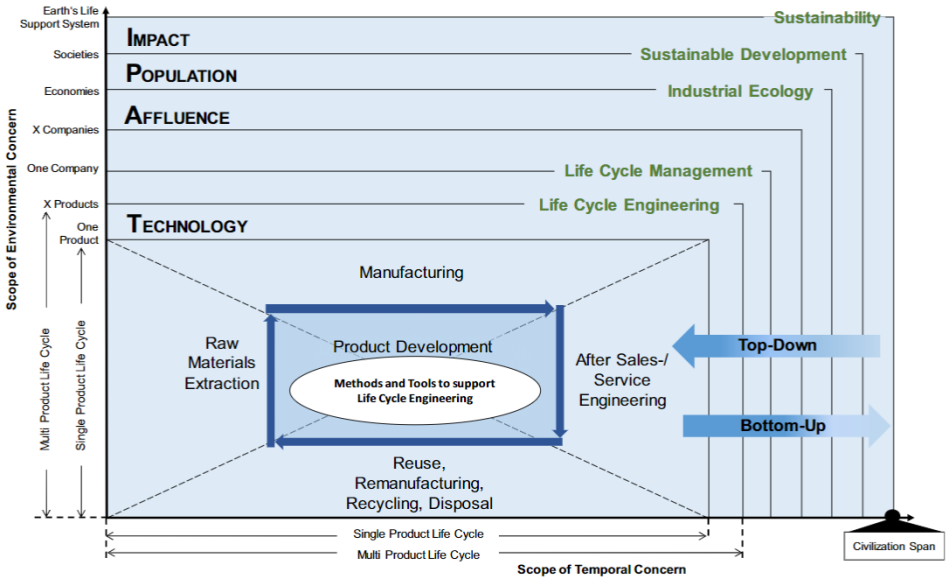


Figure 1.7. *A framework of Life Cycle Engineering* (Hauschild et al., 2017)

Thanks to the LCE tools, improvement areas in the life cycles of the products can be determined and as a result, environmentally sustainable products can be developed. Examples of commonly used LCE tools are Eco-Design, Life Cycle Assesment, and Life Cycle Costing. These tools enable the ability to measure the environmental impacts and associated costs of products to identify the highest improvement potentials to the product's lifecycle. For example, if the main environmental footprint for the production of a product is excessive energy consumption, it can be produced by designing alternative production processes that can produce the same product with less energy consumption. In this way, the environmental impact of the product is also reduced.

After the products are designed and produced, a number of LCE tools such as Eco-Labelling and Eco-Profilling are used. In this way, customers are encouraged to buy more environmentally friendly products and to consume less energy and water after purchasing the products.

Products lose their functionality at the end of their life cycle and are released into nature as EOL products. In order to manage the life cycle in an environmentally friendly way, it is designed by considering the products to be EOL at the design stage. LCE focuses on developing products that are easy to collect, disassembly, reuse, remanufacture

and recycle EOL products. For reuse, reproduction and recycling, first disassembly is required. Therefore, disassembly is critical in this process.

Disassembly activities can be done in a single workstation, or in a Disassembly Cell or a Disassembly Line consisting of more than one serial station (Güngör & Gupta, 1999a; Wiendahl et al., 1999). Although a single disassembly workstation or a Disassembly Cell is more flexible in terms of offering the possibility to sort the sub-parts resulting from disassembly activities according to their quality, the highest productivity rate and the lowest environmental impact (due to efficiency) are achieved with the Disassembly Line.

Disassembly facility consisting of a single station and employing a single worker is as given in Figure 1.8 (a). In such facilities, the EOL product enters the system and a number of disassembly operations are performed to obtain Disassembled Semi Products. The Disassembly Cell facility, which is formed by the clustering of more than one machine along a single station and where a single worker works, is as given in Figure 1.8 (b). In such facilities, the EOL product enters the system and a number of disassembly operations are performed on all machines, respectively, to obtain Disassembled Semi Products. The Disassembly Line facility, consisting of more than one station and one or more worker at each station, is as given in Figure 1.8 (c). In such facilities, the EOL product enters the system continuously and a set of disassembly operations is performed at each station, resulting in a set of Disassembled Semi Products at each station.

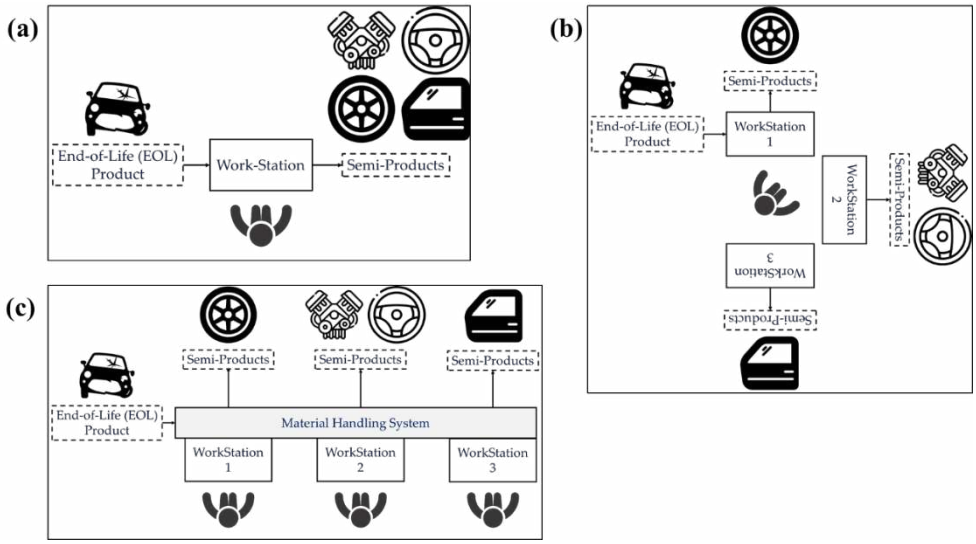


Figure 1.8. *Disassembly systems: (a) single workstation system, (b) disassembly cell system, (c) disassembly line system*

Obtained Disassembled Semi Products are sent directly to reuse, to re-production with Product Recovery or to recycling with Product Recycle, depending on their suitability in terms of quality. In case of chemical damage such as burning, Disassembled Semi Product becomes Disposal Product directly and is released into nature.

Disassembly Lines are used for rapid disassembly of bulk-packed EOL products. In order to minimize the environmental impact, as the production and use of easy-to-disassemble products are increasing day by day, there will be a need for Disassembly Lines in the future more than today. In order to minimize the environmental impact during disassembly of these lines, the lines must be designed very efficiently. In this context, the Disassembly Line Balancing (DLB) problem was first defined by Güngör and Gupta in 1999 (see. (Güngör & Gupta, 1999a)). Thanks to the solution of the DLB problem, the use of environmentally and materially valuable resources such as time and labor is minimized, while the profit obtained from selling the recycled parts to the suppliers is maximized and the amount of resources taken from nature is minimized. In general, the DLB problem is defined as the assignment of disassembly tasks to certain workstations, taking into account the priority relations between them, in a way that optimizes a number of performance indicators such as cycle time, number of workstations, level of meeting demands. In line with this definition, although the DLB problem seems to be similar to the Assembly Line Balancing (ALB) problem, there are important differences between the two problems, especially the priority relations, stock problems, and serious uncertainties in the quality and structure of the products. Although there are many differences between ALB and DLB, different line arrangements such as Straight, U-Shaped, Parallel, Two-Sided, model variants such as single, mixed or multi model and so on in Disassembly Lines, just as in Assembly Lines. Depending on the variations, different balancing models can be used.

In this study, a Mixed-Model Two-Sided Disassembly Line system, which can be used for the disassembly of different models of high-volume EOL products that are quite similar to each other, is defined using AND/OR Precedence Relations, and the Mixed-Model Two-Sided Disassembly Line system is defined in order to minimize the sum of line design cost, labor usage cost and disassembly times. A Mixed-Integer Linear Programming (MILP) based mathematical model has been developed for the Two-Sided Disassembly Line Balancing (MTDLB) problem. First, the three objective functions were

hierarchically converted to a single objective function and the mathematical model for small and medium-sized cases could not be solved with LINGO and GAMS commercial optimization programs. However, when the mathematical model is coded in Python using the GUROBI solver, a solution is obtained in reasonable time. However, it has been analyzed that when the case size increases, the solution times increase at a high rate. For this reason, a Memetic Algorithm (MA) is designed to minimize the hierarchical objective function for solving even larger cases. Considering the results of the MA, it was found that the solution time was not affected even if the case size increased. At the same time, a Multi Objective Memetic Algorithm (MOMA) has been developed in order to obtain Pareto Optimum results by considering each objective function individually. Compared to the Non-Dominated Sorted Genetic Algorithm - II (NSGA-II) algorithm, which is frequently preferred in the literature, MOMA results were found to provide better results thanks to the Iterated Local Search it contains. So far, the MTDLB problem in the literature has been investigated only in this study, which is the greatest originality of this study.

In the next section of this study, detailed information about disassembly activities is given, and information is given about disassembly problems in the literature, especially DLB, and various Precedence Relations, especially AND/OR Precedence Relations, which play a critical role in solving the DLB problem. In the third section of this study, a detailed literature review about DLB is presented, emphasizing the importance of multi-objective MTDLB. In the fourth part of the study, the multi-objective MTDLB problem is explained in detail and a MILP based mathematical model is developed. In the fifth section of this study, the MA and MOMA algorithms, which will be used for the solution of the multi-objective MTDLB problem, was introduced, and the adaptation of the MTDLB problem and the selection of the parameters used in the algorithm were made. In the sixth section of this study, the effectiveness of the MA and MOMA algorithms is compared with the other well-known algorithms used as reference in the light of some data from the literature and some data produced within the scope of the study. In the seventh section of this study, the results were explained, the necessity of the study was emphasized once again, and a few suggestions were made for future studies.

2. DISASSEMBLY LINE BALANCING

One area of focus of the LCE approach, which emerged with the aim of minimizing the damage to nature, is to design the product so that it is easy to collect, disassemble, reuse, re-manufacture and recycle when the designed products lose their functionality, that is, when they become EOL products. In this way, it is aimed to obtain a sustainable environment and production environment by minimizing the amount of resources taken from nature and waste given to nature. In particular, it is of great importance to bring back the sub-assembly parts of electronic devices and large-sized products (refrigerators, automobiles, airplanes, etc.) that contain many sub-assembly parts into production, with the necessary processes, in terms of protecting the environment and creating a sustainable production environment. The ever-increasing demand for EOL products and disassembled products necessitates efficient installation and management of disassembly systems in order to protect the environment by disassembling EOL products, to meet the demand for disassembled parts, and to have a flexible system. In this context, the Disassembly Line system has been introduced for the rapid and systematic disassembly of EOL products, which increase in quantity every day, and the Disassembly Line Balancing (DLB) problem for assigning disassembly tasks to this system in a way that optimizes certain performance criteria. Küçükkoç defines the DLB problem in 2019 as "*assigning disassembly tasks of any product or product group to workstations in a way that optimizes one or more performance indicator*" (see. (Kucukkoc et al., 2019)). Performance indicators for DLB problems are divided into Type-1 (minimization of the number of workstations) and Type-2 (minimization of cycle time). However, there are various performance indicators, including the smoothness index, which takes into account the smoothness between workstations, and the earliest possible exit of hazardous materials (Özceylan et al., 2019).

Güngör and Gupta first described the DLB problem in 1999 (Güngör & Gupta, 1999a). In 2001, a Shortest Route formulation was proposed to solve the DLB problem. Since the DLB problem was shown to be NP-hard by McGovern and Gupta in 2007 (see. (McGovern & Gupta, 2007b)), interest in heuristic and metaheuristic studies has increased. Indeed, in the same year, the same authors proposed a Genetic Algorithm (GA) for solving the DLB problem (McGovern & Gupta, 2007a). Today, although the solution methods of DLB problems are heuristic and metaheuristic methods, Linear/Nonlinear Programming (LP/NLP) using exact methods (see. (He et al., 2020; Li et al., 2020)) or

solvers such as CPLEX/LINGO/GUROBI (see. (Edis et al., 2019; Mete et al., 2019)) methods are also available. However, LP/NLP solvers are insufficient for solving large-scale problems and it requires large computation time to find the optimal solution and prove it to be the optimum solution.

A wide variety of precedence relationships are defined for the DLB problem. When the DLB problem first appeared, geometric removal constraints were determined. Although this is a correct approach, it becomes impossible to calculate when the problem size increases. Traditional precedence relations, which are frequently used in Assembly Line Balancing (ALB) problems, are used in some DLB problem studies with strict disassembly sequence. Although this is a relatively correct usage, it is more correct to use flexible relations instead of rigid ones, as functionality is not taken into account in disassembly unlike assembly. Although disassembly is flexible compared to assembly because it includes relations such as AND Predecessor, OR Predecessor, Complex AND/OR Predecessor, OR Successor, it can sometimes be very complex (Güngör & Gupta, 2002; Koc et al., 2009). This precedence relation, called AND/OR Graph (AOG), is a more valid approach to solving the DLB problem since it takes into account all possible disassembly sequences. However, just like in the geometric precedence relation, the calculation becomes impossible when the problem size increases in the AOG precedence relation. For this, Transformed AOG (TAOG) precedence relation consisting of normal nodes and artificial nodes was defined (see. (Koc et al., 2009)). In the literature, the use of TAOG precedence relation is discussed in two types, with and without OR Successor relation. In diagrams without OR Successors, "*In order for any task to be performed, all previous AND Predecessors must have been built and at least one of all previous OR Predecessors must have been made.*" condition is sought. In the diagrams in the OR Successor, "*After any task is done, one of the OR Successors must be done (no more or less).*" condition is sought.

Although assembly and disassembly seem to be quite similar in terms of general features, the operational and technical differences between them are discussed as given in Table 2.1 by Gupta and Güngör in 2001 (see. (Gupta & Güngör, 2001)).

Table 2.1. *Main differences between assembly and disassembly lines* (Gupta & Güngör, 2001)

Line Considerations	Assembly Lines	Disassembly Lines
Demand	Dependent	Dependent
Demand Sources	Single	Multiple
Demand Entity	End Product	Invidual Parts / Subassemblies
Precedence Relations	Yes	Yes
Complexity related to precedence relationship	High (Physical and functional precedence constraints)	Moderate (mostly physical constraints)
Uncertainty related to quality of parts	Low	High
Uncertainty related to quantity of parts	Low	High
Uncertainty related to workstations and the material handling system	Low to Moderate	High
Reliability of the workstations and the material handling system	High	Low
Multiple products	Yes	Yes
Flow process	Convergent	Divergent
Line flexibility	Low to moderate	High
Layout alternatives	Multiple	Multiple
Complexity of performance measures	Moderate	High
Known performance measures	Numerous	Numerous
Disappearing Work-pieces Phenomena (DWP)	N/A	Yes
Exploding Work-pieces Phenomena (EWP)	N/A	Yes
Required line robustness	Moderate	High
Complexity of "between workstations inventory" handling	Moderate	High
Known techniques for line optimization	Numerous	Numerous
Problem complexity	NP-hard	NP-hard

Looking at the main differences between assembly line and disassembly line in Table 2.1:

- It is seen that they are opposite to each other in terms of demand and demand source,
- Disassembly line is higher than assembly line in terms of uncertainty, line flexibility and material flow are the opposite of each other,

- It is seen that they are similar to each other in terms of line layout,
- Generally similar to each other as performance indicators,
- Generally similar to each other in terms of inventory problems and solution approaches.

Disappearing Work-pieces Phenomena (DWP) and Exploding Work-pieces Phenomena (EWP), which are on the disassembly line but not on the assembly line, are defined as follows:

- DWP, "*A task that fails during disassembly prevents the disassembly of all other parts remaining on the workpiece. In other words, the workpiece is lost. Stations after the workpiece failing station remain hungry.*" explained as. In order to overcome this situation, the failures to be experienced on the workpiece are minimized by completing the trainings for the workers/robots who perform the disassembly operations.
- EWP, "*A workpiece moves as two or more workpieces after it is disassembled. In the disassembly line, each piece acts as a separate work piece.*" explained as. In order to overcome this situation, the disassembly sequences are well adjusted to prevent movement as two separate parts on the material handling system.

Within the scope of this thesis, the DLB problem was classified in accordance with the diagram given in Figure 2.1 and the necessary information was collected by scanning the literature in this direction. It is possible to collect DLB problems in four main classes: problem characteristic, type of parameters used, measured performance indicators and methods/methods used for solution.

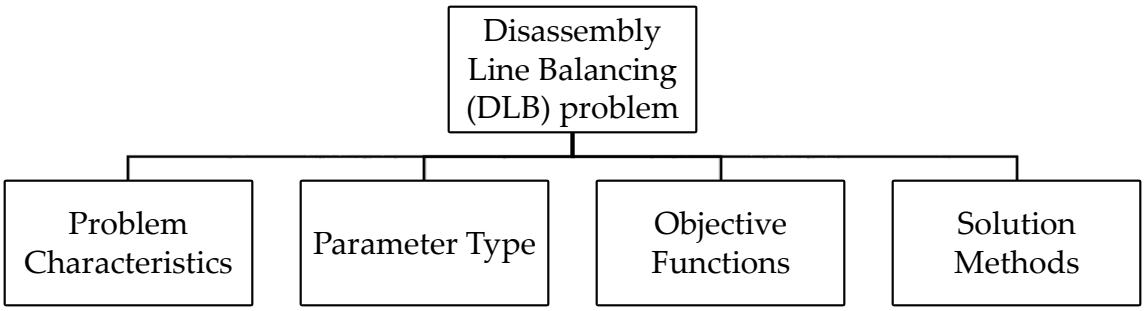


Figure 2.1. *Main classification scheme of DLB problem*

2.1. Problem Characteristics

They are the characteristics that reveal the basic structure of the DLB problem. It basically resembles the characteristics used for the ALB problem. Basically, the design of the line, the number and shape of the model entering the line, the type of worker (human or robot) used on the line, the priority relations used for disassembly, and the disassembly level (partial disassembly or complete disassembly) that arises specific to the DLB problem are collected in five main classes. The classification scheme is as given in Figure 2.2.

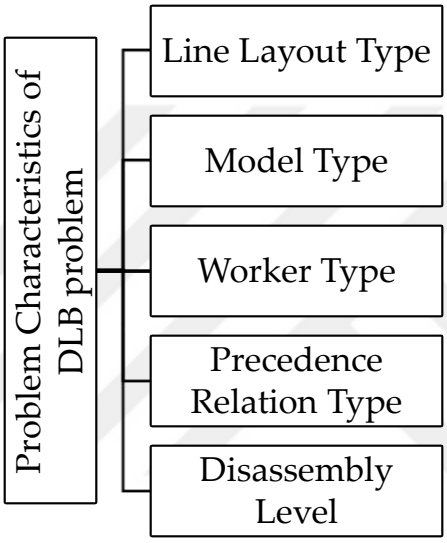


Figure 2.2. *Problem characteristics scheme of DLB problem*

2.1.1. Line layout types for DLB problem

Just like in the ALB problem, the design of the line must be determined in the DLB problem. Straight layout, U-Shape layout, two-sided layout and parallel layout used in ALB problems are also used in DLB problems. Line design is a strategic decision and is decided by the decision makers during line installation. Therefore, if there is an installed line, it is very difficult to change the layout of this line; sometimes it is not even possible. For this, it is a very critical decision to decide the line design based on the products to be assembled/disassembled. The DLB problem has been investigated using very different line layouts in the literature (see. (Özceylan et al., 2019)). The line layout type's scheme generally applicable to the DLB problem is as given in Figure 2.3.

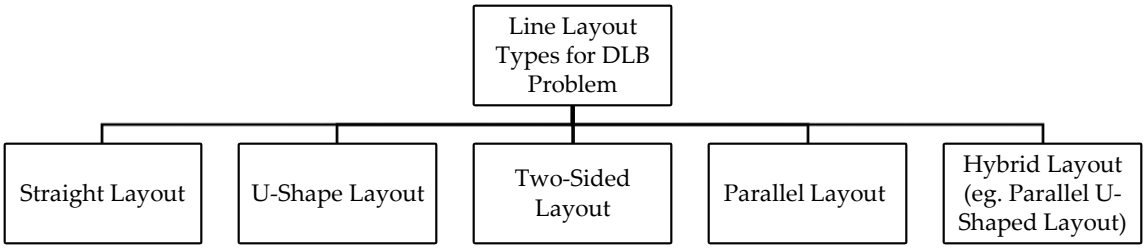


Figure 2.3. *Line layout types scheme for DLB problem*

Straight layout for DLB problem

Since it is easy to install and follow, the most preferred line layout type in the literature and practice is straight layout (Özceylan et al., 2019). It consists of workstations lined up in a straight line and a material handling system between stations. Products entering the line for disassembly are removed by visiting all stations respectively. The sub-assembly parts disassembled from the product are transported to the warehouses by means of the material handling system or collected at the stations for public transportation. As an example, the disassembly line using a straight layout for a disassembly consisting of 8 operations is as given in Figure 2.4. Here, four consecutive stations/workforce perform their disassembly tasks in sequence. After each disassembly operation, a sub-assembly part is removed from the main part and transported to the warehouse by the material handling system it belongs to.

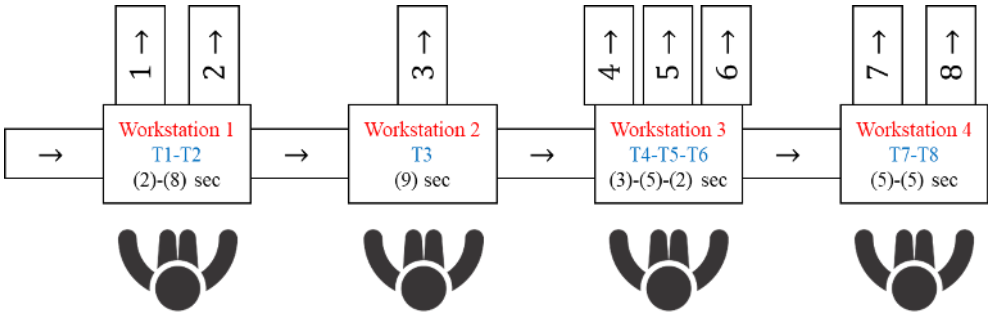


Figure 2.4. *A typical straight layout disassembly line*

U-Shaped layout for DLB problem

U-shaped layout is the most preferred layout type after straight layout as it provides space and efficiency advantage compared to straight layout (Agrawal & Tiwari, 2008; Avikal, Jain, et al., 2013; Li, Kucukkoc, & Zhang, 2019; Özceylan et al., 2019). The line is designed in a U shape. In this way, a single operator/robot can work synchronously at two different stations at the same time. The total working time of the synchronized

operator/robots is the cycle time. Therefore, the duration of the stations where the operator works can be designed to be lower than the cycle time. Since this situation does not reduce the efficiency, higher efficiency or less operator usage can be achieved compared to the straight layout. The products entering the line for disassembly are disassemled by visiting all the stations in the front, and then they return to the stations at the back, where disassembly operations are carried out. Therefore, the places where the product enters the line and where the last disassembling is done are very close to each other. The sub-assembly parts disassembled from the product are transported to the warehouses by means of the material handling system or collected at the stations for public transportation. As an example, the disassembly line using the U-Shaped layout for a disassembly consisting of 10 operations is as given in Figure 2.5. Here, the second operator works synchronously at both stations 2 and 5. This operator spends 7 seconds for station 2 and 3 seconds for station 5 on a line with a cycle time of 10 seconds (walking time between stations is accepted as 0). After each disassembly operation, a sub-assembly part is removed from the main part and transported to the warehouse by the material handling system it belongs to.

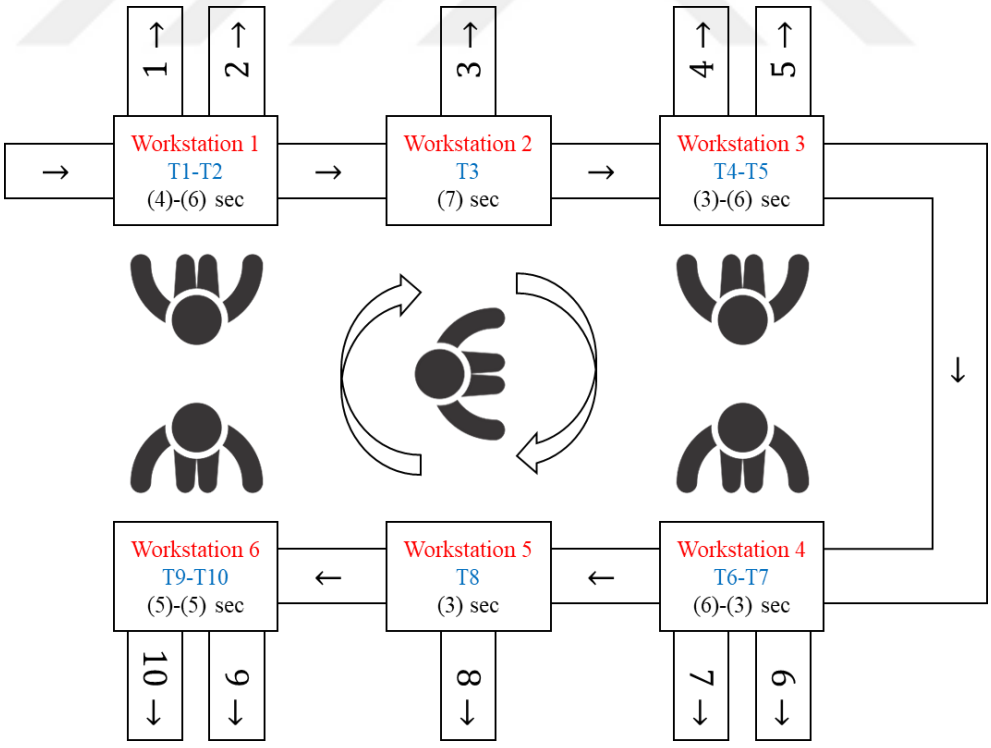


Figure 2.5. *A typical U-Shaped layout disassembly line*

Two-Sided layout for DLB problem

Two-Sided Disassembly Lines are generally designed specifically for disassembly of large volume products (automobile, truck, and refrigerator). It is advantageous when it is costly to rotate the product on the line or to change the position of the worker on the line. The increase in technological developments and affluence level day by day leads to the emergence of large products in size and volume. Among the most important examples of this is the increase in phone sizes and television sizes day by day. Studies for two-sided layout in the literature have intensified in the last two or three years (see. (Kucukkoc, 2020; Liang et al., 2021; Wang et al., 2019b)). The line consists of a material handling system that runs right through the middle and stations located on the right and left of this system. Stations that are positioned opposite each other to the right and left are called mated stations. The short length of the line depends on the number of stations opened. In some cases (considering the priority relations), it happens that while transactions are made on one station on the right or on the left, simultaneous transactions cannot be made on the other side. This is a factor that increases the length of the line. Again, considering the priority relations, it is possible to wait for a transaction to be completed on the other side so that the operation can be performed at the station deployed on the right or left side. Therefore, disassembly lines using two-sided layout should be planned well. As an example, the disassembly line using a two-sided layout for a disassembly consisting of 12 operations is as given in Figure 2.6. Here, the third station was opened only on the right, and the necessary precedence relations could not be provided for the opening of the station on the left. After each disassembly operation is done at the stations on the right and left, a sub-assembly part is removed from the main part and transported to the warehouse from the right or left side by the material handling system it belongs to. For the same example, when looking at Station 2 on the right, it is seen that the processing time is 5 seconds, half of the cycle time. The reason for this is that the station is located on the opposite side of some of its predecessors. Until these missions are completed, the mission in the station cannot be dismounted. Figure 2.7 explains this situation. Figure 2.7 (a) shows an example priority diagram up to task 7 and Figure 2.7 (b) shows the task schedule for the first two stations. As can be seen here, even if there is free time for the disassembling of task 5, the opposite side is waiting for the disassembly of task 4 to be completed. Similarly, for the disassembling of the task number 7 on the right, the disassembling of the task number 5 on the opposite side is expected. When the two-sided

layout is used, not only the priority relations within the station, but also the priority relations between the reciprocal stations are taken into account.

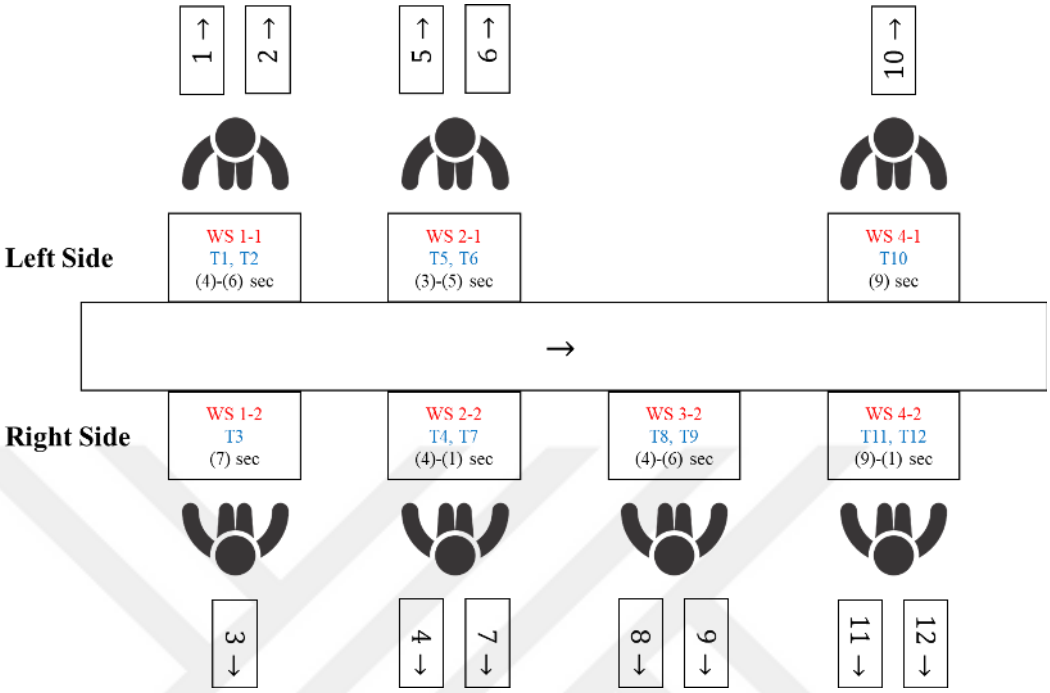
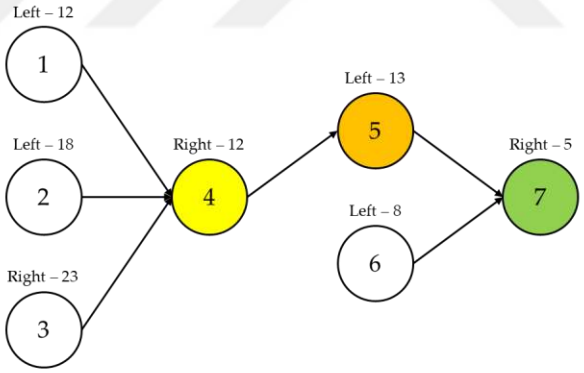
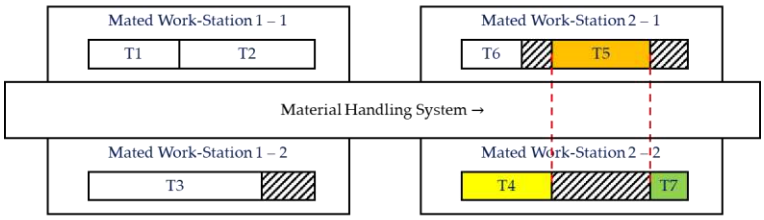


Figure 2.6. *A typical Two-Sided layout disassembly line*



(a) An example precedence diagram.



(b) An example solution of line balancing.

Figure 2.7. *Workstation schedule for Figure 2.6: (a) precedence diagram, (b) schedule of balanced disassembly line*

Parallel layout for DLB problem

Generally, more than one parallel line is used by businesses that want to increase their production/disassembling volume. In this case, the lines can be thought of as independent from each other, as well as in interaction with each other, that is, as a parallel layout. Considered independently of each other, each line will display straight layout features. However, considering that they are in interaction with each other, efficiency gains can be achieved since the operators will be used synchronized on both lines. Since disassembly was not a very common area in the early days, parallel layout was not preferred much. However, the increasing number of EOL products has intensified the research of DLB problems for parallel layout (see. (Fang et al., 2019; Hezer & Kara, 2015)). In the Production/Disassembly area, the lines are placed in parallel (each line shows straight features) and serial operators/robots are placed on the lines or the operator/robot working synchronized on both lines is placed between the lines. As an example, the disassembly line using a parallel layout consisting of nine operations on a total of two separate lines is as given in Figure 2.8. Although both lines work separately here, the fifth Operator works synchronously on both lines. On a line with a cycle time of 10 seconds, this operator spends 4 seconds on line 1 and 6 seconds on line 2 (movement time between stations is accepted as 0). After each disassembly operation, a sub-assembly part is removed from the main part and transported to the warehouse by the material handling system it belongs to.

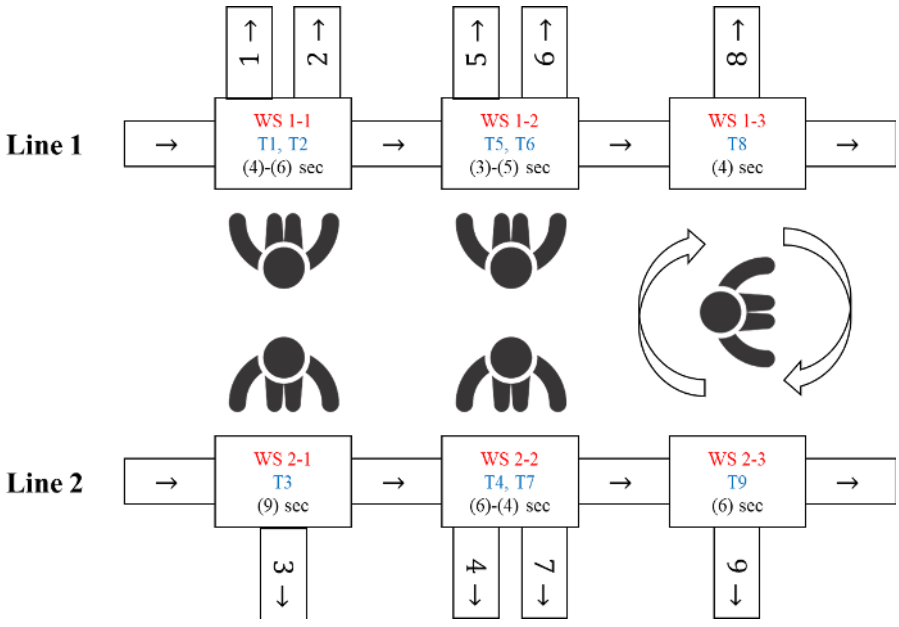


Figure 2.8. *A typical Parallel layout disassembly line*

Hybrid layout for DLB problem

Each line layout used for assembly/disassembly lines has its own advantages. In some assembly/disassembly works, the line layout is designed as a mixture of more than one layout. As an example, Küçükkoç and Zhang designed a parallel u-shaped layout for assembly in 2015 (see. (Kucukkoc & Zhang, 2015)). This layout is as given in Figure 2.9. In this way, labor savings and increased productivity have been achieved in cases where the production volume is high.

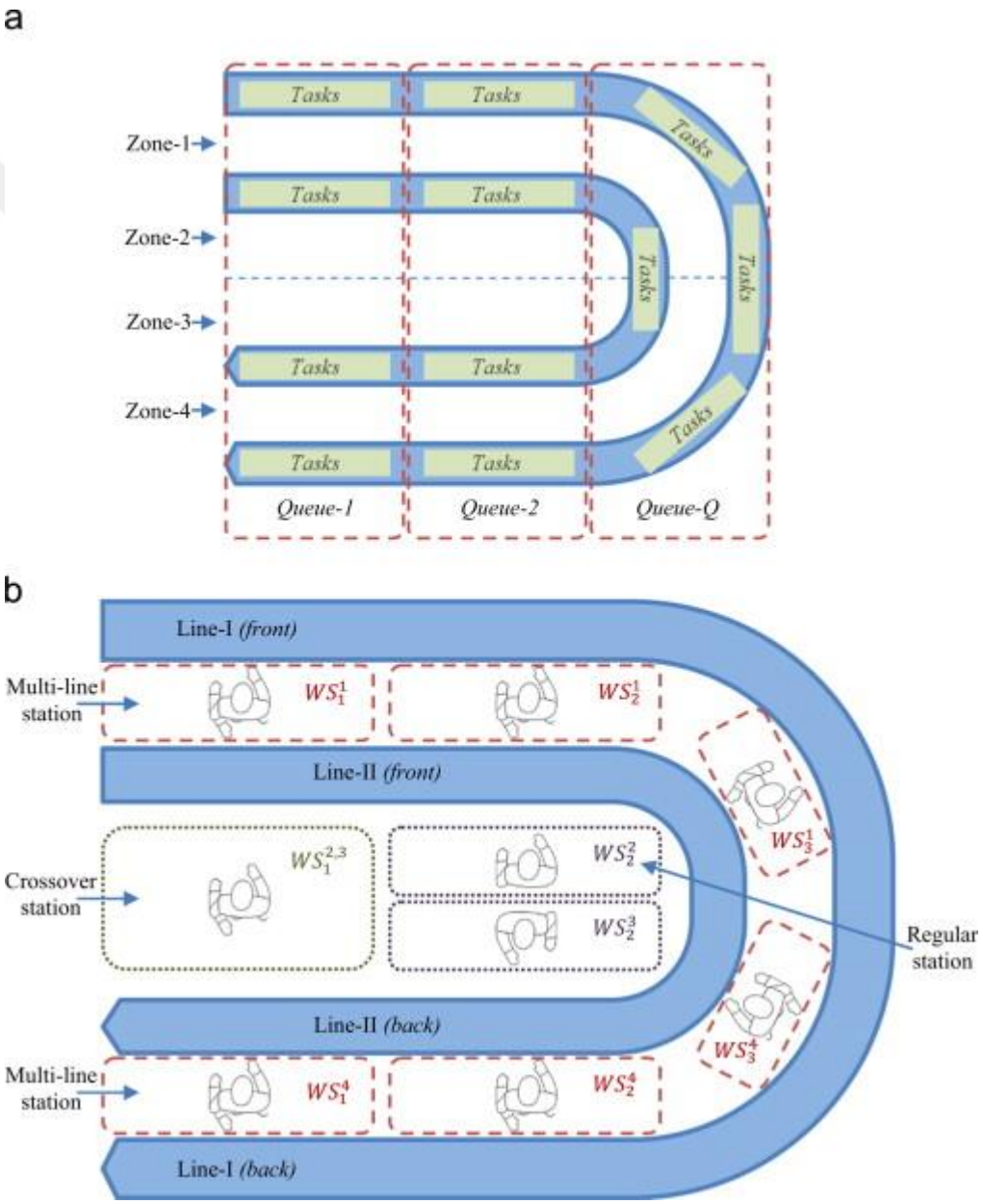


Figure 2.9. *Schematic representation of the parallel U-shaped assembly line system: (a) zoning of the work area and (b) allocation of the workstations* (Kucukkoc & Zhang, 2015).

2.1.2. Model type for DLB problem

Just like in the ALB problem, the input types of the products (model type) entering the line must be determined in the DLB problem. Single model, mixed model and multi model used in ALB problems are also used in DLB problems. Since assembly lines are generally designed to assemble and sell one type or similar type of products to customers, the format in which the mixed model is converted into a single model is generally used by combining the precedence relations of single model or products that are quite similar to each other. However, EOL products entering the disassembly lines are quite different from each other, even if they are quite similar to each other since the parameters such as place/time etc. are different. At the same time, since the line to be created for the disassembly of a single type of product is much higher than the cost of the line to be created for the assembly of a single type of product, even if single model is preferred in theory, mixed or multi model lines are established in practice. In general, the model type's scheme applicable to the DLB problem is as given in Figure 2.10.

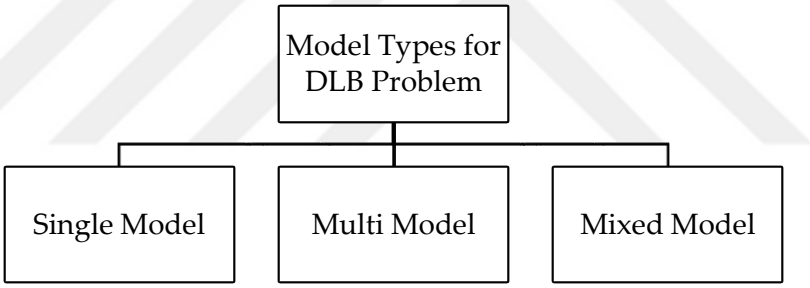


Figure 2.10. *Model types scheme for DLB problem*

Single model for DLB problem

Single Model is the most preferred model type, which is not seen in real life disassembly systems, but offers ease of calculation in theory (see. (Özceylan et al., 2019)). The uniformity of the EOL product type entering the line and the tool/equipment and workforce expertise on the line are only for one type of product. Its type is as given in Figure 2.11.

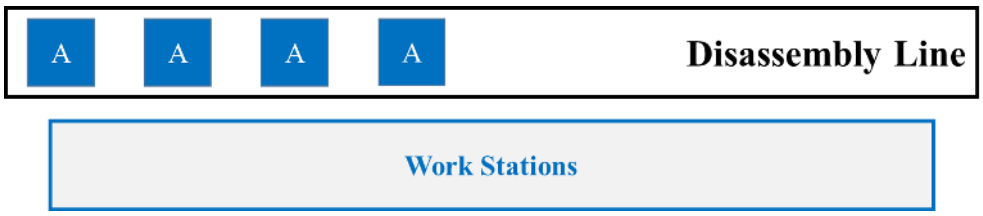


Figure 2.11. *A typical single model disassembly line*

Multi model for DLB problem

Multi-model is another type of model encountered in real-life disassembly systems. It is generally preferred in multi-model lines in terms of ease of tracking and calculation. It is sufficient for the operators to have relatively less competence compared to the mixed model. It is generally used when a setup time is required during model migrations. For example, if there is a dyed process, it will take time to switch from red to blue. In such a case, the products are disassembled in batches and the setup time is shared by the amount of product in the batch. It is the least researched model type in the literature for the DLB problem (see. (Özceylan et al., 2019)). A multi model disassembly line consisting of two models is as given in Figure 2.12.

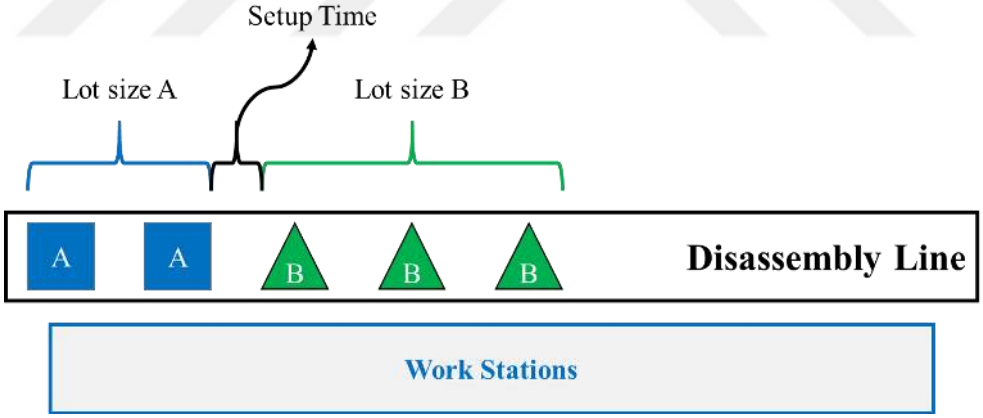


Figure 2.12. *A typical multi model disassembly line*

Mixed model for DLB problem

Mixed-model is the type of model often encountered in real-life disassembly systems. In the case of mixed models encountered in assembly lines, since the products are generally similar to each other, the precedence relation is converted into a single model format and a solution is sought. However, since a wide variety of products come to the disassembly lines, the tools/equipment specific to the incoming products must be deployed at the stations and the disassembly operations specific to the incoming product

must be assigned to the stations. Compared to the single model situation, it is expected that the operators at the stations will have more competencies. While the operator is disassembling the A model on the line, the next product B may be the other product C model. In this case, trainings are given so that the operator does not react too quickly and cause the line to stop. Generally, there are relatively few studies in the literature compared to the single model, as the modeling and solution difficulties are higher than the single model (see. (Fang et al., 2019, 2020; Xia et al., 2019)). A mixed model disassembly line consisting of three models is as given in Figure 2.12.

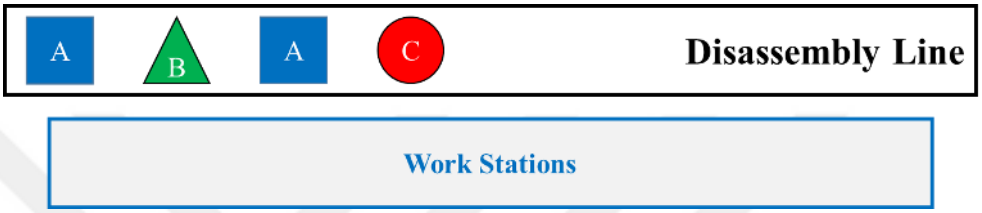


Figure 2.13. *A typical mixed model disassembly line*

2.1.3. Worker type for DLB problem

In the stations located on the disassembly line, a human can work as well as a robot. Since robots are faster and have a lower error rate than humans, they are frequently preferred in assembly lines today. However, the high uncertainty in the disassembly lines creates a question mark for the use of robots instead of humans. Although disassembly lines consisting of robots are not very common in real life, the increase in the reaction capabilities of robots in the face of errors is a sign that the use of robots in real life disassembly lines will increase. In fact, studies have been started for the use of robots in disassembly lines (see. (Fang et al., 2019; Ming et al., 2019)). In some lines, human and robot interactive work is also available (Liu et al., 2019). In general, the worker type scheme applicable to the DLB problem is as given in Figure 2.3.

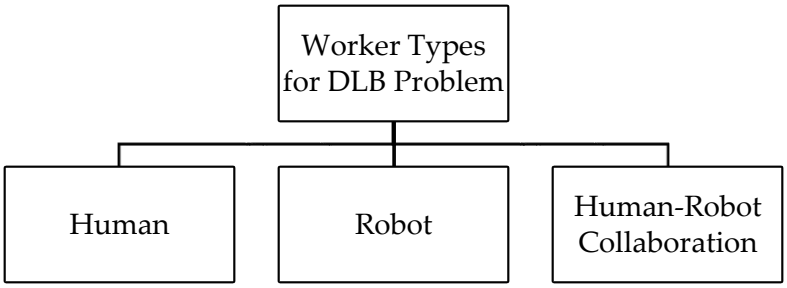


Figure 2.14. *Worker types scheme for DLB problem*

2.1.4. Precedence relation type for DLB problem

Precedence Diagrams should be used to provide physical constraints for DLB problem, similar to the Precedence Diagrams used to provide physical and functional constraints in ALB problem that are frequently studied in the literature. However, although a stricter Precedence Diagram occurs because both physical and functional constraints are taken into consideration in ALB problems, flexible Precedence Diagrams emerge because the functionality is not important in DLB problems. ALB problemlerinde fiziksel ve fonksiyonel kısıtların her ikisi de göz önüne alındığından dolayı daha katı bir Precedence Diagram oluşmasına ragmen, DLB problemlerinde fonksiyonellik göz önüne alınmadığı için daha esnek Precedence Diagram ortaya çıkmaktadır. Fakat, öncelik ilişkilerinin esnetilebilir oluşu bir çok olası disassembly sequence oluşmasına, dolayısıyla da çözümün zorlaşmasına yol açmaktadır.

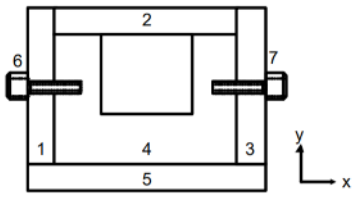
Generally, four types of Precedence Diagrams are used in DLB problem in the literature. These Precedence Diagram types:

1. Geometrically based Precedence Diagram, first used for DLB problem by Güngör and Gupta in 1999 (Güngör & Gupta, 1999b).
2. Traditional based Precedence Diagram defined by Salveson in 1955 (Kriengkorakot & Pianthong, 1955).
3. AND/OR Graph (AOG) based Precedence Diagram defined by Homem de Mello and Sanderson in 1990 (L.S. Homem de Mello & Sanderson, 1990; Luiz S. Homem de Mello & Sanderson, 1991a, 1991b).
4. Transformed AND/OR Graph (TAOG) based Precedence Diagram defined by Koç et al. in 2009 (Koc et al., 2009).

Geometrically based precedence relation

Geometrically precedence relation is the precedence relation method used when the DLB problem first appeared. The disassembly direction of the sub-assemblies on the main part is indicated as +x, -x, +y, -y, +z, -z in three axes. In this direction, a solution is sought. The Geometrically based precedence relation table revealed for a product recommended by Güngör and Gupta in 1999 is as seen in Figure 2.15. The reading of the R matrix is as follows. All tasks of each sub-assembly must be completed on a column-by-column basis. Considering product number 1 in this direction, it is sufficient to disassemble product

number 6 in any way. Looking at the product number 2, no disassembling rule is valid, that is, the product can be disassembled. Looking at the product number 3, the product number 7 must be disassembled in any way. When looking at product number 4, product number 1 should be removed in the -x direction, product number 2 in the y direction, product number 3 in the +x direction, product number 5 in the -y direction, product number 6 and 7 in any direction. There is no disassembly requirement for products 5, 6 and 7, that is, these products can be disassembled.



a. Sample Product

$$R = \begin{array}{c} n \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & -x & 0 & 0 & 0 \\ 0 & 0 & 0 & y & 0 & 0 & 0 \\ 0 & 0 & 0 & x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -y & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

b. R of the Product

Figure 2.15. *Geometrically based precedence relation example: (a) sample product, (b) precedence matrix* (Güngör & Gupta, 1999b)

Traditional precedence relation

Salveson first defined the Traditional Precedence Diagram, which is often used in ALB problems, in 1955. All the tasks to be done in this Precedence Diagram and the priorities of the tasks are determined. Without all predecessor tasks of a task, that task cannot be done. Erel et al. (2005), a Task based Precedence Diagram used in the ALB problem is as given in Figure 2.16. In this diagram, task 7 must be completed in order to perform task 9, and similarly, tasks 3, 4 and 5 must be completed in order to perform task number 7.

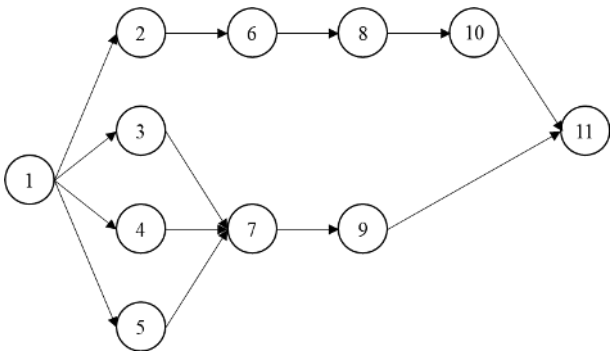


Figure 2.16. *A typical traditional precedence relation for assembly line balancing problem*

Similar to the Traditional Precedence Diagram used in ALB problems, the Traditional Precedence Diagram can also be used in DLB problems. Typical Traditional Precedence Diagram used in the DLB problem is as given in Figure 2.17 (Wang et al., 2019a). In this way, disassembly tasks numbered 12 and 13 must be completed in order to carry out the disassembly task number 9, and similarly task number 15 must be completed for task number 12, and task number 16 must be completed in order to perform task number 13. The Traditional Precedence Diagram used in DLB problems is an inverted formation of the TPD used in ALB problems.

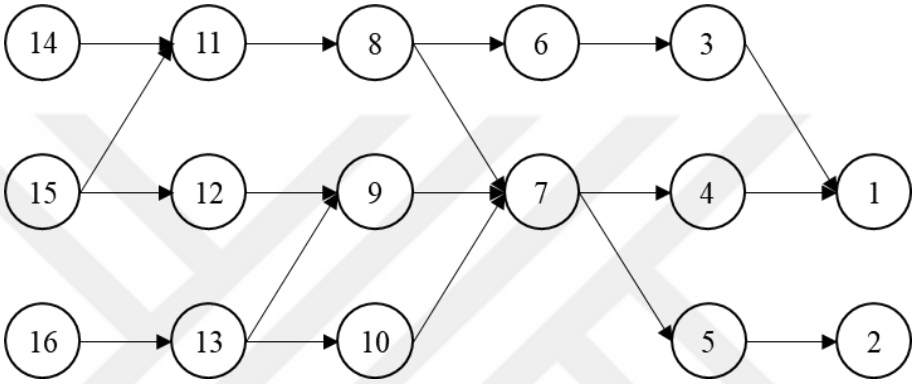


Figure 2.17. *A typical traditional precedence relation for disassembly line balancing problem*

AND/OR Graph (AOG) based precedence relation

The AND/OR Graph (AOG) precedence diagram ensures that all possible assembly/disassembly sequences are taken into account when the best assembly/disassembly sequence is not known, taking into account the necessary physical and functional constraints. That is, a typical AOG is a graph showing all possible ways a product can be disassembled down to all the components it is composed of. Nodes in this graph correspond to subassembly product and arcs correspond to assembly/disassembly operations. In DLB problems, it becomes difficult to choose among all possible disassembly sequences when functionality is secondary and only physical constraints are considered. For this, AOG precedence diagram is frequently used especially in DLB problems. The AOG precedence diagram was originally designed by De Mello and Sanderson (1990) to consider all possible assembly sequences. In order to better understand the AOG precedence diagram, the example created by De Mello and Sanderson (1990) is explained. Considering the physical and functional constraints on a simple product, which consists of four separate parts named cap, stick, receptacle and handle, given in Figure 2.18 and defined by Homem de Mello and Sanderson, it is known

that all possible assembly sequences are as in Figure 2.19 (L.S. Homem de Mello & Sanderson, 1990).

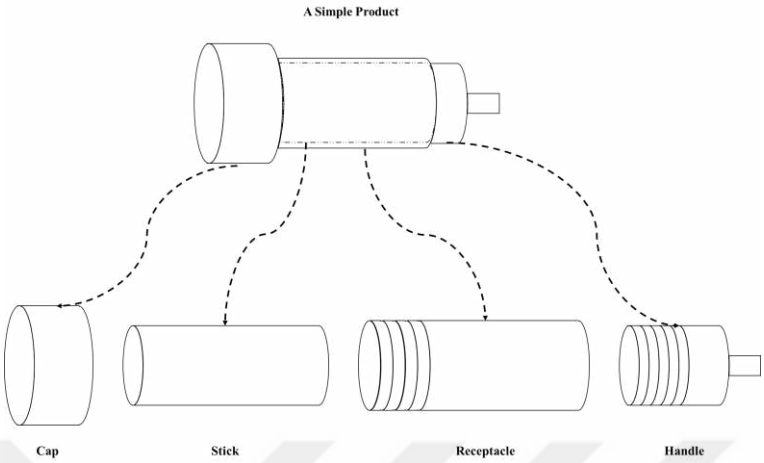


Figure 2.18. *A simple product* (L.S. Homem de Mello & Sanderson, 1990)

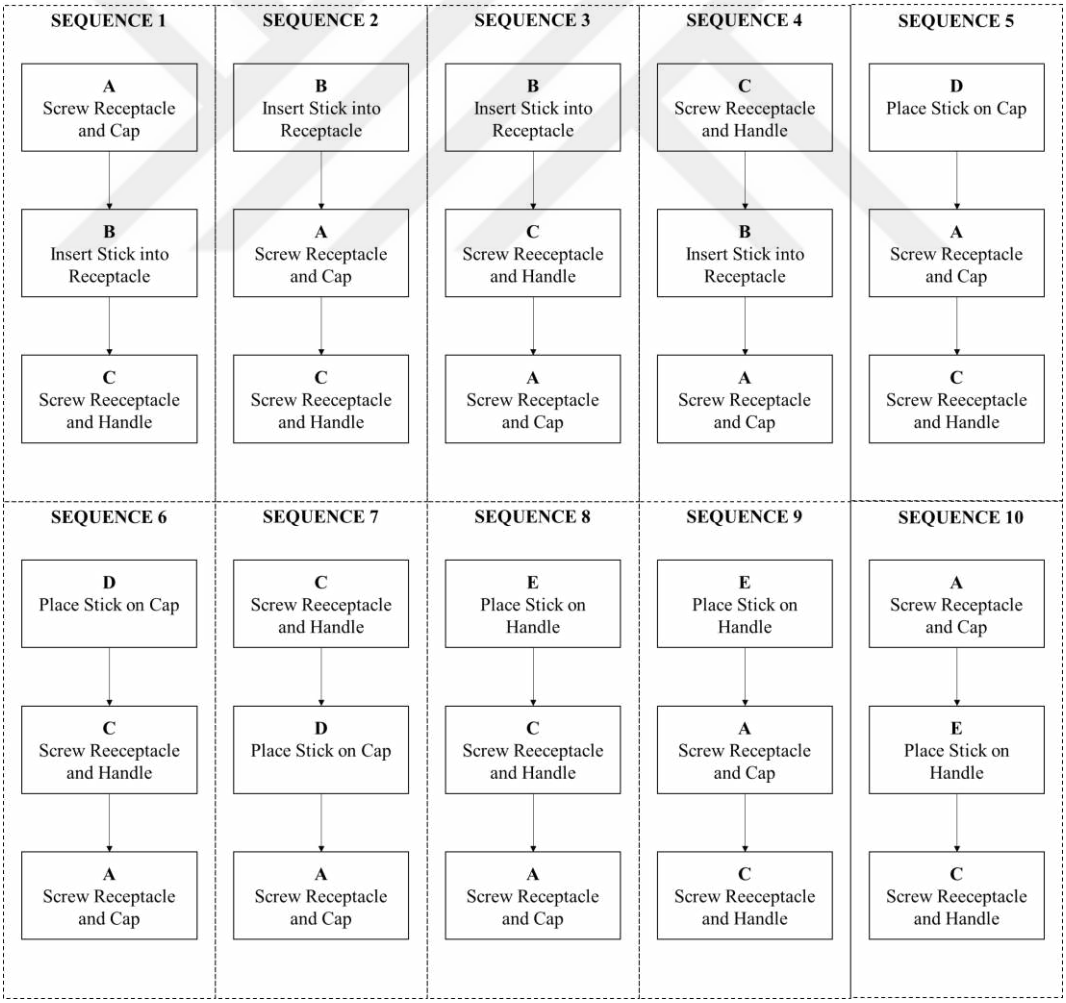


Figure 2.19. *Possible assembly sequences of simple product which given by Figure 2.18* (L.S. Homem de Mello & Sanderson, 1990)

In this way, an AOG precedence diagram given in Figure 2.20 is defined because it is difficult to follow all possible assembly sequences and is not suitable for the programming format.

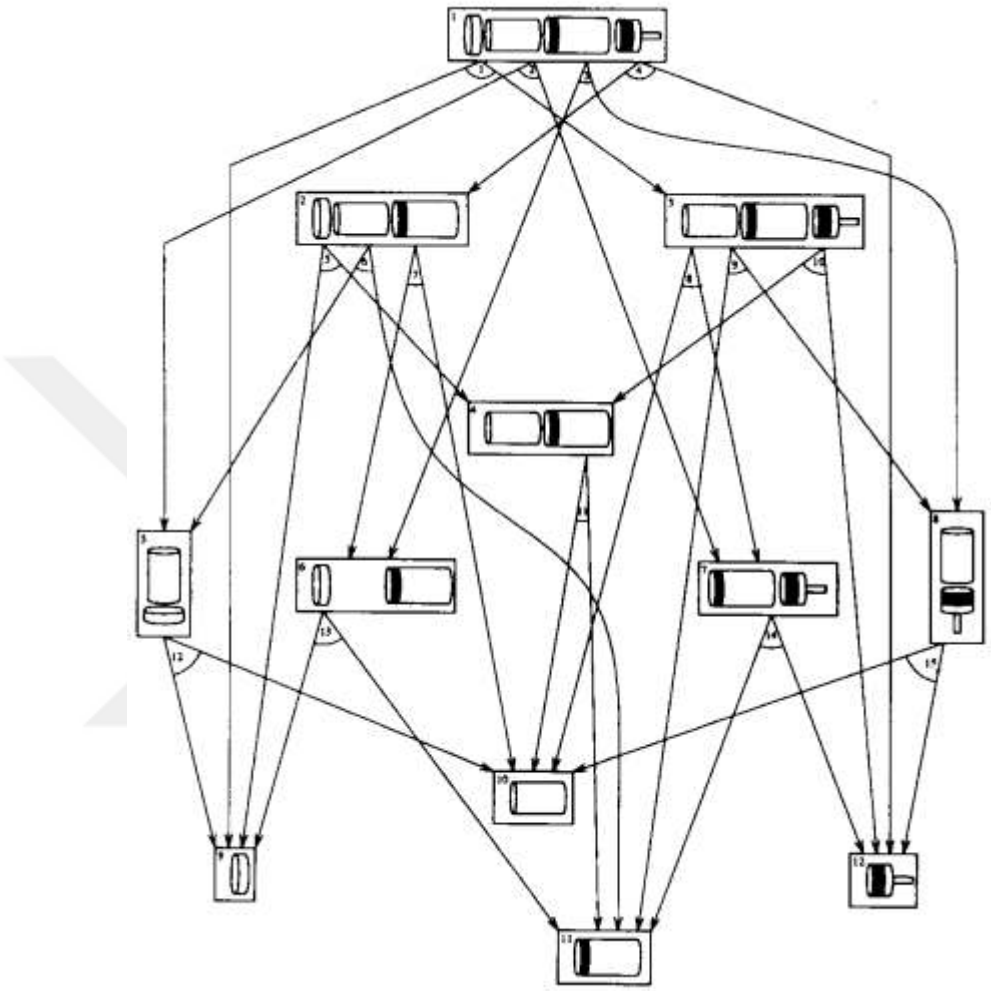


Figure 2.20. *AND/OR Graph of simple product which given by Figure 2.18* (L.S. Homem de Mello & Sanderson, 1990)

With the AOG precedence diagram, for a sample product shown in Figure 2.18, all assembly sequences shown in Figure 2.20 can be accessed. The assembly sequence of "Screw Receptacle and Handle – Inset Stick into Receptacle – Screw Receptacle and Cap" described as Sequence 4 in Figure 2.19, the steps to be reached from the AOG precedence diagram given in Figure 2.21. The assembly sequence of "Place Stick on Handle – Screw Receptacle and Handle – Screw Receptacle and Cap" is shown in Figure 2.22.

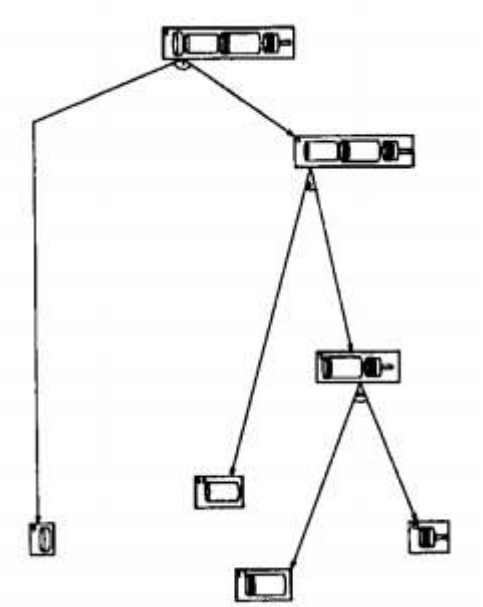


Figure 2.21. *A possible assembly sequence for AND/OR Graph of simple product which given by Figure 2.18*

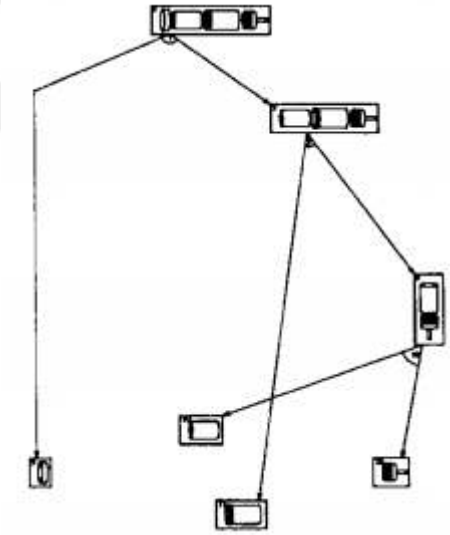


Figure 2.22. *A possible assembly sequence for AND/OR Graph of simple product which given by Figure 2.18*

The AOG precedence diagram is also adapted for DLB problems, as there are multiple disassembly sequences in DLB problems where functional constraints are unimportant and only physical constraints are taken into account. However, some flexibility, such as the lack of functionality in disassembly unlike assembly, has led to the differentiation of the AOG precedence diagram created. Moore (2001) and Altekin (2008) address these differences with the priority relations put forward. In this context, the

priority relations in the AOG precedence diagram used for solving DLB problems are listed as follows (Kalaycılar, 2012):

- AND Predecessor
- OR Predecessor
- Complex AND/OR Predecessor
- OR Successors
- AND within OR
- OR within AND

AND Predecessor

It is generally the most used relationship type for the AOG precedence diagram. In this relationship type, each of the AND Predecessor tasks of any task must be performed. Moore (2001) first described it for DLB problems. The AND Predecessor relationship is as shown in Figure 2.23. In this kind of relationship, it is emphasized that both tasks 2 and 3 must be done in order for task number 1 to be performed.

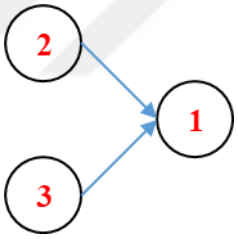


Figure 2.23. *AND Predecessor relation*

OR Predecessor

Another most commonly used relationship type for the AOG precedence diagram is the "OR Predecessor" relationship. For a task with this relationship, at least one of the tasks connected with this relationship must be performed before it. Moore (2001) first described it for DLB problems. A typical "OR Predecessor" relationship is as described in Figure 2.24. Here, at least one of the tasks numbered 2 or 3 must be completed in order for task number 1 to be completed.

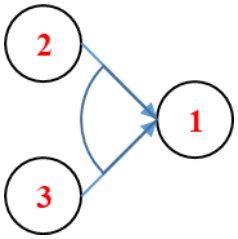


Figure 2.24. *OR Predecessor relation*

Complex AND/OR Predecessor

Another type of relationship used for the AOG precedence diagram is the “Complex AND/OR Predecessor” relationship. For a task with this relationship, all AND-bound tasks must be performed before it, and at least one of all OR-bound tasks must be performed. Moore (2001) first described it for DLB problems. A typical “Complex AND/OR Predecessor” relationship is as described in Figure 2.25. Here, in order to complete the task number 1, task number 2 and at least one of the tasks 3 and 4 must be completed.

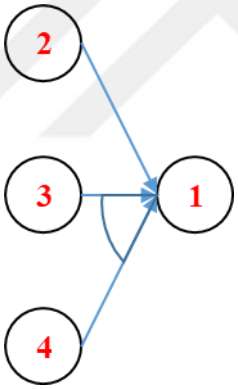


Figure 2.25. *Complex AND/OR Predecessor relation*

OR Successor

The most commonly used relationship type in the disassembly sequence selection for DLB problem solving is the "OR Successor" relationship. A task with this relationship can only do one of the tasks connected to it with this relationship after itself. Moore (2001) first described it. A typical “OR Successor” relationship is as described in Figure 2.26. Here, after task 1 is done, only one of tasks 2 or 3 will be able to be done.

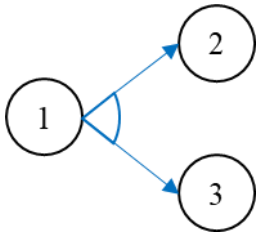


Figure 2.26. *OR Successor relation*

AND within OR

Another type of relationship used for the AOG precedence diagram is the "AND within OR" relationship. In this relationship, after two or more tasks connected with AND predecessor are completed, only one task connected with OR Successor can be completed. A typical AND within OR relationship is as described in Figure 2.27. Here, after completing the task 1 and 2, only one (at least and at most) of the tasks 3 and 4 be completed.

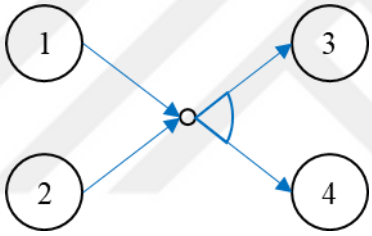


Figure 2.27. *AND within OR relation*

OR within AND

Another type of relationship used for the AOG precedence diagram is the "OR within AND" relationship. In this relationship, after one of two or more tasks connected with OR predecessor is completed, two or more tasks connected with AND Predecessor can be completed. A typical "OR within AND" relationship is as described in Figure 2.28. Here, after completing one of the tasks 1 and 2, the tasks 3 and 4 can be completed.

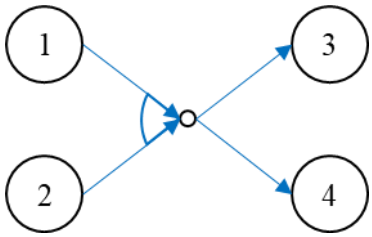


Figure 2.28. *OR within AND relation*

Transformed AND/OR Graph (TAOG) based precedence relation

Although AOG provides access to all possible disassembly sequences, it is not suitable for mathematical models due to computational difficulties. Therefore, another method, Transformed AND/OR Graph (TAOG) based Precedence Diagram, which is very similar to AOG, was developed using normal node (task) and artificial node. The TAOG based Precedence Diagram for the DLB problem was first published in 2009 by Koç et al. As a typical usage, TAOG is based on selecting only one normal node after each artificial node and selecting all artificial nodes that go after each normal node. Therefore, there is an OR Successor relationship between artificial nodes and normal nodes, and an AND Successor relationship between normal nodes and artificial nodes. However, in some studies, an AND Successor relationship is observed between the artificial node and the normal node (Bentaha et al., 2013).

A typical TAOG based Precedence Diagram with an OR Successor relationship between an artificial node and a normal node and an AND Successor relationship between a normal node and an artificial node is given in Figure 2.29. There are 23 normal nodes (disassembly task) and 13 artificial nodes in total here. Starting from the artificial node A_0, the process continues until only one normal node is selected from each OR Successor relationship, and there are no normal and artificial nodes to be selected. The normal nodes selected at the end are used as the priority relationship.

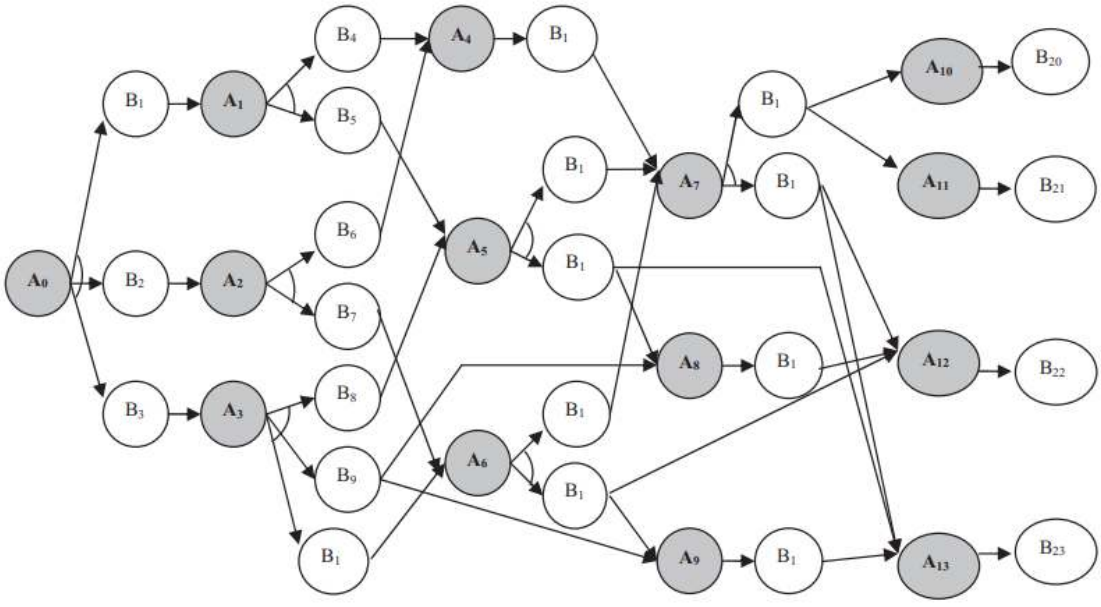


Figure 2.29. *A typical Transformed AND/OR Graph (TAOG) based precedence diagram*

The first Integer Linear Programming (ILP) mathematical model developed by Koç et al in 2009, which reached the optimum result using the TAOG based Precedence Diagram method for the Single-Model Straight DLB (SSDLB) problem. Thanks to this model, the best line balancing is achieved by considering all possible disassembly sequences. This model also forms the basis for the Mixed-Model Two-Sided DLB (MTDLB) problem mathematical model developed within the scope of the thesis.

Indices and Sets:

$k \in \{0, \dots, h\}$	Index of artificial nodes in AOG.
$i \in \{1, \dots, I\}$	Index of normal nodes (tasks) in AOG.
$j \in \{1, \dots, J\}$	Index of stations.
$P(k, i)$	Immediate predecessor set of k, i, respectively.
$S(k, i)$	Immediate successor set of k, i, respectively.

Scalars and Parameters:

C	Cycle time of disassembly line.
t_i	Task time of normal node (task) i.

Decision and Auxiliary Variables:

$x_{ij} \in \{0,1\}$	If normal node i is assigned to station j, 1, otherwise 0
$f_j \in \{0,1\}$	If station j is opened, 1, otherwise 0
$z_i \in \{0,1\}$	If normal nodel (task) i is performed, 1, otherwise 0

Objective Function:

$$objective = \boldsymbol{Min} \sum_{j=1}^{M} j \times f_j \tag{2.1}$$

The objective function defined by Equation 2.1 aims to minimize the number of opened workstations. The j index is multiplied by the work because it is desired to open the workstations in order. In this way, stations will start to open starting from the smallest index.

Constraints:

$$\sum_{i \in S(k,i)} z_i = 1 \qquad \forall k \in \{0\} \qquad (2.2)$$

$$\sum_{i \in S(k,i)} z_i = \sum_{i \in P(k,i)} z_i \qquad \forall k \in \{1, \dots, h\} \qquad (2.3)$$

$$\sum_{j=1}^{M} x_{ij} = z_i \qquad \forall i \in \{1, \dots, I\} \qquad (2.4)$$

$$\sum_{i \in P(k,i)} \sum_{j=1}^{v} x_{ij} = \sum_{i \in S(k,i)} x_{iv} \qquad \forall k \in \{1, \dots, h\}, \forall v \in \{1, \dots, M\} \qquad (2.5)$$

$$\sum_{i=1}^{I} x_{ij} \times t_i = C \times f_j \qquad \forall j \in \{1, \dots, M\} \qquad (2.6)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in \{1, \dots, I\}, \forall j \in \{1, \dots, M\} \qquad (2.7)$$

$$z_i \in \{0,1\} \qquad \forall i \in \{1, \dots, I\} \qquad (2.8)$$

$$f_j \in \{0,1\} \qquad \forall j \in \{1, \dots, M\} \qquad (2.9)$$

The constraints given by Equation 2.2 and 2.3 are the constraints that allow only one of the OR Successors to be selected to ensure that only one disassembly sequence is selected from the TAOG. With Equation 2.2, when the artificial node in the TAOG is equal to zero, that is, in the initial state, only one OR Successor is selected, the first step is taken to determine the disassembly sequences. With Equation 2.3, it explains that if one of the predecessors of artificial nodes is selected in TAOG, one of the OR successors should also be selected. Thus, line balancing is achieved by choosing the best disassembly sequence in accordance with the objective function. The constraint given by Equation 2.4 defines that if a task is selected, that task must be assigned to an absolute station. This is the updated format of the constraint of each task assigned to a station in ALB problems, using TAOG. The constraint given in Equation 2.5 defines the assignment according to the priority relationship between the two selected tasks. The predecessor of any task must be assigned to the same or previous station. It is the updated format of the priority constraint in ALB problems using TAOG. The constraint given by Equation 2.6 explains that the sum of the durations of the tasks assigned to any station cannot exceed the cycle time if the station is opened. It is similar to the cycle time constraint found in ALB

problems. The constraints given by Equation 2.7 – 2.9 determine the data type of decision variables and auxiliary variables.

An Illustrative Example:

The mathematical model of the DLB problem using TAOG, which was defined by Koc et al in 2009 and explained above, has been used by many researchers in the literature or transformed into different formats. This mathematical model is critical in terms of reaching the optimum result by considering all possible disassembly sequences. Therefore, this mathematical model forms the basis of the MTDLB problem developed within the scope of this thesis. In order to better understand the working logic of this mathematical model, an example do Koc et al create designed using the TAOG (given in Figure 2.29) in 2009, the task periods given in Table 2.2 and the cycle time being 10.

The SSDLB mathematical model created by Koc et al in 2009 was coded with LINGO 9.0 and given with Appendix-01. The resulting best disassembly sequence is as given by Figure 2.30 and Table 2.3 gives the best DLB solution.

Table 2.2. *Disassembly tasks and processing times of Figure 2.24 for illustrative TAOG example*

Task i	t_i	Task i (continued)	t_i
1	3	13	8
2	9	14	3
3	5	15	3
4	7	16	8
5	3	17	8
6	10	18	6
7	4	19	6
8	4	20	6
9	5	21	7
10	6	22	8
11	6	23	3
12	7		

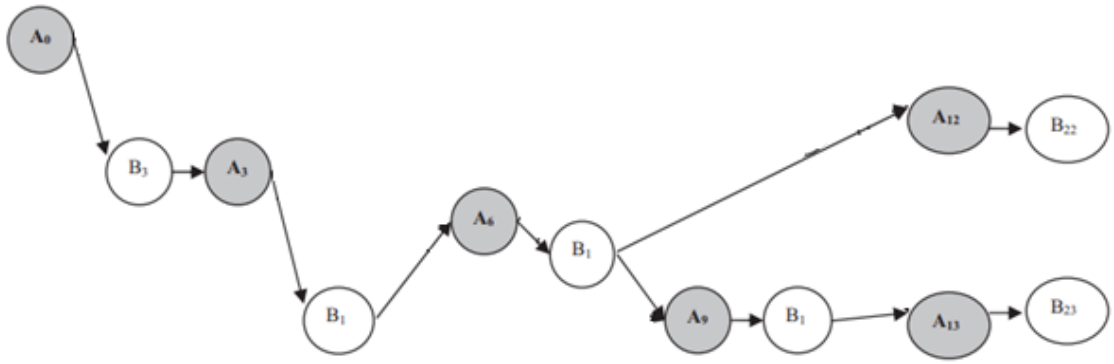


Figure 2.30. *Optimal disassembly sequence for illustrative TAOG example*

Table 2.3. *Optimal disassembly line assignment for illustrative TAOG example*

Stations	Assigned Tasks
1	3
2	10, 15
3	19, 23
4	22

2.1.5. Disassembly Level for DLB problem

DLB problems are divided into disassembly levels as complete and partial. Complete disassembly is the type in which all parts of the product are disassembled and all tasks in the precedence relation diagram are performed in sequence. Partial disassembly, on the other hand, is the type in which only certain parts of the product are disassembled, adhering to the precedence relation for profit.

2.2. Parameter Types

The types of parameters used in defining and solving the DLB problem are similar to the parameter types used in the ALB problem. It is collected in four main classes in total: Deterministic, where all parameters are considered to be completely known, Stochastic, which is considered to fit a certain statistical distribution, and Fuzzy, which is accepted to be valid at certain intervals, that is, has a membership function. The classification scheme is as given in Figure 2.31.

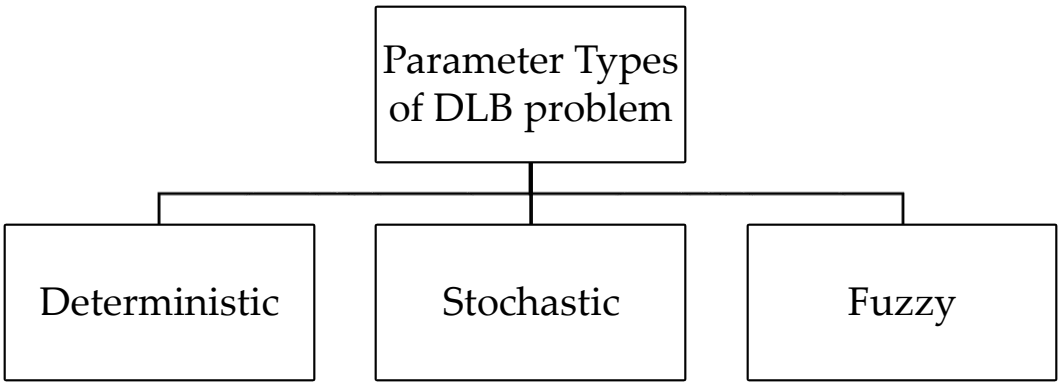


Figure 2.31. *Parameter types scheme of DLB problem*

The main differences between the ALB problem and the DLB problem are given in Table 2.1. Here, it is seen that the uncertainty is higher in the DLB problem compared to the ALB problem. The degree of damage to the incoming products and possible errors that may occur during disassembling increase the uncertainty for the DLB problem. For this reason, the use of stochastic parameters for the DLB problem in the literature (see. (Bentaha et al., 2013; Ming et al., 2019; Tuncel et al., 2014)) moreover, the use of fuzzy parameters (see. (Kalayci et al., 2015; Seidi & Saghari, 2016; Zhang et al., 2017)) are quite common. However, the number of DLB problem studies in which the parameters are deterministic under certain assumptions in terms of modeling and computational convenience is the highest (Özceylan et al., 2019). In addition to all these, there are studies in the literature where some parameters are deterministic, some parameters are stochastic, and some parameters are fuzzy.

2.3. Objective Functions

The selection of the performance indicator to be optimized plays a critical role as well as the definition of the DLB problem. There are a wide variety of performance indicators used in DLB problem research in the literature. In some studies, more than one conflicting performance indicator can be used instead of one performance indicator. Such studies are called multi-objective. Classification according to objective function type is as given in Figure 2.32.

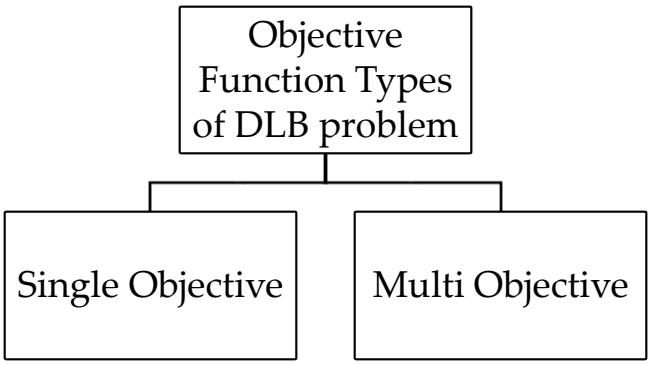


Figure 2.32. *Objective function types scheme of DLB problem*

Some objective functions that are widely used in the literature are as given in Figure 2.33.

Mostly Used Objective Functions of DLB problem	
	Number of Workstations / Idle Time / Efficiency
	Cycle Time
	Profit / Revenue / Demand
	Balance / Smoothness
	Hazardous Parts
	Direction
	Inventory Level
	Space Allocation
	Disassembly Cost / Disposal Cost
	Total Disassembly Operations Time
	Total Failing Task Cost
	Shipping Cost

Figure 2.33. *Mostly used objective functions of DLB problem*

Some of the frequently used objective functions for the DLB problem are given below. For all other objective functions, literature research studies can be looked at (see. (Özceylan et al., 2019)).

Number of Workstations

The most used objective function in the DLB literature is Number of Workstation. The goal is to perform certain disassembly tasks on the fewest number of workstations. Although it is modeled in different formats according to the way the problem is handled, when W refers to the number of workstations, it is usually modeled as in Equation 2.10.

$$\boldsymbol{Minimize}\ f_1 = W \tag{2.10}$$

Smoothness

Another objective function most used in the DLB literature is the Smoothness Index. The aim is to reduce the idle time of the workstations as much as possible and to establish a balance between the workstations.

Bir DL sisteminde, Cycle Time (CT) ve her bir $j \in \{1, \dots, W\}$ iş istasyonun çalışma süresi (WT_j) bilindiğinde, her bir $j \in \{1, \dots, W\}$ iş istasyonu için Idle Time (IT_j) Equation 2.11'deki gibi hesaplanır. Bu aynı zamanda istasyon sayısının en küçüklenmesi ile aynı anlamı taşımaktadır.

In a DL system, given the Cycle Time (CT) and the uptime (WT_j) of each $j \in \{1, \dots, W\}$ workstation, the Idle Time (IT_j) for each $j \in \{1, \dots, W\}$ workstation is known. IT_j is calculated as in Equation 2.11. This also has the same meaning as minimizing the number of stations.

$$IT_j = CT - WT_j \qquad \forall j \in \{1, \dots, W\} \tag{2.11}$$

In a DL system, the sum of the Idle Time(IT_j) squares of each $j \in \{1, \dots, W\}$ workstation must be minimized in order to ensure smoothness between workstations, that is, for the DL system to be Smooth. The modeling generally used in DLB studies aiming to minimize the Smoothness Index is as given in Equation 2.12.

$$\boldsymbol{Minimize}\ f_2 = \sum_{j=1}^{W} {IT_j}^2 = \sum_{j=1}^{W} \left(CT - WT_j\right)^2 \tag{2.12}$$

2.4. Solution Methods

Due to its nature, the DLB problem is included in the NP-Hard class. In other words, when the size of the case to be solved increases, the solution becomes impossible after a

certain case size, since the problem solution space increases exponentially. When an example of the NP-Hard class is given through the Traveling Salesman Problem (TSP), the number of alternative tours for the 5-city TSP is 5 factorial, ie 120 alternatives, while the number of alternative tours for the 10-city TSP is 10 factorial, ie 3628800, and for the 20-city TSP, 2432902008176640000 alternative tours. Therefore, even if the distance calculation for each round takes 1 millisecond for a TSP problem with 20 cities, distance calculations can be made for all alternative solutions in 77146816.5962 years. In this and similar cases, heuristic/metaheuristic algorithms that give very close results are applied, although it is not known for sure that the optimum solution is. Sometimes, different transformations are made for the problem and a solution is sought with exact methods. Therefore, there is no limit to the solutions for the defined DLB problem. Within the scope of this study, the literature was scanned and the solution approaches for the DLB problem were classified as given in Figure 2.5.

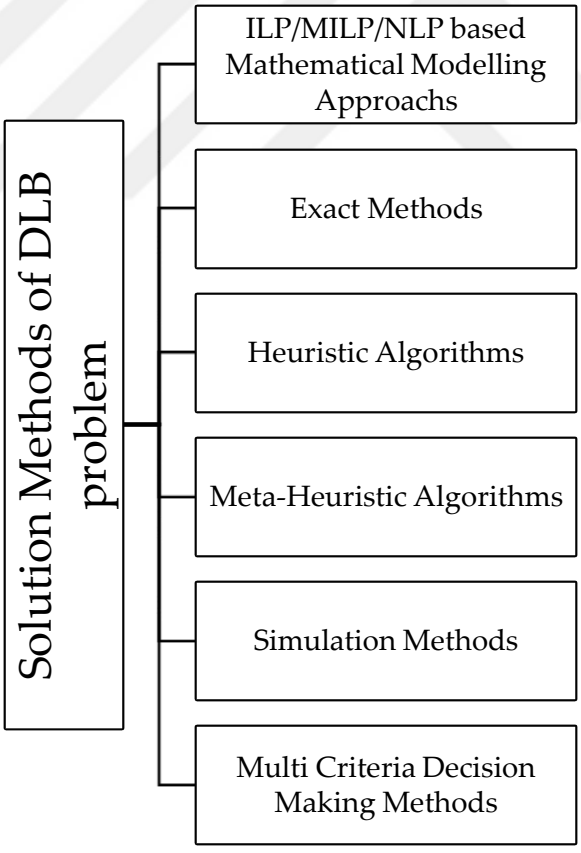


Figure 2.34. *Solution methods scheme of DLB problem*

Integer Linear Programming (ILP) / Mixed Integer Linear Programming (MILP) / Non Linear Programming (NLP) based mathematical programming approaches are based

on solving the mathematical model developed for the problem using commercial software or coding the Branch-and-Bound algorithm. Since this approach guarantees the optimum solution, the method should be tried first.

Exact methods are based on solving the problem with certain algorithms by relaxing or remodeling under certain assumptions in cases where a solution cannot be obtained with ILP/MILP/NLP based mathematical programming approaches. A typical example is searching for a solution with the Branch-and-Cut algorithm, which is the relaxed version of the Branch-and-Bound algorithm.

Heuristic algorithms are preferred in cases where solutions cannot be obtained with ILP/MILP/NLP based mathematical programming approaches or for problems that cannot be mathematically modeled. These methods, which are based on a specific procedure, give the solution reached as the best solution when the stopping criterion ends. It can be developed according to a predefined procedure or according to the problem structure.

Metaheuritic algorithms are frequently encountered solution methods in cases where solutions cannot be obtained with ILP/MILP/NLP based mathematical programming approaches. These algorithms allow optimization based on certain systems found in nature. For example, the Genetic Algorithm (GA) has emerged based on the theory of evolution found in nature. GA is a method of changing the coded solutions with Selection, Crossover and Mutation in each generation, resulting in new generations, and thus reaching the optimum solution in a short time by intelligently scanning the solution space. This and similar (Tabu Search algorithm, Particle Swarm Optimization algorithm, Artificial Bee Colony algorithm etc.)

Generally, when the parameters are stochastic, simulation method is used to solve the DLB problem. Simulation is the repetitive testing of a system by modeling it in a computer environment. In this way, statistical information can be provided to the decision maker by finding average results because of stochastic parameters. Another method used to solve the DLB problem is Multi Criteria Decision Making (MCDM) methods. With this method, possible scenarios are designed for the solution of the problem and it is based on the selection of the best scenario by considering multiple criteria. In some multi-objective studies, it is also used in the form of choosing the most appropriate scenario by

evaluating the undecided solution scenarios by multi-criteria.In addition to all these methods, some matheuristic methods have been used. This method is a hybrid of the exact method and metaheuristic approaches. The problem is relaxed and the optimum solution is sought, and the solution result is given as an input to the metaheuristic algorithm, where the rest of the problem is searched. The output found in the metaheuristic result is given as an input to the exact method and the process continues like this. Although there are not many matheuristic approaches for the DLB problem in the literature, the effectiveness of matheuristic approaches will be a good tool to be used for solving the DLB problem in the future.

3. LITERATURE REVIEW

Güngör and Gupta first described the DLB problem in 1999 (Güngör & Gupta, 1999b). McGovern and Gupta proved interest in the DLB problem has increased after it in 2007 that the DLB problem belongs to the NP-Hard class (McGovern & Gupta, 2007b). Today, the DLB problem is investigated and solved in a wide variety of ways. In this section, a detailed literature search has been made considering the DLB problem classification given in Section 2.

The literature for the DLB problem was searched using Web of Science and Scopus databases. The search was carried out with "disassembly line balancing" in any part of the study in the literature. In this context, 147 studies, the oldest of which were published in 2001 and the latest in 2022, were found in the Web of Science database, and 217 studies were found in the Scopus database, the oldest of which was published in 2001 and the newest in 2022. The status of these studies being included in the databases, in turn, is as given in Table 3.1.

Table 3.1. *All DLB studies in literature*

Year	Cite	Conference Proceedings	Journal Article	Web of Science	Scopus
2001	(Güngör et al., 2001)	✓		✓	✓
	(Gupta & Güngör, 2001)	✓		✓	✓
	(Tang et al., 2001)	✓		✓	✓
	(Güngör & Gupta, 2001)		✓	✓	✓
2003	(McGovern & Gupta, 2003)	✓		✓	✓
2004	(McGovern & Gupta, 2004b)	✓		✓	✓
	(Altekin et al., 2004)	✓		✓	✓
	(McGovern & Gupta, 2004a)	✓			✓
	(Kizilkaya & Gupta, 2004)	✓		✓	✓

	(McGovern & Gupta, 2004c)	✓			✓	✓
2005	(Duta et al., 2005)	✓				✓
	(McGovern & Gupta, 2005a)			✓		✓
	(Kizilkaya & Gupta, 2005)	✓				✓
	(Lambert & Gupta, 2005a)	✓				✓
	(Lambert & Gupta, 2005b)	✓				✓
	(McGovern & Gupta, 2005b)	✓				✓
	(Prakash & Tiwari, 2005)	✓			✓	✓
	(Turowski et al., 2005)	✓			✓	✓
2006	(Johar & Gupta, 2006)	✓			✓	✓
	(McGovern & Gupta, 2006a)	✓			✓	✓
	(McGovern & Gupta, 2006b)			✓	✓	✓
	(McGovern & Gupta, 2006c)			✓	✓	✓
2007	(Lambert, 2007)	✓				✓
	(McGovern & Gupta, 2007a)	✓				✓
	(McGovern & Gupta, 2007b)			✓	✓	✓
	(McGovern & Gupta, 2007c)			✓	✓	✓
2008	(Agrawal & Tiwari, 2008)			✓	✓	✓
	(Altekin et al., 2008)			✓	✓	✓
	(Duta et al., 2008)	✓				✓
2009	(Ding et al., 2009)			✓		✓

	(Koc et al., 2009)		✓	✓	✓
2010	(Ding et al., 2010)		✓	✓	✓
2011	(Kalayci & Gupta, 2011)	✓			✓
2013	(Aydemir-Karadag & Turkbey, 2013)		✓	✓	✓
	(Bentaha et al., 2013a)	✓			✓
	(Bentaha et al., 2013b)	✓			✓
	(Bentaha et al., 2013c)	✓			✓
	(Bentaha et al., 2013d)	✓		✓	✓
	(Bentaha et al., 2013e)	✓		✓	✓
	(Kalayci & Gupta, 2013a)		✓	✓	✓
	(Kalayci & Gupta, 2013b)	✓			✓
	(Kalayci & Gupta, 2013c)		✓		✓
	(Kalayci & Gupta, 2013d)		✓	✓	✓
	(Kalayci & Gupta, 2013e)		✓	✓	✓
	(B. Y. Liu et al., 2013)	✓			✓
	(Özceylan & Paksoy, 2013)		✓	✓	✓
	(Paksoy et al., 2013)		✓	✓	✓
2014	(Avikal, Jain, & Mishra, 2014)		✓	✓	✓
	(Avikal, Jain, Yadav, et al., 2014)	✓		✓	✓
	(Avikal, Mishra, et al., 2014)		✓	✓	✓

	(Bentaha et al., 2014a)	✓		✓	✓
	(Bentaha et al., 2014b)	✓			✓
	(Bentaha et al., 2014c)	✓		✓	✓
	(Bentaha et al., 2014d)		✓	✓	✓
	(Habibi et al., 2014)	✓		✓	✓
	(Igarashi, Yamada, & Inoue, 2014)		✓		✓
	(Igarashi, Yamada, Itsubo, et al., 2014)		✓		✓
	(Kalayci & Gupta, 2014)		✓	✓	✓
	(Minca et al., 2014)		✓	✓	✓
	(Özceylan & Paksoy, 2014a)		✓	✓	✓
	(Özceylan & Paksoy, 2014b)		✓	✓	✓
	(Özceylan et al., 2014)		✓	✓	✓
	(Tuncel et al., 2014)		✓	✓	✓
	(X. Zhu et al., 2014)		✓		✓
2015	(Bentaha et al., 2015)		✓	✓	✓
	(Hezer & Kara, 2015)		✓	✓	✓
	(Kalayci, Hancilar, et al., 2015)		✓	✓	✓
	(Kalayci, Polat, et al., 2015)		✓	✓	✓
	(McGovern & Gupta, 2015)		✓	✓	✓
	(Süleyman Mete et al., 2015)	✓			✓

	(Riggs et al., 2015)		✓	✓	✓
2016	(Altekin, 2016)	✓		✓	✓
	(Avikal et al., 2016)		✓		✓
	(Avikal, 2016)		✓		✓
	(Duta et al., 2016)	✓		✓	✓
	(Filipescu et al., 2016)	✓		✓	✓
	(Hao & Hasan, 2016)		✓		✓
	(Igarashi et al., 2016)		✓	✓	✓
	(Kalaycılar et al., 2016)		✓	✓	✓
	(Kalayci et al., 2016)		✓	✓	✓
	(Süleyman Mete, Çil, Ağpak, et al., 2016)		✓	✓	✓
	(Süleyman Mete, Çil, Özceylan, et al., 2016)	✓		✓	✓
	(Seidi & Saghari, 2016)		✓	✓	✓
	(Z. Zhang et al., 2016)		✓		✓
2017	(Altekin, 2017)		✓	✓	✓
	(Ilgin et al., 2017)		✓	✓	✓
	(Jia Liu & Wang, 2017)		✓	✓	✓
	(Jia & Shuwei, 2017)	✓		✓	✓
	(Kannan et al., 2017)		✓	✓	✓
	(L. Li et al., 2017)		✓		✓
	(Süleyman Mete et al., 2017)	✓			✓
	(Qiang et al., 2018)	✓		✓	

	(Ren et al., 2017)		✓	✓	✓
	(K. Wang, Zhang, Mao, et al., 2017)		✓		✓
	(K. Wang, Zhang, Zhu, et al., 2017)		✓		✓
	(S. Xiao et al., 2017)		✓	✓	✓
	(Z. Zhang et al., 2017)		✓	✓	✓
	(Zou et al., 2017)		✓		✓
2018	(Battaïa et al., 2018)		✓	✓	✓
	(Bentaha et al., 2018)		✓	✓	✓
	(Y. Gao et al., 2018)		✓	✓	✓
	(Jia Liu & Wang, 2018)		✓		✓
	(Jiayi Liu et al., 2018)		✓	✓	✓
	(Kazancoglu & Ozturkoglu, 2018)		✓	✓	✓
	(L. Zhu et al., 2018)		✓	✓	✓
	(L. Li, Zhang, Guan, et al., 2018)		✓		✓
	(L. Li, Zhang, Zhu, et al., 2018)		✓		✓
	(Liuke et al., 2018)		✓		✓
	(Süleyman Mete et al., 2018)		✓	✓	✓
	(Ning et al., 2018)		✓		✓
	(Pistolesi et al., 2018)		✓	✓	✓
	(Qiang et al., 2018)	✓			✓
	(Ren et al., 2018)		✓	✓	✓
	(Xia et al., 2018)		✓		✓
	(Z. Zhang, Cai, et al., 2018)		✓		✓

	(Z. Zhang, Wang, Li, et al., 2018)		✓		✓
	(Z. Zhang, Wang, Zhu, et al., 2018)		✓		✓
	(Zheng et al., 2018)		✓	✓	✓
	(Zou, Zhang, Cai, et al., 2018)		✓		✓
	(Zou, Zhang, Li, et al., 2018)		✓		✓
2019	(B. Liu, Xu, Liu, et al., 2019)	✓		✓	✓
	(Cai, Zhang, Zhang, et al., 2019)		✓		✓
	(Cai, Zhang, Zou, et al., 2019)		✓		✓
	(Cao et al., 2019)		✓	✓	✓
	(Deniz & Ozcelik, 2019)		✓	✓	✓
	(Edis et al., 2019)		✓	✓	✓
	(Fang, Liu, et al., 2019)		✓	✓	✓
	(Fang, Wei, et al., 2019)	✓		✓	✓
	(Ilgin, 2019)		✓	✓	✓
	(J. Li, Chen, & Chu, 2019)	✓		✓	✓
	(J. Li, Chen, Zhu, et al., 2019)		✓	✓	✓
	(K. Wang, Li, & Gao, 2019a)		✓	✓	✓
	(K. Wang, Li, & Gao, 2019b)		✓	✓	✓
	(K. Wang, Li, Gao, et al., 2019)		✓	✓	✓
	(M. Liu, Liu, & Chu, 2019)	✓		✓	✓
	(M. Liu, Liu, Liu, et al., 2019)	✓		✓	✓

	(Ming et al., 2019)	✓		✓	✓
	(Özceylan et al., 2019)		✓	✓	✓
	(Q. Liu, Li, Fang, et al., 2019)	✓		✓	✓
	(Ren et al., 2019)	✓		✓	✓
	(S. Wang et al., 2019)		✓	✓	✓
	(X. Li, Laili, Zhang, et al., 2019)	✓			✓
	(Yang et al., 2019)		✓	✓	✓
	(Z. Li, Kucukkoc, & Zhang, 2019)		✓	✓	✓
	(Zeng et al., 2019)	✓			✓
	(Z. Zhang et al., 2019)		✓		✓
	(R. Zhou et al., 2019)	✓		✓	✓
2020	(Budak, 2020)		✓	✓	✓
	(C. Xu et al., 2020)	✓		✓	✓
	(Cevikcan et al., 2020)		✓	✓	✓
	(Q. Chen et al., 2020)	✓			✓
	(Çil et al., 2020)		✓	✓	✓
	(Diri Kenger et al., 2020)		✓	✓	✓
	(Fang & Xu, 2020)		✓		✓
	(Fang, Ming, et al., 2020)		✓	✓	✓
	(Fang, Xu, et al., 2020)		✓	✓	✓
	(Fang, Zhang, et al., 2020)	✓			✓
	(He et al., 2020)		✓	✓	✓
	(Jiayi Liu et al., 2020)		✓	✓	✓

(K. Wang, Gao, et al., 2020)		✓	✓	✓
(K. Wang, Li, et al., 2020)		✓	✓	✓
(K. Z. Gao et al., 2020)		✓	✓	✓
(Kaya et al., 2020)	✓		✓	✓
(Kazancoglu & Ozkan-Ozen, 2020)		✓	✓	✓
(Kucukkoc et al., 2019)		✓	✓	✓
(Kucukkoc, 2020)		✓	✓	✓
(L. Zhu, Zhang, & Guan, 2020)		✓	✓	✓
(L. Zhu, Zhang, Wang, et al., 2020)		✓	✓	✓
(Laili et al., 2020)		✓	✓	✓
(Luo et al., 2020)	✓			✓
(M. Liu et al., 2020)		✓	✓	✓
(Meng & Zhang, 2020)		✓	✓	✓
(Gui Bin Qin et al., 2020)	✓		✓	✓
(Ren et al., 2020)		✓	✓	✓
(T. Y. Wang et al., 2020)	✓		✓	✓
(Wu et al., 2020)	✓		✓	✓
(Xia et al., 2020)		✓	✓	✓
(Y. Xu et al., 2020)	✓			✓
(Ying Zhang et al., 2020)		✓		✓
(Y. Zhou et al., 2020)		✓	✓	✓
(Yuan et al., 2020)	✓		✓	✓
(Z. Li et al., 2020)		✓	✓	✓
(Z. W. Zhang et al., 2020)	✓		✓	✓

	(Zeng et al., 2020)		✓		✓
2021	(ÇİL, 2021)		✓	✓	✓
	(Dalle Mura et al., 2021)		✓	✓	✓
	(Diri Kenger et al., 2021)		✓	✓	✓
	(Edis, 2021)		✓	✓	✓
	(Goksoy Kalaycilar et al., 2021)		✓	✓	✓
	(GuiBin Qin et al., 2021)	✓			✓
	(He, Chu, Dolgui, et al., 2021)		✓	✓	✓
	(He, Chu, Zheng, et al., 2021)		✓	✓	✓
	(Hu et al., 2021)		✓	✓	✓
	(J. C. Chen et al., 2021)		✓	✓	✓
	(J. Liang et al., 2021)		✓	✓	✓
	(Jiang et al., 2021)		✓		✓
	(K. Wang, Li, Gao, Li, et al., 2021b)		✓		✓
	(K. Wang, Li, Gao, & Li, 2021a)		✓		✓
	(K. Wang, Li, Gao, & Li, 2021b)		✓		✓
	(K. Wang, Li, Gao, Li, et al., 2021a)		✓	✓	✓
	(Kanagaraj et al., 2021)		✓	✓	
	(L. Zhang et al., 2021)		✓	✓	✓
	(M. Liu et al., 2021)		✓	✓	✓
	(Mei & Fang, 2021)	✓			✓

	(Mutlu & Güner, 2021)	✓			✓
	(P. Liang et al., 2021)		✓	✓	
	(Q. Xiao et al., 2021)		✓	✓	✓
	(Suleyman Mete & Serin, 2021)	✓			✓
	(X. Liu et al., 2021)		✓	✓	✓
	(Xie et al., 2021)		✓		✓
	(Y. Wang et al., 2021)		✓		✓
	(Yılmaz & Yazıcı, 2021)		✓	✓	✓
	(Yin et al., 2021)		✓	✓	✓
	(Yolmeh & Saif, 2021)		✓	✓	✓
	(Yu Zhang et al., 2021)		✓	✓	✓
	(Z. Li & Janardhanan, 2021)		✓	✓	✓
2022	(Yin et al., 2022)		✓		✓

When we look at the DLB studies scanned in Web of Science and Scopus, it is seen that a record has been broken with 37 studies in 2020 and 32 studies have been published as of 2021, which we are currently in. When we look at the distribution of DLB studies based on years given in Figure 3.1, it can be deduced that DLB studies are given importance today.

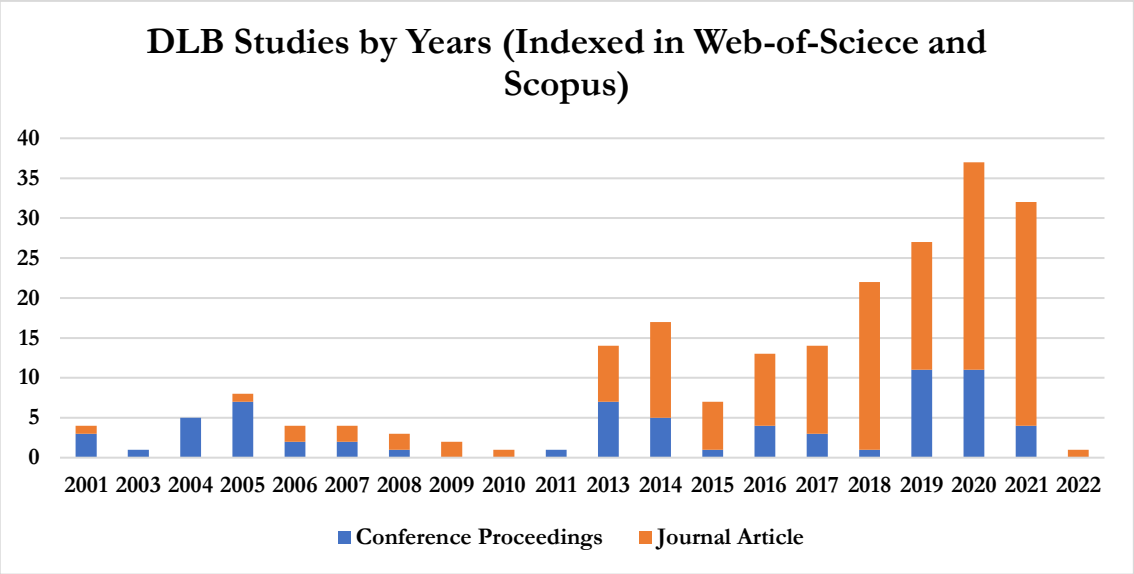


Figure 3.1. *DLB studies by years (indexed in web-of-science and scopus)*

91% of the DLB studies in the literature, that is, the majority of them, have the straight-line format. 4% of the DLB studies in the literature have U-shaped line, 3% two-sided line and 2% parallel line format. This statistic is visualized in Figure 3.2.

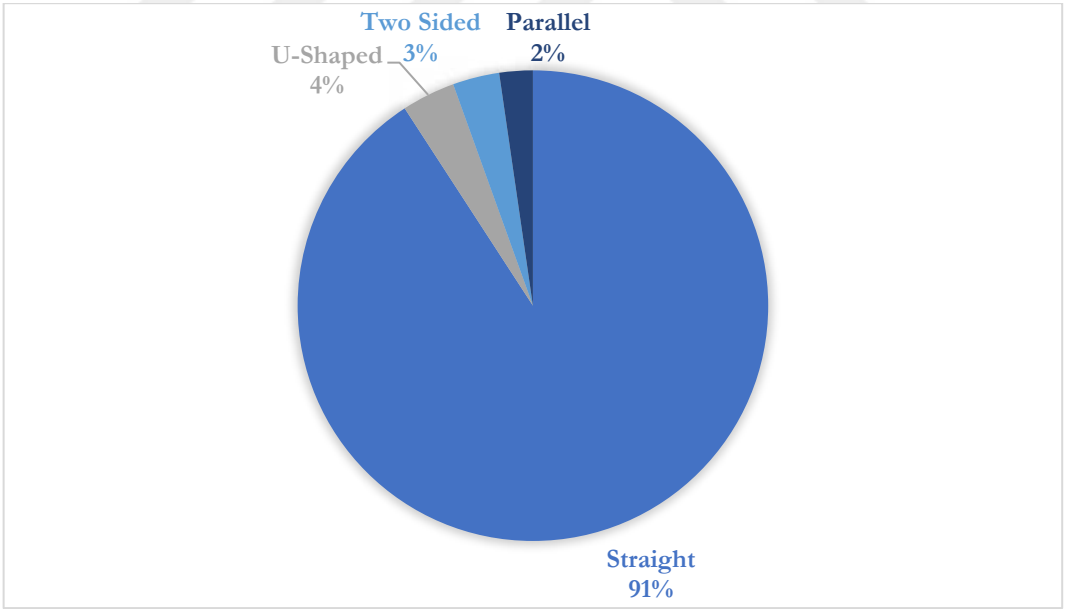


Figure 3.2. *Line layout types piechart for DLB studies in literature*

94% of the DLB studies in the literature, that is, the majority of them, have the single-model format. 0% of the DLB studies in the literature have multi-model line, 6% mixed-model format. This statistic is visualized in Figure 3.3.

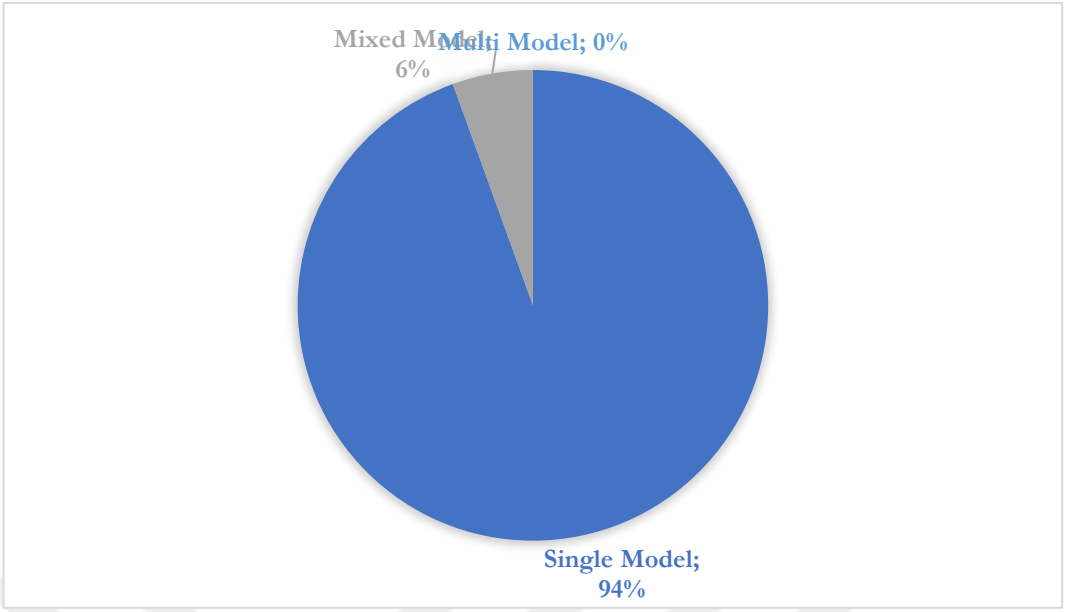


Figure 3.3. *Model types piechart for DLB studies in literature*

83% of the DLB studies in the literature, that is, the majority of them, have the completely disassembly format. 17% of the DLB studies in the literature have partially disassemblys format. This statistic is visualized in Figure 3.4.

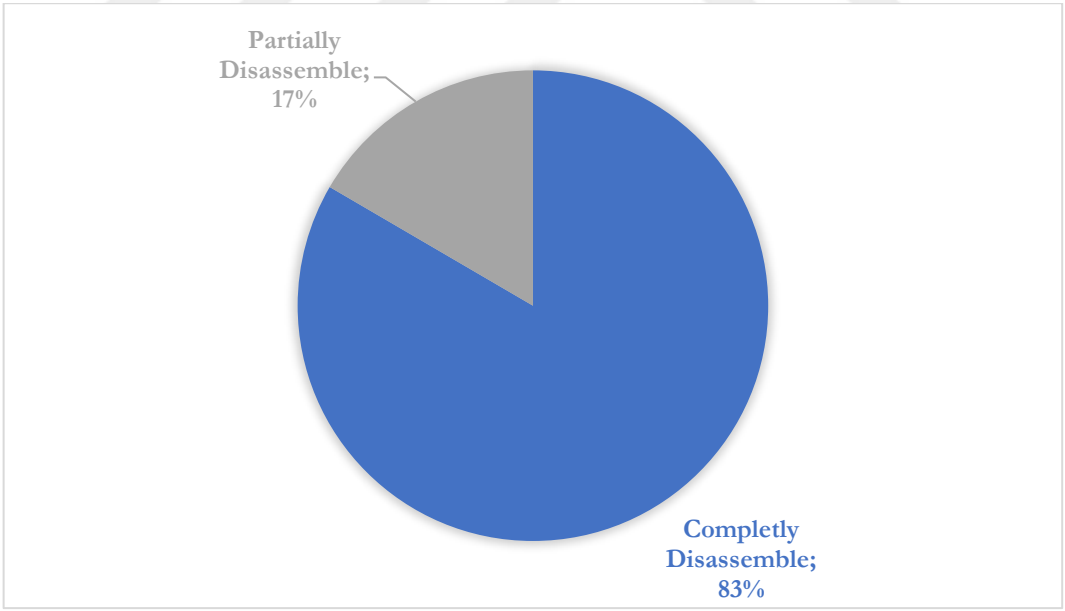


Figure 3.4. *Disassembly level piechart for DLB studies in literature*

61% of the DLB studies in the literature, that is, the majority of them, have the single objective format. 39% of the DLB studies in the literature have multi objective format. This statistic is visualized in Figure 3.5.

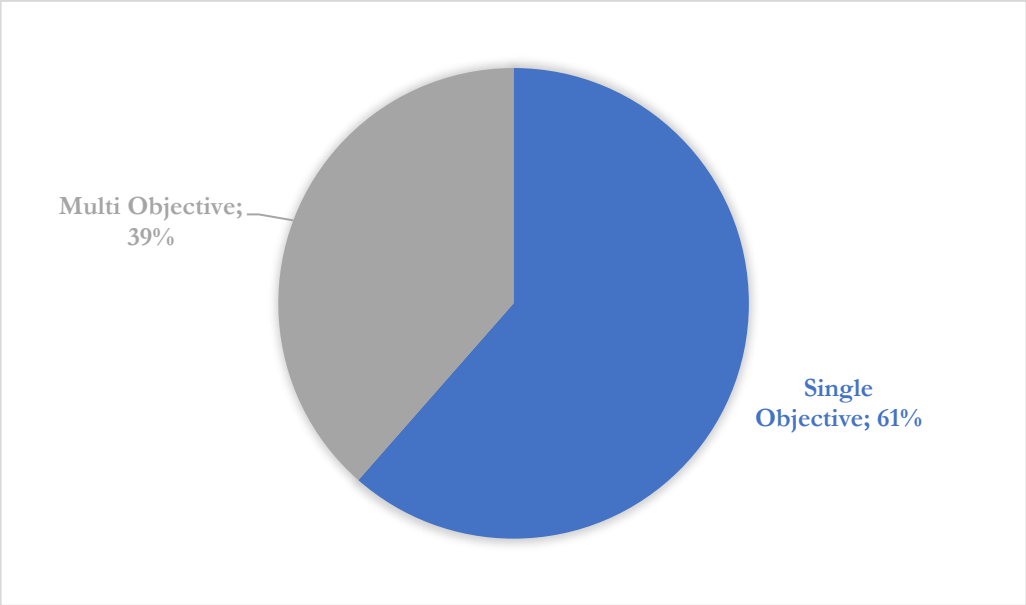


Figure 3.5. *Objective function type piechart for DLB studies in literature*

88% of the DLB studies in the literature, that is, the majority of them, have the human worker format. 11% of the DLB studies in the literature have robotic workers, 1% human-robot collabroation format. This statistic is visualized in Figure 3.6.

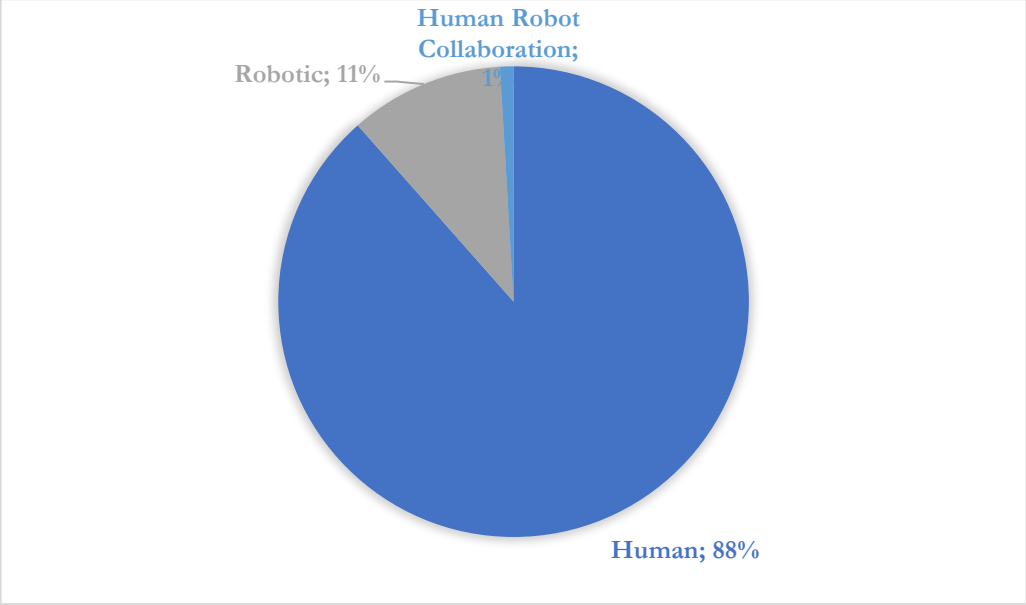


Figure 3.6. *Worker type piechart for DLB studies in literature*

There are only 7 studies in the literature for the TDLB problem. When we look at the distribution of TDLB studies based on years given in Figure 3.7, it can be deduced that DLB studies are given importance today.

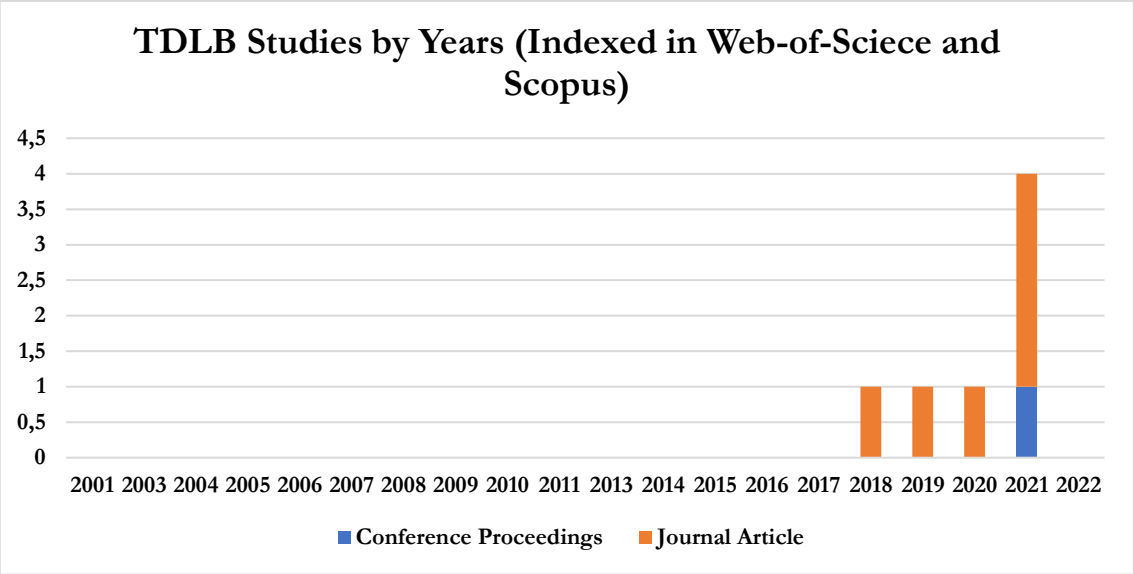


Figure 3.7. *TDLB studies by years (indexed in web-of-science and scopus)*

For the TDLB problem, the studies in the literature were examined in detail and each study was classified as given in Table 3.2. Looking at this table, it is seen that the MTDLB problem was handled for the first time in the study published within the scope of this study and this is the only study that used TAOG for the TDLB problem. It adequately explains the importance and originality of this work.

Table 3.2. *Classification of TDLB studies in literature*

Study	Model Type	Disassembly Level	Precedence Diagram Type	Objective Function Type	Solution Method
(Zou, Zhang, Li, et al., 2018)	Single	Completely	Traditional	Multi Objective	Pareto Bat Algorithm
(K. Wang, Li, Gao, et al., 2019)	Single	Partially	Traditional	Multi Objective	Discrete Flower Pollination Algorithm
(Kucukkoc, 2020)	Single	Completely	AOG	Single Objective	2 - Genetic Algorithm
(Mutlu & Güner, 2021)	Mixed	Completely	TAOG	Multi Objective (Scalar)	Memetic Algorithm
(Yu Zhang et al., 2021)	Single	Completely	Traditional	Single Objective	Improved Whale Optimization Algorithm
(J. Liang et al., 2021)	Single	Completely	Traditional	Multi Objective	Simulated Annealing Algorithm
(Xie et al., 2021)	Single	Completely	Traditional	Multi Objective	NSGA-II

4. MIXED-MODEL TWO-SIDED DISASSEMBLY LINE BALANCING PROBLEM

In line with the ever-increasing technological developments and affluence, the size of the products is also increasing, as CRT televisions turn into LCD / LED televisions. At the same time, thanks to the increasing technological developments, the use of airplanes weighing more than 285 tons in air transportation, ships with a weight of more than 600 thousand tons in sea transportation, and trucks that offer a transportation volume of up to 100 cubic meters on the road have become widespread. According to the published statistical data, approximately 3 million trucks are produced worldwide in 2019, and approximately 79 million refrigerators are produced only in China in 2019 (Statista, 2020a, 2020b). Considering that the average life of the products is 10-15 years and the maximum life is 25-30 years on average, when large-size products complete their life and become idle, that is, when they become EOL products, a high amount of resources will be idle and cause a high amount of pollution for nature. The recycling of these products is of great importance both financially and environmentally, and its importance is increasing day by day.

The Two-Sided Assembly Line system has been developed in order to save time and cost for high-volume production of large-sized products and to prevent unnecessary movements such as rotation movements, and its applications in the literature (see. (Kim et al., 2009; Özcan & Toklu, 2009; Roshani et al., 2012; Tapkan et al., 2012)) provides very efficient production. Similarly, Two-Sided Disassembly Line systems are needed for disassembly of same/similar products and it is thought that there will be more requirements in the future. Indeed, a MILP mathematical model for the Single-Model Two-Sided Disassembly Line Balancing (STDLB) problem, in which only a strict disassembly sequence is taken into account using the Traditional Precedece Diagram and investigated as partial disassembly, has been developed by Wang et al. It was developed by Wang et al in 2019 (Wang et al., 2019b). After this study, 7 more studies dealing with the TDLB problem were presented to the literature. When we look at the studies in the literature for the TDLB problem, TAOG precedence diagram and mixed-model formation were used only in the study that emerged within the scope of this thesis (Mutlu & Güner, 2021). There is great clarity in the literature for MTDLB problems using the TAOG precedence diagram, since the TAOG precedence diagram takes into account all possible disassembly sequences and it is necessary to disassemble more than one model on the

same line for real-life disassembly. Within the scope of this thesis, the MTDLB problem, which takes into account all possible disassembly sequences and allows disassembly of more than one product in the same disassembly line, has been investigated and a MILP mathematical model has been developed to define the problem. The considered disassembly line is as shown in Figure 4.1. In this disassembly system, the EOL A product with 12 disassembly processes and the EOL B product with 11 disassembly processes are disassembled on the same disassembly line working in two-sided synchrony and sent to the required warehouses. For example, on this figure, since the premise of the B_4 operation on the right side for the EOL A product is the B_1 operation on the left side, although the B_3 operation on the right was finished, the process could not be started directly and the B_1 operation on the left was expected to be completed. Although these and similar restrictions cause idle times in stations, this system provides more efficient results than straight or U-shape lines as it completely removes the right-left rotations caused by large-sized products and provides faster disassembly by reducing the cycle time. provides the formation. In this way, the environmental impact for disassembly of large-sized products is minimized as much as possible.

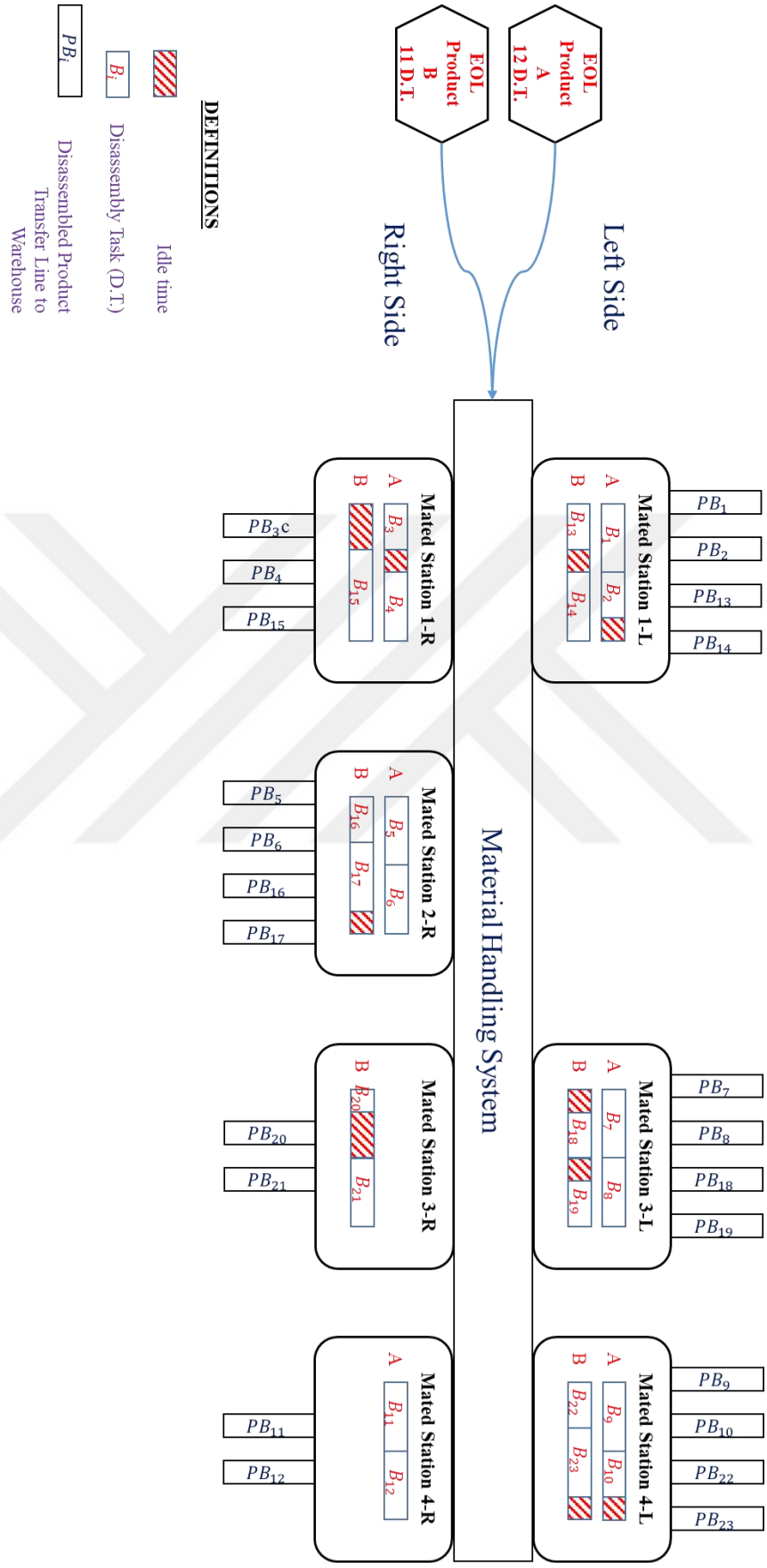


Figure 4.1. *Representation of mixed-model two-sided disassembly line*

In this context, a MILP mathematical model has been defined in order to optimize the line design cost, disassembly process times and line stability for the MTDLB problem defined only by us in the literature. While using the mathematical model, 3 main sources were used. These resources are:

1. ILP mathematical model by Koc et al in 2009, using the TAOG precedence diagram, which reaches the optimum result for the Single-Model Straight Disassembly Line Balancing (SSDLB) problem by considering all possible disassembly sequences.
2. MILP mathematical model created for the Mixed-Model Two-Sided Assembly Line Balancing (MTALB) problem defined by Özcan & Toklu in 2009.
3. MILP mathematical model created by Paksoy et al in 2013 for the Mixed-Model Straight Disassembly Line Balancing (MSDLB) problem, using the AOG precedence diagram, in which all possible sequences are considered.

In this section, respectively;

1. The TAOG precedence diagram used for the MTDLB problem is defined.
2. The general scheme for the problem was presented by making the necessary assumptions for the MTDLB problem.
3. Using the TAOG precedence diagram and assumptions defined for the MTDLB problem, the MILP mathematical model was developed by adding mandatory constraints and additional constraints in order to optimize the line design cost, disassembly process times and line stability.
4. An illustrative example was created for the MTDLB problem and the defined MILP mathematical model was solved by arranging the objective functions as a single objective function in a hierarchical order.

4.1. Transformed AND/OR Graph (TAOG) based Precedence Diagram for MTDLB Problem

As mentioned in Sections 2 and 3, there are various precedence diagrams in the literature for the DLB problem. However, the insignificance of functionality for disassembly creates a flexible precedence diagram for assembly lines. AND/OR Graph based precedence diagram reflects real life more as it provides access to all possible

disassembly sequences. However, it has not been widely used in the literature, as it is difficult in terms of programming/modelling. For this reason, in 2009 Koç et al. suggested the Transformed AND/OR Graph (TAOG) based precedence diagram, which is easy for programming/modelling. Actually, the DLB problem, which reaches the optimum result in a very short time by considering all possible disassembly sequences, has shown its competence with the emergence of mathematical models. However, there is no mathematical model of the TDLB problem used in the TAOG precedence diagram in the literature. Therefore, within the scope of this study, the TAOG precedence diagram defined by Koç et al., 2009 was developed for the TDLB problem is given in Figure 4.2 for the Flashlight product as an example application.

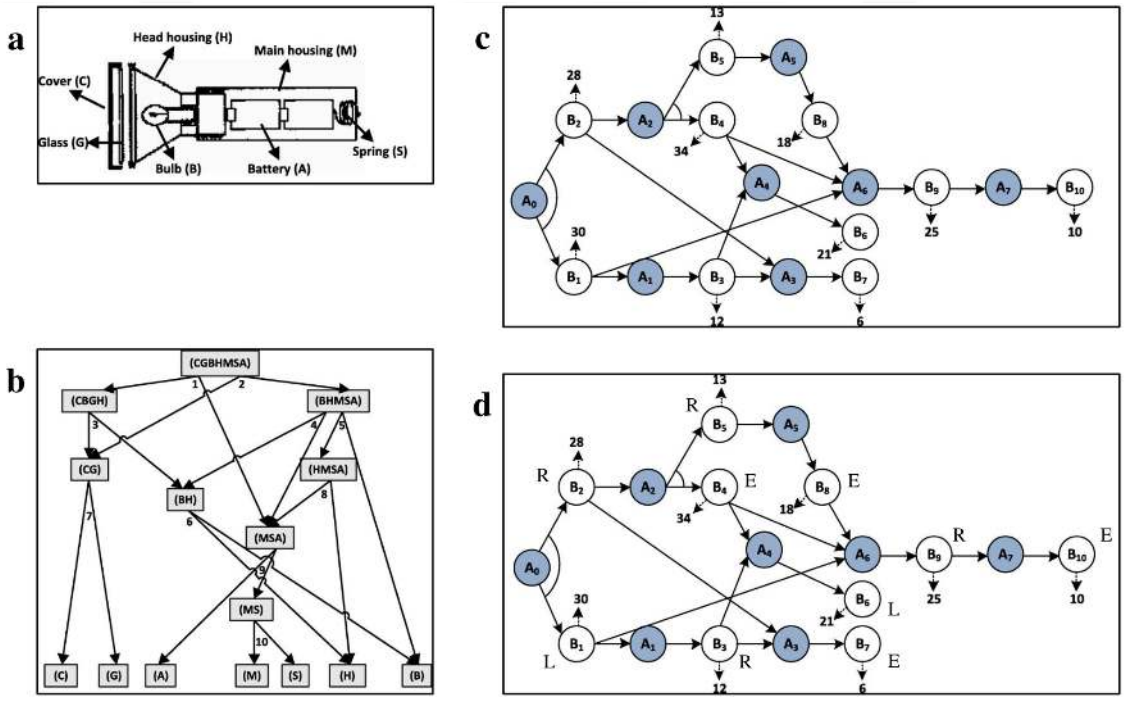


Figure 4.2. *Used TAOG based precedence diagram for Mixed-Model Two-Sided Disassembly Line Balancing (MTDLB) problem: (a) a sample product (flashlight), (b) AND/OR graph of flashlight, (c) Transformed AND/OR graph of flashlight, (d) TAOG of flashlight for MTDLB problem*

In the TAOG diagram for the TDLB problem given in Figure 4.2, the sub-assembly parts of the Flashlight product are given with a, AOG with b, TAOG with c and TAOG for the TDLB problem developed within the scope of this study. White nodes represent normal nodes (disassembly tasks), blue nodes represent artificial nodes. The only difference between the developed TAOG precedence diagram and the traditional TAOG precedence diagram is that it has a side that can be made for each of the normal nodes. Starting from the artificial node A_0, only one of the normal nodes of each artificial node

connected with the OR Successor relationship is selected and continued. The precedence diagram is obtained when there is no other normal node to be selected.

4.2. Assumptions for MTDLB Problem

Some assumptions or definitions have been made for the MTDLB problem investigated in this thesis. These assumptions/descriptions in order:

- The disassembly line has two-sided layout.
- More than one different product can be disassembled in the disassembly line.
- Each product has more than one disassembly sequence alternative and the mathematical model decides the best disassembly sequence.
- Each job in the selected disassembly sequence must be assigned straight or reverse to a station.
- There is a difference in processing times between jobs assigned to the same station.
- If the predecessor jobs are assigned to reciprocal stations, the predecessor job cannot be started before the predecessor job is finished.
- Each job has two types of assignability, straight or reverse, and jobs must be assigned to the appropriate station for the type to which they can be assigned.
- The completion times of the jobs assigned to the station for each model cannot exceed the cycle time.

4.3. Mixed-Integer Linear Programming (MILP) based Mathematical Model for MTDLB Problem

A Mixed-Integer Linear Programming (MILP) based mathematical model has been developed for the MTDLB problem investigated in this thesis. First, the notation is specified for the developed mathematical model, and then the objective functions and constraints are specified.

Scalars:

$nA = \max\{nA_m\}$	Number of total artificial nodes
$nN = \max\{nN_m\}$	Number of total normal nodes (tasks)
nJ	Number of available work-stations

nS	Number of sides
nM	Number of models

Indices and Sets:

$k \in \{1, \dots, nA\}$	Set of artificial nodes.
$i \in \{1, \dots, nN\}$	Set of normal nodes (tasks).
$j \in \{1, \dots, nJ\}$	Set of mated-stations.
$s \in \{1, \dots, nS\}$	Set of station sides.
$PRE(m, k, i)$	Set of immediate normal node i predecessors of artificial nodes k for each model m.
$SUC(m, k, i)$	Set of immediate normal node i successors of artificial nodes k for each model m.
$\theta(m, i, s)$	Set of station-sides s where normal node i can be made for each model m.

Parameters:

nA_m	Number of artificial nodes for each model m.
nN_m	Number of normal nodes for each model$m$.
C_m	Cycle time for each model m.
t_{im}	Processing time of each normal task i for each model m.

Decision and Auxiliary Variables:

F_j	If mated-station j is opened from both side, 1, otherwise, 0.
G_j	If mated-station j is opened from only one side, 1, otherwise, 0.
U_{js}	If station j is opened from station-side s, 1, otherwise, 0.
δ_{hi}	If task h is made before task i, 1, otherwise, 0.
γ_{mjs}	If station-side s of station j for model m is opened, 1, otherwise, 0.
x_{mijs}	If station j is opened from station-side s, 1, otherwise, 0.

4.3.1. Objective Functions of MTDLB Problem

As mentioned in Section 2 and Section 3, a wide variety of objective functions can be used for the DLB problem. In this thesis, three objective functions, which are considered the most necessary for two-sided lines, are used. These objective functions are, respectively, minimizing the cost of line design (muting stations as much as possible for two-sided lines), minimizing the total processing time since TAOG precedence relations are used (minimizing the processing time will also minimize the resources used, such as electricity) and achieving line balance. The objective functions defined for the MTDLB problem are given in Equation 4.1 – 4.3, respectively.

$$objective_1 = \boldsymbol{Min} \sum_{j=1}^{nJ} j \times \left(F_j + G_j\right) \tag{4.1}$$

Equation 4.1 expresses the minimization of the design cost of the resulting disassembly line. In two-sided lines, it is desirable to open the stations as mutually as possible. In this way, the line is made as short as possible and space is saved. For example, while eight stations occupy a minimum of eight units in a straight disassembly line, a minimum of four units in a double-sided disassembly line.

The second objective function for the MTDLB problem is to minimize the number of opened stations. This objective function is as given in Equation 4.2.

$$objective_2 = \boldsymbol{Min} \sum_{j=1}^{nJ} \sum_{s=1}^{nS} U_{js} \tag{4.2}$$

Equation 4.2 expresses the minimization of the number of opened stations in the disassembly line. While it is desired that the opened stations are mutual in the first objective function, it is aimed to minimize the number of opened stations. For example, if there are seven stations and eight stations as two-sided disassembly line alternatives that take up four units of space, the line with seven stations is preferred. However, while there are seven stations in the two-sided disassembly line, which takes up five units of space, there can be eight stations in the two-sided disassembly line that takes up four units of space. Which alternative is better in this case depends on the decision maker. Therefore, the second objective function of the MTDLB problem is to minimize the number of opened stations.

The third objective function for the MTDLB problem is to minimize the sum of the processing times of the selected disassembly operations. This objective function is as given in Equation 4.3.

$$objective_3 = \boldsymbol{Min} \sum_{m=1}^{nM} \sum_{i=1}^{nN_m} t_{mi} \times z_{mi} \tag{4.3}$$

Equation 4.3 refers to the minimization of the sum of the disassembly operations times selected from the TAOG precedence relations of each model. The insignificance of functionality for disassembly leads to stretching of the priority relations, which in turn causes the tasks and task sequences to change. Although the number of studies in which a single disassembly sequence is used, just like in assembly lines, is quite high in the literature, it does not reflect the truth. TAOG, used in this study and developed by Koç et al in 2009, provides a systematic display of all disassembly sequences in one place using artificial and normal nodes. The aim is to minimize the processing times of the disassembly tasks in the selected disassembly sequence.

4.3.2. Constraints of MTDLB Problem

The constraints constituting the general lines of the MTDLB problem investigated within the scope of this thesis are as given in Equation 4.4 – 4.22. These constraints are indispensable constraints and in the absence of one of them, the problem ceases to be an MTDLB problem.

$$\sum_{i \in SUC(m,k)} z_{mi} = 1 \qquad \forall m \in \{1, \dots, nM\}, \forall k \in \{0\} \tag{4.4}$$

$$\sum_{i \in SUC(m,k)} z_{mi} = \sum_{i \in PRE(m,k)} z_{mi} \qquad \forall m \in \{1, \dots, nM\}, \forall k \in \{1, \dots, nA_m\} \tag{4.5}$$

$$\sum_{j=1}^{nJ} \sum_{s \in \theta(m,i)} x_{mijs} = z_{mi} \qquad \forall m \in \{1, \dots, nM\}, \forall i \in \{1, \dots, nN_m\} \tag{4.6}$$

$$\sum_{i \in PRE(m,k)} \sum_{v=1}^{j} \sum_{s \in \theta(m,i)} x_{mivs} \geq \sum_{i \in SUC(m,k)} \sum_{s \in \theta(m,i)} x_{mijs} \qquad \forall m \in \{1, \dots, nM\}, \forall j \in \{1, \dots, nJ\}, \forall k \in \{1, \dots, nA_m\} \tag{4.7}$$

$$tf_{mi} \leq C \times z_{mi} \qquad \forall m \in \{1, \dots, nM\}, \forall i \in \{1, \dots, nN_m\} \tag{4.8}$$

$$tf_{mi} \geq t_{mi} \times z_{mi} \qquad \forall m \in \{1, \dots, nM\}, \forall i \in \{1, \dots, nN_m\} \tag{4.9}$$

$$tf_{mi} - tf_{mh} + BM \times \left(2 - \sum_{s \in \theta(m,i)} x_{mijs} - \sum_{s \in \theta(m,h)} x_{mhjs}\right) \geq t_{mi} \qquad \forall m \in \{1, \dots, nM\}, \forall j \in \{1, \dots, nJ\}, \forall k \in \{1, \dots, nA_m\}, \forall i \in SUC(m,k), \forall h \in PRE(m,k) \tag{4.10}$$

$$tf_{mi} - tf_{mh} + BM \times \left(3 - x_{mijs} - x_{mhjs} - \delta_{hi}\right) \geq t_{mi} \qquad \forall m \in \{1, \dots, nM\}, \forall j \in \{1, \dots, nJ\}, \forall s \in \{1, \dots, nS\}, \forall i \in \{1, \dots, nN_m\}, \forall h \in \{i > h \mid 1, \dots, nN_m\} \tag{4.11}$$

$$tf_{mh} - tf_{mi} + BM \times \left(2 - x_{mijs} - x_{mhjs} + \delta_{hi}\right) \geq t_{mh} \qquad \forall m \in \{1, \dots, nM\}, \forall j \in \{1, \dots, nJ\}, \forall s \in \{1, \dots, nS\}, \forall i \in \{1, \dots, nN_m\}, \forall h \in \{i > h \mid 1, \dots, nN_m\} \tag{4.12}$$

$$\sum_{i \in \theta(m,s) \cap i \leq nN_m} x_{mijs} - nN_m \times \gamma_{mjs} \leq 0 \qquad \forall m \in \{1, \dots, nM\}, \forall j \in \{1, \dots, nJ\}, \forall s \in \{1, \dots, nS\} \tag{4.13}$$

$$\sum_{m=1}^{nM} \gamma_{mjs} - nM \times U_{js} \leq 0 \qquad \forall j \in \{1, \dots, nJ\}, \forall s \in \{1, \dots, nS\} \tag{4.14}$$

$$\sum_{s=1}^{nS} U_{js} - 2 \times F_j - G_j = 0 \qquad \forall j \in \{1, \dots, nJ\} \tag{4.15}$$

$$F_j, G_j \in \{0,1\} \qquad \forall j \in \{1, \dots, nJ\} \tag{4.16}$$

$$U_{js} \in \{0,1\} \qquad \forall j \in \{1, \dots, nJ\}, \forall s \in \{1, \dots, nS\} \tag{4.17}$$

$$\delta_{ih} \in \{0,1\} \qquad \forall i, h \in \{1, \dots, nN_m\} \tag{4.18}$$

$$z_{mi} \in \{0,1\} \qquad \forall m \in \{1, \dots, nM\}, \forall i \in \{1, \dots, nN_m\} \tag{4.19}$$

$$tf_{mi} \geq 0 \qquad \forall m \in \{1, \dots, nM\}, \forall i \in \{1, \dots, nN_m\} \tag{4.20}$$

$$\gamma_{mjs} \in \{0,1\} \qquad \forall m \in \{1, \dots, nM\}, \forall j \in \{1, \dots, nJ\}, \forall s \in \{1, \dots, nS\} \tag{4.21}$$

$$x_{mijs} \in \{0,1\} \qquad \forall m \in \{1, \dots, nM\}, \forall i \in \{1, \dots, nN_m\}, \forall j \in \{1, \dots, nJ\}, \forall s \in \{1, \dots, nS\} \tag{4.22}$$

Equations 4.4 – 4.22 are the main constraints describing the MTDLB problem discussed in this thesis. Equations 4.4 – 4.6 ensures that a single disassembly sequence is selected for each model over the TAOG it belongs to, and that each selected normal node, that is, disassembly tasks, is assigned to a station on the side where it can be done. Equation 4.4 ensures that for each model, one of the normal nodes of the first artificial node ($k = 0$) on the TAOG it belongs to, which has a Successor relationship, is selected. Equation 4.5 ensures that for each model, except for the first artificial node on the TAOG it belongs to, if any of the normal nodes of the other artificial nodes ($k > 0$) with a Precedence relationship is selected, one of the normal nodes of the same artificial node with a Successor relationship is selected. Equation 4.6 ensures that if any normal node (disassembly task) is selected on the TAOG it belongs to for each model, that task is definitely assigned to a station on the side where it can be done. Equation 4.7 provides precedence relationships for each model. Artificial nodes on TAOG provide priority relations. For each model, except for the first artificial node on the TAOG it belongs to, the normal nodes of the other artificial nodes with a Precedence relationship are assigned to the same station before the normal nodes with the Successor relationship. Equation 4.8 ensures that the end times of the normal nodes selected for each model are less than or equal to the cycle time. Equation 4.9 ensures that the expiry times of the normal nodes selected for each model are greater than or equal to the transaction time. Equation 4.10 ensures that if two tasks with a predecessor relationship are assigned to the same station on the same side or mutually, the difference in finish times between the predecessor task and the other task is equal to the processing time of the other task. For each model, except for the first artificial node on the TAOG it belongs to, if each normal node of the other artificial nodes with a Precedence relationship and each normal node with a Successor relationship are assigned to the same station on the same side or reciprocally, the ending time of the normal node with the Successor relationship is calculated. Ensures that the finish time of the normal node with the Precedence relationship is as high as or more than the execution time of the normal node with the Successor relationship. In Equations 4.11

– 4.12, the difference in finish times between tasks assigned to the same station is adjusted according to which job has been done before. That is, only one task can be performed on the station at a time in its zone. Equations 4.13 – 4.15 are the necessary constraints for calculating objective functions. Equation 4.13, for each model, if there is a task processed on any mated station, information is if that mated station is opened on the transacted side. In Equation 4.14, if each mated station is opened on any side for any model, information is if that mated station is opened on that side. In Equation 4.15, for each mated station, if the number of stations opened on each side is two, it is mutually opened, if it is one, it is unilaterally opened. Equations 4.16 – 4.22 are the constraints by which the type of decision variables are determined.

4.4. An Illustrative Example for MTDLB Problem

An illustrative example was designed and solved by Paksoy et al by using Flashlight and Radio EOL products released by TAOG precedence relations in 2013 in order to test the accuracy of the model and to better understand the MTDLB problem, which was defined within the scope of this thesis and whose MILP-based mathematical model was developed.

4.4.1. Description of Data

The Flashlight EOL product has 10 disassembly tasks and the radio EOL product has 30 disassembly tasks. TAOG consisting of 10 normal nodes (disassembly task) and 8 artificial nodes of the Flashlight product is given in Figure 4.3 and TAOG consisting of 30 normal nodes (disassembly task) and 20 artificial nodes of the Radio EOL product is given in Figure 4.4.

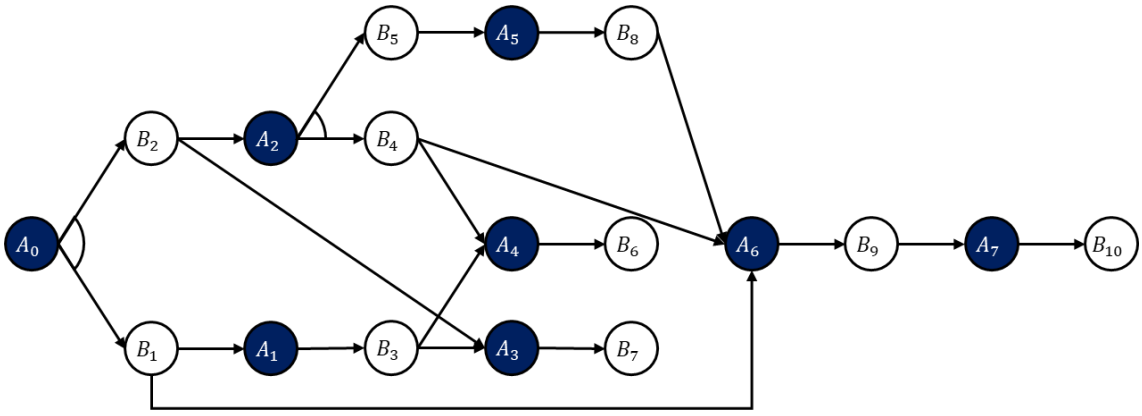


Figure 4.3. *TAOG for EOL Flashlight product*

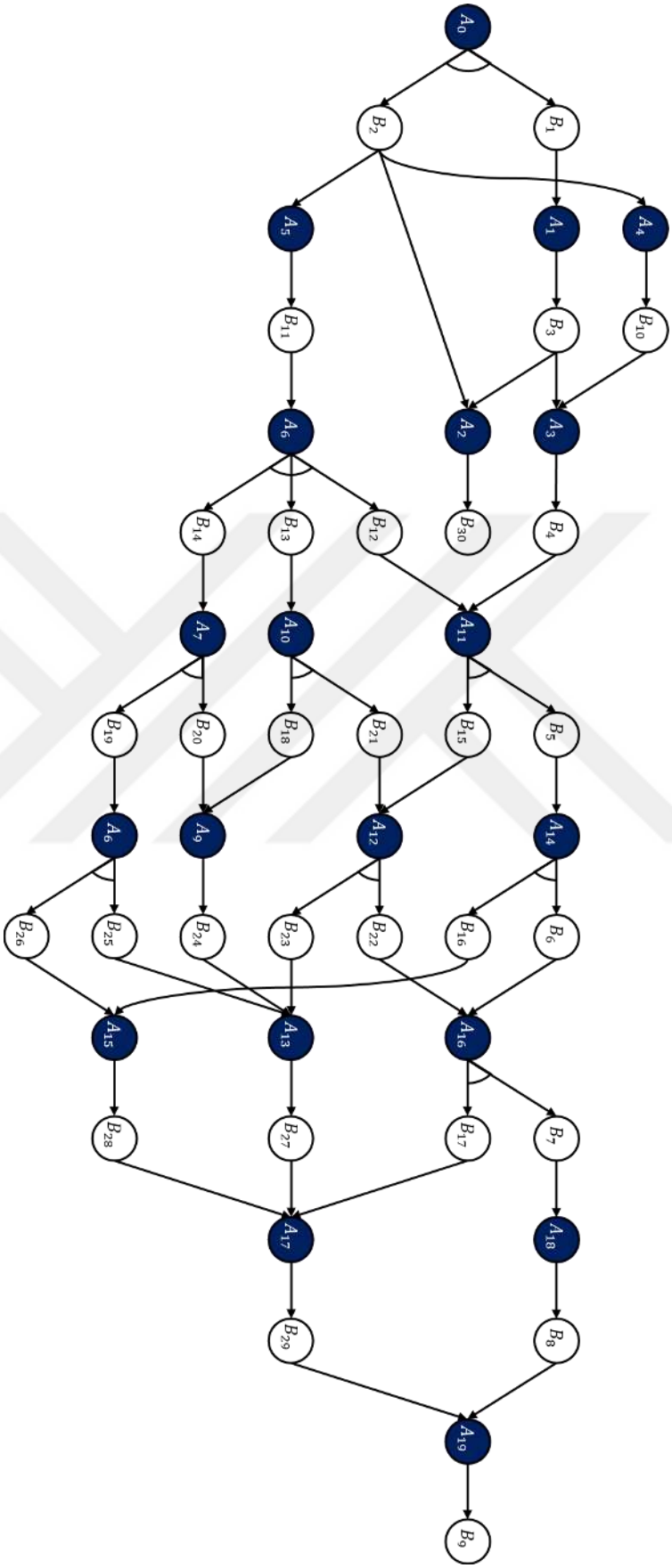


Figure 4.4. *TAOG for EOL Radio product*

The processing times of the Flashlight EOL product are given in Table 4.1 in seconds as defined by Paksoy et al (2013). Also in this table are given the randomly generated possible sides for each disassembly task. Similarly, the processing times of the Radio EOL product are given in Table 4.2, as defined by Paksoy et al (2013), in seconds, and randomly generated sides for each disassembly task. Cycle times for both models are assumed 40 seconds.

Table 4.1. *The side information and processing times of the Flashlight EOL product*

# Tasks	Side (1 – Left, 2 - Right)	Processing Time (second)
1	1	3
2	2	28
3	1	12
4	1	34
5	1, 2	13
6	1, 2	21
7	2	6
8	2	18
9	1	25
10	1, 2	10

Since commercial solvers such as GAMS, LINGO, Gurobi, which are available in the market, cannot give the solutions of multi-objective mathematical models, the objectives of the MTDLB problem defined in this thesis have been converted to hierarchical form as given in Equation 4.23. Here ε denotes a very small number. In this way, a degree of importance was given to each objective function and the multi-objective mathematical model was transformed into a single-objective mathematical model.

$$objective_{ie} = \boldsymbol{Min}(objective_1 + \varepsilon \times objective_2 + \varepsilon^2 \times objective_3)$$

$$objective_{ie} = \boldsymbol{Min}\left(\sum_{j=1}^{nJ} j \times (F_j + G_j) + \varepsilon \times \sum_{j=1}^{nJ}\sum_{s=1}^{nS} U_{js} + \varepsilon^2 \times \sum_{m=1}^{nM}\sum_{i=1}^{nN_m} t_{mi} \times z_{mi}\right) \quad (4.23)$$

Equation 4.23 is the hierarchical objective function for the MTDLB problem. First, it is aimed to minimize the design cost of the disassembly line. Secondly, it is aimed to

minimize the number of opened stations in the disassembly line. Finally, it is aimed to minimize the sum of the disassembly operations times selected from the TAOG precedence relations of each model.

Table 4.2. *The side information and processing times of the Radio EOL product*

# Tasks	Side (1 – Left, 2 - Right)	Processing Time (second)
1	2	11
2	1	20
3	1	20
4	1, 2	14
5	2	19
6	2	1
7	2	7
8	1	6
9	1	6
10	1, 2	7
11	1, 2	19
12	1, 2	11
13	1, 2	18
14	1	13
15	1	5
16	1,2	11
17	2	6
18	2	4
19	1, 2	6
20	1	8
21	2	18
22	1	15
23	1	15
24	2	6
25	2	10
26	1, 2	20
27	1	13
28	1	1
29	1, 2	4
30	1, 2	5

4.4.2. Solution of Illustrative Example

The MILP-based mathematical model created for the MTDLB problem was solved using Gurobi solver on an Intel i7 2.00 GHz processor with 8 GB ram computer. Python codes for Gurobi are given in Appendix 2. The optimum solution was reached in 1.33 seconds. As a result of the solution, four stations were opened. While one mated station was opened mutually, two mated stations were opened unilaterally. Total disassembly line length was obtained as three units. Normal nodes (disassembly tasks) selected over TAOG for both models are given in Figure 4.5 and Figure 4.6.

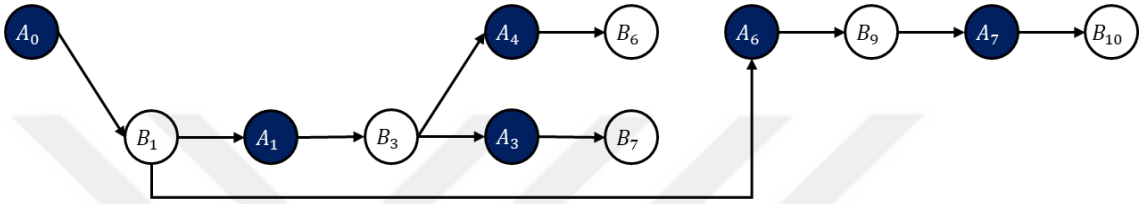


Figure 4.5. *Optimal disassembly sequence of flashlight EOL product for illustrative MTDLB problem example*

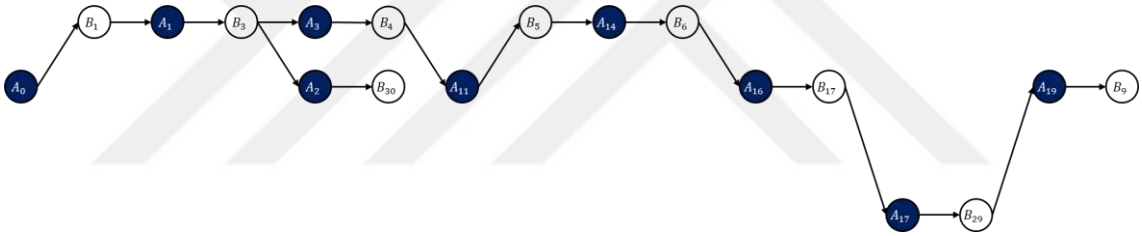


Figure 4.6. *Optimal disassembly sequence of radio EOL product for illustrative MTDLB problem example*

Detailed solution of MTDLB problem for Illustrative example is given in Table 4.3 for Flashlight EOL product and Table 4.4 for Radio EOL product. At the same time, the resulting disassembly line is shown in Figure 4.7.

Table 4.3. *Disassembly line assignments and solution information of the Flashlight EOL product*

# Task	Station	Side	Processing Time	Starting Time	Ending Time
1	1	1	30	0	30
3	2	2	12	25	37
6	3	1	21	0	21
7	3	1	6	21	27
9	2	2	25	0	25
10	3	1	10	27	37

Table 4.4. *Disassembly line assignments and solution information of the Radio EOL product*

# Task	Station	Side	Processing Time	Starting Time	Ending Time
1	1	2	11	0	11
3	1	1	20	11	31
4	2	2	14	0	14
5	2	2	19	14	33
6	2	2	1	33	34
9	3	1	6	9	15
17	2	2	6	34	40
29	3	1	4	0	4
30	3	1	5	4	9

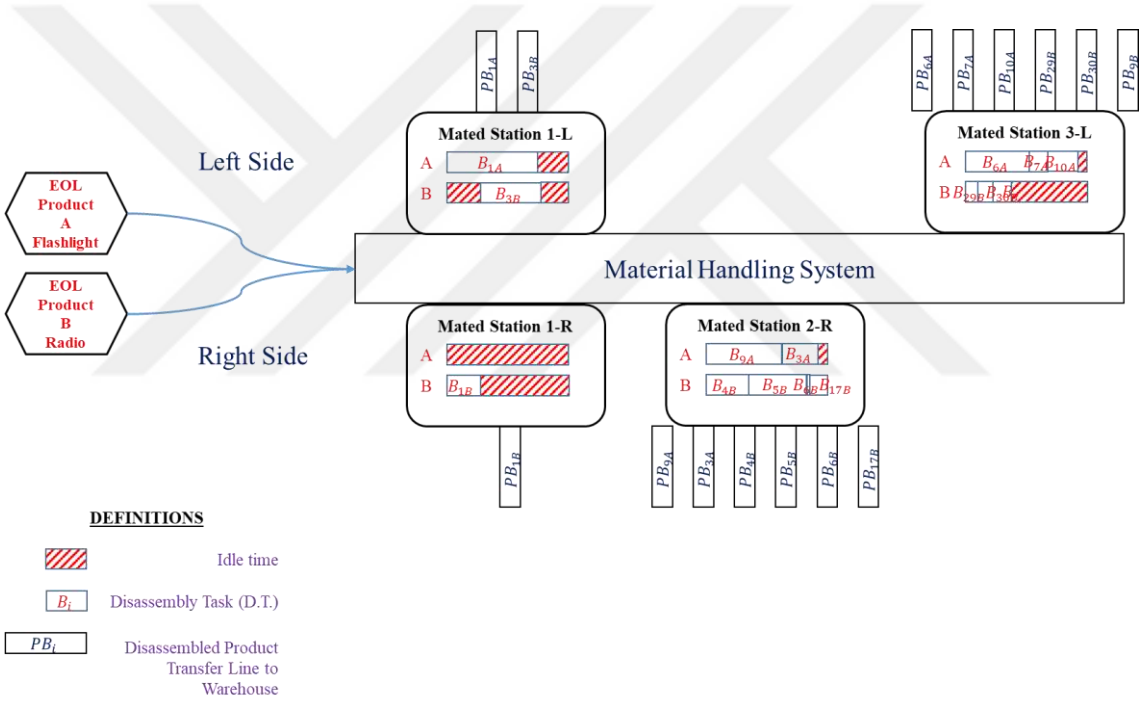


Figure 4.7. *Optimal disassembly line for illustrative MTDLB problem example*

For the Flashlight EOL product, three stations in three mated stations are used. Total disassembly time is 104 seconds. Therefore, the line efficiency for the Flashlight EOL product is 86.67%. For the Radio EOL product, four stations, which are in three mated stations, are used. Total disassembly time is 86 seconds. Therefore, the line efficiency for the Flashlight EOL product is 53.75%. Considering that equal amount of product comes from both EOL products, the efficiency of the line becomes 70.21%. However, since the efficiency of the line depends on the processing times and the number of stations opened, and one of the objectives of the MTDLB problem is to minimize the processing times,

the low efficiency remains inexpressive based on this problem. In order to minimize the environmental impact, the process times spent for disassembly should be minimized rather than the efficient operation of the line.

The illustrative example that was resolved was for two small-volume industrial products. However, since the defined MTDLB problem takes into account large volume, multi-model products, when the number of models and product content increases, it will not be possible to reach the optimum result with commercial software due to the NP-hard nature of the problem. At the same time, although the MTDLB problem described is multi-objective, since commercial software does not have a multi-objective solver, the objectives were converted into a single objective format and optimum results were sought. However, the desired situation is to find the best results for each objective and show the results in the region called Pareto Frontier to the decision maker. The best solution among them is left to the decision maker. For this reason, a Multi-Objective Memetic Algorithm has been defined to solve the MTDLB problem described in the other section.

5. SOLUTION APPROACH – MULTI-OBJECTIVE MEMETIC ALGORITHM (MOMA)

For the MTDLB problem defined in this thesis, when large-volume multi-model products are considered, it is not possible to reach the optimum result with commercial software. At the same time, the multi-objective nature of the problem limits the consideration of each objective function individually. Commercial software only obtains solutions for one objective. For all these reasons, a Multi-Objective Memetic Algorithm (MOMA) has been defined to solve the MTDLB problem described in this section.

Memetic Algorithm (MA) is an advanced version of Genetic Algorithm (GA). GA is a metaheuristic algorithm that takes into account the evolutionary process in nature. It basically consists of Selection, Crossover and Mutation processes. Thanks to all these processes, populations similar to the old population but constantly improving themselves by taking the good characteristics of the old population are obtained. MA is the addition of Local Search algorithms to GA. Local Search is based on iterative improvement of an existing solution using a specific methodology. By adding any Local Search algorithm to the GA, an MA is obtained. In this section:

1. Genetic Algorithm defined,
2. Local Search algorithms defined,
3. Memetic Algorithm defined,
4. Multi-objective structure and Non-Dominated Sorted Genetic Algorithm – II (NSGA-II) algorithm are defined in metaheuristic algorithms,
5. Multi-Objective Memetic Algorithm (MOMA) algorithm defined,
6. The representation is defined for the solution of the MTDLB problem with MOMA.

5.1. Genetic Algorithm (GA)

GA is an artificial intelligence-based optimization algorithm used to obtain near-optimal results thanks to biologically inspired operators such as Selection, Crossover and Mutation, taking into account the evolutionary process in nature. Alan Turing first discovered it in 1950 (Whitley, 1994). Each solution/individual has a feature that can change and/or mutate. Although these solutions are usually represented by zero and one (binary coding), different encoding types (permutation, integer coding) are also available

in the literature. GA enables solutions/individuals within a given population to evolve for the better.

GA usually starts with an initial solution/population that is randomly generated or generated according to a specific procedure. Each iteration in the GA is called a generation. In each generation, the fitness value (objective function) is calculated for each solution/individual in the population. Among the individuals in the current population, solutions/individuals with a better fitness value are selected to create the new generation. Selected solutions/individuals are crossed among themselves according to certain procedures and new solutions/individuals are obtained. Usually, when the maximum number of generations is achieved, the algorithm is terminated and the best solution/solutions obtained so far are shown to the user.

In order to implement GA, the representation of solutions/individuals (binary, permutation, integer etc. coding) and a fitness function are required. By standard, the representation of each solution/individual is a bit string of one and zero. However, the most appropriate display type is determined according to the nature of the problem. For example, the notation for the Traveling Salesman Problem (TSP), which is frequently encountered in the literature, is permutation coding. Permutation coding is preferred because each city is visited only once and the main important thing is to reveal the order in which the cities are visited. In this way, fitness function calculations can be made very easily over the representations of the population. It can also be used by combining all display formats according to the structure of the problem.

After the representation and fitness value calculation for the GA is completed, the GA is started with an initial population and the solution is iteratively improved with the Selection, Crossover and Mutation operators. A typical GA is given in Figure 5.1.

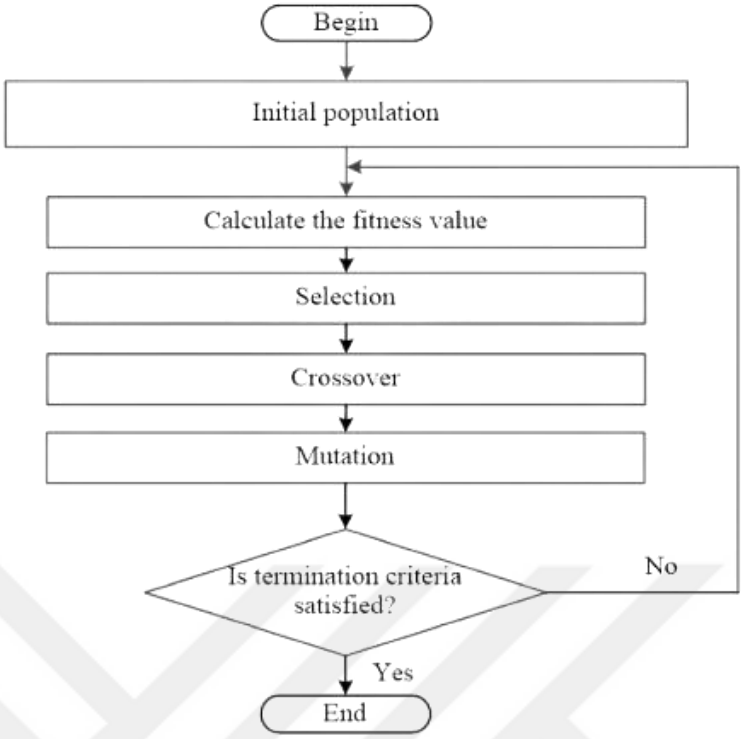


Figure 5.1. *Flowchart of Genetic Algorithm (GA)* (Albadr et al., 2020)

5.1.1. Selection Operator in GA

In order to obtain new generations, two individuals are selected from the individuals in the current generation, and then crossover and mutation are applied. Selection of individuals in the current generation plays a critical role so that new individuals can develop the best solution.

There are a wide variety of selection strategies developed for specific objectives for the new generations in the literature.

The most used selection strategies are as follows:

- Roulette Wheel Selection
- Stochastic Universal Sampling (SUS)
- Tournament Selection
- Rank Selection
- Random Selection

Roulette Wheel Selection

Individuals in the current generation get a share from the circular wheel according to their fitness values. The sum of the shares of individuals in the current generation is 100% as the whole of the circular wheel, or 360 degrees. The wheel is randomly rotated and the individual corresponding to the fixed point of the wheel is selected for crossover. The first individual selected for the second individual to be crossed is removed from the wheel and the vacant share is distributed to all individuals in proportion to their shares. The wheel is rotated randomly again and the individual corresponding to the fixed point of the wheel is selected for the second individual of the crossover. This is repeated for half the number of individuals in the generation (two parents are selected for two offspring in each iteration).

In Roulette Wheel Selection calculations, when in and in' represent individuals in the current generation, nIN represents the total number of individuals, p_{in} represents the selection probability for each individual (the area it occupies on the wheel), and f_{in} represents the fitness value for each individual, Equation 5.1 is used in the first individual selection. 5.2 is used for second individual selection.

$$p_{in} = \frac{f_{in}}{\sum_{in'=1}^{nIN} f_{in'}} \quad \forall in \in \{1, \dots, nIN\}\ for\ maximization\ problems$$
$$p_{in} = \frac{\frac{1}{f_{in}}}{\sum_{in'=1}^{nIN} \frac{1}{f_{in'}}} \quad \forall in \in \{1, \dots, nIN\}\ for\ minimization\ problems \tag{5.1}$$

$$p_{in} = \frac{p_{in}}{\sum_{in'=1 \cap in' \neq slctd\_inv}^{nIN} p_{in'}} \quad \forall in \in \{1, \dots, nIN\} \tag{5.2}$$

A typical example for Roulette Wheel Selection is given in Figure 5.2.

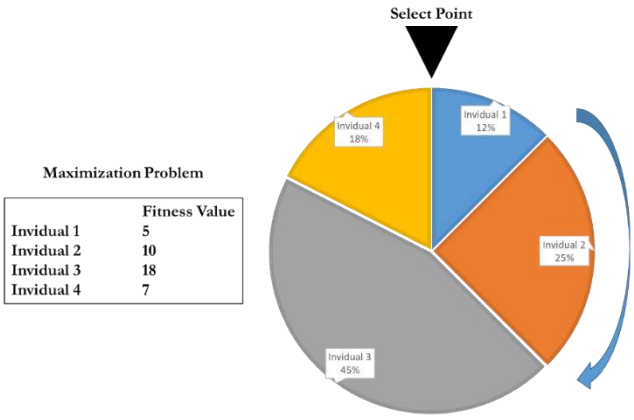


Figure 5.2. *Representation of Roulette Wheel Selection method in GA*

Stochastic Universal Sampling (SUS) Selection

Stochastic Universal Sampling (SUS) Selection is the same as Roulette Wheel Selection. The only difference between the two is that while Roulette Wheel Selection has a single select point, Stochastic Universal Sampling (SUS) Selection has two select points. This results in a higher selection of individuals with low probability. The Stochastic Universal Sampling (SUS) Selection for a typical maximization problem is given in Figure 5.3.

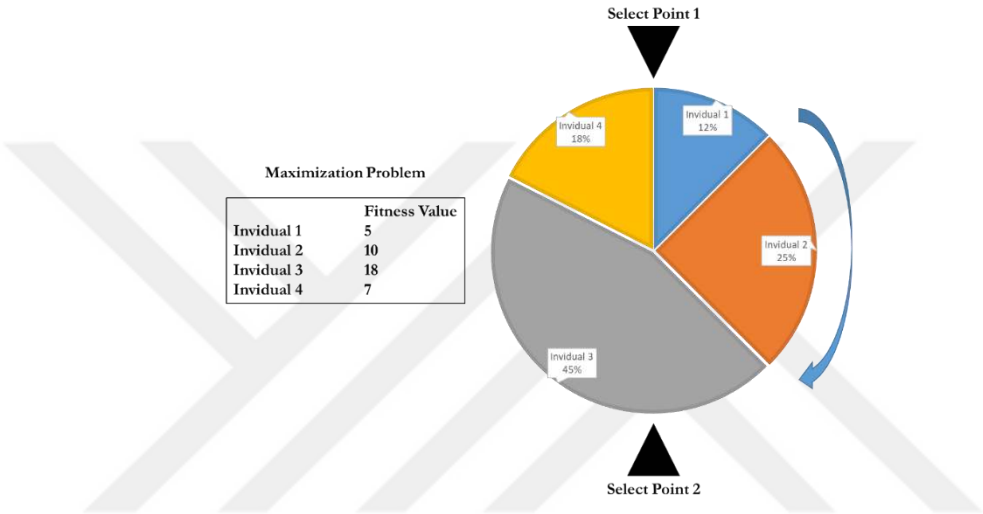


Figure 5.3. *Representation of Stochastic Universal Sampling (SUS) Selection method in GA*

Tournament Selection

Tournament Selection or K-Way Tournament Selection is the selection of k individuals from the population and choosing the best of the selected individuals as parents. The individual selected for the second parent is removed from the list and the process is repeated. Tournament Selection is very common in the literature as it can be applied for all kinds of fitness values. The Tournament Selection for a typical maximization problem is given in Figure 5.4.

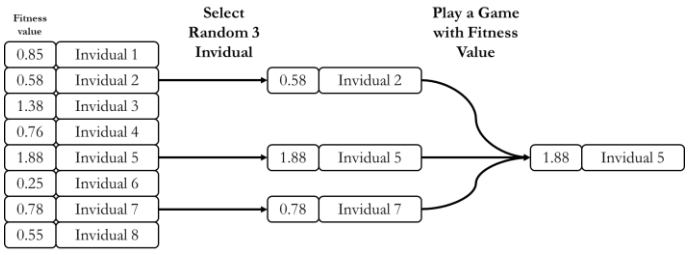


Figure 5.4. *Representation of Tournament Selection method in GA*

Rank Selection

Rank Selection is generally used when individuals in the population have very close fitness values. Individuals in the population are ranked according to their fitness values. The ranked individuals have the probability of being selected according to their rank. It is the same as Roulette Wheel Selection. However, all individuals have a share according to their rank instead of a share according to their fitness value. Therefore, no matter how much the fitness value value is above the others, the selection rate is constant. The Rank Selection for a typical maximization problem is given in Figure 5.5.

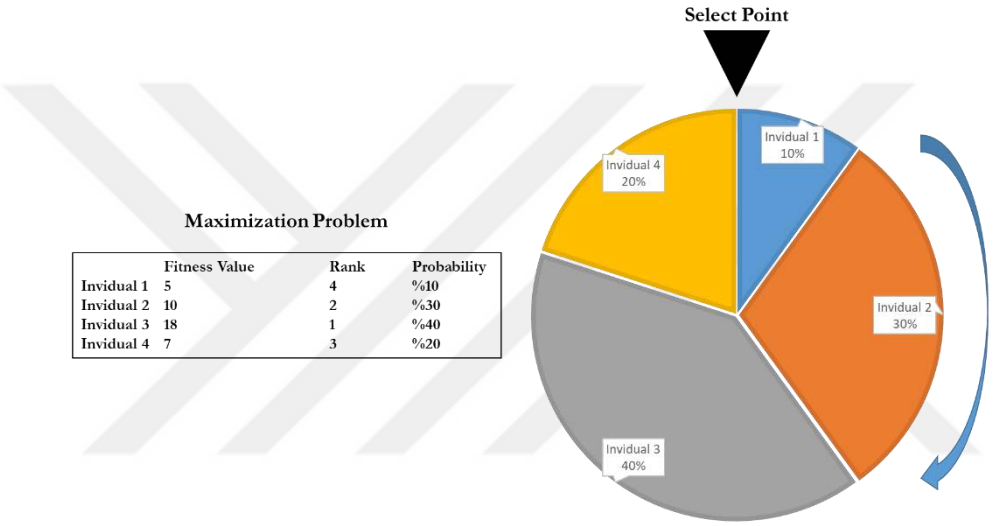


Figure 5.5. *Representation of Rank Selection method in GA*

Random Selection

Random Selection is the random selection of parents from the population. It is generally not preferred because good individuals are deleted from the population after a period.

5.1.2. Crossover Operator in GA

Obtaining new offspring individuals by crossing the individuals in the population among themselves can be defined as "crossover". In this way, the population is diversified and different points in the solution space are searched.

There are many different crossover strategies in the literature. The choice of this strategy, which differs from the problem and coding style being addressed, is critical for an accurate scan of the solution space.

The most commonly used crossover strategies are as follows:

- One/Single Point Crossover
- Multi Point Crossover
- Uniform Crossover
- Whole Arithmatic Recombination Crossover
- Davis' Order Crossover (OX1)

One/Single Point Crossover

Two individuals selected for crossover are obtained by cutting from a predetermined crossover point or from a randomly determined crossover point in each crossover operation and adding them to each other to obtain two new individuals. Typically One/Single Point Crossover as given in Figure 5.6.

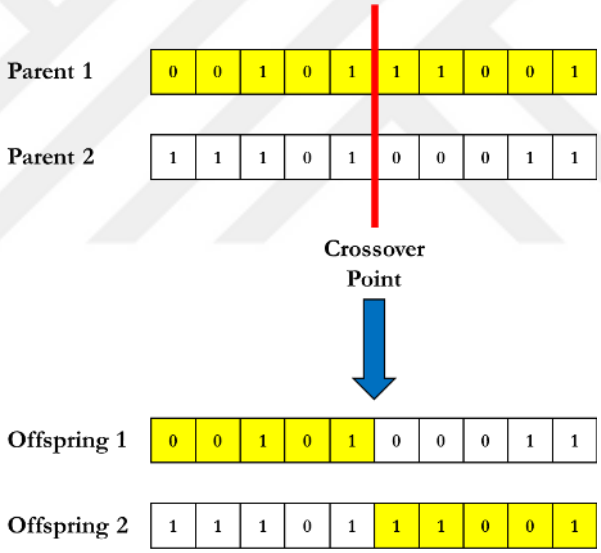


Figure 5.6. *A typically One/Single Point Crossover*

Multi Point Crossover

Two individuals selected for crossover are two or more predetermined crossover points, or two new individuals are obtained by cutting them from randomly determined crossover points in each crossover operation and adding them to each other. Typically a Two Point Crossover is as given in Figure 5.7.

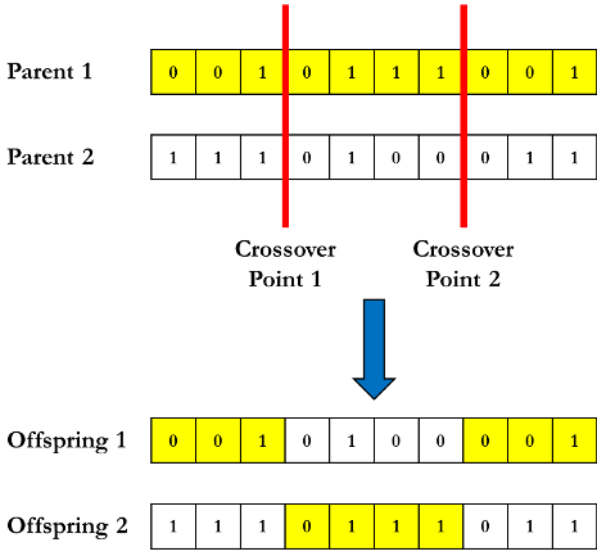


Figure 5.7. *A typically Multi Point Crossover*

Uniform Crossover

It is to obtain two new individuals by taking each nucleotide of the two individuals selected for crossover from the first or second individual in line with the randomly determined number and giving the remaining nucleotide to the other individual. Typically a Uniform Crossover is as given in Figure 5.8.

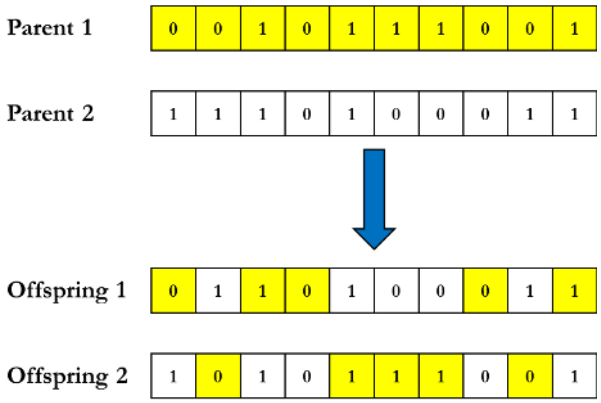


Figure 5.8. *A typically Uniform Crossover*

Whole Arithmetic Recombination Crossover

Whole Arithmetic Recombination Crossover is generally used for continuous number coding. The chromosomes of the two individuals selected for crossover are formed by Equation 5.3. That is, the newly formed individuals are the arithmetic mean of the parent individuals depending on the parameter a, which takes a value between 0 and

1. A typical Whole Arithmetic Recombination Crossover for the $a = 0.5$ case is as given in Figure 5.2.

$$Offspring_1 = a \times x + (1 - a) \times y$$
$$Offspring_2 = a \times y + (1 - a) \times x \quad (5.3)$$

The a parameter used in Whole Arithmetic Recombination Crossover varies between 0 and 1. If this parameter is selected as 0 or 1, the new individuals formed are the same as the parent individuals, and if it is selected as 0.5 in the middle, the two newly formed individuals are the same. This situation is shown visually in Figure 5.9. In the case of choosing this crossover type, the selection of parameter a plays a critical role.

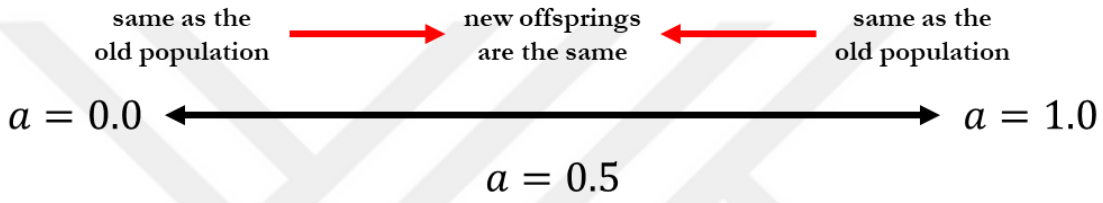


Figure 5.9. *Alpha parameter problem in Whole Arithmetic Recombination Crossover*

Typically a Whole Arithmetic Recombination is as given in Figure 5.10.

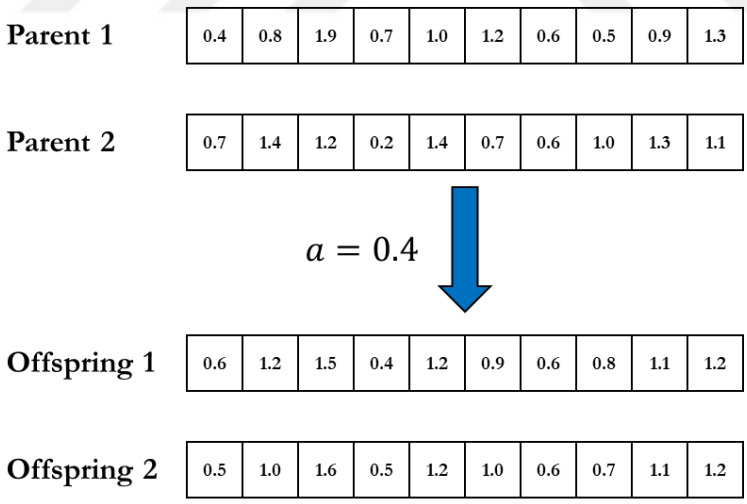


Figure 5.10. *A typically Whole Arithmetic Recombination Crossover*

Davis' Order Crossover (OX1)

The OX1 crossover method is generally preferred for permutation coding. The working algorithm is as follows:

1. Randomly generate two crossover points on the parent chromosomes and copy the parental segment between the two crossover points to the new offspring.
2. Fill in the remaining blanks for the first offspring created from the second parent, in order not to be in the first offspring.
3. Repeat step two for the second offspring.

A typical example for the OX1 crossover method is as given in Figure 5.11.

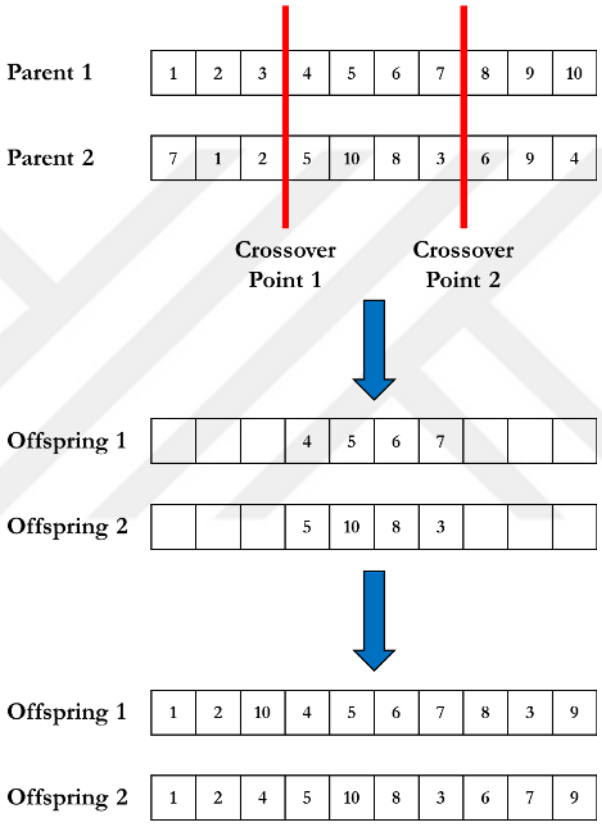


Figure 5.11. *A typically Davis' Order Crossover (OX1)*

5.1.3. Mutation Operator in GA

Making a small random change in the chromosomes of individuals in order to provide diversity in the population and obtain new solutions can be defined as "mutation". The mutation is carried out within a certain rate. If this ratio is too small, the solutions will gather around the local optimum after a while, on the contrary, if it is too high, the algorithm will turn into a random search and it will be difficult to find local optimum points and global optimum points. Therefore, finding a very good mutation rate will both ensure that all local optimum points are found and escape from these local optimum points.

In the literature, there are various strategies related to mutation, just like crossover strategies. Although these strategies are used singularly in most studies, many studies combine multiple strategies.

The most commonly used mutation strategies are as follows:

- Bit Flip Mutation
- Random Resetting Mutation
- Swap Mutation
- Scramble Mutation
- Inversion Mutation

Bit Flip Mutation

It is the most used method in solutions that are usually coded with the Binary Coding method. This method is based on the conversion of one or more randomly selected nucleotides from the chromosome to their opposite. A typical Bit Flip Mutation is as given in Figure 5.12.

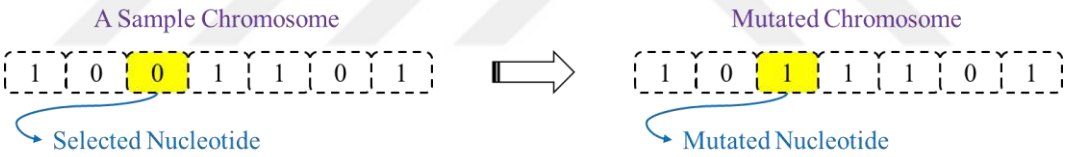


Figure 5.12. *A typically Bit Flip Mutation*

Random Resetting Mutation

It is the converted format for Integer Coding of the Bit Flip Mutation method, which is frequently used for the Binary Coding method. This method is based on assigning a random value from a set of allowed values to one or more randomly selected nucleotides from the chromosome. A typical Bit Flip Mutation is as given in Figure 5.13.

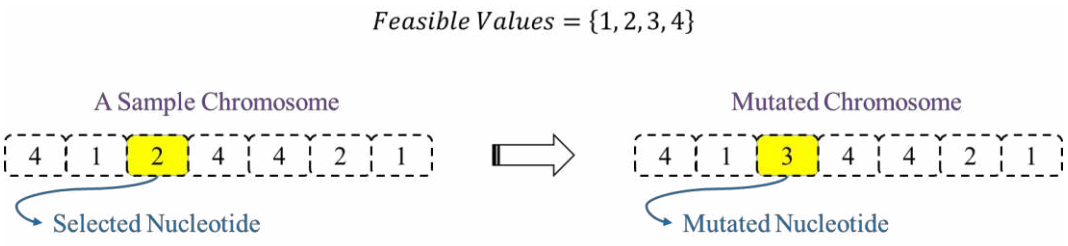


Figure 5.13. *A typically Random Resetting Mutation*

SWAP Mutation

It is the most common mutation strategy in the literature, which can be used for any coding method. Usually used for Permutation Coding, this strategy is based on swapping two randomly selected nucleotides. A typical Swap Mutation is as given in Figure 5.14.

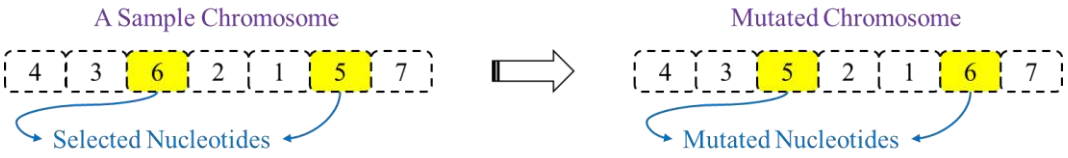


Figure 5.14. *A typically SWAP Mutation*

Scramble Mutation

Scramble Mutation is another mutation strategy that can be used for any coding method such as the Swap Mutation strategy. Usually used for Permutation Coding, this strategy is based on randomly shuffling a randomly selected nucleotide subset. A typical Scramble Mutation is as given in Figure 5.15.

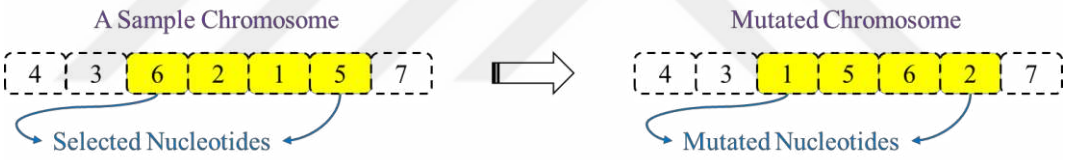


Figure 5.15. *A typically Scramble Mutation*

Inversion Mutation

Inversion Mutation is another mutation strategy that can be used for any coding method such as Swap Mutation and Scramble Mutation strategies. Usually used for Permutation Coding, this strategy is based on inversion of a randomly selected subset of nucleotides. A typical Inversion Mutation is as given in Figure 5.16.

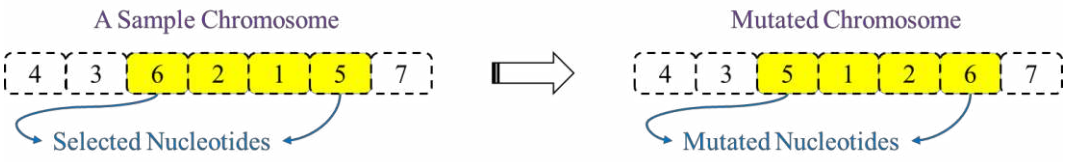


Figure 5.16. *A typically Inversion Mutation*

5.2. Local Search Algorithms

A Local Search algorithm is an optimization algorithm that starts with a solution and searches for the best solution by iteratively moving to a neighboring solution, adhering to a certain procedure. Local Search algorithms can only be applied when a neighborhood is defined. In this neighborhood graph theory, while the arcs between the nodes can be removed and connected to other nodes, it can be the candidate solution, while in the Traveling Salesman Problem (TSP), which is coded with permutation coding; two nucleotides can be changed with SWAP. The SWAP applied for TSP and the arc change applied for Graph Theory are actually the same, but there are structural differences. An example neighborhood with Graph Theory for the TSP problem is shown in Figure 5.17 (this is also known as three-opt algorithm), an example neighborhood for permuation coding is shown in Figure 5.18 (this is also known as SWAP). It is seen that the neighborhood shown in Figure 5.17 and the neighborhood shown in Figure 5.18 are the same.

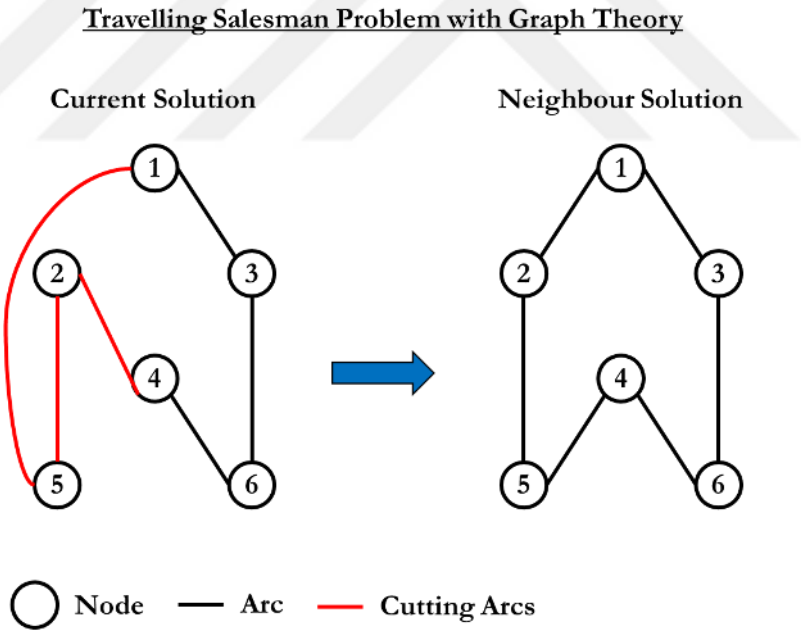


Figure 5.17. *A neigbourhood solution with 3-opt algorithm for Travelling Salesman Problem*

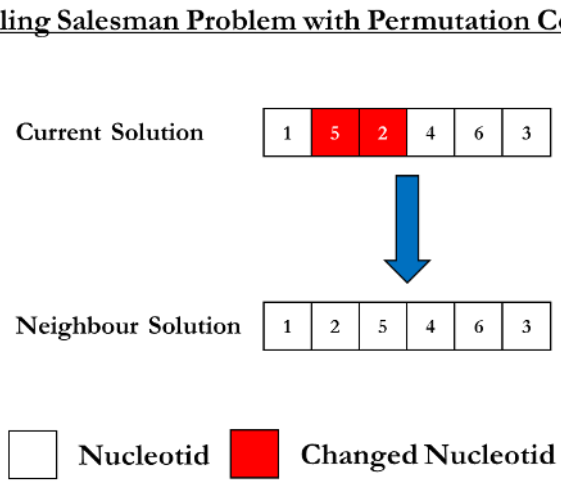


Figure 5.18. *A neigbourhood solution with SWAP operator for Travelling Salesman Problem*

In Graph Theory, each solution has more than one neighbor solution when considering which arcs will be cut, which nucleotides will be replaced or changed in solutions with permutation or any encoding. In a 10-city TSP problem using permuation coding, approximately 55 neighbors are obtained with SWAP. The decision of which neighborhood to move to in the next iteration is made based on the current solution, neighboring solutions and the procedure used. It is stuck at the local optimum point when there are no optimizer neighbors left for the solution. The local optimum problem can be solved by switching to a completely different solution, using iterative algorithms/procedures such as Iterated Local Search (ILS).

Termination of local search may be due to an iteration limit, not optimizing the best solution in a given iteration. Local search algorithms are typically approximate or incomplete algorithms, as the algorithm may terminate even before the solution found is global optimum. Because the global optimum solution may be too far from the current solution neighbors.

Local search algorithms that are frequently preferred in the literature:

- Hill Climbing Algorithm
- k-Opt Algorithm
- Simulated Annealing (SA) Algorithm
- Tabu Search (TS) Algorithm
- Variable Neighbourhood Search (VNS) Algorithm
- Guided Local Search (GLS) Algorithm

Hill Climbing Algorithm

Hill Climbing algorithm is a local search algorithm used for solving mathematical optimization problems. It is called Hill Climbing because it is based on searching for the highest points for a maximization problem on a two-dimensional graph. It is based on the principle of looking at the neighborhoods of the current solution, and when a better neighborhood is found, moving to that point and searching for the neighborhoods of that point. The Hill Climbing Algorithm pseudocode for a maximization problem is as given by Algorithm 5.1.

Algorithm 5.1. Pseudocode of Hill Climbing Algorithm for a Maximization Problem

```
1: i ← initial solution
2: while f(s) ≤ f(i)  ∀s ∈ Neighbours(i) do
3:     s ← ∃Neighbours(i)
4:     if f(s) > f(i) do
5:         i ← s
6:     end if
7: end while
```

k-Opt Algorithm

The k-Opt algorithm is a local search algorithm generally used for solving mathematical optimization problems using Graph Theory. The general usage in the literature is for solving the TSP problem. k is the number of arcs to be cut from the Graph created for problem solving. Neighboring solutions are obtained by connecting the cut arcs to each other in a different way. A typical two-opt algorithm, when n represents the total number of arcs, is as given by Algorithm 5.2.

Algorithm 5.2. Pseudocode of two-opt Algorithm for a Minimization Problem

```
1: define initial solution with Graph Theory
2: for (i = 1:n − 2); (j = i + 2:n) do
3:     d1 ← total lenght of 2 arc
4:     d2 ← total lenght of 2 arc when swaped
5:     if d1 > d2 do
6:         swap arcs
7:     end if
8: end while
```

Simulated Annealing (SA) Algorithm

A local search algorithm is often used for combinatorial optimization problems. The algorithm takes its name from annealing, a technique in metallurgical science where materials are heated and then controlled cooled to increase the size of crystals and reduce defects. Kirkpatrick et al. used to solve the TSP problem first introduced the Simulated Annealing (SA) algorithm in 1983 (Kirkpatrick et al., 1983). With the concept of slow cooling applied in the SA algorithm, as the solution space is explored, the rate of acceptance of bad solutions decreases. At the beginning of the algorithm, bad solutions are also accepted and it is desired to search for the global optimum point, but when the number of iterations increases and the temperature decreases, the aim is to search for the local optimum point instead of the global optimum. SA algorithm for a minimization problem s current solution, s' neighbour solution, T_0 initial temperature, T current temperature, T_f final temperature, $f(s)$ and $f(s')$ fitness value for candidate solutions, k temperature coefficient, Δ is the difference in fitness value between the current and neighboring solution and a is the temperature reduction coefficient as given by Algorithm 5.3.

Algorithm 5.3. Pseudocode of Simulated Annealing Algorithm for a Minimization Problem

```
s ← initial solution
T ← T_0
while T < T_f do
    for it = 1:nIT do
        s' ← Neighbour(s)
        if f(s') < f(s) do
            s ← s'
        else
            Δ ← f(s') − f(s)
            r ← random(0; 1)
            if r < exp(−Δ/(k×T)) do
                s ← s'
            end if
        end if
    end for
    T ← a × T
end while
```

Tabu Search (TS) Algorithm

Tabu Search (TS) algorithm is a local search algorithm used for solving mathematical optimization problems. It was first proposed by Fred W. Glover in 1986 (Glover, 1986) and developed in 1989 (Glover, 1989). As with other local search algorithms, the TS algorithm checks its neighbors to find a better result. If there is no improvement, that is, if it is stuck at the local optimum, the algorithm loosens the ground rule and starts to accept worse solutions. In addition, certain penalties (taboos) are applied to prevent visiting previously visited points. If a neighboring solution has been visited before in a short time, it is included in the Tabu List. This improves search performance. The TS algorithm for a minimization problem is as given by Algorithm 5.4 when s is the current solution, s' is the neighboring solution, $f(s)$ and $f(s')$ is the fitness value for the candidate solutions, and nT is the maximum number of Tabu List elements.

Algorithm 5.4. Pseudocode of Tabu Search Algorithm for a Minimization Problem

$s \leftarrow initial\ solution$
$Tabu\ List \leftarrow \emptyset$
for $it = 1{:}nIT$ **do**
    $s' \leftarrow Neighbour(s)$
    if $(s' \notin Tabu\ List) \cap (f(s') < f(s)$ **do**
        $s \leftarrow s'$
    end if
    $Tabu\ List \leftarrow Tabu\ List \cup s'$
    if $n(Tabu\ List) > nT$ **do**
        $Tabu\ List \leftarrow Tabu\ List - Tabu\ List\ (1)$
    end if
end for

Iterated Local Search (ILS) Algorithm

Iterated Local Search (ILS) is a local search algorithm often used for combinatorial optimizations problem solving. The typical feature of local search algorithms is that they are stuck at the local optimum point. A local search is performed, starting from different initial solutions, with a change made in the ILS algorithm each time. This is called "Iterated" local search and the information produced in previous local search stages is not used. Typically in the ILS algorithm:

1. The local optimum point is searched for the current solution (other local search algorithms can be used).

2. With Perturbation (Mutation), the existing solution is disrupted and a different point is obtained.
3. Step 1 is repeated.

The ILS algorithm for a minimization problem is as given by Algorithm 5.5 when s is the current solution, s' is the neighboring solution, $f(s)$ and $f(s')$ is the fitness value for the candidate solutions, and nIT is the maximum number of iterations.

Algorithm 5.5. Pseudocode of Iterated Local Search (ILS) Algorithm for a Minimization Problem

```
s ← initial solution
s ← Local Search (s)
for it = 1:nIT do
    s' ← Mutation(s)
    s' ← Local Search (s')
    if f(s') < f(s) do
        s ← s'
    end if
end for
```

Variable Neighbourhood Search (VNS) Algorithm

The Variable Neighborhood Search (VNS) algorithm is a local search algorithm that is generally used to solve combinatorial optimization problems. Mladenovic and Hansen first proposed it in 1997 (Mladenović & Hansen, 1997). It uses multiple neighborhood structures instead of using a single neighborhood structure as in other local search algorithms. Local search is performed with this randomly selected neighborhood structure in each iteration. The VNS algorithm for a minimization problem is as given by Algorithm 5.6 when s is the current solution, s' is the neighboring solution, $f(s)$ and $f(s')$ is the fitness value for the candidate solutions, and nIT is the maximum number of iterations.

Algorithm 5.6. Pseudocode of Variable Neighbourhood Search (VNS) Algorithm for a Minimization Problem

1: $s \leftarrow initial\ solution$
2: $N_{it} \quad \forall it \in \{1, \dots, nIT\}$
$\leftarrow set\ of\ neighbourhood\ structures\ for\ all\ iterations$
3: **for** $it = 1{:}\,nIT$ **do**
4: $s' \leftarrow Shaking(s, N_{it})$
5: $s'' \leftarrow Local\ Search\ (s')$
6: **if** $f(s'') < f(s)$ **do**
7: $s \leftarrow s''$
8: **end if**
9: **end for**

Guided Local Search (GLS) Algorithm

Guided Local Search (GLS) algorithm is a local search algorithm used for solving mathematical optimization problems. Penalties are created during the search in the GLS algorithm. It is similar to the TS algorithm due to its structure. However, while the solutions with penalties in the TS algorithm are the same, the penalties in the GLS algorithm are stuck at the local optimum point. The objective function is changed when the GLS is fitted at a local optimum point. The algorithm then uses an augmented objective function to extract the search from the local optimum point. The basic logic of GLS is to change the objective function according to the predetermined features. The GLS algorithm for a minimization problem is as given by Algorithm 5.7 when s is the current solution, s' is the neighboring solution, p_i is the penalty value for each feature, nF is the maximum number of features, and nIT is the maximum number of iterations.

Algorithm 5.7. Pseudocode of Guided Local Search Algorithm for a Minimization Problem

1: $s \leftarrow initial\ solution$
2: $p_i \leftarrow 0 \quad \forall i \in \{1, \dots, nF\}$
3: $h \leftarrow g + \lambda \times \sum p_i \times I_i$
4: **for** $it = 1{:}\,nIT$ **do**
5: $s' \leftarrow LocalSearch(s, h)$
6: $util_i \leftarrow I_i(s') \times \frac{c_i}{1 + p_i} \quad \forall i \in \{1, \dots, nF\}$
7: $p_i \leftarrow p_i + 1 \quad \forall i \in \{u_i\ is\ maximum \mid 1, \dots, nF\}$
8: $s \leftarrow s'$
9: **end for**

5.3. Memetic Algorithm (MA)

Memetic Algorithm (MA) is an extension of GA. A local search technique is used in addition to GA to reduce the possibility of early convergence (Cotta et al., 2018). MA represents one of the areas of research that has grown quite recently, as it has achieved quite good results by combining discrete individual learning and composite population learning in solving optimization problems.

Pablo Moscato introduced MA in 1989, influenced by Darwin's principles of natural evolution and Dawkins' concept of memes (P. Moscato & Cotta Porras, 2003). The MA is a GA combined with an individual learning procedure for which local improvements are made. In other words, it is a Hybrid Genetic Algorithm. Darwinian evolution on the one hand, Dawkinsian memes and local searches on the other, allow the use of two types of metaheuristic/heuristic algorithms together. In this way, a good balance is established between generality and problem specificity.

MAs are also searched in the literature with names such as Hybrid Evolutionary Algorithm, Baldwinian Evoluationary Algorithm, Lamarckian Evolutionary Algorithm, Cultural Algorithm and Genetic Local Search. Many different examples have been reported in various optimization problems that converge more efficiently and with higher quality than traditional evolutionary algorithms (Pablo Moscato & Mathieson, 2019).

An MA flowchart in general terms is as given in Figure 5.19.

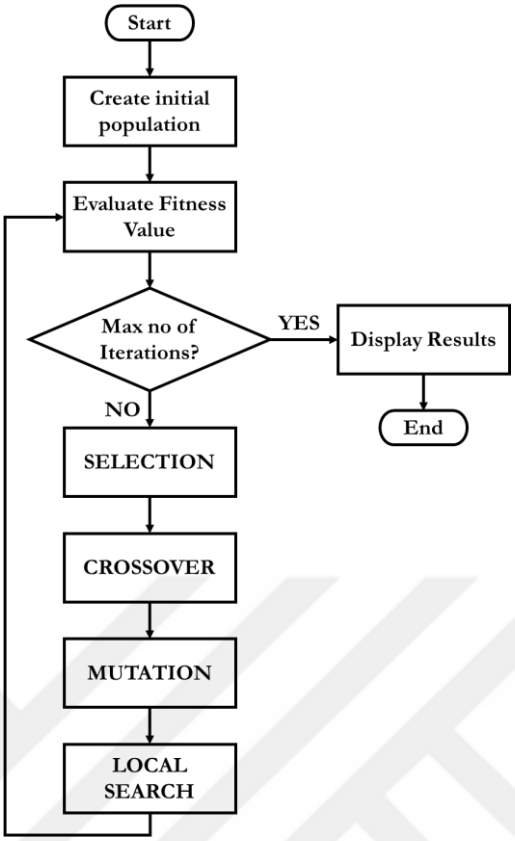


Figure 5.19. *Flowchart of Memetic algorithm*

The frequency and intensity of individual learning in MA provides greater chances of convergence with the local optimum, but computational difficulties limit the amount of evolution. Therefore, these two parameters should be finely tuned to maximize search performance and minimize computational cost. When individual learning is applied to some of the individuals in the population, it is necessary to make a good decision on which individuals to apply individual learning in order to maximize the benefit of MA.

How often to implement individual learning is still a researched issue in the literature. In some studies, it has been revealed that the effect of individual learning on each individual in each evolutionary stage/iteration, while in some studies it is thought that it is necessary to apply individual learning to each individual at each evolutionary stage. Within the scope of this thesis, it was decided to implement individual learning for each individual in the entire evolution stage/iteration.

Another problem is how long individual learning will take. That is, how long will be the maximum time allowed for the development of the individual solution taken into

individual learning. Within the scope of this thesis, 25% of the maximum number of generations/iterations is allocated.

Another unsolved problem that often makes MAs diversify is which self-learning method or meme to use. At this stage, one of the Local Search algorithms given in Section 5.2 or another local search algorithm in the literature is used. Within the scope of this thesis, MA was designed for several methods and the results were compared.

5.4. Multi Objective Metaheuristic Algorithms and Non-Dominated Sorting Genetic Algorithm – II (NSGA-II)

In this section, the basis of multi-objective metaheuristic algorithms and Non-Dominated Sorting Genetic Algorithm – II (NSGA-II), which is the most used multi-objective optimization heuristic algorithm in the literature, are explained.

5.4.1. Multi-objective Optimization Problem (MOP)

In general, a Multi-Objective Optimization Problem (MOP) occurs when $\bar{x} = [x_1, x_2, \dots, x_n]^T$ represents a vector of decision variables, vector g represents m inequality constraints, vector h represents p equality constraints, and vector f represents k objective functions, as given by Equation 5.4 – 5.6.

$$f(\bar{x}) = [f_1(\bar{x}), f_2(\bar{x}), \dots, f_k(\bar{x})]^T \tag{5.4}$$

Subject to;

$$g_i(\bar{x}) \leq 0 \qquad \forall i \in \{1, \dots, m\} \tag{5.5}$$

$$h_i(\bar{x}) = 0 \qquad \forall i \in \{1, \dots, p\} \tag{5.6}$$

In an optimization problem with several objective functions, the concept of optimum is different. Because in MOPs, good compromises or balances are tried to be found between the objective functions instead of a single solution as in the global optimum. Francis Ysidro Edgeworth proposed the most widely used optimum concept in 1881. Vifredo Pareto generalized this concept in 1896. This concept is generally referred to as Pareto Optimum or Edgeworth-Pareto Optimum in the literature. Considering any minimization problem, in order for the $\bar{x}^* \in F$ decision variable vector reached as a result of the solution to be Pareto Optimum, for all other $\bar{x} \in F$ decision variables vector, for

each objective function $i \in \{1, \ldots, k\}$, $f_i(\bar{x}^*) \leq f_i(\bar{x})$ and any objective function $j \in \{1, \ldots, k\}$ the conditions $f_i(\bar{x}^*) < f_i(\bar{x})$ must be satisfied. By this definition, it is understood that the Pareto Optimum solution is not a single solution and consists of a series of solutions called the Pareto Optimum Set. Non-dominant solutions are shown as Pareto Front in the graph of objective functions. For a typical two-objective optimization problem, the Pareto Front is as shown in Figure 5.20. Here, the x-axis represents the first objective function, the y-axis represents the second objective function, the green dots represent the non-dominated solutions, and the blue dots represent the dominated solutions.

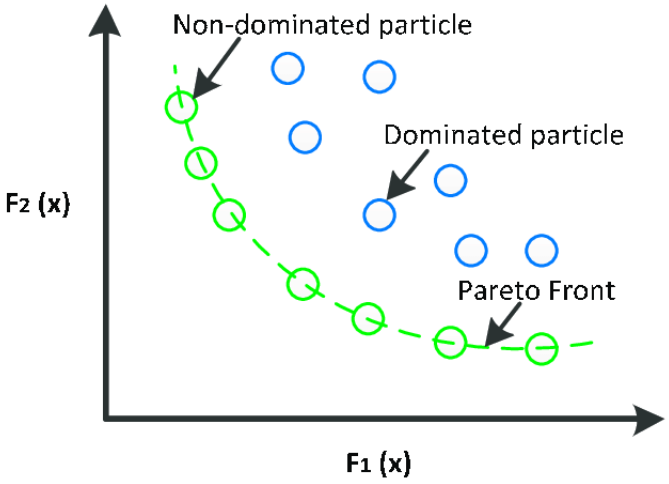


Figure 5.20. *A typical pareto front* (Mahesh et al., 2016)

In the literature, there are various mathematical programming-based approaches for solving multi-objective optimization problems. However, these methods are generally not used much, especially since they are very sensitive to the shape of the Pareto Front (when the Pareto Front is concave, etc.). On the other hand, metaheuristic methods are generally preferred for solving MOPs because they are less sensitive to the shape of the Pareto Front (works even when the Pareto Front is concave or discontinuous, etc.). It is possible to reach a few elements of Pareto Front in a single study with population-based metaheuristic approaches.

5.4.2. Non-dominated Sorted Genetic Algorithm – II (NSGA-II) for MOPs

Deb, Pratab and Agarwal proposed the NSGA-II algorithm in 2002. It is the most preferred algorithm for solving MOPs in the literature (Deb et al., 2002). In general, two

calculations are made in addition to the classical GA. With the Fast Non-Dominated Sort approach, the non-dominated frontier is found quickly, and with the Crowding-Distance Assignment, solutions are chosen.

5.4.2.1. Fast Non-Dominated Sort approach

In a simple approach, whether each solution in the population is dominant can be compared with all other solutions to determine the first non-dominant front. At this stage, all individuals in the first non-dominant front are found. For the next non-dominant front, individuals in the first front are temporarily removed from the population and the process to determine the first front is repeated. Considering that there is only one solution on each front, this will require a great deal of processing power. For this, a fast method is discussed in the NSGA-II algorithm. In this method, first two parameters are calculated for each individual:

- Domination count, which represents the number of solutions dominated
- Domination set that contains the solutions it dominates

For the first non-dominated front, the domination count counts as zero. Afterwards, if there are other frontier individuals in the domination set for each individual, the domination count of the individual is reduced by one. If any domination count is zero at this time, it is placed in a separate list. These members belong to other non-dominated front. This process continues until all fronts are determined. In general terms, the Fast Non-Dominated Sort Algorithm pseudocode is as given with Algorithm 5.8. Here, p and q represent each individual, P the set of individuals, n_p domination count, S_p domination set, F_i each front, p_{rank} the rank of each individual, that is, on which front they are located, and ⟨ domination.

Algorithm 5.8. Pseudocode of Fast Non-Dominated Sort Algorithm

```
for p ∈ P do
    S_p ← ∅
    n_p ← 0
    for q ∈ P do
        if p ⟨ q do
            S_p ← S_p ∪ {q}
        elseif q ⟨ p do
            n_p ← n_p + 1
        end if
    end for
    if n_p = 0 do
        p_rank ← 1
        F_1 ← F_1 ∪ {p}
    end if
    i ← 1
    while F_i ≠ ∅ do
        Q ← ∅
        for p ∈ F_i do
            for q ∈ S_p do
                n_q ← n_q − 1
                if n_q = 0 do
                    q_rank ← i + 1
                    Q ← Q ∪ {q}
                end if
            end for
        end for
        i ← i + 1
        F_i ← Q
    end while
end for
```

5.4.2.2. Crowding-Distance Assignment

The expectation from the algorithms is to maintain a good solution spread as well as converge to the Pareto-Optimal set. In the classical NSGA algorithm, sustainable diversity is provided in the population depending on a sharing parameter determined by the user. However, this brings with it a great computational cost in maintaining the propagation of the solutions depending on the value chosen and since each individual has to be compared with other individuals in the population.

Two difficulties were overcome with a non-user-defined approach in the NSGA-II algorithm. For this, the Density Estimation metric was used. This metric is the mean distance of the point on either side of that point along each target to estimate the density

of solutions located around a particular solution in the population. This is represented by a cuboid, where the nearest neighbors represent the vertices. The general representation of the approach is as given in Figure 5.21. Here, for individual i, the nearest neighbors $i-1$ and $i+1$ represent the corners of the cuboid. Dashed lines indicate the edges of the cuboid, and the Density Estimation metric is the average of the cuboid's edge lengths.

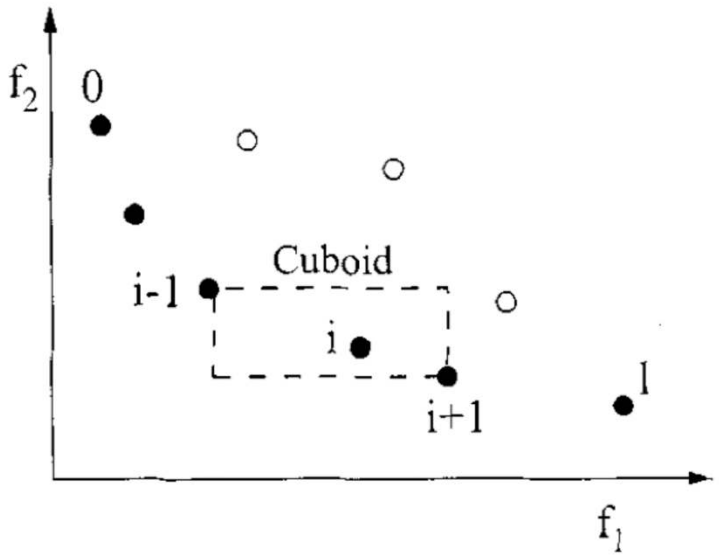


Figure 5.21. *Representation of crowding-distance computation*

Crowding-Distance Compuation requires the population to be ordered by each objective function value. For each objective function, the individuals with the smallest and largest values are assigned an infinite distance value. All remaining intermediate individuals are assigned a distance equal to the absolute value of the difference of functions of two adjacent individuals. This calculation is done for all objective functions. The overall Crowding-Distance value is the sum of the calculated Crowding-Distance values for each objective function. Crowding-Distance Assignment Algorithm pseudocode is as given with Algorithm 5.9. Here I is the solution of each individual, m is each objective function, $n0$ is the number of objective functions, and f_m^{max}, f_m^{min} is the maximum and minimum values for each objective function.

Algorithm 5.9. Pseudocode of Crowding-Distance Assignment Algorithm

1: $l \leftarrow |I|$
2: $I_{distance}(i) \leftarrow 0 \quad \forall i \in \{1, \dots, l\}$
3: **for** $m \in nO$ **do**
4: $I \leftarrow sort(I, m)$
5: $I_{distance}(1), I_{distance}(l) \leftarrow \infty$
6: **for** $i \in \{2, \dots, l-1\}$ **do**
7: $I_{distance}(i) \leftarrow I_{distance}(i) + \frac{(I(i+1).m - I(i-1).m)}{\left(f_m^{max} - f_m^{min}\right)}$
8: **end for**
9: **end for**

After assigning a crowding-distance metric to all individuals in the population, the crowding value between both solutions can be compared. An individual with a smaller value of this distance metric can be interpreted as being more crowded than other individuals.

5.4.2.3. Main Loop

Initially, a population of parents is created randomly or using a specific algorithm. The generated population is sorted by non-domination. From this step, each individual's fitness value is equal to their rank (1 - best, 2 - next best, etc.). To develop the solution, the offspring population is created by using the Selection, Crossover and Mutation operators in GA. The offspring and parent population are combined. The combined population is sorted by non-domination. Elitism is achieved at this point, as the previous and newly formed populations are involved. All non-domination fronteirs are determined in turn. After this stage, individuals in non-domination frontiers are selected for the parent population, respectively. If all individuals in the non-domination frontier are selected, if the parent population does not exceed the limit, all individuals are taken and passed to the other non-domination frontier. However, if all individuals in the non-domination frontier are selected, if the parent population limit is exceeded, the Crowding-Distance Assignment algorithm is activated for all individuals in that frontier. In this way, the best of the remaining individuals is tried to be selected. A typical representation of the NSGA-II algorithm is as shown in Figure 5.22. Here, P represents the parent population, Q represents the offspring population, $Rt = P \cup Q$ the combination of all populations, and F represents the Pareto Frontier.

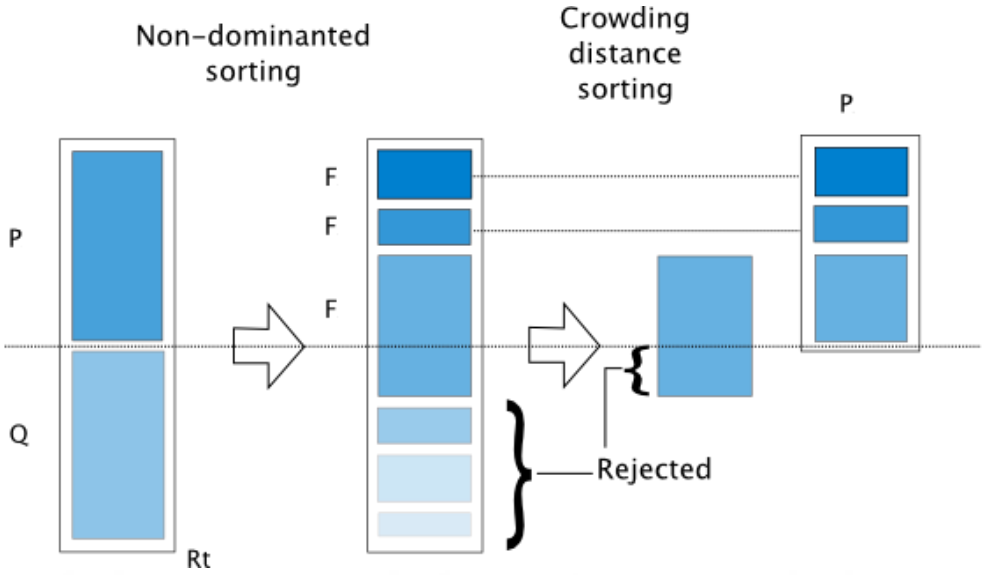


Figure 5.22. *Procedure of NSGA-II algorithm*

The pseudocode of the NSGA-II algorithm is as given by Algorithm 5.10. Where t is each iteration or generation, P_t is the parent population in each generation t, Q_t is the offspring population in each t generation, R_t is the total population in each t generation, and F_i represents the each i Pareto Frontier. The $MakeNewPopulation$ operation refers to the standard Selection - Crossover - Mutation operations in GA. $|P_t|$ is the size of the parent population in generation t, $|F_i|$ refers to the size of the each i Pareto Frontier.

Algorithm 5.10. Pseudocode of Non-dominated Sorted Genetic Algorithm – II (NSGA-II)

$t \leftarrow 0$
$P_t \leftarrow InitialPopulation$
$Q_t \leftarrow \emptyset$
while $t \leq nT$ **do**
    $R_t \leftarrow P_t \cup Q_t$
    $F \leftarrow FastNonDominatedSort(R_t)$
    $P_{t+1} \leftarrow \emptyset$
    $i \leftarrow 1$
    while $|P_{t+1}| + |F_i| \leq N$ **do**
        $CrowdingDistanceAssignment(F_i)$
        $P_{t+1} \leftarrow P_{t+1} \cup F_i$
        $i \leftarrow i + 1$
    end while
    $Sort(F_i, \langle_n)$
    $P_{t+1} \leftarrow P_{t+1} \cup F_i[1:(N - |P_{t+1}|)]$
    $Q_{t+1} \leftarrow MakeNewPopulation(P_{t+1})$
    $t \leftarrow t + 1$
end while

5.5. Multi Objective Memetic Algorithm (MOMA)

Multi Objective Memetic Algorithm (MOMA) developed within the scope of this study, Memetic Algorithm (MA) general flow given with Section 5.3, Iterated Local Search (ILS) Algorithm local search given with Section 5.2.5 and Non-Dominated Sorted Genetic Algorithm – II (NSGA-II) given with Section 5.4 is a combination of multi-objective structure. The developed algorithm performs both global and local searches for multi-objective optimization and searches the solution space more efficiently and more intelligently than other multi-objective optimization algorithms. The developed MOMA pseudocode is as given with Algorithm 5.11. All abbreviations here are the same as those defined for the Pseudocode of the NSGA-II algorithm.

Algorithm 5.11. Pseudocode of Multi Objective Memetic Algorithm (MOMA)

$t \leftarrow 0$
$P_t \leftarrow InitialPopulation$
$Q_t \leftarrow \emptyset$
while $t \leq nT$ **do**
    $R_t \leftarrow P_t \cup Q_t$
    $F \leftarrow FastNonDominatedSort(R_t)$
    $P_{t+1} \leftarrow \emptyset$
    $i \leftarrow 1$
    while $|P_{t+1}| + |F_i| \leq N$ **do**
        $CrowdingDistanceAssignment(F_i)$
        $P_{t+1} \leftarrow P_{t+1} \cup F_i$
        $i \leftarrow i + 1$
    end while
    $Sort(F_i, \langle_n)$
    $P_{t+1} \leftarrow P_{t+1} \cup F_i[1:(N - |P_{t+1}|)]$
    $Q'_{t+1} \leftarrow Selection(P_{t+1}, rank_{t+1})$
    $Q''_{t+1} \leftarrow Crossover(Q'_{t+1})$
    $Q'''_{t+1} \leftarrow Mutation(Q''_{t+1})$
    $Q_{t+1} \leftarrow IteratedLocalSearch(Q''_{t+1})$
    $t \leftarrow t + 1$
end while

The developed MOMA flowchart is as given in Figure 5.23.

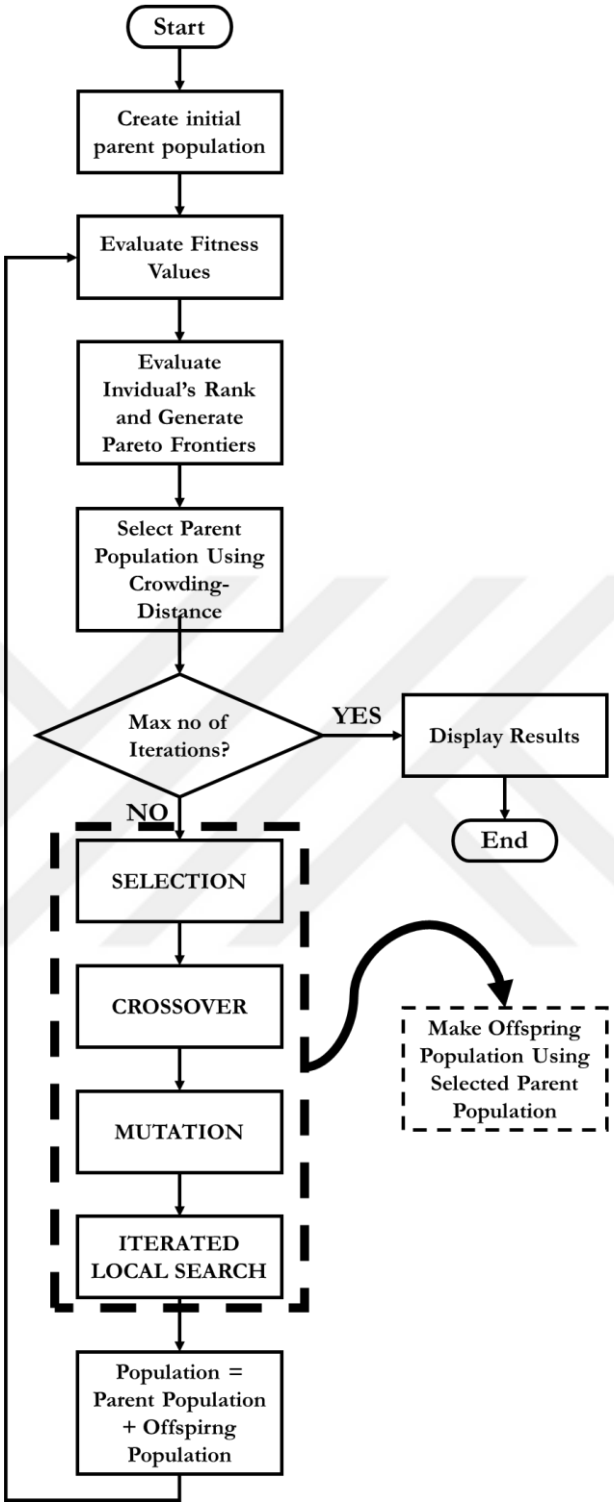


Figure 5.23. *Flowchart of MOMA*

5.6. Encoding/Decoding Algorithms for MTDLB Problem

In order to solve the MTDLB problem, which was defined in this thesis and whose MILP-based mathematical model was developed, with the MOMA developed within the scope of this thesis, the fitness values of the MTDLB problem should be calculated for

each individual in the population in the algorithm in each iteration/generation. For this, firstly, the chromosome structure of each individual was determined, then the disassembly sequence was determined for each model through the TAOG precedence diagram, which is the first stage of the problem, and then the selected tasks were determined by a procedure that allows very few stations to be opened on the disassembly line, again considering the TAOG precedence diagram has been appointed.

5.6.1. Chromosome Structure Phase

Each individual in the MOMA algorithm used to solve the MTDLB problem has the chromosome structure given in Figure 5.24. Chromosome consists of three parts. The first part is used for the selection of OR Successors on TAOG, which is encoded with binary coding. The second part is used to choose which side the selected tasks will be performed on. Here, if there is more than one feasible side for the task, it is done on the left side if the number it belongs to on the code is zero, and on the right if it is one. If there is only one, viable party for the task, a direct assignment is made to the party to which it belongs. The last part expresses the order of assignment of the selected tasks to the disassembly line and is coded with permuation coding. In this part, a list of tasks that can be assigned to the station is created, taking into account the TAOG precedence diagram for the tasks selected with the first part and assigned to the parties with the second part. If the list of tasks that can be assigned to the station is more than one, a task is selected by considering the priorities in the last section and assigned to the most suitable location. Afterwards, the process is repeated by updating the list of tasks that can be assigned, taking into account the TAOG precedence diagram and the assigned tasks.

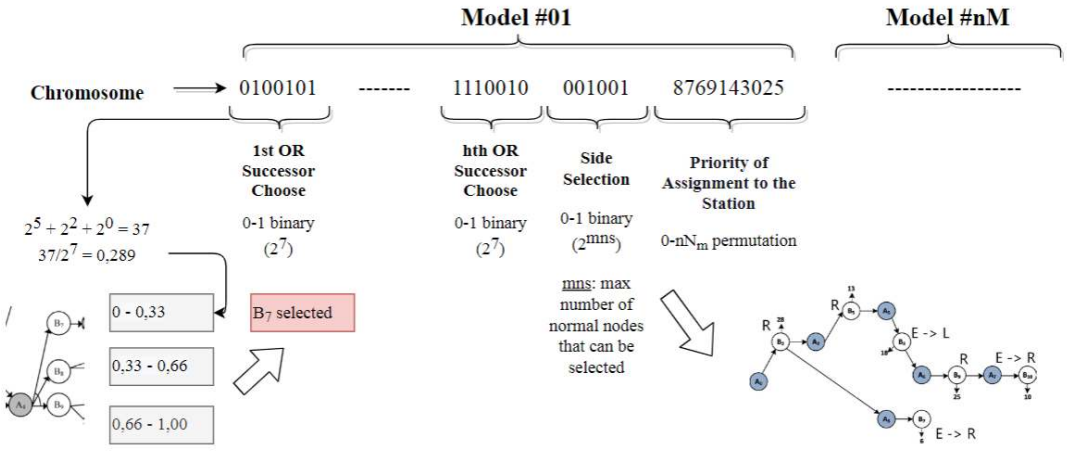


Figure 5.24. *Chromosome structure of proposed MOMA*

5.6.2. Choose Disassembly Sequence Phase

The first step in solving the MTDLB problem is to select the tasks for disassembly on the TAOG precedence diagram. Considering the chromosome structure given in Section 5.6.1, Choose Disassembly Sequence (Algo-CDS) is as given with Algorithm 5.12.

Algorithm 12. Choose Disassembly Sequence (Algo-CDS)

```
/* The algorithm must be repeated for each p invidual.        */
Data: nM, nA, SUC, PRE, dec_sol_p ;      /* Algo-CDS inputs */
Result: sc ;                             /* Algo-CDS output */
begin
    /* Start of Algo-CDS || Repeat for each m model.          */
    for m ← 1 to nM do
        /* Set initial values for each model m.               */
        sc_m ← ∅ ;                       /* Selected normal nodes */
        cA ← [A_0] ;                     /* Encountered artificial nodes */
        cOR ← 1 ;                        /* Encountered OR Successor relations */
        /* Continues until no artificial node is encountered
           */
        while cA ≠ ∅ do
            /* Assign assignable normal nodes for each
               encountered articial node.                     */
            for k ∈ cA do
                /* If there is OR Successor relation in the
                   encountered artificial node k, select normal
                   node using SUC_mk, dec_sol_pm and cOR,
                   otherwise select directly with SUC_mk1.    */
                if s(SUC_mk) > 1 and SUC_mk ∉ sc_m then
                    sc_m ← sc_m + {SUC_mk(ceil(s(SUC_mk) × dec_sol_pm(cOR)))};
                    cOR ← cOR + 1
                else if s(SUC_mk) = 1 and SUC_mk1 ∉ sc_m then
                    sc_m ← sc_m + {SUC_mk1};
            end
            /* Add the artificial nodes k in PRE_m to the cA,
               considering the normal nodes added to sc_m list
               in the last iteration.                         */
            /* If a normal node in SUC_mk of artificial nodes
               k in cA is selected in sc_m, remove the
               artifical node k from cA list.                 */
            Update cA with sc_m;
        end
    end
    return sc
end
```

5.6.3. Assign Disassembly Line Phase

The second step for solving the MTDLB problem is to assign the selected disassembly tasks to the appropriate stations on the disassembly line. Considering the

chromosome structure given in Section 5.6.1 and Algo-CDS given with Algorithm 1, the Assign Disassembly Line (Algo-ADL) is as given with Algorithm 5.13.

Algorithm 13. Assign Disassembly Line (Algo-ADL)

```
/* The algorithm must be repeated for each p invidual.    */
Data: nM, nA, nJ, nS, nORSuc, SUC, PRE, sol_p, sc, θ, t, C ;
      /* Algo-ADL inputs */
Result: X, U, tf ;                          /* Algo-ADL output */
begin
    X_{m(i∈sc_m)js} ← [0] ;  /* (0,1) assign of tasks to stations */
    U_{js} ← [0] ;                        /* (0,1) opening stations */
    tf_{m(i∈sc_m)} ← [0] ;   /* ≥ t_{m(i∈sc_m)} finishing time of tasks */
    task_pre, task_theta ← Algo-SPT ;      /* Set selected tasks'
     precedence and theta lists */
    /* Start of Algo-ADL || Repeat for each m model.      */
    for m ← 1 to nM do
        j ← 0 ;       /* Initial value of station number is 0 */
        assignable ← ∅;
        for i ∈ sc_m if task_pre_{mi} = ∅ then Append i in assignable;
        assigned ← ∅;
        while |selected_m| ≠ |assigned| do
            if |assignable| = 1 then
                r ← 0;
                j, X, U, tf, assignable, assigned ← Algo-AR
            else if |assignable| ≥ 2 then
                r ← 0 for l ← 1 to |assignable| do
                    if index(assignable_l, sol_{pm(nORSuc_m+2)}) <
                     index(assignable_r, sol_{pm(nORSuc_m+2)}) then
                        r ← l
                    end
                end
                j, X, U, tf, assignable, assigned ← Algo-AR
            for i ∈ sc_m do
                if i ∉ assigned ∩ i ∉ assignable then
                    for i' ∈ task_pre_{mi} do
                        if i' ∉ assigned then
                            Break
                        else if i' is last element in task_pre_{mi} then
                            Append i in assignable
                    end
                end
            end
        end
    end
    return X, U, tf
end
```

5.6.4. Main Representation Phase

Considering the MTDLB problem representation phases detailed in Section 5.6, an individual follows the flowchart given in Figure 5.25 to calculate the required fitness values.

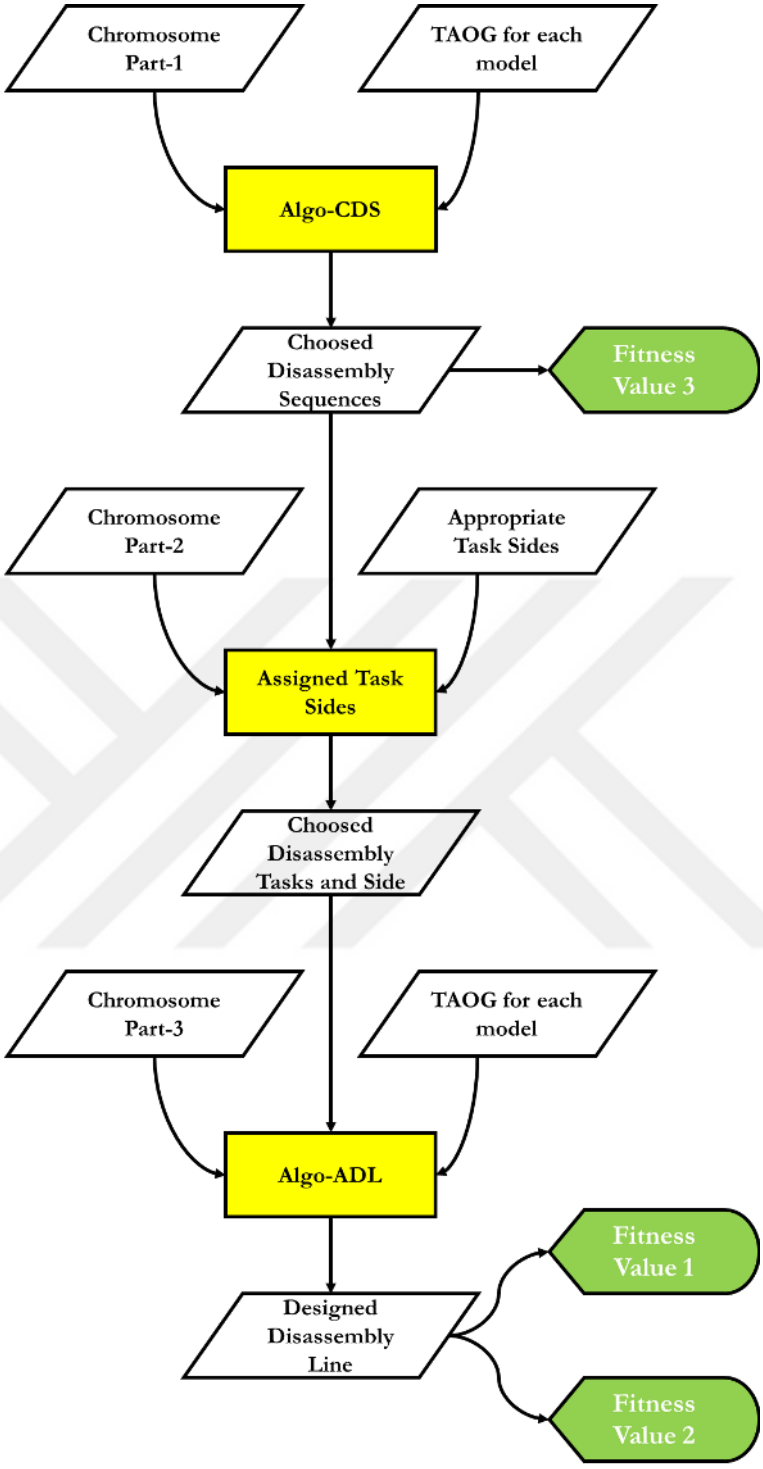


Figure 5.25. *Main representation of MTDLB problem for MOMA*

6. NUMERICAL EXAMPLES AND EXPERIMENTAL RESULTS

The MTDLB problem given in Section 4 is solved using MOMA given in Section 5 using some generated cases, literature works data, and the performance of the algorithm is tested. The MOMA algorithm developed for the solution of the MTDLB problem was coded using the Python programming language and given with Appendix-3.

In this section, the cases that were created were introduced, then the solutions were made on the created cases, and the solution results were given.

6.1. Generate Cases

Within the scope of this thesis, 11 cases with 2, 3 and 4 models were created by using Flashlight, Radio, Toy Car and Ball Point Pen products, for which the TAOG precedence diagram was previously created in the literature.

TAOG precedence diagram of Flashlight product is as given in Figure 6.1, TAOG precedence diagram of Radio product is given in Figure 6.2, TAOG precedence diagram of Toy Car product is given in Figure 6.3 and TAOG precedence diagram of Ball Point Pen product is given in Figure 6.4.

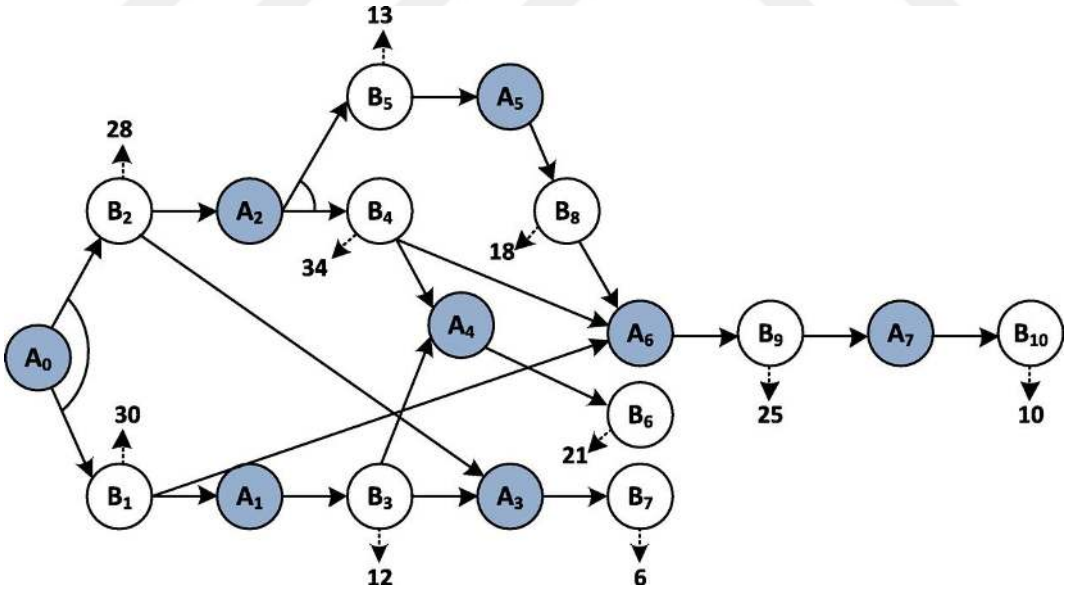


Figure 6.1. *TAOG precedence diagram for flashlight* (Paksoy et al., 2013)

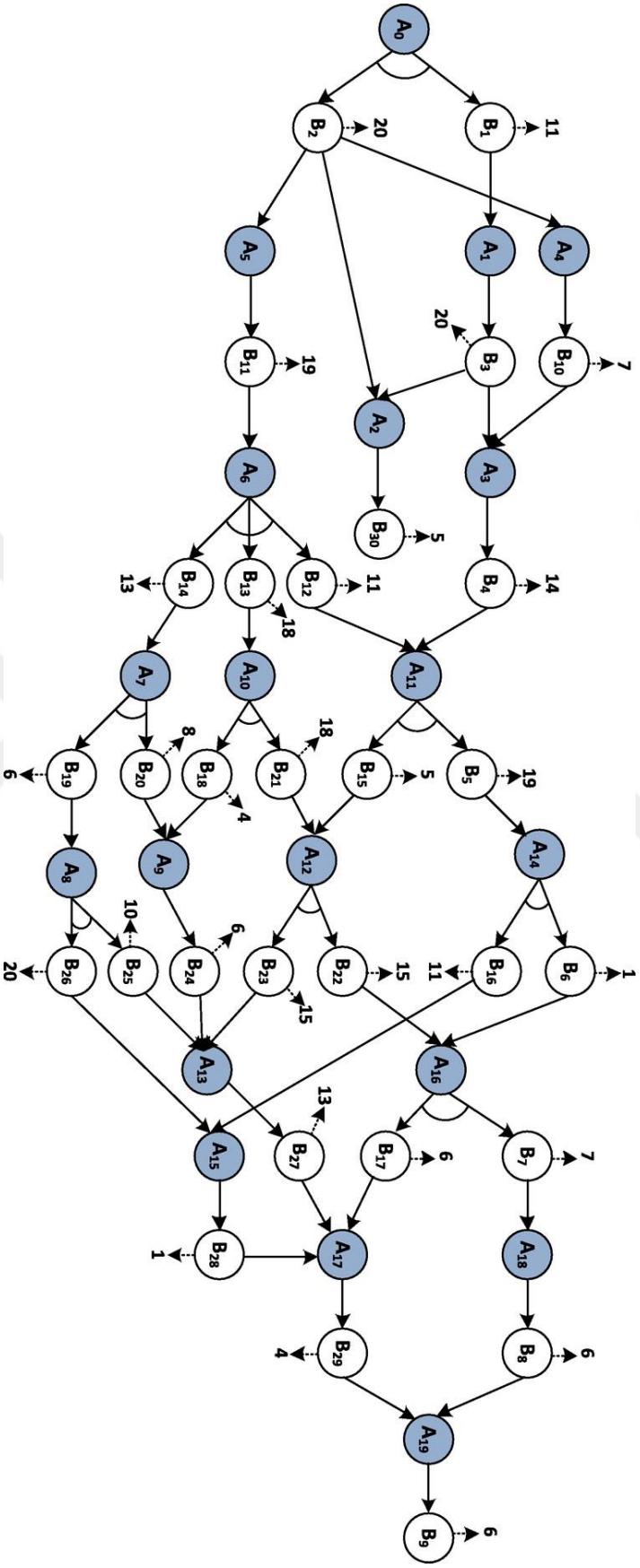


Figure 6.2. *TAOG precedence diagram for radio* (Paksoy et al., 2013)

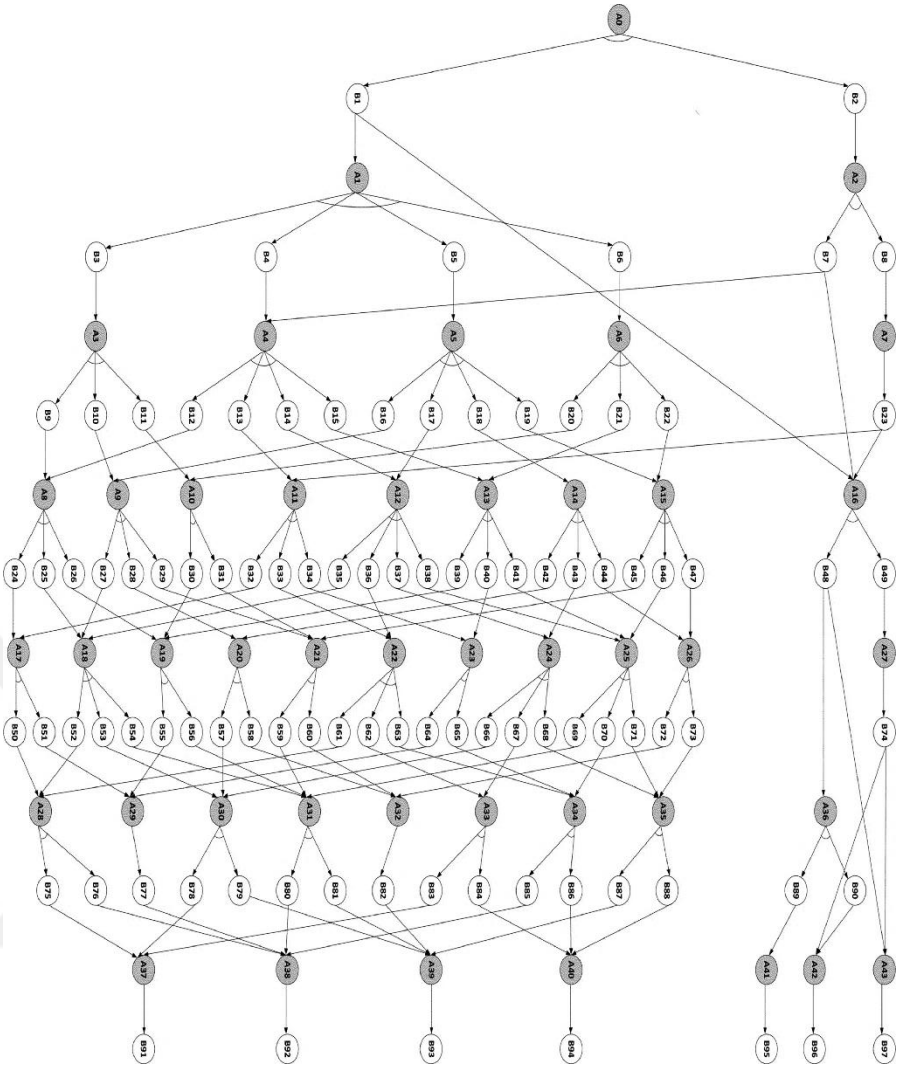


Figure 6.3. *TAOG precedence diagram for toy car* (Çil et al., 2020)

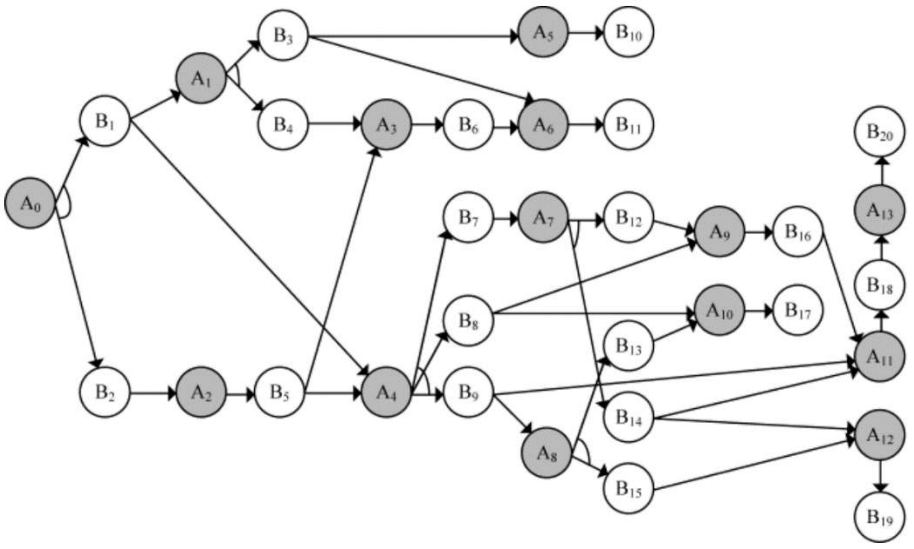


Figure 6.4. *TAOG precedence diagram for ball point pen*

Information such as party information, transaction time's information, etc. for all products are included in the Python code given with Appendix-3B. Statistical information for the 11 cases created is as given in Table 6.1.

Table 6.1. *Statistical information of generated cases*

Case Name	**Products**	$\sum nM$	$\sum nA$	$\sum nN$
#01	Flashlight Radio	2	30	40
#02	Flashlight Toy Car	2	53	107
#03	Flashlight Ball Point Pen	2	22	30
#04	Radio Toy Car	2	65	127
#05	Radio Ball Point Pen	2	34	50
#06	Toy Car Ball Point Pen	2	57	117
#07	Flashlight Radio Toy Car	3	74	137
#08	Flashlight Radio Ball Point Pen	3	43	60
#09	Flashlight Toy Car Ball Point Pen	3	66	127
#10	Radio Toy Car Ball Point Pen	3	78	147
#11	Flashlight Radio Toy Car Ball Point Pen	4	87	157

6.2. Results for Generated Cases

The MTDLB problem was solved in two ways for the created cases. Firstly, when the objective function is hierarchical, the results of the Memetic Algorithm (MA) are compared with the Genetic Algorithm (GA), which is frequently used in the literature, and the Gurobi solver, which gives optimum results. Secondly, the problem is solved with Multi Objective Memetic Algorithm (MOMA) by preserving its multi-objective structure and its results are compared with the NSGA-II algorithm, which is frequently used in the literature.

Each algorithm is coded in Python environment and given with Appendix 3. Caseler has been solved on a computer with Intel i7 2.00 GHz processor and 8 GB RAM.

In case the purpose is hierarchical, the solution results are as given in Table 6.2. Looking at the results, it is seen that MA gives better results than GA for each case, and even reaches the optimum result directly for some cases.

Table 6.2. *Comparative results of MA, GA and Gurobi (Obj Func — Sol Time).*

Case Name	Memetic Algorithm (Genetic Algorithm + Iterated Local Search)	Genetic Algorithm	Gurobi Solver (Optimum Result)
#01	**3.007190—13**	**3.007190—10**	**3.007190—01**
#02	**3.011274—15**	3.011285—11	**3.011274—10**
#03	**3.009240—14**	3.010243—09	**3.009240—01**
#04	3.013278—14	3.014266—15	**3.012250—43**
#05	3.011228—12	3.012234—12	**3.010229—01**
#06	**3.012310—15**	3.013294—14	**3.012310—25**
#07	3.012442—16	3.013344—13	**3.012360—11**
#08	3.011328—13	3.012336—12	**3.010329—02**
#09	**3.012413—16**	3.013492—15	**3.012413—10**
#10	3.012413—17	3.013374—16	**3.012399—15**
#11	3.012678—18	3.012793—17	**3.012499—28**

The results of the MOMA algorithm, which was developed by preserving the multi-objective structure of the problem, are given in Figure 6.5 for Case1 – Case6 and Figure 6.6 for Case7 – Case11.

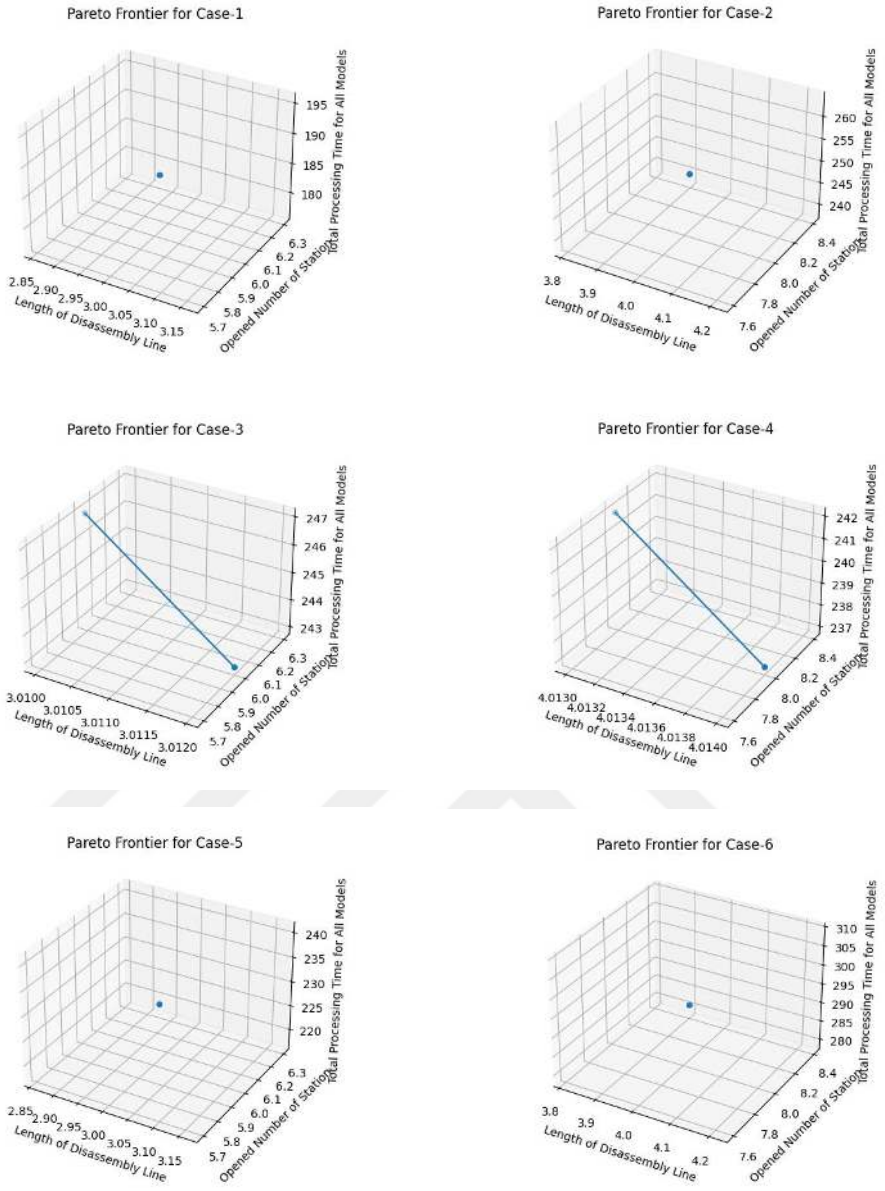


Figure 6.5. *Pareto Frontiers of MOMA for generated cases 1-6*

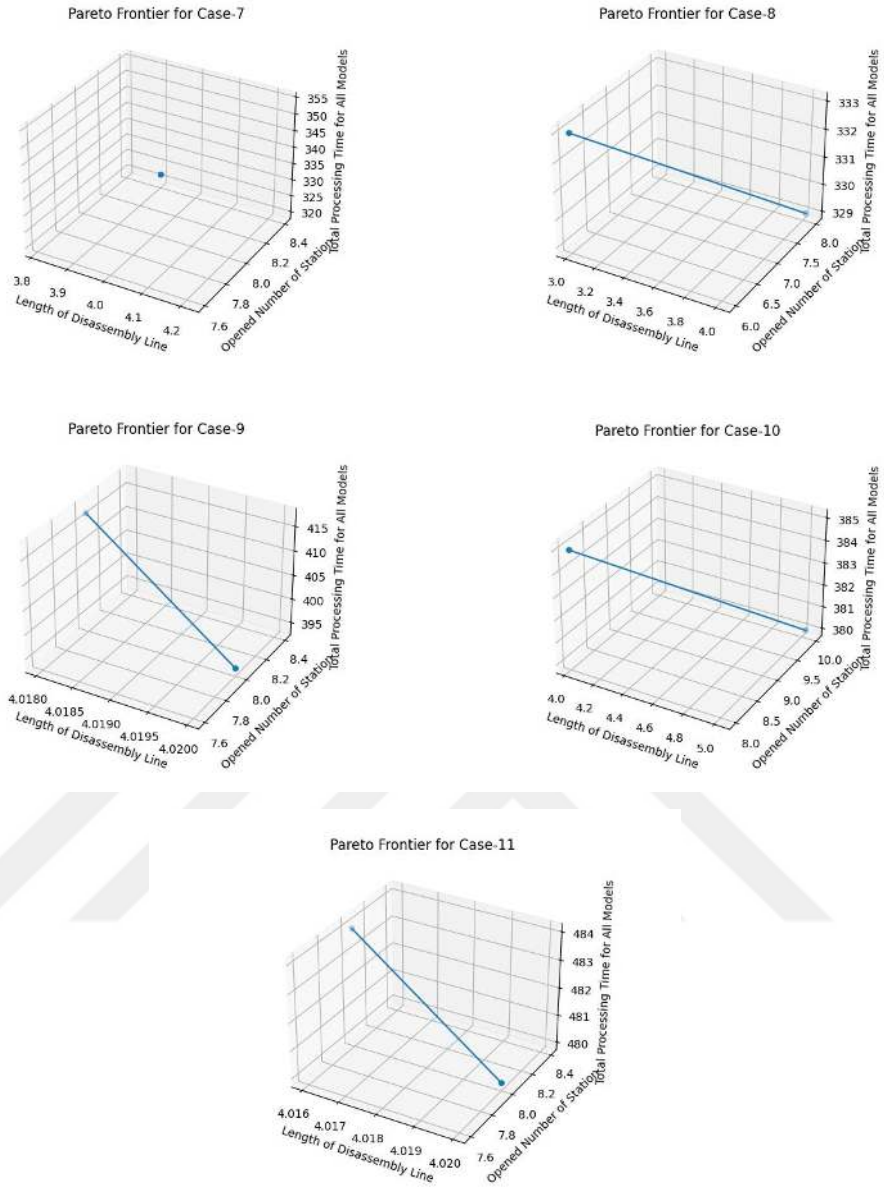


Figure 6.6. *Pareto Frontiers of MOMA for generated cases 7-11*

For comparison purposes, NSGA-II algorithm results are given in Figure 6.7 for Case1 – Case6 and Figure 6.8 for Case7 – Case11.

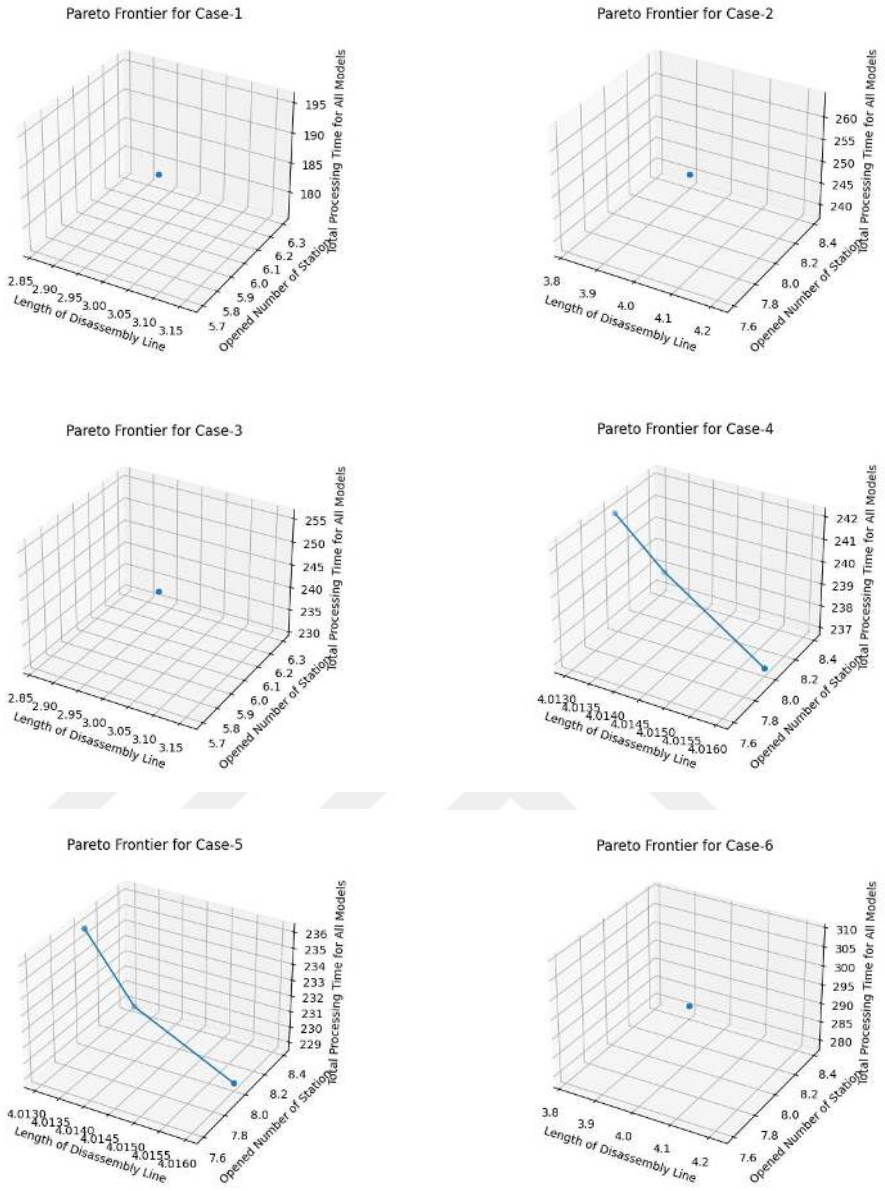


Figure 6.7. *Pareto Frontiers of NSGA-II for generated cases 1-6*

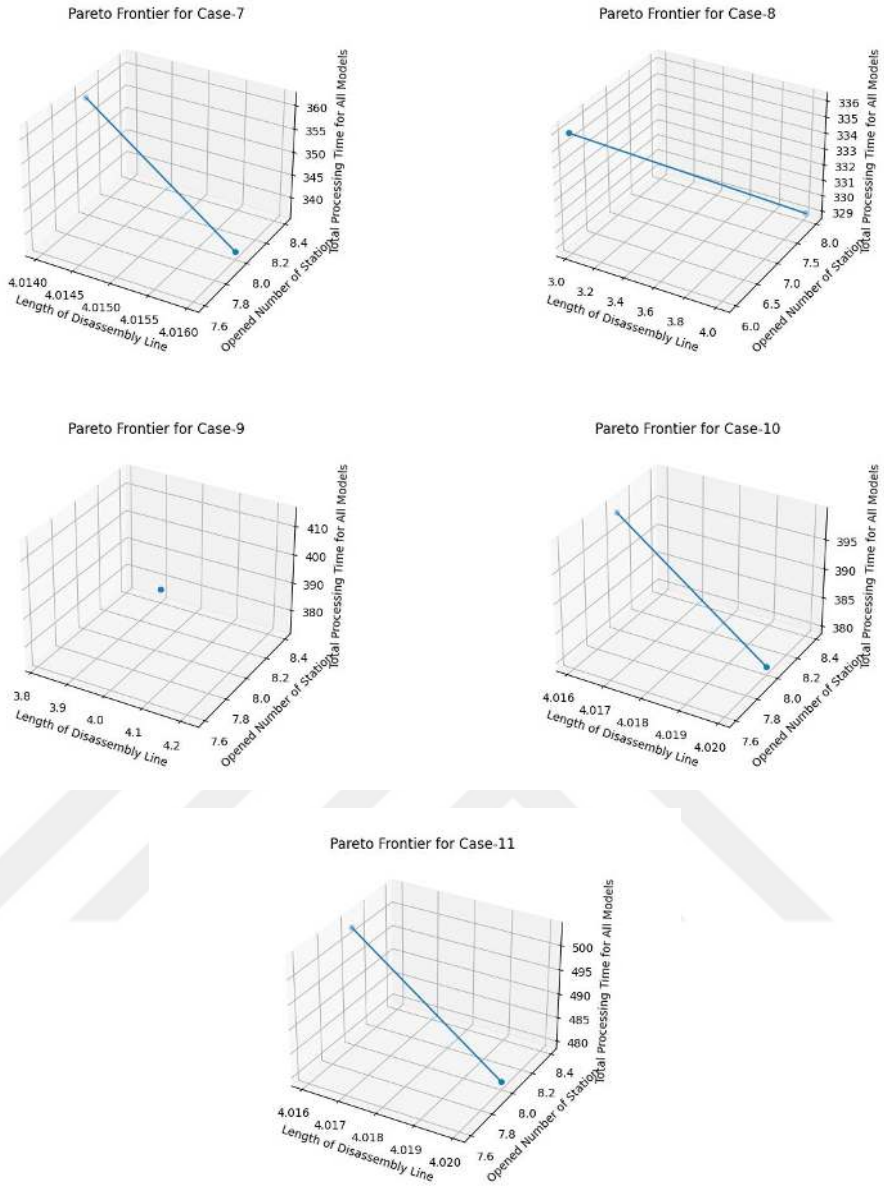


Figure 6.8. *Pareto Frontiers of NSGA-II for generated cases 7-11*

Looking at the Pareto Frontier graphs, it is seen that the NSGA-II algorithm generally produces 1 Pareto Optimum point for the solution of MTDLB problem cases, and in some cases, the Pareto Optimum solution found by MOMA dominates the Pareto Optimum solution found in the NSGA-II algorithm. It is seen that the MOMA algorithm reveals multiple Pareto Optimum points for each case and offers more options to the decision maker. Therefore, it can be said that the MOMA algorithm is better than the NSGA-II algorithm for solving the multi-objective MTDLB problem.

6.3. Compare with Literature Works

In order to see the situation of the study against other studies in the literature, the methods developed by considering the Kucukkoc, 2020 study, which is the closest study and which deals with the TDLB problem, were compared. However, since the AOG precedence diagram in Kucukkoc's 2020 study was handled differently than the diagram used in this thesis (like the precedence diagram of the 2P25 example given in Figure 6.8), the mathematical model was revised on Algo-CDS and Algo-ADL, and the studies were carried out accordingly way compared. In this context, Equation 6.1 instead of Equation 4.5, and Equation 6.2 instead of Equation 4.7.

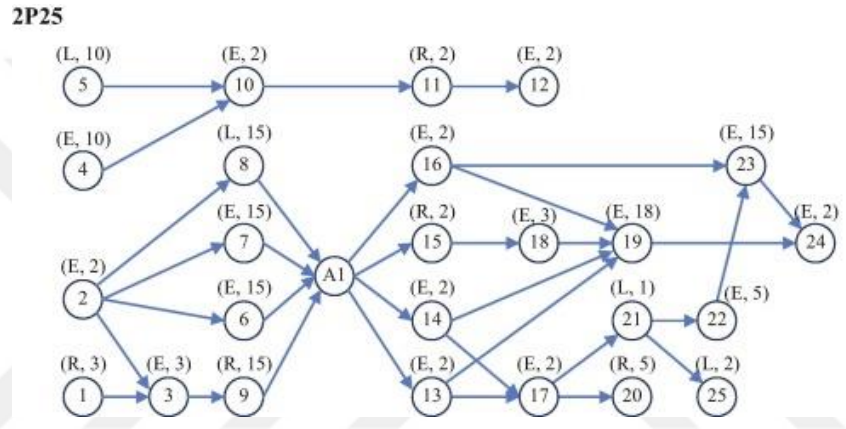


Figure 6.9. *Precedence diagram for 2P25* (Kucukkoc, 2020)

$$BM \times \sum_{i \in SUC(m,k)} z_{mi} \geq \sum_{i \in PRE(m,k)} z_{mi} \qquad \forall m \in \{1, \dots, nM\}, \forall k \in \{1, \dots, nA_m\} \tag{6.1}$$

$$\sum_{v=1}^{j} \sum_{s \in \theta(m,i)} x_{mivs} \geq \sum_{s \in \theta(m,i)} x_{mi'js} \qquad \forall m \in \{1, \dots, nM\}, \forall j \in \{1, \dots, nJ\}, \forall k \in \{1, \dots, nA_m\}, \forall i \in PRE(m,k), \forall i' \in SUC(m,k) \tag{6.2}$$

Since the (Kucukkoc, 2020) study was developed for a single purpose (number of stations opened and number of mated-stations opened), the MA algorithm developed within the scope of this thesis was evaluated in this direction. The Gurobi solution and the MA solution for the MTDLB mathematical model developed within the scope of this thesis were tested against the CPLEX solution and GA developed in the Küçükkoç 2020

study, and the results given in Table 6.3 were obtained. Considering the solution results, the Gurobi solution of the MTDLB mathematical model created within the scope of this thesis found better results than Küçükkoç's study (the optimum result was achieved), and the MA developed within the scope of this thesis gave very good results compared to the GA developed by Küçükkoç, even optimum for some cases. Appears to have achieved the result.

Table 6.3. *Comparative results with TDLB studies in literature (Number of Mated Station (Number of Total Station) — Sol Time); bold is optimum.*

Case	*C*	Gurobi	MA (GA + ILS)	Küçükkoç 2020 [10] CPLEX	2-GA
2P8	36	**5(6)**—≤0.1	**5(6)**—≤0.1	**5(6)**—≤1	**5(6)**—≤0.01
	38	**4(4)**—≤0.2	**4(4)**—≤0.1	**4(4)**—≤1	**4(4)**—≤0.01
	40	**4(4)**—≤0.1	**4(4)**—≤0.1	**4(4)**—≤1	**4(4)**—≤0.01
	38	**4(4)**—≤0.2	**4(4)**—≤0.1	**4(4)**—≤1	**4(4)**—≤0.01
2P10	36	**4(5)**—≤0.2	**4(5)**—≤0.1	**4(5)**—≤1	**4(5)**—≤0.01
	42	**3(5)**—≤0.5	**3(5)**—≤0.1	**3(5)**—≤1	**3(5)**—≤0.01
	44	**3(5)**—≤0.5	**3(5)**—≤0.1	**3(5)**—≤1	**3(5)**—≤0.01
	48	**2(4)**—≤0.5	**2(4)**—≤0.1	**2(4)**—≤1	**2(4)**—≤0.01
2P25	18	**5(9)**—2.49	**5(9)**—1	**5(9)**—110	6(10)—≤0.1
	24	**4(7)**—4.19	**4(7)**—2	**4(7)**—1800	**4(7)**—≤0.1
	28	**3(6)**—7.85	**3(6)**—1	**3(6)**—32	**3(6)**—≤0.1
2P47-A	98	4(8)—1800	5(8)—7	5(9)—1800	5(8)—≤1
	104	**4(7)**—209	4(8)—12	4(8)—1800	4(8)—≤1
	107	**4(7)**—150	4(8)—8	4(8)—1800	4(8)—≤1
	113	4(7)—1800	4(7)—11	4(7)—1800	4(7)—≤1

6.4. Case Study

Case #11, one of the cases created for a better understanding of the MTDLB problem solutions, was solved with MOMA and a possible Disassembly Line is given in Figure 6.10. Here, Task Station Assign Plan is shown above and Schedule of Disassembly Line below, respectively. In total, four models were assigned to six stations and all stations were opened to each other. Total line length was found to be three. In the Schedule of Disassembly Line section, the parts painted in blue represent the duties, the parts painted in yellow represent the lines that cannot be opened for that model, and the parts painted in red represent the idle time of the line.

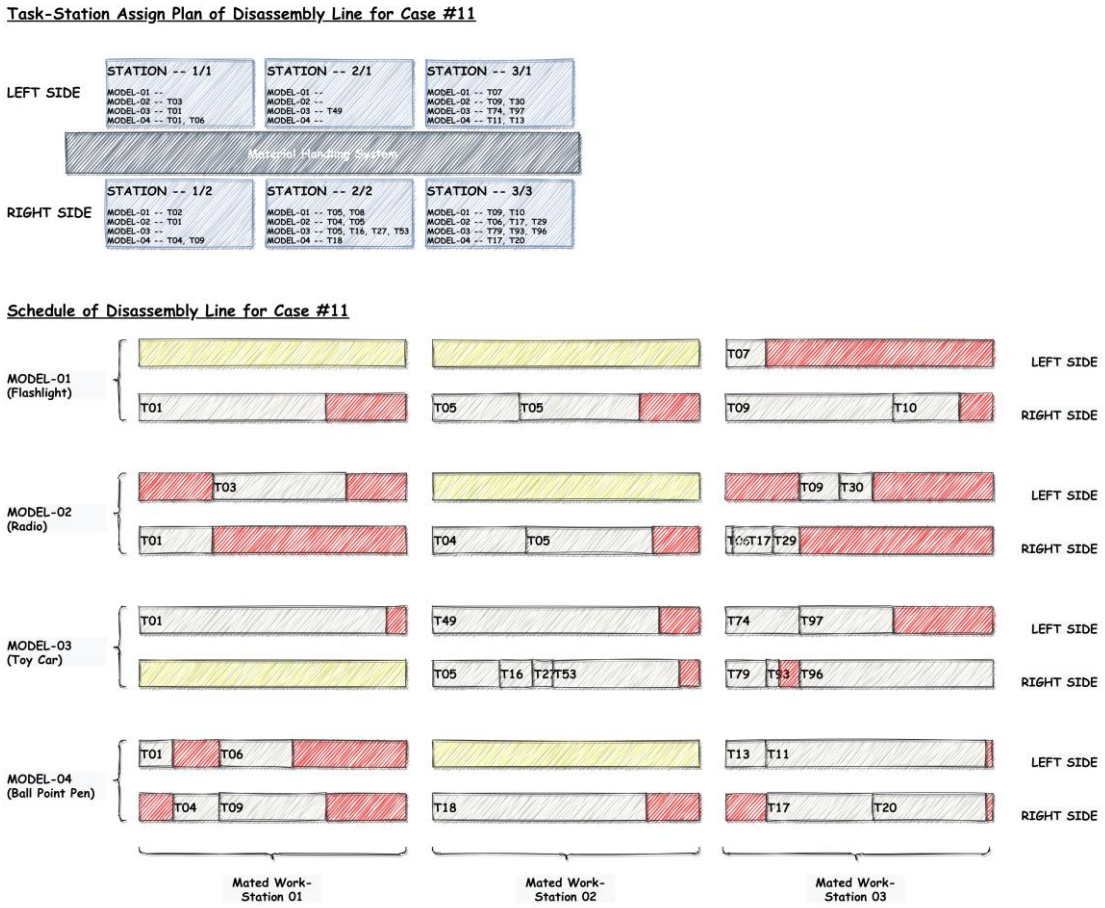


Figure 6.10. *Case study result (Case #11)*

7. CONCLUSION AND DISCUSSIONS

The fact that the increase in the level of Carbon Dioxide, which is visualized using technologies such as satellites, is at the peak of the last 800,000 years has revealed the necessity of radically changing our understanding of production. Contrary to popular belief, increasing population, welfare and technological developments increase the environmental impact even more. In order to minimize the environmental impact, it is necessary to use the resources efficiently and to minimize the wastes released into the nature. With the Life Cycle Engineering (LCE) methodology that emerged in this context, it is ensured that the environmental impact is minimized by directly intervening in all the Life Cycle stages of the product. One of these stages is the stage where the use of the product ends, that is, the product becomes End-of-Life (EOL). At this stage, the product is usually either completely released into the nature or recycled and the whole product is offered for use again. Product reuse is critical to minimizing environmental impact. At this point, with the LCE methodology, the product is designed as an easily disassembled product while it is still in the design phase. However, the design of the product alone is not enough, it is necessary to establish systems to disassemble these products and to minimize the environmental impact caused by the system by effectively managing them. In this context, the need for research on the Disassembly Line, which emerged for the disassembly of bulk EOL products, and the Disassemly Line Balancing (DLB) problem, which allows them to be designed and managed effectively, is increasing day by day. There are very few studies in the literature especially for the Two-Sided Disassembly Line Balancing (TDLB) problem, which can be used for disassembly of large volume products. Especially the Mixed-Model Two-Sided Disassembly Line Balancing (MTDLB) problem, which allows mixed model disassembly on these lines, was presented for the first time within the scope of this thesis.

In this study, a Mixed-Integer Linear Programming (MILP) based mathematical model has been developed for the MTDLB problem and a Multi Objective Memetic Algorithm (MOMA) has been developed to solve this model with high-dimensional cases. The developed mathematical model was tested with the Gurobi solver and the cases created by coding the MOMA and MA algorithms with Python and the literature comparison. It was found that all the methods developed within the scope of the study gave much better results compared to the GA and NSGA-II algorithms, which are

frequently used in the literature, and as a result of the literature comparison, optimum results were achieved that other studies could not reach.

In future studies, the developed MOMA algorithm will be made more efficient, it will be investigated how it gives results with different local search algorithms.

REFERENCES

Agrawal, S., & Tiwari, M. K. (2008). A collaborative ant colony algorithm to stochastic mixed-model U-shaped disassembly line balancing and sequencing problem. *International Journal of Production Research*, *46*(6), 1405–1429. https://doi.org/10.1080/00207540600943985

Albadr, M. A., Tiun, S., Ayob, M., & Al-Dhief, F. (2020). Genetic algorithm based on natural selection theory for optimization problems. Symmetry. https://doi.org/10.3390/sym12111758

Altekin, F. T. (2016). A Piecewise Linear Model for Stochastic Disassembly Line Balancing. *IFAC-PapersOnLine*. https://doi.org/10.1016/j.ifacol.2016.07.895

Altekin, F. T. (2017). A comparison of piecewise linear programming formulations for stochastic disassembly line balancing. *International Journal of Production Research*, *55*(24), 7412–7434. https://doi.org/10.1080/00207543.2017.1351639

Altekin, F. T., Kandiller, L., & Ozdemirel, N. E. (2004). Disassembly line balancing with limited supply and subassembly availability. In S. M. Gupta (Ed.), *Environmentally Conscious Manufacturing III* (pp. 59–70). https://doi.org/10.1117/12.516073

Altekin, F. T., Kandiller, L., & Ozdemirel, N. E. (2008). Profit-oriented disassembly-line balancing. *International Journal of Production Research*, *46*(10), 2675–2693. https://doi.org/10.1080/00207540601137207

Avikal, S. (2016). A heuristic based on AHP and TOPSIS for disassembly line balancing. *Advances in Intelligent Systems and Computing*. https://doi.org/10.1007/978-981-10-0448-3_69

Avikal, S., Jain, R., & Mishra, P. K. (2014). A Kano model, AHP and M-TOPSIS method-based technique for disassembly line balancing under fuzzy environment. *Applied Soft Computing*, *25*, 519–529. https://doi.org/10.1016/j.asoc.2014.08.002

Avikal, S., Jain, R., Yadav, H. C., & Mishra, P. K. (2014). A New Heuristic for Disassembly Line Balancing Problems with AND/O"R Precedence Relations.

Proceedings of the Second International Conference on Soft Computing, 519–525.

Avikal, S., Mishra, P. K., & Jain, R. (2014). A Fuzzy AHP and PROMETHEE method-based heuristic for disassembly line balancing problems. *International Journal of Production Research*, *52*(5), 1306–1317. https://doi.org/10.1080/00207543.2013.831999

Avikal, S., Sharma, S., Kalra, J. S., Varma, D., & Rohit Pandey. (2016). A Fuzzy AHP Approach for Calculating the Weights of Disassembly Line Balancing Criteria. In *Advances in Intelligent Systems and Computing* (Vol. 436, pp. 723–727). https://doi.org/10.1007/978-981-10-0448-3_60

Aydemir-Karadag, A., & Turkbey, O. (2013). Multi-objective optimization of stochastic disassembly line balancing with station paralleling. *Computers and Industrial Engineering*, *65*(3), 413–425. https://doi.org/10.1016/j.cie.2013.03.014

Battaïa, O., Dolgui, A., Heragu, S. S., Meerkov, S. M., & Tiwari, M. K. (2018). Design for manufacturing and assembly/disassembly: joint design of products and production systems. In *International Journal of Production Research*. https://doi.org/10.1080/00207543.2018.1549795

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2013). Chance Constrained Programming Model for Stochastic Profit–Oriented Disassembly Line Balancing in the Presence of Hazardous Parts. IFIP Advances in Information and Communication Technology, 414, 103–110. https://doi.org/10.1007/978-3-642-41266-0_13

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2013a). A cone programming approach for stochastic Disassembly Line Balancing in the presence of hazardous parts. *22nd International Conference on Production Research, ICPR 2013*.

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2013b). L-shaped Algorithm for Stochastic Disassembly Line Balancing Problem. *IFAC Proceedings Volumes*, *46*(9), 407–411. https://doi.org/10.3182/20130619-3-RU-3018.00500

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2013c). A decomposition method for stochastic partial disassembly line balancing with profit maximization. *2013*

IEEE International Conference on Automation Science and Engineering (CASE), 404–409. https://doi.org/10.1109/CoASE.2013.6654016

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2013d). A Stochastic Formulation of the Disassembly Line Balancing Problem. *IFIP Advances in Information and Communication Technology*, *397*(PART 1), 397–404. https://doi.org/10.1007/978-3-642-40352-1_50

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2013e). Chance Constrained Programming Model for Stochastic Profit–Oriented Disassembly Line Balancing in the Presence of Hazardous Parts. *IFIP Advances in Information and Communication Technology*, *414*, 103–110. https://doi.org/10.1007/978-3-642-41266-0_13

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2014a). Disassembly Line Balancing and Sequencing under Uncertainty. *Procedia CIRP*, *15*, 239–244. https://doi.org/10.1016/j.procir.2014.06.016

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2014b). Disassembly Line Balancing Problem with Fixed Number of Workstations under Uncertainty. *IFAC Proceedings Volumes*, *47*(3), 3522–3526. https://doi.org/10.3182/20140824-6-ZA-1003.02788

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2014c). Lagrangian Relaxation for Stochastic Disassembly Line Balancing Problem. *Procedia CIRP*, *17*, 56–60. https://doi.org/10.1016/j.procir.2014.02.049

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2014d). A sample average approximation method for disassembly line balancing problem under uncertainty. *Computers & Operations Research*, *51*, 111–122. https://doi.org/10.1016/j.cor.2014.05.006

Bentaha, M. L., Battaïa, O., & Dolgui, A. (2015). An exact solution approach for disassembly line balancing problem under uncertainty of the task processing times. *International Journal of Production Research*, *53*(6), 1807–1818. https://doi.org/10.1080/00207543.2014.961212

Bentaha, M. L., Dolgui, A., Battaïa, O., Riggs, R. J., & Hu, J. (2018). Profit-oriented partial disassembly line design: dealing with hazardous parts and task

processing times uncertainty. *International Journal of Production Research*, *56*(24), 7220–7242. https://doi.org/10.1080/00207543.2017.1418987

Budak, A. (2020). Sustainable reverse logistics optimization with triple bottom line approach: An integration of disassembly line balancing. *Journal of Cleaner Production*. https://doi.org/10.1016/j.jclepro.2020.122475

Cai, N., Zhang, Z., Zhang, Y., & Zhu, L. (2019). Modeling and Improved SFLA for Disassembly Line Balancing under Resource Constraints. *Zhongguo Jixie Gongcheng/China Mechanical Engineering*. https://doi.org/10.3969/j.issn.1004-132X.2019.17.011

Cai, N., Zhang, Z., Zou, B., & Li, L. (2019). Multi-objective disassembly line balancing optimization and analytic hierarchy process decision-making considering energy consumption. *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*. https://doi.org/10.13196/j.cims.2019.01.012

Cao, J., Xia, X., Wang, L., Zhang, Z., & Liu, X. (2019). A Novel Multi-Efficiency Optimization Method for Disassembly Line Balancing Problem. *Sustainability*, *11*(24), 6969. https://doi.org/10.3390/su11246969

Cevikcan, E., Aslan, D., & Yeni, F. B. (2020). Disassembly line design with multi-manned workstations: a novel heuristic optimisation approach. *International Journal of Production Research*, *58*(3), 649–670. https://doi.org/10.1080/00207543.2019.1587190

Chen, J. C., Chen, Y. Y., Chen, T. L., & Yang, Y. C. (2021). An adaptive genetic algorithm-based and AND/OR graph approach for the disassembly line balancing problem. *Engineering Optimization*. https://doi.org/10.1080/0305215X.2021.1957468

Chen, Q., Yao, B., & Pham, D. T. (2020). Sequence-dependent robotic disassembly line balancing problem considering disassembly path. *ASME 2020 15th International Manufacturing Science and Engineering Conference, MSEC 2020*. https://doi.org/10.1115/MSEC2020-8268

Cotta, C., Mathieson, L., & Moscato, P. (2018). Memetic algorithms. In Handbook of Heuristics. https://doi.org/10.1007/978-3-319-07124-4_29

ÇİL, Z. A. (2021). An exact solution method for multi-manned disassembly line design with AND/OR precedence relations. *Applied Mathematical Modelling*. https://doi.org/10.1016/j.apm.2021.07.013

Çil, Z. A., Mete, S., & Serin, F. (2020). Robotic disassembly line balancing problem: A mathematical model and ant colony optimization approach. *Applied Mathematical Modelling*. https://doi.org/10.1016/j.apm.2020.05.006

Dalle Mura, M., Pistolesi, F., Dini, G., & Lazzerini, B. (2021). End-of-life product disassembly with priority-based extraction of dangerous parts. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-020-01592-z

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation. https://doi.org/10.1109/4235.996017

Deniz, N., & Ozcelik, F. (2019). An extended review on disassembly line balancing with bibliometric & social network and future study realization analysis. In *Journal of Cleaner Production*. https://doi.org/10.1016/j.jclepro.2019.03.188

Ding, L. P., Feng, Y., Tan, J.-R., & Gao, Y. (2010). A new multi-objective ant colony algorithm for solving the disassembly line balancing problem. *The International Journal of Advanced Manufacturing Technology*, *48*(5–8), 761–771. https://doi.org/10.1007/s00170-009-2303-5

Ding, L. P., Tan, J. R., Feng, Y. X., & Gao, Y. C. (2009). Multiobjective optimization for disassembly line balancing based on Pareto ant colony algorithm. *Computer Integrated Manufacturing Systems*, *15*(7).

Diri Kenger, Z., Koç, Ç., & Özceylan, E. (2020). Integrated disassembly line balancing and routing problem. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2020.1740346

Diri Kenger, Z., Koç, Ç., & Özceylan, E. (2021). Integrated disassembly line balancing and routing problem with mobile additive manufacturing. *International Journal of Production Economics*. https://doi.org/10.1016/j.ijpe.2021.108088

Duta, L., Caciula, I., & Patic, P. C. (2016). Column generation approach for disassembly line balancing. *IFAC-PapersOnLine*, *49*(12), 916–920. https://doi.org/10.1016/j.ifacol.2016.07.892

Duta, L., Filip, F. G., & Caciula, I. (2008). Real Time Balancing of Complex Disassembly Lines. *IFAC Proceedings Volumes*, *41*(2), 913–918. https://doi.org/10.3182/20080706-5-KR-1001.00156

Duta, L., Filip, F. G., & Henrioud, J. M. (2005). APPLYING EQUAL PILES APPROACH TO DISASSEMBLY LINE BALANCING PROBLEM. *IFAC Proceedings Volumes*, *38*(1), 152–157. https://doi.org/10.3182/20050703-6-CZ-1902.01450

Edis, E. B. (2021). Constraint programming approaches to disassembly line balancing problem with sequencing decisions. *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2020.105111

Edis, E. B., Ilgin, M. A., & Edis, R. S. (2019). Disassembly line balancing with sequencing decisions: A mixed integer linear programming model and extensions. *Journal of Cleaner Production*, *238*, 117826. https://doi.org/10.1016/j.jclepro.2019.117826

Fang, Y., & Xu, H. (2020). Constraint Handling Methods for Resource-Constrained Robotic Disassembly Line Balancing Problem. *Journal of Physics: Conference Series*. https://doi.org/10.1088/1742-6596/1576/1/012039

Fang, Y., Liu, Q., Li, M., Laili, Y., & Pham, D. T. (2019). Evolutionary many-objective optimization for mixed-model disassembly line balancing with multi-robotic workstations. *European Journal of Operational Research*, *276*(1), 160–174. https://doi.org/10.1016/j.ejor.2018.12.035

Fang, Y., Ming, H., Li, M., Liu, Q., & Pham, D. T. (2020). Multi-objective evolutionary simulated annealing optimisation for mixed-model multi-robotic disassembly line balancing with interval processing time. *International Journal of Production Research*, *58*(3), 846–862. https://doi.org/10.1080/00207543.2019.1602290

Fang, Y., Ming, H., Li, M., Liu, Q., & Pham, D. T. (2020). Multi-objective evolutionary simulated annealing optimisation for mixed-model multi-robotic disassembly line balancing with interval processing time. International Journal of Production Research. https://doi.org/10.1080/00207543.2019.1602290

Fang, Y., Wei, H., Liu, Q., Li, Y., Zhou, Z., & Pham, D. T. (2019). Minimizing Energy Consumption and Line Length of Mixed-Model Multi-Robotic Disassembly Line Systems Using Multi-Objective Evolutionary Optimization. *ASME 2019*

14th International Manufacturing Science and Engineering Conference, *1*. https://doi.org/10.1115/MSEC2019-2773

Fang, Y., Xu, H., Liu, Q., & Pham, D. T. (2020). Evolutionary optimization using epsilon method for resource-constrained multi-robotic disassembly line balancing. *Journal of Manufacturing Systems*. https://doi.org/10.1016/j.jmsy.2020.06.006

Fang, Y., Zhang, H., Liu, Q., Zhou, Z., Yao, B., & Pham, D. T. (2020). Interval multi-objective evolutionary optimization for disassembly line balancing with uncertain task time. *ASME 2020 15th International Manufacturing Science and Engineering Conference, MSEC 2020*. https://doi.org/10.1115/MSEC2020-8265

Filipescu, A., Filipescu, A., Voda, A., & Minca, E. (2016). Hybrid modeling, balancing and control of a mechatronics line served by two mobile robots. *2016 20th International Conference on System Theory, Control and Computing, ICSTCC 2016 - Joint Conference of SINTES 20, SACCS 16, SIMSIS 20 - Proceedings*. https://doi.org/10.1109/ICSTCC.2016.7790671

Gao, K. Z., He, Z. M., Huang, Y., Duan, P. Y., & Suganthan, P. N. (2020). A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing. *Swarm and Evolutionary Computation*. https://doi.org/10.1016/j.swevo.2020.100719

Gao, Y., Wang, Q., Feng, Y., Zheng, H., Zheng, B., & Tan, J. (2018). An Energy-Saving Optimization Method of Dynamic Scheduling for Disassembly Line. *Energies*, *11*(5), 1261. https://doi.org/10.3390/en11051261

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. Computers and Operations Research. https://doi.org/10.1016/0305-0548(86)90048-1

Glover, F. (1989). Tabu Search—Part I. ORSA Journal on Computing. https://doi.org/10.1287/ijoc.1.3.190

Goksoy Kalaycilar, E., Batun, S., & Azizoğlu, M. (2021). A stochastic programming approach for the disassembly line balancing with hazardous task failures. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2021.1916119

Gupta, S. M., & Güngör, A. (2001). Product recovery using a disassembly line: challenges and solution. *Proceedings of the 2001 IEEE International Symposium on Electronics and the Environment. 2001 IEEE ISEE (Cat. No.01CH37190)*, 36–40. https://doi.org/10.1109/ISEE.2001.924499

Güngör, A., & Gupta, S. M. (1999a). Disassembly line balancing. Proceedings of the 1999 Annual Meeting of the Northeast Decision Sciences Institute, 193–195.

Güngör, A., & Gupta, S. M. (1999b). Disassembly Line Balancing. Proceedings of the 1999 Annual Meeting of the Northeast Decision Sciences Institute, 193–195.

Güngör, A., & Gupta, S. M. (2001). A solution approach to the disassembly line balancing problem in the presence of task failures. *International Journal of Production Research*, *39*(7), 1427–1467. https://doi.org/10.1080/00207540110052157

Güngör, A., & Gupta, S. M. (2002). Disassembly line in product recovery. International Journal of Production Research, 40(11), 2569–2589. https://doi.org/10.1080/00207540210135622

Güngör, A., Gupta, S. M., Pochampally, K., & Kamarthi, S. V. (2001). Complications in disassembly line balancing. *Proceedings of SPIE International Conference on Environmentally Concious Manufacturing*, *4193*, 289–298.

Habibi, M. K. K., Battaïa, O., Cung, V.-D., & Dolgui, A. (2014). Integrated Procurement–Disassembly Problem. In *IFIP Advances in Information and Communication Technology* (Vol. 439, Issue PART 2, pp. 482–490). https://doi.org/10.1007/978-3-662-44736-9_59

Hao, Y. K., & Hasan, S. (2016). The improvement of line efficiency on disassembly line balancing problem: An HRRCD's heuristic rule. *ARPN Journal of Engineering and Applied Sciences*, *11*(10), 6428–6433.

Hauschild, M. Z., Herrmann, C., & Kara, S. (2017). An Integrated Framework for Life Cycle Engineering. Procedia CIRP, 61, 2–9. https://doi.org/10.1016/j.procir.2016.11.257

He, J., Chu, F., Dolgui, A., Zheng, F., & Liu, M. (2021). Integrated stochastic disassembly line balancing and planning problem with machine specificity. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2020.1868600

He, J., Chu, F., Zheng, F., & Liu, M. (2021). A green-oriented bi-objective disassembly line balancing problem with stochastic task processing times. *Annals of Operations Research*. https://doi.org/10.1007/s10479-020-03558-z

He, J., Chu, F., Zheng, F., Liu, M., & Chu, C. (2020). A multi-objective distribution-free model and method for stochastic disassembly line balancing problem. *International Journal of Production Research*, *58*(18), 5721–5737. https://doi.org/10.1080/00207543.2019.1656841

Hezer, S., & Kara, Y. (2015). A network-based shortest route model for parallel disassembly line balancing problem. *International Journal of Production Research*, *53*(6), 1849–1865. https://doi.org/10.1080/00207543.2014.965348

Holdren, J. P. (1993). A Brief History of "IPAT" (IMPACT = POPULATION X AFFLUENCE X TECHNOLOGY). In Population and Sustainability (Vol. 2, Issue 2).

Homem de Mello, L.S., & Sanderson, A. C. (1990). AND/OR graph representation of assembly plans. IEEE Transactions on Robotics and Automation, 6(2), 188–199. https://doi.org/10.1109/70.54734

Homem de Mello, Luiz S., & Sanderson, A. C. (1991a). A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences. IEEE Transactions on Robotics and Automation. https://doi.org/10.1109/70.75905

Homem de Mello, Luiz S., & Sanderson, A. C. (1991b). Representations of Mechanical Assembly Sequences. IEEE Transactions on Robotics and Automation. https://doi.org/10.1109/70.75904

Hu, P., Chu, F., Fang, Y., & Wu, P. (2021). Novel distribution-free model and method for stochastic disassembly line balancing with limited distributional information. *Journal of Combinatorial Optimization*. https://doi.org/10.1007/s10878-020-00678-x

Igarashi, K., Yamada, T., & Inoue, M. (2014). 2-Stage Optimal Design and Analysis for Disassembly System with Environmental and Economic Parts Selection Using the Recyclability Evaluation Method. *Industrial Engineering and Management Systems*, *13*(1), 52–66. https://doi.org/10.7232/iems.2014.13.1.052

Igarashi, K., Yamada, T., Gupta, S. M., Inoue, M., & Itsubo, N. (2016). Disassembly system modeling and design with parts selection for cost, recycling and CO2 saving rates using multi criteria optimization. *Journal of Manufacturing Systems*, *38*, 151–164. https://doi.org/10.1016/j.jmsy.2015.11.002

Igarashi, K., Yamada, T., Itsubo, N., & Inoue, M. (2014). Optimal disassembly system design with environmental and economic parts selection for CO2 saving rate and recycling cost. *International Journal of Supply Chain Management*, *3*(3), 159–171.

Ilgin, M. A. (2019). A DEMATEL-Based Disassembly Line Balancing Heuristic. *Journal of Manufacturing Science and Engineering*, *141*(2). https://doi.org/10.1115/1.4041925

Ilgin, M. A., Akçay, H., & Araz, C. (2017). Disassembly line balancing using linear physical programming. *International Journal of Production Research*, *55*(20), 6108–6119. https://doi.org/10.1080/00207543.2017.1324225

Jia, L., & Shuwei, W. (2017). A proposed multi-objective optimization model for sequence-dependent disassembly line balancing problem. *2017 3rd International Conference on Information Management (ICIM)*, 421–425. https://doi.org/10.1109/INFOMAN.2017.7950420

Jiang, J., Zhang, Z., Xie, M., & Cai, N. (2021). Improved wolf pack algorithm for multi-objective disassembly line balancing problem under space constraints. *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*. https://doi.org/10.13196/j.cims.2021.06.004

Johar, B. O., & Gupta, S. M. (2006). Balancing inventory generated from a disassembly line: mathematical approach. In S. M. Gupta (Ed.), *Environmentally Conscious Manufacturing VI* (Vol. 6385, p. 638502). https://doi.org/10.1117/12.686072

Kalaycılar, E. G., Azizoğlu, M., & Yeralan, S. (2016). A disassembly line balancing problem with fixed number of workstations. *European Journal of Operational Research*, *249*(2), 592–604. https://doi.org/10.1016/j.ejor.2015.09.004

Kalayci, C. B., & Gupta, S. M. (2011). Tabu search for disassembly line balancing with multiple objectives. *41st International Conference on Computers and Industrial Engineering 2011*, 160–165.

Kalayci, C. B., & Gupta, S. M. (2013a). Balancing a sequencedependent disassembly line using simulated annealing algorithm. In *Applications of Management Science* (Vol. 16, pp. 81–103). Springer Netherlands. https://doi.org/10.1108/S0276-8976(2013)0000016008

Kalayci, C. B., & Gupta, S. M. (2013b). Simulated Annealing Algorithm for Solving Sequence-Dependent Disassembly Line Balancing Problem. *IFAC Proceedings Volumes*, *46*(9), 93–98. https://doi.org/10.3182/20130619-3-RU-3018.00064

Kalayci, C. B., & Gupta, S. M. (2013c). Ant colony optimization for sequence-dependent disassembly line balancing problem. *Journal of Manufacturing Technology Management*, *24*(3), 413–427. https://doi.org/10.1108/17410381311318909

Kalayci, C. B., & Gupta, S. M. (2013d). A particle swarm optimization algorithm with neighborhood-based mutation for sequence-dependent disassembly line balancing problem. *The International Journal of Advanced Manufacturing Technology*, *69*(1–4), 197–209. https://doi.org/10.1007/s00170-013-4990-1

Kalayci, C. B., & Gupta, S. M. (2013e). Artificial bee colony algorithm for solving sequence-dependent disassembly line balancing problem. *Expert Systems with Applications*, *40*(18), 7231–7241. https://doi.org/10.1016/j.eswa.2013.06.067

Kalayci, C. B., & Gupta, S. M. (2014). A tabu search algorithm for balancing a sequence-dependent disassembly line. *Production Planning & Control*, *25*(2), 149–160. https://doi.org/10.1080/09537287.2013.782949

Kalayci, C. B., Hancilar, A., Güngör, A., & Gupta, S. M. (2015). Multi-objective fuzzy disassembly line balancing using a hybrid discrete artificial bee colony algorithm. *Journal of Manufacturing Systems*, *37*, 672–682. https://doi.org/10.1016/j.jmsy.2014.11.015

Kalayci, C. B., Polat, O., & Gupta, S. M. (2015). A variable neighbourhood search algorithm for disassembly lines. *Journal of Manufacturing Technology Management*, *26*(2), 182–194. https://doi.org/10.1108/JMTM-11-2013-0168

Kalayci, C. B., Polat, O., & Gupta, S. M. (2016). A hybrid genetic algorithm for sequence-dependent disassembly line balancing problem. *Annals of Operations Research*, *242*(2), 321–354. https://doi.org/10.1007/s10479-014-1641-3

Kanagaraj, G., Naresh, R., & Yu, V. F. (2021). ENUMERATIVE SEARCH ALGORITHM FOR ROBOTIC DISASSEMBLY LINE BALANCING PROBLEM. *International Journal of Robotics and Automation*, *36*(1). https://doi.org/10.2316/J.2021.206-0427

Kannan, D., Garg, K., Jha, P. C., & Diabat, A. (2017). Integrating disassembly line balancing in the planning of a reverse logistics network from the perspective of a third party provider. *Annals of Operations Research*, *253*(1), 353–376. https://doi.org/10.1007/s10479-016-2272-7

Kaya, Ü., Koruca, H. İ., & Chehbi-Gamoura, S. (2020). Development of a Flexible Software for Disassembly Line Balancing with Heuristic Algorithms. In *Lecture Notes on Data Engineering and Communications Technologies*. https://doi.org/10.1007/978-3-030-36178-5_90

Kazancoglu, Y., & Ozkan-Ozen, Y. D. (2020). Sustainable disassembly line balancing model based on triple bottom line. *International Journal of Production Research*, *58*(14), 4246–4266. https://doi.org/10.1080/00207543.2019.1651456

Kazancoglu, Y., & Ozturkoglu, Y. (2018). Integrated framework of disassembly line balancing with Green and business objectives using a mixed MCDM. *Journal of Cleaner Production*, *191*, 179–191. https://doi.org/10.1016/j.jclepro.2018.04.189

Kim, Y. K., Song, W. S., & Kim, J. H. (2009). A mathematical model and a genetic algorithm for two-sided assembly line balancing. Computers and Operations Research. https://doi.org/10.1016/j.cor.2007.11.003

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. Science. https://doi.org/10.1126/science.220.4598.671

Kizilkaya, E. A., & Gupta, S. M. (2004). Modeling Operational Behavior of a Disassembly Line. In S. M. Gupta (Ed.), *Proceedings of the SPIE International Conference on Environmentally Conscious Manufacturing IV* (pp. 79–93). https://doi.org/10.1117/12.580419

Kizilkaya, E. A., & Gupta, S. M. (2005). Impact of different disassembly line balancing algorithms on the performance of dynamic kanban system for disassembly line. In S. M. Gupta (Ed.), *Environmentally Conscious Manufacturing V* (Vol. 5997, pp. 59970D-59970D – 8). https://doi.org/10.1117/12.631371

Koc, A., Sabuncuoglu, I., & Erel, E. (2009). Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an AND/OR graph. *IIE Transactions*, *41*(10), 866–881. https://doi.org/10.1080/07408170802510390

Kriengkorakot, N., & Pianthong, N. (1955). The assembly line balancing problem: Review Problem. Journal of Industrial Engineering, 6(3), 18–25. http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+Assembly+Line+Balancing+Problem+:#0%5Cnhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+assembly+line+balancing+problem%231

Kucukkoc, I. (2020). Balancing of two-sided disassembly lines: Problem definition, MILP model and genetic algorithm approach. *Computers & Operations Research*, *124*, 105064. https://doi.org/10.1016/j.cor.2020.105064

Kucukkoc, I., & Zhang, D. Z. (2015). Balancing of parallel U-shaped assembly lines. Computers and Operations Research, 64, 233–244. https://doi.org/10.1016/j.cor.2015.05.014

Kucukkoc, I., Li, Z., & Li, Y. (2019). Type-E disassembly line balancing problem with multi-manned workstations. *Optimization and Engineering*. https://doi.org/10.1007/s11081-019-09465-y

Laili, Y., Li, Y., Fang, Y., Pham, D. T., & Zhang, L. (2020). Model review and algorithm comparison on multi-objective disassembly line balancing. In *Journal of Manufacturing Systems*. https://doi.org/10.1016/j.jmsy.2020.07.015

Lambert, A. J. D. (2007). A HEURISTIC FOR ASSEMBLY AND DISASSEMBLY LINE BALANCING. *IFAC Proceedings Volumes*, *40*(2), 69–74. https://doi.org/10.3182/20070523-3-ES-4907.00013

Lambert, A. J. D., & Gupta, S. M. (2005a). A heuristic solution for the disassembly line balancing problem incorporating sequence dependent costs. In S. M. Gupta (Ed.), *Environmentally Conscious Manufacturing V* (Vol. 5997, pp. 59970A-59970A – 8). https://doi.org/10.1117/12.637368

Lambert, A. J. D., & Gupta, S. M. (2005b). Determining optimum and suboptimum disassembly sequences with an application to a cell phone. *(ISATP 2005). The 6th IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, 2005.*, *2005*, 260–265. https://doi.org/10.1109/ISATP.2005.1511483

Li, J., Chen, X., & Chu, C. (2019). The disassembly line design problem. *Proceedings of the 2019 International Conference on Industrial Engineering and Systems Management, IESM 2019*. https://doi.org/10.1109/IESM45758.2019.8948107

Li, J., Chen, X., Zhu, Z., Yang, C., & Chu, C. (2019). A branch, bound, and remember algorithm for the simple disassembly line balancing problem. *Computers and Operations Research*, *105*, 47–57. https://doi.org/10.1016/j.cor.2019.01.003

Li, L., Zhang, Z., Guan, C., & Jia, L. (2018). Multi-objective Optimization for Partial Disassembly Line Balancing with Goal- driven Discrete Cuckoo Search. *Jisuanji Fuzhu Sheji Yu Tuxingxue Xuebao/Journal of Computer-Aided Design and Computer Graphics*. https://doi.org/10.3724/SP.J.1089.2018.16439

Li, L., Zhang, Z., Hu, Y., & Zou, B. (2017). Optimization of Disassembly Line Balancing Problems with Setup Times Based on Multi-objective Algorithm and Dynamic Simulation. *Zhongguo Jixie Gongcheng/China Mechanical Engineering*. https://doi.org/10.3969/j.issn.1004-132X.2017.17.016

Li, L., Zhang, Z., Zhu, L., & Zou, B. (2018). Modeling and Optimizing for Multi-objective Partial Disassembly Line Balancing Problem. *Jixie Gongcheng Xuebao/Journal of Mechanical Engineering*. https://doi.org/10.3901/JME.2018.03.125

Li, X., Laili, Y., Zhang, L., & Ren, L. (2019). Modeling for component relations in robotic disassmebly. *31st European Modeling and Simulation Symposium, EMSS 2019*. https://doi.org/10.46354/i3m.2019.emss.036

Li, Z., & Janardhanan, M. N. (2021). Modelling and solving profit-oriented U-shaped partial disassembly line balancing problem. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2021.115431

Li, Z., Çil, Z. A., Mete, S., & Kucukkoc, I. (2020). A fast branch, bound and remember algorithm for disassembly line balancing problem. *International Journal of Production Research*, *58*(11), 3220–3234. https://doi.org/10.1080/00207543.2019.1630774

Li, Z., Kucukkoc, I., & Zhang, Z. (2019). Iterated local search method and mathematical model for sequence-dependent U-shaped disassembly line balancing problem. *Computers & Industrial Engineering*, *137*, 106056. https://doi.org/10.1016/j.cie.2019.106056

Liang, J., Guo, S., Du, B., Li, Y., Guo, J., Yang, Z., & Pang, S. (2021). Minimizing energy consumption in multi-objective two-sided disassembly line balancing problem with complex execution constraints using dual-individual simulated annealing algorithm. *Journal of Cleaner Production*, *284*, 125418. https://doi.org/10.1016/j.jclepro.2020.125418

Liang, P., Fu, Y., Gao, K., & Sun, H. (2021). An enhanced group teaching optimization algorithm for multi-product disassembly line balancing problems. *Complex & Intelligent Systems*. https://doi.org/10.1007/s40747-021-00478-8

Lindsey, R. (2020). Climate Change: Atmospheric Carbon Dioxide. NOAA Climate.Gov.

Liu, B. Y., Chen, W. Da, & Huang, S. (2013). Research on disassembly line balancing problem in the presence of uncertain cycle time. *International Asia Conference on Industrial Engineering and Management Innovation: Core Areas of Industrial Engineering, IEMI 2012 - Proceedings*, 497–507. https://doi.org/10.1007/978-3-642-38445-5-51

Liu, B., Xu, W., Liu, J., Yao, B., Zhou, Z., & Pham, D. T. (2019). Human-Robot Collaboration for Disassembly Line Balancing Problem in Remanufacturing. *ASME 2019 14th International Manufacturing Science and Engineering Conference, MSEC 2019*, *1*. https://doi.org/10.1115/MSEC2019-2919

Liu, Jia, & Wang, S. (2017). Balancing Disassembly Line in Product Recovery to Promote the Coordinated Development of Economy and Environment. *Sustainability*, *9*(2), 309. https://doi.org/10.3390/su9020309

Liu, Jia, & Wang, S. W. (2018). A hybrid artificial bee colony algorithm for solving sequence-dependent disassembly line balancing problem. *Kongzhi Yu Juece/Control and Decision*. https://doi.org/10.13195/j.kzyjc.2017.0003

Liu, Jiayi, Zhou, Z., Pham, D. T., Xu, W., Ji, C., & Liu, Q. (2020). Collaborative optimization of robotic disassembly sequence planning and robotic disassembly line balancing problem using improved discrete Bees algorithm in remanufacturing☆. *Robotics and Computer-Integrated Manufacturing*, *61*, 101829. https://doi.org/10.1016/j.rcim.2019.101829

Liu, Jiayi, Zhou, Z., Pham, D. T., Xu, W., Yan, J., Liu, A., Ji, C., & Liu, Q. (2018). An improved multi-objective discrete bees algorithm for robotic disassembly line balancing problem in remanufacturing. *The International Journal of Advanced Manufacturing Technology*, *97*(9–12), 3937–3962. https://doi.org/10.1007/s00170-018-2183-7

Liu, M., Liu, R., & Chu, F. (2019). Distribution-free and Risk-averse Disassembly Line Balancing Problem. *Proceedings of the 2019 International Conference on Industrial Engineering and Systems Management, IESM 2019*. https://doi.org/10.1109/IESM45758.2019.8948079

Liu, M., Liu, X., Chu, F., Zheng, F., & Chu, C. (2020). Robust disassembly line balancing with ambiguous task processing times. *International Journal of Production Research*, *58*(19), 5806–5835. https://doi.org/10.1080/00207543.2019.1659520

Liu, M., Liu, X., Chu, F., Zheng, F., & Chu, C. (2021). An exact method for disassembly line balancing problem with limited distributional information. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2019.1704092

Liu, M., Liu, Z., Liu, X., & Chu, F. (2019). Entropy-based bi-objective disassembly line balancing problem. *Proceedings of the 2019 International Conference on Industrial Engineering and Systems Management, IESM 2019*. https://doi.org/10.1109/IESM45758.2019.8948166

Liu, Q., Li, Y., Fang, Y., Laili, Y., Lou, P., & Pham, D. T. (2019). Many-objective best-order-sort genetic algorithm for mixed-model multi-robotic disassembly line balancing. *Procedia CIRP*, *83*, 14–21. https://doi.org/10.1016/j.procir.2019.04.076

Liu, X., Chu, F., Zheng, F., Chu, C., & Liu, M. (2021). Distributionally robust and risk-averse optimisation for the stochastic multi-product disassembly line balancing problem with workforce assignment. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2021.1881648

Liuke, U., Zeqiang, Z., Binsen, Z., & Ning, C. (2018). Optimization of Multi-objective Disassembly Line Balancing Problem Using Immune Mechanism Cooperative Genetic Algorithm. *Information and Control*. https://doi.org/10.13976/j.cnki.xk.2018.7217

Luo, W., Zhou, M. C., Guo, X. W., Wei, H., Qi, L., & Zhao, Z. (2020). Improved Artificial Bee Colony Algorithm for Solving a Single-Objective Sequence-dependent Disassembly Line Balancing Problem. *2020 IEEE International Conference on Networking, Sensing and Control, ICNSC 2020*. https://doi.org/10.1109/ICNSC48988.2020.9238075

Mahesh, K., Nallagownden, P., & Elamvazuthi, I. (2016). Advanced Pareto front non-dominated sorting multi-objective particle swarm optimization for optimal placement and sizing of distributed generation. Energies. https://doi.org/10.3390/en9120982

McGovern, S. M., & Gupta, S. M. (2003). Greedy algorithm for disassembly line scheduling. *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483)*, *2*, 1737–1744. https://doi.org/10.1109/ICSMC.2003.1244663

McGovern, S. M., & Gupta, S. M. (2004a). Demanufacturing strategy based upon metaheuristics. *IIE Annual Conference and Exhibition 2004*.

McGovern, S. M., & Gupta, S. M. (2004b). 2-opt heuristic for the disassembly line balancing problem. In S. M. Gupta (Ed.), *Environmentally Conscious Manufacturing III* (Vol. 5262, pp. 71–84). https://doi.org/10.1117/12.516155

McGovern, S. M., & Gupta, S. M. (2004c). Combinatorial optimization methods for disassembly line balancing. In S. M. Gupta (Ed.), *Environmentally Conscious Manufacturing IV* (pp. 53–66). https://doi.org/10.1117/12.570493

McGovern, S. M., & Gupta, S. M. (2005a). Local search heuristics and greedy algorithm for balancing a disassembly line. *International Journal of Operations and Quantitative Management*, *11*(2), 91–114.

McGovern, S. M., & Gupta, S. M. (2005b). Uninformed and probabilistic distributed agent combinatorial searches for the unary NP-complete disassembly line balancing problem. In S. M. Gupta (Ed.), *Environmentally Conscious Manufacturing V* (Vol. 5997, pp. 59970B-59970B – 12). https://doi.org/10.1117/12.629121

McGovern, S. M., & Gupta, S. M. (2006a). Computational complexity of a reverse manufacturing line. *Environmentally Conscious Manufacturing VI*. https://doi.org/10.1117/12.686371

McGovern, S. M., & Gupta, S. M. (2006b). Deterministic hybrid and stochastic combinatorial optimization treatments of an electronic product disassembly line. *Applications of Management Science*, *12*, 175–197. https://doi.org/10.1016/S0276-8976(06)12013-1

McGovern, S. M., & Gupta, S. M. (2006c). Ant colony optimization for disassembly sequencing with multiple objectives. *The International Journal of Advanced Manufacturing Technology*, *30*(5–6), 481–496. https://doi.org/10.1007/s00170-005-0037-6

McGovern, S. M., & Gupta, S. M. (2007a). A balancing method and genetic algorithm for disassembly line balancing. European Journal of Operational Research, 179(3), 692–708. https://doi.org/10.1016/j.ejor.2005.03.055

McGovern, S. M., & Gupta, S. M. (2007a). BENCHMARK DATA SET FOR EVALUATION OF LINE BALANCING ALGORITHMS. *IFAC Proceedings Volumes*, *40*(2), 48–53. https://doi.org/10.3182/20070523-3-ES-4907.00009

McGovern, S. M., & Gupta, S. M. (2007b). A balancing method and genetic algorithm for disassembly line balancing. *European Journal of Operational Research*, *179*(3), 692–708. https://doi.org/10.1016/j.ejor.2005.03.055

McGovern, S. M., & Gupta, S. M. (2007b). Combinatorial optimization analysis of the unary NP-complete disassembly line balancing problem. International Journal of Production Research, 45(18–19), 4485–4511. https://doi.org/10.1080/00207540701476281

McGovern, S. M., & Gupta, S. M. (2007c). Combinatorial optimization analysis of the unary NP-complete disassembly line balancing problem. *International Journal of Production Research*, *45*(18–19), 4485–4511. https://doi.org/10.1080/00207540701476281

McGovern, S. M., & Gupta, S. M. (2015). Unified assembly- and disassembly-line model formulae. *Journal of Manufacturing Technology Management*, *26*(2), 195–212. https://doi.org/10.1108/JMTM-11-2013-0169

Mei, K., & Fang, Y. (2021). Multi-robotic disassembly line balancing using deep reinforment learning. *Proceedings of the ASME 2021 16th International Manufacturing Science and Engineering Conference, MSEC 2021*. https://doi.org/10.1115/MSEC2021-63522

Meng, W., & Zhang, X. (2020). Optimization of remanufacturing disassembly line balance considering multiple failures and material hazards. *Sustainability (Switzerland)*. https://doi.org/10.3390/SU12187318

Mete, S., Çil, Z. A., Celik, E., & Özceylan, E. (2019). Supply-driven rebalancing of disassembly lines: A novel mathematical model approach. Journal of Cleaner Production, 213, 1157–1164. https://doi.org/10.1016/j.jclepro.2018.12.265

Mete, Suleyman, & Serin, F. (2021). A Reinforcement Learning Approach for Disassembly Line Balancing Problem. *2021 International Conference on Information Technology, ICIT 2021 - Proceedings*. https://doi.org/10.1109/ICIT52682.2021.9491689

Mete, Süleyman, Çil, Z. A., & Özceylan, E. (2017). A heuristic approach for joint design of assembly and disassembly line balancing problem. *2017 8th International Conference on Information Technology (ICIT)*, 418–423. https://doi.org/10.1109/ICITECH.2017.8080036

Mete, Süleyman, Çil, Z. A., Ağpak, K., Özceylan, E., & Dolgui, A. (2016). A solution approach based on beam search algorithm for disassembly line balancing problem. *Journal of Manufacturing Systems*, *41*, 188–200. https://doi.org/10.1016/j.jmsy.2016.09.002

Mete, Süleyman, Çil, Z. A., Özceylan, E., & Ağpak, K. (2015). A beam search approach to the disassembly line balancing problem. *Proceedings - CIE 45: 2015 International Conference on Computers and Industrial Engineering*.

Mete, Süleyman, Çil, Z. A., Özceylan, E., & Ağpak, K. (2016). Resource Constrained Disassembly Line Balancing Problem. *IFAC-PapersOnLine*, *49*(12), 921–925. https://doi.org/10.1016/j.ifacol.2016.07.893

Mete, Süleyman, Çil, Z. A., Özceylan, E., Ağpak, K., & Battaïa, O. (2018). An optimisation support for the design of hybrid production lines including assembly and disassembly tasks. *International Journal of Production Research*, *56*(24), 7375–7389. https://doi.org/10.1080/00207543.2018.1428774

Minca, E., Filipescu, A., & Voda, A. (2014). Modelling and control of an assembly/disassembly mechatronics line served by mobile robot with manipulator. *Control Engineering Practice*. https://doi.org/10.1016/j.conengprac.2014.06.005

Ming, H., Liu, Q., & Pham, D. T. (2019). Multi-Robotic Disassembly Line Balancing with Uncertain Processing Time. *Procedia CIRP*, *83*, 71–76. https://doi.org/10.1016/j.procir.2019.02.140

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. Computers & Operations Research, 24(11), 1097–1100. https://doi.org/10.1016/S0305-0548(97)00031-2

Moscato, P., & Cotta Porras, C. (2003). An Introduction to Memetic Algorithms. INTELIGENCIA ARTIFICIAL. https://doi.org/10.4114/ia.v7i19.721

Moscato, Pablo, & Mathieson, L. (2019). Memetic Algorithms for Business Analytics and Data Science: A Brief Survey. In Business and Consumer Analytics: New Ideas. https://doi.org/10.1007/978-3-030-06222-4_13

Mutlu, S., & Güner, B. (2021). A memetic algorithm for mixed-model two-sided disassembly line balancing problem. *Procedia CIRP*, *98*, 67–72. https://doi.org/10.1016/j.procir.2021.01.007

NCA. (2009). 800,000 Year record of CO2 Concentration. Global Climate Change Impacts in the United States 2009 Reports. https://nca2009.globalchange.gov/800000-year-record-co2-concentration/index.html

Ning, C., Zeqiang, Z., Lixia, Z., & Lin, J. (2018). Improved Fruit-fly Fuzzy Optimization Algorithm for Multi-constrained Disassembly-line Balancing Problem Considering Energy Consumption. *Information and Control*. https://doi.org/10.13976/j.cnki.xk.2018.7390

Özcan, U., & Toklu, B. (2009). Balancing of mixed-model two-sided assembly lines. Computers & Industrial Engineering, 57(1), 217–227. https://doi.org/10.1016/j.cie.2008.11.012

Özceylan, E., & Paksoy, T. (2013). Reverse supply chain optimisation with disassembly line balancing. *International Journal of Production Research*, *51*(20), 5985–6001. https://doi.org/10.1080/00207543.2013.784405

Özceylan, E., & Paksoy, T. (2014a). Fuzzy mathematical programming approaches for reverse supply chain optimization with disassembly line balancing problem. *Journal of Intelligent & Fuzzy Systems*, *26*(4), 1969–1985. https://doi.org/10.3233/IFS-130875

Özceylan, E., & Paksoy, T. (2014b). Interactive fuzzy programming approaches to the strategic and tactical planning of a closed-loop supply chain under uncertainty. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2013.865852

Özceylan, E., Kalayci, C. B., Güngör, A., & Gupta, S. M. (2019). Disassembly line balancing problem: a review of the state of the art and future directions. *International Journal of Production Research*, *57*(15–16), 4805–4827. https://doi.org/10.1080/00207543.2018.1428775

Özceylan, E., Paksoy, T., & Bektaş, T. (2014). Modeling and optimizing the integrated problem of closed-loop supply chain network design and disassembly line balancing. *Transportation Research Part E: Logistics and Transportation Review*, *61*, 142–164. https://doi.org/10.1016/j.tre.2013.11.001

Paksoy, T., Güngör, A., Özceylan, E., & Hancilar, A. (2013). Mixed model disassembly line balancing problem with fuzzy goals. *International Journal of Production Research*, *51*(20), 6082–6096. https://doi.org/10.1080/00207543.2013.795251

Pistolesi, F., Lazzerini, B., Mura, M. D., & Dini, G. (2018). EMOGA: A Hybrid Genetic Algorithm With Extremal Optimization Core for Multiobjective Disassembly Line Balancing. *IEEE Transactions on Industrial Informatics*, *14*(3), 1089–1098. https://doi.org/10.1109/TII.2017.2778223

Prakash, & Tiwari, M. K. (2005). Solving a Dissassembly Line Balancing Problem With Task Failure Using a Psycho-Clonal Algorithm. *Volume 4b: Design for Manufacturing and the Life Cycle Conference*, *2005*, 393–399. https://doi.org/10.1115/DETC2005-84999

Qiang, Y., Lin, Y., & Tian, G. (2018). A novel MCDM-based approach for disassembly line balancing problem. *2017 International Conference on Advanced Mechatronic Systems (ICAMechS)*, *2017-Decem*, 151–156. https://doi.org/10.1109/ICAMechS.2017.8316567

Qin, Gui Bin, Guo, X. W., Zhou, M. C., Liu, S. X., & Qi, L. (2020). Multi-Objective Discrete Migratory Bird Optimizer for Stochastic Disassembly Line Balancing Problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. https://doi.org/10.1109/SMC42975.2020.9283371

Qin, GuiBin, Guo, X., Liu, S., Qi, L., Zhao, J., Zhao, Z., & Tang, Y. (2021). *Multi-objective Discrete Migrating Birds Optimizer Solving Multiple-product Partial U-shaped Disassembly Line Balancing Problem*. https://doi.org/10.1109/med51440.2021.9480304

Ren, Y., Meng, L., Zhang, C., Lu, Q., & Tian, G. (2019). Multi-criterion decision making for disassembly line balancing problem. *Procedia CIRP*, *80*, 542–547. https://doi.org/10.1016/j.procir.2019.01.008

Ren, Y., Yu, D., Zhang, C., Tian, G., Meng, L., & Zhou, X. (2017). An improved gravitational search algorithm for profit-oriented partial disassembly line balancing problem. *International Journal of Production Research*, *55*(24), 7302–7316. https://doi.org/10.1080/00207543.2017.1341066

Ren, Y., Zhang, C., Zhao, F., Tian, G., Lin, W., Meng, L., & Li, H. (2018). Disassembly line balancing problem using interdependent weights-based multi-criteria decision making and 2-Optimal algorithm. *Journal of Cleaner Production*, *174*, 1475–1486. https://doi.org/10.1016/j.jclepro.2017.10.308

Ren, Y., Zhang, C., Zhao, F., Triebe, M. J., & Meng, L. (2020). An MCDM-Based Multiobjective General Variable Neighborhood Search Approach for Disassembly Line Balancing Problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–14. https://doi.org/10.1109/TSMC.2018.2862827

Riggs, R. J., Battaïa, O., & Hu, S. J. (2015). Disassembly line balancing under high variety of end of life states using a joint precedence graph approach. *Journal of Manufacturing Systems*, *37*, 638–648. https://doi.org/10.1016/j.jmsy.2014.11.002

Roshani, A., Fattahi, P., Roshani, A., Salehi, M., & Roshani, A. (2012). Cost-oriented two-sided assembly line balancing problem: A simulated annealing approach. International Journal of Computer Integrated Manufacturing. https://doi.org/10.1080/0951192X.2012.664786

Seidi, M., & Saghari, S. (2016). The Balancing of Disassembly Line of Automobile Engine Using Genetic Algorithm (GA) in Fuzzy Environment. *Industrial Engineering and Management Systems*, *15*(4), 364–373. https://doi.org/10.7232/iems.2016.15.4.364

Statista. (2020a). Heavy truck production worldwide from 2011 to 2019, by region.

Statista. (2020b). Industrial production volume of household refrigerators in China from 2009 to 2019. https://www.statista.com/statistics/408899/china-household-refrigerator-production/

Tang, Y., Zhou, M.-C., & Caudill, R. J. (2001). A systematic approach to disassembly line design. *Proceedings of the 2001 IEEE International Symposium on Electronics and the Environment. 2001 IEEE ISEE (Cat. No.01CH37190)*, 173–178. https://doi.org/10.1109/ISEE.2001.924522

Tapkan, P., Ozbakir, L., & Baykasoglu, A. (2012). Modeling and solving constrained two-sided assembly line balancing problem via bee algorithms. Applied Soft Computing Journal. https://doi.org/10.1016/j.asoc.2012.06.003

Tuncel, E., Zeid, A., & Kamarthi, S. (2014). Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. *Journal of Intelligent Manufacturing*, *25*(4), 647–659. https://doi.org/10.1007/s10845-012-0711-0

Turowski, M., Morgan, M., & Ying Tang. (2005). Disassembly Line Design with Uncertainty. *2005 IEEE International Conference on Systems, Man and Cybernetics*, *1*, 954–959. https://doi.org/10.1109/ICSMC.2005.1571269

Wang, K., Gao, L., & Li, X. (2020). A multi-objective algorithm for U-shaped disassembly line balancing with partial destructive mode. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-020-04721-0

Wang, K., Li, X., & Gao, L. (2019a). Modeling and optimization of multi-objective partial disassembly line balancing problem considering hazard and profit. *Journal of Cleaner Production*, *211*, 115–133. https://doi.org/10.1016/j.jclepro.2018.11.114

Wang, K., Li, X., & Gao, L. (2019b). A multi-objective discrete flower pollination algorithm for stochastic two-sided partial disassembly line balancing problem. *Computers & Industrial Engineering*, *130*, 634–649. https://doi.org/10.1016/j.cie.2019.03.017

Wang, K., Li, X., Gao, L., & Garg, A. (2019). Partial disassembly line balancing for energy consumption and profit under uncertainty. *Robotics and Computer-Integrated Manufacturing*, *59*, 235–251. https://doi.org/10.1016/j.rcim.2019.04.014

Wang, K., Li, X., Gao, L., & Li, P. (2020). Energy consumption and profit-oriented disassembly line balancing for waste electrical and electronic equipment. *Journal of Cleaner Production*, *265*, 121829. https://doi.org/10.1016/j.jclepro.2020.121829

Wang, K., Li, X., Gao, L., & Li, P. (2021a). Modeling and Balancing for Disassembly Lines Considering Workers With Different Efficiencies. *IEEE Transactions on Cybernetics*. https://doi.org/10.1109/TCYB.2021.3070122

Wang, K., Li, X., Gao, L., & Li, P. (2021b). Modeling and Balancing for Green Disassembly Line Using Associated Parts Precedence Graph and Multi-objective Genetic Simulated Annealing. *International Journal of Precision Engineering and Manufacturing - Green Technology*. https://doi.org/10.1007/s40684-020-00259-7

Wang, K., Li, X., Gao, L., Li, P., & Gupta, S. M. (2021a). A genetic simulated annealing algorithm for parallel partial disassembly line balancing problem. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2021.107404

Wang, K., Li, X., Gao, L., Li, P., & Sutherland, J. W. (2021b). A Discrete Artificial Bee Colony Algorithm for Multiobjective Disassembly Line Balancing of End-of-Life Products. *IEEE Transactions on Cybernetics*. https://doi.org/10.1109/TCYB.2020.3042896

Wang, K., Zhang, Z., Mao, L., & Li, L. (2017). Pareto artificial fish swarm algorithm for multi-objective disassembly line balancing problems. *Zhongguo Jixie Gongcheng/China Mechanical Engineering*. https://doi.org/10.3969/j.issn.1004-132X.2017.02.010

Wang, K., Zhang, Z., Zhu, L., & Zou, B. (2017). Pareto genetic simulated annealing algorithm for multi-objective disassembly line balancing problem. *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*. https://doi.org/10.13196/j.cims.2017.06.013

Wang, S., Guo, X., & Liu, J. (2019). An efficient hybrid artificial bee colony algorithm for disassembly line balancing problem with sequence-dependent part removal times. *Engineering Optimization*, *51*(11), 1920–1937. https://doi.org/10.1080/0305215X.2018.1564918

Wang, T. Y., Guo, X. W., Liu, S. X., Qi, L., & Zhao, Z. Y. (2020). A Stochastic Sequence-dependent Multi-objective Disassembly Line Balancing Model Subject to Task Failure and Resource Constraint via Multi-objective Cuckoo Search. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. https://doi.org/10.1109/SMC42975.2020.9283012

Wang, Y., Xie, Y., Ren, Y., & Zhang, C. (2021). A MCDM-based meta-heuristic approach for U-shaped disassembly line balancing problem. *Journal of Physics: Conference Series*. https://doi.org/10.1088/1742-6596/1828/1/012159

Whitley, D. (1994). A genetic algorithm tutorial. Statistics and Computing. https://doi.org/10.1007/BF00175354

Wiendahl, H.-P., Seliger, G., Perlewitz, H., & Bürkner, S. (1999). A general approach to disassembly planning and control. Production Planning & Control, 10(8), 718–726. https://doi.org/10.1080/095372899232542

Wu, K., Guo, X. W., Zhou, M. C., Liu, S. X., & Qi, L. (2020). Multi-objective Discrete Brainstorming Optimizer for Stochastic Disassembly Line Balancing Problem Subject to Disassembly Failure. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. https://doi.org/10.1109/SMC42975.2020.9282908

Xia, X., Liu, W., Zhang, Z., & Wang, L. (2020). Partial disassembly line balancing problem analysis based on sequence-dependent stochastic mixed-flow. *Journal of Computing and Information Science in Engineering*. https://doi.org/10.1115/1.4046993

Xia, X., Liu, W., Zhang, Z., Wang, L., Cao, J., & Liu, X. (2019). A Balancing Method of Mixed-model Disassembly Line in Random Working Environment. Sustainability, 11(8), 2304. https://doi.org/10.3390/su11082304

Xia, X., Zhou, M., Wang, L., & Cao, J. (2018). Remanufacturing disassembly service line and balancing optimization method. *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*. https://doi.org/10.13196/j.cims.2018.10.011

Xiao, Q., Guo, X., & Li, D. (2021). Partial disassembly line balancing under uncertainty: robust optimisation models and an improved migrating birds optimisation algorithm. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2020.1744765

Xiao, S., Wang, Y., Yu, H., & Nie, S. (2017). An Entropy-Based Adaptive Hybrid Particle Swarm Optimization for Disassembly Line Balancing Problems. *Entropy*, *19*(11), 596. https://doi.org/10.3390/e19110596

Xie, M., Zhang, Z., & Jiang, J. (2021). Modeling and optimization of two-sided disassembly line balance problem considering station constraint and energy consumption. *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*. https://doi.org/10.13196/j.cims.2021.03.005

Xu, C., Wei, H., Guo, X. W., Liu, S. X., Qi, L., & Zhao, Z. Y. (2020). Human-Robot Collaboration Multi-objective Disassembly Line Balancing Subject to Task Failure via Multi-objective Artificial Bee Colony Algorithm. *IFAC-PapersOnLine*. https://doi.org/10.1016/j.ifacol.2021.04.076

Xu, Y., Yao, B., & Pham, D. T. (2020). Research on intelligent optimization algorithm for multi-objective disassembly line balancing problem. *ASME 2020 15th International Manufacturing Science and Engineering Conference, MSEC 2020*. https://doi.org/10.1115/MSEC2020-8269

Yang, Y., Yuan, G., Zhuang, Q., & Tian, G. (2019). Multi-objective low-carbon disassembly line balancing for agricultural machinery using MDFOA and fuzzy AHP. *Journal of Cleaner Production*, *233*, 1465–1474. https://doi.org/10.1016/j.jclepro.2019.06.035

Yılmaz, Ö. F., & Yazıcı, B. (2021). Tactical level strategies for multi-objective disassembly line balancing problem with multi-manned stations: an optimization model and solution approaches. *Annals of Operations Research*. https://doi.org/10.1007/s10479-020-03902-3

Yin, T., Zhang, Z., & Jiang, J. (2021). A Pareto-discrete hummingbird algorithm for partial sequence-dependent disassembly line balancing problem considering tool requirements. *Journal of Manufacturing Systems*. https://doi.org/10.1016/j.jmsy.2021.07.005

Yin, T., Zhang, Z., Zhang, Y., Wu, T., & Liang, W. (2022). Mixed-integer programming model and hybrid driving algorithm for multi-product partial disassembly line balancing problem with multi-robot workstations. *Robotics and Computer-Integrated Manufacturing*, *73*, 102251. https://doi.org/10.1016/j.rcim.2021.102251

Yolmeh, A., & Saif, U. (2021). Closed-loop supply chain network design integrated with assembly and disassembly line balancing under uncertainty: an enhanced decomposition approach. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2020.1736723

Yuan, G., Yang, Y., & Pham, D. T. (2020). Multiobjective Ecological Strategy Optimization for Two-Stage Disassembly Line Balancing with Constrained-Resource. *IEEE Access*. https://doi.org/10.1109/ACCESS.2020.2994065

Zeng, Y., Zhang, Z., & Zhu, L. (2019). Modeling and optimization of partial destructive disassembly line balancing problem considering profit and energy consumption. *Conference Proceedings of the 7th International Symposium on Project Management, ISPM 2019*.

Zeng, Y., Zhang, Z., Zhang, Y., Liu, S., & Li, Y. (2020). Pareto flower pollination algorithm for multi-objective bucket brigade mixed-model disassembly line balancing problem. *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*. https://doi.org/10.13196/j.cims.2020.03.018

Zhang, L., Zhao, X., Ke, Q., Dong, W., & Zhong, Y. (2021). Disassembly Line Balancing Optimization Method for High Efficiency and Low Carbon Emission. *International Journal of Precision Engineering and Manufacturing - Green Technology*. https://doi.org/10.1007/s40684-019-00140-2

Zhang, Ying, Zhang, Z., Zeng, Y., & Cai, N. (2020). Improved wind driven optimization algorithm for multi-objective disassembly line balancing problem concerning human factors. *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*. https://doi.org/10.13196/j.cims.2020.05.003

Zhang, Yu, Zhang, Z., Guan, C., & Xu, P. (2021). Improved whale optimisation algorithm for two-sided disassembly line balancing problems considering part characteristic indexes. *International Journal of Production Research*, 1–19. https://doi.org/10.1080/00207543.2021.1897178

Zhang, Z. W., Guo, X. W., Zhou, M. C., Liu, S. X., & Qi, L. (2020). Multi-objective Discrete Grey Wolf Optimizer for Solving Stochastic Multi-objective Disassembly Sequencing and Line Balancing Problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. https://doi.org/10.1109/SMC42975.2020.9283184

Zhang, Z., Cai, N., Zeng, Y., Li, L., & Zou, B. (2018). Review of Modeling Theory and Solution Method for Disassembly Line Balancing Problems for Remanufacturing. *Zhongguo Jixie Gongcheng/China Mechanical Engineering*. https://doi.org/10.3969/j.issn.1004-132X.2018.21.017

Zhang, Z., Hu, Y., & Chen, C. (2016). Improved Artificial Bee Colony Algorithm for Disassembly Line Balancing Problem. *Xinan Jiaotong Daxue Xuebao/Journal of Southwest Jiaotong University*. https://doi.org/10.3969/j.issn.0258-2724.2016.05.013

Zhang, Z., Li, L., Cai, N., & Jia, L. (2019). Local neighborhood genetic algorithm for stochastic disassembly line balancing problem. *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*. https://doi.org/10.13196/j.cims.2019.03.008

Zhang, Z., Wang, K., Li, L., & Mao, L. (2018). Multi-objective optimization for U-shaped disassembly line balancing problem with stochastic operation times. *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*. https://doi.org/10.13196/j.cims.2018.01.009

Zhang, Z., Wang, K., Zhu, L., & Cheng, W. (2018). Pareto Hybrid Ant Colony and Genetic Algorithm for Multi-Objective U-Shaped Disassembly Line Balancing Problem. *Xinan Jiaotong Daxue Xuebao/Journal of Southwest Jiaotong University*. https://doi.org/10.3969/j.issn.0258-2724.2018.03.026

Zhang, Z., Wang, K., Zhu, L., & Wang, Y. (2017). A Pareto improved artificial fish swarm algorithm for solving a multi-objective fuzzy disassembly line balancing problem. *Expert Systems with Applications*, *86*, 165–176. https://doi.org/10.1016/j.eswa.2017.05.053

Zheng, F., He, J., Chu, F., & Liu, M. (2018). A new distribution-free model for disassembly line balancing problem with stochastic task processing times. *International Journal of Production Research*, *56*(24), 7341–7353. https://doi.org/10.1080/00207543.2018.1430909

Zhou, R., Guo, X., Fu, Y., & Qi, L. (2019). Solving sequence-dependent disassembly line balancing problem with improved cuckoo search algorithm. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*. https://doi.org/10.1109/SMC.2019.8914273

Zhou, Y., Guo, X., & Li, D. (2020). A dynamic programming approach to a multi-objective disassembly line balancing problem. *Annals of Operations Research*. https://doi.org/10.1007/s10479-020-03797-0

Zhu, L., Zhang, Z., & Guan, C. (2020). Multi-objective partial parallel disassembly line balancing problem using hybrid group neighbourhood search algorithm. *Journal of Manufacturing Systems*. https://doi.org/10.1016/j.jmsy.2020.06.013

Zhu, L., Zhang, Z., & Wang, Y. (2018). A Pareto firefly algorithm for multi-objective disassembly line balancing problems with hazard evaluation. *International Journal of Production Research*, *56*(24), 7354–7374. https://doi.org/10.1080/00207543.2018.1471238

Zhu, L., Zhang, Z., Wang, Y., & Cai, N. (2020). On the end-of-life state oriented multi-objective disassembly line balancing problem. *Journal of Intelligent Manufacturing*, *31*(6), 1403–1428. https://doi.org/10.1007/s10845-019-01519-3

Zhu, X., Zhang, Z., Zhu, X., & Hu, J. (2014). An ant colony optimization algorithm for multi-objective disassembly line balancing problem. *China Mechanical Engineering*, *8*, 1075–1079. https://doi.org/10.3969/j.issn.1004-132X.2014.08.016

Zou, B., Zhang, Z., Cai, N., & Zhu, L. (2018). Cat swarm simulated annealing algorithm for disassembly line balancing problem under tool constraints. *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*. https://doi.org/10.13196/j.cims.2018.09.009

Zou, B., Zhang, Z., Li, L., & Cai, N. (2018). Modeling and Optimization for Two-sided Disassembly Line Balancing Problems. *Zhongguo Jixie Gongcheng/China Mechanical Engineering*. https://doi.org/10.3969/j.issn.1004-132X.2018.09.013

Zou, B., Zhang, Z., Li, L., & Zhu, L. (2017). Multi-objective disassembly line balancing problem based on pareto improved cat swarm optimization algorithm. *Information and Control*. https://doi.org/10.13976/j.cnki.xk.2017.0503

APPENDIX – 01. LINGO Code of TAOG Based DLB Problem

```
!Ali Koc , Ihsan Sabuncuoglu & Erdal Erel (2009) Two exact formulations for disassembly
line balancing problems with task precedence diagram construction using an AND/OR graph,
IIE Transactions, 41:10, 866-881, DOI: 10.1080/07408170802510390;

!Data is generated to me, but precedence diagram taken from article;

MODEL:

SETS:

!SET OF ARTIFICIAL NODES;
ARTIFICIAL/1..14/;
!SET OF NORMAL NODES;
NORMAL/1..23/:D, Z;

!SET OF STATIONS;
STATIONS/1..5/:F;

!PRECESSORS SET;
P(ARTIFICIAL, NORMAL)/2 1, 3 2, 4 3, 5 4, 5 6, 6 5, 6 8, 7 7, 7 10, 8 11, 8 12, 8 14, 9 9,
9 13, 10 9, 10 15, 11 16, 12 16, 13 15, 13 17, 13 18, 14 13, 14 17, 14 19/;

!SUCCESSOR SET;
S(ARTIFICIAL, NORMAL)/1 1, 1 2, 1 3, 2 4, 2 5, 3 6, 3 7, 4 8, 4 9, 4 10, 5 11, 6 12, 6 13,
7 14, 7 15, 8 16, 8 17, 9 18, 10 19, 11 20, 12 21, 13 22, 14 23/;

AUXSET1(NORMAL, STATIONS):X;

ENDSETS

DATA:

!TASK TIME OF NORMAL NODES;
D     =     3     9     5     7     3     10    4     4     5     6     6     7     8
      3     3     8     8     6     6     6     7     8     3;

!CYCLE TIME;
T     =     10;

ENDDATA

!IF TAKS I IS ASSIGNED TO STATION J, 1, OTHERWISE, 0;
@FOR(AUXSET1(I,J): @BIN(X(I,J)));

!IF TASK I IS PERFORM;
@FOR(NORMAL(I): @BIN(Z(I)));

!IF STATION J IS OPENEDED, 1, OTHERWISE, 0;
@FOR(STATIONS(J): @BIN(F(J)));

!OBJECTIVE FUNCTION
MINIMIZE TO OPENED STATIONS;
MIN = @SUM(STATIONS(J): J*F(J));

!CONSTRAINT ONE AND CONSTRAINT TWO
ONE OF THE OR SUCCESSORS IS SELECTED;
@FOR(ARTIFICIAL(K) | K#EQ#1 : @SUM(S(K2, I) | K2#EQ#K : Z(I)) = 1);
@FOR(ARTIFICIAL(K) | K#NE#1 : @SUM(S(K2, I) | K2#EQ#K : Z(I)) = @SUM(P(K2, I) | K2#EQ#K :
Z(I)));

!CONSTRAINT THREE
SELECTED TASKS IS ASSIGNED TO ONE OF STATIONS;
@FOR(NORMAL(I): @SUM(STATIONS(J): X(I,J)) = Z(I));

!CONSTRAINT FOUR
PRECEDENCE RELATIONS;
@FOR(ARTIFICIAL(K) | K#NE#1 : @FOR(STATIONS(V): @SUM(P(K2, I) | K2#EQ#K : @SUM(STATIONS(J)
| J#LE#V : X(I,J))) >= @SUM(S(K2, I) | K2#EQ#K : X(I,V))));
```

```
!CONSTRAINT FIVE
TOTAL WORKLOAD OF A STATION TO BE LESS THAN CYCLE TIME, IF THAT STATION IS OPENED;
@FOR(STATIONS(J): @SUM(NORMAL(I): X(I,J)*D(I)) <= T*F(J));
```

APPENDIX – 02. GUROBI Code of MTDLB Problem

C:\Users\yusuf\Desktop\28CIRPLCL-master\gurobi_opt\main.py

```
# ************************
# AUTHORS: Serkan MUTLU, Banu GUNER.
# Department of Industrial Engineering, Eskisehir Technical University, Eskisehir, Turkey.
# ************************
# Mixed-Model Two-Sided Disassembly Line Balancing (MTDLB) problem optimization codes ↵
↳by GUROBI.
# ************************

# -------------------------
# Import Library and Python Functions
# -------------------------

import gurobipy as grb
import time
import inputs

# -------------------------
# Inputs
# -------------------------

nA = { ("m1"): 9, ("m2"): 21 }
nN = { ("m1"): 10, ("m2"): 30 }
MODELS = ["m1", "m2"]
STATIONS = [(0), (1), (2), (3)]
SIDES = {(1), (2)}
MONO = {(m,i) for m in MODELS for i in range(1, nN[m]+1)}
MONONO = {(m,i,i2) for m in MODELS for i in range(1, nN[m]+1) for i2 in range(1, ↵
↳nN[m]+1) if i2 < i}
PRE = { ("m1", 1): [1], ("m1", 2): [2], ("m1", 3): [2, 3], ("m1", 4): [3, 4], ("m1", ↵
↳5): [5], ("m1", 6): [1, 4, 8], ("m1", 7): [9], ("m2", 1): [1], ("m2", 2): [2, 3], ↵
↳("m2", 3): [3, 10], ("m2", 4): [2], ("m2", 5): [2], ("m2", 6): [11], ("m2", 7): [14], ↵
↳("m2", 8): [19], ("m2", 9): [18, 20], ("m2", 10): [13], ("m2", 11): [4, 12], ("m2", ↵
↳12): [15, 21], ("m2", 13): [23, 24, 25], ("m2", 14): [5], ("m2", 15): [16, 26], ("m2", ↵
↳16): [6, 22], ("m2", 17): [17, 27, 28], ("m2", 18): [7], ("m2", 19): [8, 29] }
SUC = { ("m1", 0): [1, 2], ("m1", 1): [3], ("m1", 2): [4, 5], ("m1", 3): [7], ("m1", ↵
↳4): [6], ("m1", 5): [8], ("m1", 6): [9], ("m1", 7): [10], ("m2", 0): [1, 2], ("m2", ↵
↳1): [3], ("m2", 2): [30], ("m2", 3): [4], ("m2", 4): [10], ("m2", 5): [11], ("m2", 6): ↵
↳[12, 13, 14], ("m2", 7): [19, 20], ("m2", 8): [25, 26], ("m2", 9): [24], ("m2", 10): ↵
↳[18, 21], ("m2", 11): [5, 15], ("m2", 12): [22, 23], ("m2", 13): [27], ("m2", 14): [6, ↵
↳16], ("m2", 15): [28], ("m2", 16): [7, 17], ("m2", 17): [29], ("m2", 18): [8], ("m2", ↵
↳19): [9] }
THETA = { ("m1", 1): [1], ("m1", 2): [2], ("m1", 3): [2], ("m1", 4): [1, 2], ("m1", 5): ↵
↳[2], ("m1", 6): [1], ("m1", 7): [1, 2], ("m1", 8): [1, 2], ("m1", 9): [2], ("m1", 10): ↵
↳[1, 2], ("m2", 1): [2], ("m2", 2): [1], ("m2", 3): [1], ("m2", 4): [1, 2], ("m2", 5): ↵
↳[2], ("m2", 6): [2], ("m2", 7): [2], ("m2", 8): [1], ("m2", 9): [1], ("m2", 10): [1, ↵
↳2], ("m2", 11): [1, 2], ("m2", 12): [1, 2], ("m2", 13): [1, 2], ("m2", 14): [1], ↵
↳("m2", 15): [1], ("m2", 16): [1, 2], ("m2", 17): [2], ("m2", 18): [2], ("m2", 19): [1, ↵
↳2], ("m2", 20): [1], ("m2", 21): [2], ("m2", 22): [1], ("m2", 23): [1], ("m2", 24): ↵
↳[2], ("m2", 25): [2], ("m2", 26): [1, 2], ("m2", 27): [1], ("m2", 28): [1], ("m2", ↵
```

Page 1, last modified 20 Aug 2021 15:26:11

```
ʢ29): [1, 2], ("m2", 30): [1, 2] }
C = 40
t = { ("m1",1): 30, ("m1",2): 28, ("m1",3): 12, ("m1",4): 34, ("m1",5): 13, ("m1",6): ₴
ʢ21, ("m1",7): 6, ("m1",8): 18, ("m1",9): 25, ("m1",10): 10, ("m2",1): 11, ("m2",2): ₴
ʢ20, ("m2",3): 20, ("m2",4): 14, ("m2",5): 19, ("m2",6): 1, ("m2",7): 7, ("m2",8): 6, ₴
ʢ("m2",9): 6, ("m2",10): 7, ("m2",11): 19, ("m2",12): 11, ("m2",13): 18, ("m2",14): 13, ₴
ʢ("m2",15): 5, ("m2",16): 11, ("m2",17): 6, ("m2",18): 4, ("m2",19): 6, ("m2",20): 8, ₴
ʢ("m2",21): 18, ("m2",22): 15, ("m2",23): 15, ("m2",24): 6, ("m2",25): 10, ("m2",26): ₴
ʢ20, ("m2",27): 13, ("m2",28): 1, ("m2",29): 4, ("m2",30): 5 }

#EPSILON:           a very small number
#BIG_M:             a very big number
EPSILON = 0.001
BIG_M = 99999

# -------------------------
# Start of Gurobi Model
# -------------------------

time1 = time.time()
opt_model = grb.Model(name="MILP Model")

# -------------------------
# Variables -- MTDLB Variables
# -------------------------

F = opt_model.addVars(STATIONS, vtype=grb.GRB.BINARY)
G = opt_model.addVars(STATIONS, vtype=grb.GRB.BINARY)
U = opt_model.addVars(STATIONS, SIDES, vtype=grb.GRB.BINARY)
Z = opt_model.addVars(MONO, vtype=grb.GRB.BINARY)
GAMMA = opt_model.addVars(MODELS, STATIONS, SIDES, vtype=grb.GRB.BINARY)
X = opt_model.addVars(MONO, STATIONS, SIDES, vtype=grb.GRB.BINARY)
DELTA = opt_model.addVars(MONONO, vtype=grb.GRB.BINARY)
TF = opt_model.addVars(MONO, lb=0)

# -------------------------
# Objective Function
# -------------------------

# Equation (4.23)
objective = (grb.quicksum(F[j] + G[j] for j in STATIONS)
             + EPSILON*grb.quicksum(j*U[j,s] for j in STATIONS for s in SIDES)
             + EPSILON*EPSILON*grb.quicksum(t[m,i]*Z[m,i] for (m,i) in MONO))

# -------------------------
# Constraints
# -------------------------

# Equation (4.4)
```

```
opt_model.addConstrs(grb.quicksum(Z[m,i] for i in SUC[m,0]) == 1 for m in MODELS)

# Equation (4.5)
opt_model.addConstrs(grb.quicksum(Z[m,i] for i in SUC[m,k]) == grb.quicksum(Z[m,i] for ↵
↳i in PRE[m,k]) for m in MODELS for k in range(1, nA[m]) if ((m,k) in SUC) and ((m,k) ↵
↳in PRE))

# Equation (4.6)
opt_model.addConstrs(grb.quicksum(grb.quicksum(X[m,i,j,s] for s in THETA[m,i]) for j in ↵
↳STATIONS) == Z[m,i] for (m,i) in MONO)

# Equation (4.7)
opt_model.addConstrs(grb.quicksum(X[m,i,v,s] for i in PRE[m,k] for s in THETA[m,i] for ↵
↳v in range(1, j+1)) >= grb.quicksum(X[m,i,j,s] for i in SUC[m,k] for s in THETA[m,i]) ↵
↳for m in MODELS for k in range(1, nA[m]) for j in STATIONS if (((m,k) in PRE) and ↵
↳((m,k) in SUC)))

# Equation (4.8)
opt_model.addConstrs(TF[m,i] <= C*Z[m,i] for (m,i) in MONO)

# Equation (4.9)
opt_model.addConstrs(TF[m,i] >= t[m,i]*Z[m,i] for (m,i) in MONO)

# Equation (4.10)
opt_model.addConstrs(TF[m,i] - TF[m,h] + BIG_M*(2 - grb.quicksum(X[m,i,j,s] for s in ↵
↳THETA[m,i]) - grb.quicksum(X[m,h,j,s] for s in THETA[m,h])) >= t[m,i] for m in MODELS ↵
↳for j in STATIONS for k in range(1, nA[m]) if (((m,k) in SUC) and ((m,k) in PRE)) for ↵
↳i in SUC[m,k] for h in PRE[m,k])

# Equation (4.11)
opt_model.addConstrs(TF[m,i] - TF[m,h] + BIG_M*(3 - X[m,i,j,s] - X[m,h,j,s] - ↵
↳DELTA[m,i,h]) >= t[m,i] for (m,i,h) in MONONO for j in STATIONS for s in SIDES)

# Equation (4.12)
opt_model.addConstrs(TF[m,h] - TF[m,i] + BIG_M*(2 - X[m,i,j,s] - X[m,h,j,s] + ↵
↳DELTA[m,i,h]) >= t[m,h] for (m,i,h) in MONONO for j in STATIONS for s in SIDES)

# Equation (4.13)
opt_model.addConstrs(grb.quicksum(X[m,i,j,s] for i in range(1, nN[m]+1) if s in ↵
↳THETA[m,i]) - nN[m]*GAMMA[m,j,s] <= 0 for m in MODELS for j in STATIONS for s in SIDES)

# Equation (4.14)
opt_model.addConstrs(grb.quicksum(GAMMA[m,j,s] for m in MODELS) - len(MODELS)*U[j,s] <= ↵
↳0 for j in STATIONS for s in SIDES)

# Equation (4.15)
opt_model.addConstrs(grb.quicksum(U[j,s] for s in SIDES) - 2*F[j] - G[j] == 0 for j in ↵
↳STATIONS)
```

C:\Users\yusuf\Desktop\28CIRPLCL-master\gurobi_opt\main.py

```
#-------------------------
# Model Setup
# -------------------------

opt_model.ModelSense = grb.GRB.MINIMIZE
opt_model.setObjective(objective)
opt_model.update()
opt_model.optimize()

print(opt_model)

print("\nSolution Results\n")
print("Time = ", time.time() - time1, "second")
print("Total number of stations opened from both sides\t\t:\t", sum([F[j].X for j in ↵
↳STATIONS]))
print("Total number of stations opened from only one side\t:\t", sum([G[j].X for j in ↵
↳STATIONS]))
print("Total number of stations opened\t\t\t\t:\t", sum([U[j,s].X for j in STATIONS for ↵
↳s in SIDES]))
for m in MODELS:
  print("#### MODEL-",m," ####")
  print("(m, i)\t\t (j,s)\t\t Processing Time\t Starting Time\t Ending Time")
  for i in range(1, nN[m]+1):
    if TF[m,i].X != 0.0:
      if i < 10:
        print((m,i), " :\t", [(j,s) for j in STATIONS for s in SIDES if X[m,i,j,s].X == ↵
↳       1.00], "\t", t[m,i], "\t\t\t", TF[m,i].X - t[m,i], "\t\t", TF[m,i].X)
      else:
        print((m,i), ":\t", [(j,s) for j in STATIONS for s in SIDES if X[m,i,j,s].X == ↵
↳       1.00], "\t", t[m,i], "\t\t\t", TF[m,i].X - t[m,i], "\t\t", TF[m,i].X)
```

APPENDIX – 03A. Main Python Code of MOMA for MTDLB Problem

```
C:\Users\yusuf\Desktop\masaüstü\Dosyalar\Projeler\28th CIRP Conference on Life Cycle
Engineering\main.py

# ************************
# AUTHORS: Serkan MUTLU, Banu GUNER.
# Department of Industrial Engineering, Eskisehir Technical University, Eskisehir, Turkey.
# ************************
# Mixed-Model Two-Sided Disassembly Line Balancing (MTDLB) problem Memetic Algorithm ↵
↳codes.
# ************************
# 28th CIRP Life Cycle Engineering Conference - Jaipur, India.
# ************************


# -------------------------
# Import Library and Python Functions
# -------------------------


# -- standart python libraries
import numpy as np
import random as rnd
import time
import matplotlib.pyplot as plt

# -- extra functions
import gurobi_opt.inputs as inputs
import initial
import algorithms
import calculation
import genetic_operators
import local


# -------------------------
# Inputs
# -------------------------


#PRE(m, k, i):      immediate normal node i predecessors of artificial node k for each ↵
↳model m
#SUC(m, k, i):      immediate normal node i successors of artificial node k for each ↵
↳model m
#THETA(m, i, s):    station-sides s where normal node i can be made for each model m
#C(m):              cycle time for each model m
#t(m,i):            processing time of each normal task i for each model m
print("MODELS\nModel-01\tFlashlight\nModel-02\tRadio\nModel-03\tToy Car\nModel-04\tBall ↵
↳Point Pen\nModel-05\t2P8\nModel-06\t2P10\nModel-07\t2P25\nModel-08\t2P48-A\n")
nA, nN, MODELS, STATIONS, SIDES, MONO, MONONO, PRE, SUC, THETA, C, t = ↵
↳inputs.MTDLBInput(int(input("If model-01 is disassemble in line -- 1; otherwise -- 0 = ↵
↳")), int(input("If model-02 is disassemble in line -- 1; otherwise -- 0 = ")), ↵

Page 1, last modified 01 Sep 2021 23:50:57
```

C:\Users\yusuf\Desktop\masaüstü\Dosyalar\Projeler\28th CIRP Conference on Life Cycle Engineering\main.py

```
int(input("If model-03 is disassemble in line -- 1; otherwise -- 0 = ")),
int(input("If model-04 is disassemble in line -- 1; otherwise -- 0 = ")),
int(input("If model-05 is disassemble in line -- 1; otherwise -- 0 = ")),
int(input("If model-06 is disassemble in line -- 1; otherwise -- 0 = ")),
int(input("If model-07 is disassemble in line -- 1; otherwise -- 0 = ")),
int(input("If model-08 is disassemble in line -- 1; otherwise -- 0 = ")))

#maximum number of OR Successor relations for each model m
nORSuc = {
    ("m1"): 2,
    ("m2"): 7,
    ("m3"): 9,
    ("m4"): 4,
    ("m5"): 0,
    ("m6"): 0,
    ("m7"): 0,
    ("m8"): 0
}

maxselnode = {
    ("m1"): 7,
    ("m2"): 15,
    ("m3"): 15,
    ("m4"): 9,
    ("m5"): 8,
    ("m6"): 10,
    ("m7"): 25,
    ("m8"): 48
}

EPSILON = 0.001

nP = 30


# -------------------------
# Memetic Algorithm Start
# -------------------------


# -- start of calculating time
t1 = time.time()

# -- set decision variables
sol = [list() for _ in range(nP)]
dec_sol = [list() for _ in range(nP)]
selected = [list() for _ in range(nP)]
X = [list() for _ in range(nP)]
```

C:\Users\yusuf\Desktop\masaüstü\Dosyalar\Projeler\28th CIRP Conference on Life Cycle Engineering\main.py

```
DLBcost = [0 for _ in range(nP)]
DLBcost2 = [0 for _ in range(nP)]
DLBcost3 = [0 for _ in range(nP)]
tf = [list() for _ in range(nP)]
U = [list() for _ in range(nP)]
F = [list() for _ in range(nP)]
G = [list() for _ in range(nP)]
bestcost = 9999
TIM = []
BEST = []

# -- generate initial solution randomly.
for p in range(nP):
    sol[p], dec_sol[p] = initial.randoninit(MODELS, nN, nORSuc, maxselnode)

# -- Algo-CDS
for p in range(nP):
    selected[p] = algorithms.AlgoCDS(MODELS, nA, SUC, PRE, dec_sol[p])

# -- Algo-ADL
for p in range(nP):
    X[p], tf[p], U[p] = algorithms.AlgoADL(sol[p], selected[p], nA, nORSuc, MODELS, ↲
↳   STATIONS, SIDES, SUC, PRE, THETA, t, C)

for p in range(nP):
    F[p], G[p], DLBcost[p], DLBcost2[p], DLBcost3[p] = calculation.cost(U[p], ↲
↳   selected[p], MODELS, STATIONS, EPSILON, t)

ParetoF, degree = calculation.FastNonDominatedSort(nP, DLBcost, DLBcost2, DLBcost3)
BestCost1 = []
BestCost2 = []
BestCost3 = []
BestCost1, BestCost2, BestCost3 = calculation.costupdate(BestCost1, BestCost2, ↲
↳BestCost3, DLBcost, DLBcost2, DLBcost3, ParetoF)
print(BestCost1, BestCost2, BestCost3)

maxiter = 100
maxils_iter = 2

for it in range(maxiter):

    # -- selection with roulette wheel method
    selection_pp = genetic_operators.roulette_wheel(nP, degree)

    # -- crossover
    sol, dec_sol = genetic_operators.crossover(nP, nN, MODELS, nORSuc, maxselnode, sol, ↲
↳   selection_pp)
```

```
    # -- Algo-CDS
    for p in range(nP):
        selected[p] = algorithms.AlgoCDS(MODELS, nA, SUC, PRE, dec_sol[p])

    # -- Algo-ADL
    for p in range(nP):
        X[p], tf[p], U[p] = algorithms.AlgoADL(sol[p], selected[p], nA, nORSuc, MODELS, ↵
↳       STATIONS, SIDES, SUC, PRE, THETA, t, C)

    for p in range(nP):
        F[p], G[p], DLBcost[p], DLBcost2[p], DLBcost3[p] = calculation.cost(U[p], ↵
↳       selected[p], MODELS, STATIONS, EPSILON, t)

    ParetoF, degree = calculation.FastNonDominatedSort(nP, DLBcost, DLBcost2, DLBcost3)
    BestCost1, BestCost2, BestCost3 = calculation.costupdate(BestCost1, BestCost2, ↵
↳   BestCost3, DLBcost, DLBcost2, DLBcost3, ParetoF)

    for mup in range(nP):
        for m in range(len(MODELS)):
            for i in range(nORSuc[MODELS[m]]+1):
                if i == nORSuc[MODELS[m]]:
                    rand_point = rnd.randint(0, maxselnode[MODELS[m]]-1)
                    if sol[mup][m][i][rand_point] == 0:
                        sol[mup][m][i][rand_point] = 1
                    else:
                        sol[mup][m][i][rand_point] = 0
                else:
                    rand_point = rnd.randint(0, 6)
                    if sol[mup][m][i][rand_point] == 0:
                        sol[mup][m][i][rand_point] = 1
                    else:
                        sol[mup][m][i][rand_point] = 0
            while True:
                point1 = rnd.randint(0, nN[MODELS[m]]-1)
                point2 = rnd.randint(0, nN[MODELS[m]]-1)
                if point1 != point2:
                    break
            sol[mup][m][nORSuc[MODELS[m]]+1][point1], ↵
↳           sol[mup][m][nORSuc[MODELS[m]]+1][point2] = ↵
↳           sol[mup][m][nORSuc[MODELS[m]]+1][point2], ↵
↳           sol[mup][m][nORSuc[MODELS[m]]+1][point1]
            dec_sol[mup] = [[sum([(2**(6-p))*sol[mup][m][n][p] for p in range(7)])/128 ↵
↳           for n in range(nORSuc[MODELS[m]])] for m in range(len(MODELS))]

    for p in range(nP):
        F[p], G[p], DLBcost[p], DLBcost2[p], DLBcost3[p] = calculation.cost(U[p], ↵
↳       selected[p], MODELS, STATIONS, EPSILON, t)
```

```
    ParetoF, degree = calculation.FastNonDominatedSort(nP, DLBcost, DLBcost2, DLBcost3)
    BestCost1, BestCost2, BestCost3 = calculation.costupdate(BestCost1, BestCost2, ↵
↳   BestCost3, DLBcost, DLBcost2, DLBcost3, ParetoF)

print(BestCost1)
print(BestCost2)
print(BestCost3)

fig = plt.figure()
ax = fig.add_subplot(projection='3d')
#ax.scatter([func1(dsol[p]) for p in range(100)], [func2(dsol[p]) for p in range(100)], ↵
↳[func3(dsol[p]) for p in range(100)])
ax.scatter(BestCost1, BestCost2, BestCost3)
ax.plot(BestCost1, BestCost2, BestCost3)
ax.set_xlabel("Length of Disassembly Line")
ax.set_ylabel("Opened Number of Station")
ax.set_zlabel("Total Processing Time for All Models")
plt.title("Pareto Frontier for Case-2")
plt.show()
```

```
!Ali Koc , Ihsan Sabuncuoglu & Erdal Erel (2009) Two exact formulations for disassembly
line balancing problems with task precedence diagram construction using an AND/OR graph,
IIE Transactions, 41:10, 866-881, DOI: 10.1080/07408170802510390;

!Data is generated to me, but precedence diagram taken from article;

MODEL:

SETS:

!SET OF ARTIFICIAL NODES;
ARTIFICIAL/1..14/;
!SET OF NORMAL NODES;
NORMAL/1..23/:D, Z;

!SET OF STATIONS;
STATIONS/1..5/:F;

!PRECESSORS SET;
P(ARTIFICIAL, NORMAL)/2 1, 3 2, 4 3, 5 4, 5 6, 6 5, 6 8, 7 7, 7 10, 8 11, 8 12, 8 14, 9 9,
9 13, 10 9, 10 15, 11 16, 12 16, 13 15, 13 17, 13 18, 14 13, 14 17, 14 19/;

!SUCCESSOR SET;
S(ARTIFICIAL, NORMAL)/1 1, 1 2, 1 3, 2 4, 2 5, 3 6, 3 7, 4 8, 4 9, 4 10, 5 11, 6 12, 6 13,
7 14, 7 15, 8 16, 8 17, 9 18, 10 19, 11 20, 12 21, 13 22, 14 23/;

AUXSET1(NORMAL, STATIONS):X;

ENDSETS

DATA:

!TASK TIME OF NORMAL NODES;
D     =     3     9     5     7     3     10    4     4     5     6     6     7     8
      3     3     8     8     6     6     6     7     8     3;

!CYCLE TIME;
T     =     10;

ENDDATA

!IF TAKS I IS ASSIGNED TO STATION J, 1, OTHERWISE, 0;
@FOR(AUXSET1(I,J): @BIN(X(I,J)));

!IF TASK I IS PERFORM;
@FOR(NORMAL(I): @BIN(Z(I)));

!IF STATION J IS OPENEDED, 1, OTHERWISE, 0;
@FOR(STATIONS(J): @BIN(F(J)));

!OBJECTIVE FUNCTION
MINIMIZE TO OPENED STATIONS;
MIN = @SUM(STATIONS(J): J*F(J));

!CONSTRAINT ONE AND CONSTRAINT TWO
ONE OF THE OR SUCCESSORS IS SELECTED;
@FOR(ARTIFICIAL(K) | K#EQ#1 : @SUM(S(K2, I) | K2#EQ#K : Z(I)) = 1);
@FOR(ARTIFICIAL(K) | K#NE#1 : @SUM(S(K2, I) | K2#EQ#K : Z(I)) = @SUM(P(K2, I) | K2#EQ#K :
Z(I)));

!CONSTRAINT THREE
SELECTED TASKS IS ASSIGNED TO ONE OF STATIONS;
@FOR(NORMAL(I): @SUM(STATIONS(J): X(I,J)) = Z(I));

!CONSTRAINT FOUR
PRECEDENCE RELATIONS;
@FOR(ARTIFICIAL(K) | K#NE#1 : @FOR(STATIONS(V): @SUM(P(K2, I) | K2#EQ#K : @SUM(STATIONS(J)
| J#LE#V : X(I,J))) >= @SUM(S(K2, I) | K2#EQ#K : X(I,V))));
```

```
!CONSTRAINT FIVE
TOTAL WORKLOAD OF A STATION TO BE LESS THAN CYCLE TIME, IF THAT STATION IS OPENED;
@FOR(STATIONS(J): @SUM(NORMAL(I): X(I,J)*D(I)) <= T*F(J));
```

APPENDIX – 02. GUROBI Code of MTDLB Problem

C:\Users\yusuf\Desktop\28CIRPLCL-master\gurobi_opt\main.py

```
# ************************
# AUTHORS: Serkan MUTLU, Banu GUNER.
# Department of Industrial Engineering, Eskisehir Technical University, Eskisehir, Turkey.
# ************************
# Mixed-Model Two-Sided Disassembly Line Balancing (MTDLB) problem optimization codes ↵
↳by GUROBI.
# ************************

# -------------------------
# Import Library and Python Functions
# -------------------------

import gurobipy as grb
import time
import inputs

# -------------------------
# Inputs
# -------------------------

nA = { ("m1"): 9, ("m2"): 21 }
nN = { ("m1"): 10, ("m2"): 30 }
MODELS = ["m1", "m2"]
STATIONS = [(0), (1), (2), (3)]
SIDES = {(1), (2)}
MONO = {(m,i) for m in MODELS for i in range(1, nN[m]+1)}
MONONO = {(m,i,i2) for m in MODELS for i in range(1, nN[m]+1) for i2 in range(1, ↵
↳nN[m]+1) if i2 < i}
PRE = { ("m1", 1): [1], ("m1", 2): [2], ("m1", 3): [2, 3], ("m1", 4): [3, 4], ("m1", ↵
↳5): [5], ("m1", 6): [1, 4, 8], ("m1", 7): [9], ("m2", 1): [1], ("m2", 2): [2, 3], ↵
↳("m2", 3): [3, 10], ("m2", 4): [2], ("m2", 5): [2], ("m2", 6): [11], ("m2", 7): [14], ↵
↳("m2", 8): [19], ("m2", 9): [18, 20], ("m2", 10): [13], ("m2", 11): [4, 12], ("m2", ↵
↳12): [15, 21], ("m2", 13): [23, 24, 25], ("m2", 14): [5], ("m2", 15): [16, 26], ("m2", ↵
↳16): [6, 22], ("m2", 17): [17, 27, 28], ("m2", 18): [7], ("m2", 19): [8, 29] }
SUC = { ("m1", 0): [1, 2], ("m1", 1): [3], ("m1", 2): [4, 5], ("m1", 3): [7], ("m1", ↵
↳4): [6], ("m1", 5): [8], ("m1", 6): [9], ("m1", 7): [10], ("m2", 0): [1, 2], ("m2", ↵
↳1): [3], ("m2", 2): [30], ("m2", 3): [4], ("m2", 4): [10], ("m2", 5): [11], ("m2", 6): ↵
↳[12, 13, 14], ("m2", 7): [19, 20], ("m2", 8): [25, 26], ("m2", 9): [24], ("m2", 10): ↵
↳[18, 21], ("m2", 11): [5, 15], ("m2", 12): [22, 23], ("m2", 13): [27], ("m2", 14): [6, ↵
↳16], ("m2", 15): [28], ("m2", 16): [7, 17], ("m2", 17): [29], ("m2", 18): [8], ("m2", ↵
↳19): [9] }
THETA = { ("m1", 1): [1], ("m1", 2): [2], ("m1", 3): [2], ("m1", 4): [1, 2], ("m1", 5): ↵
↳[2], ("m1", 6): [1], ("m1", 7): [1, 2], ("m1", 8): [1, 2], ("m1", 9): [2], ("m1", 10): ↵
↳[1, 2], ("m2", 1): [2], ("m2", 2): [1], ("m2", 3): [1], ("m2", 4): [1, 2], ("m2", 5): ↵
↳[2], ("m2", 6): [2], ("m2", 7): [2], ("m2", 8): [1], ("m2", 9): [1], ("m2", 10): [1, ↵
↳2], ("m2", 11): [1, 2], ("m2", 12): [1, 2], ("m2", 13): [1, 2], ("m2", 14): [1], ↵
↳("m2", 15): [1], ("m2", 16): [1, 2], ("m2", 17): [2], ("m2", 18): [2], ("m2", 19): [1, ↵
↳2], ("m2", 20): [1], ("m2", 21): [2], ("m2", 22): [1], ("m2", 23): [1], ("m2", 24): ↵
↳[2], ("m2", 25): [2], ("m2", 26): [1, 2], ("m2", 27): [1], ("m2", 28): [1], ("m2", ↵
```

Page 1, last modified 20 Aug 2021 15:26:11

C:\Users\yusuf\Desktop\28CIRPLCL-master\gurobi_opt\main.py

```
↳29): [1, 2], ("m2", 30): [1, 2] }
C = 40
t = { ("m1",1): 30, ("m1",2): 28, ("m1",3): 12, ("m1",4): 34, ("m1",5): 13, ("m1",6): ↵
↳21, ("m1",7): 6, ("m1",8): 18, ("m1",9): 25, ("m1",10): 10, ("m2",1): 11, ("m2",2): ↵
↳20, ("m2",3): 20, ("m2",4): 14, ("m2",5): 19, ("m2",6): 1, ("m2",7): 7, ("m2",8): 6, ↵
↳("m2",9): 6, ("m2",10): 7, ("m2",11): 19, ("m2",12): 11, ("m2",13): 18, ("m2",14): 13, ↵
↳("m2",15): 5, ("m2",16): 11, ("m2",17): 6, ("m2",18): 4, ("m2",19): 6, ("m2",20): 8, ↵
↳("m2",21): 18, ("m2",22): 15, ("m2",23): 15, ("m2",24): 6, ("m2",25): 10, ("m2",26): ↵
↳20, ("m2",27): 13, ("m2",28): 1, ("m2",29): 4, ("m2",30): 5 }

#EPSILON:           a very small number
#BIG_M:             a very big number
EPSILON = 0.001
BIG_M = 99999

# -------------------------
# Start of Gurobi Model
# -------------------------

time1 = time.time()
opt_model = grb.Model(name="MILP Model")

# -------------------------
# Variables -- MTDLB Variables
# -------------------------

F = opt_model.addVars(STATIONS, vtype=grb.GRB.BINARY)
G = opt_model.addVars(STATIONS, vtype=grb.GRB.BINARY)
U = opt_model.addVars(STATIONS, SIDES, vtype=grb.GRB.BINARY)
Z = opt_model.addVars(MONO, vtype=grb.GRB.BINARY)
GAMMA = opt_model.addVars(MODELS, STATIONS, SIDES, vtype=grb.GRB.BINARY)
X = opt_model.addVars(MONO, STATIONS, SIDES, vtype=grb.GRB.BINARY)
DELTA = opt_model.addVars(MONONO, vtype=grb.GRB.BINARY)
TF = opt_model.addVars(MONO, lb=0)

# -------------------------
# Objective Function
# -------------------------

# Equation (4.23)
objective = (grb.quicksum(F[j] + G[j] for j in STATIONS)
             + EPSILON*grb.quicksum(j*U[j,s] for j in STATIONS for s in SIDES)
             + EPSILON*EPSILON*grb.quicksum(t[m,i]*Z[m,i] for (m,i) in MONO))

# -------------------------
# Constraints
# -------------------------

# Equation (4.4)
```

```
opt_model.addConstrs(grb.quicksum(Z[m,i] for i in SUC[m,0]) == 1 for m in MODELS)

# Equation (4.5)
opt_model.addConstrs(grb.quicksum(Z[m,i] for i in SUC[m,k]) == grb.quicksum(Z[m,i] for ↵
↳i in PRE[m,k]) for m in MODELS for k in range(1, nA[m]) if ((m,k) in SUC) and ((m,k) ↵
↳in PRE))

# Equation (4.6)
opt_model.addConstrs(grb.quicksum(grb.quicksum(X[m,i,j,s] for s in THETA[m,i]) for j in ↵
↳STATIONS) == Z[m,i] for (m,i) in MONO)

# Equation (4.7)
opt_model.addConstrs(grb.quicksum(X[m,i,v,s] for i in PRE[m,k] for s in THETA[m,i] for ↵
↳v in range(1, j+1)) >= grb.quicksum(X[m,i,j,s] for i in SUC[m,k] for s in THETA[m,i]) ↵
↳for m in MODELS for k in range(1, nA[m]) for j in STATIONS if (((m,k) in PRE) and ↵
↳((m,k) in SUC)))

# Equation (4.8)
opt_model.addConstrs(TF[m,i] <= C*Z[m,i] for (m,i) in MONO)

# Equation (4.9)
opt_model.addConstrs(TF[m,i] >= t[m,i]*Z[m,i] for (m,i) in MONO)

# Equation (4.10)
opt_model.addConstrs(TF[m,i] - TF[m,h] + BIG_M*(2 - grb.quicksum(X[m,i,j,s] for s in ↵
↳THETA[m,i]) - grb.quicksum(X[m,h,j,s] for s in THETA[m,h])) >= t[m,i] for m in MODELS ↵
↳for j in STATIONS for k in range(1, nA[m]) if (((m,k) in SUC) and ((m,k) in PRE)) for ↵
↳i in SUC[m,k] for h in PRE[m,k])

# Equation (4.11)
opt_model.addConstrs(TF[m,i] - TF[m,h] + BIG_M*(3 - X[m,i,j,s] - X[m,h,j,s] - ↵
↳DELTA[m,i,h]) >= t[m,i] for (m,i,h) in MONONO for j in STATIONS for s in SIDES)

# Equation (4.12)
opt_model.addConstrs(TF[m,h] - TF[m,i] + BIG_M*(2 - X[m,i,j,s] - X[m,h,j,s] + ↵
↳DELTA[m,i,h]) >= t[m,h] for (m,i,h) in MONONO for j in STATIONS for s in SIDES)

# Equation (4.13)
opt_model.addConstrs(grb.quicksum(X[m,i,j,s] for i in range(1, nN[m]+1) if s in ↵
↳THETA[m,i]) - nN[m]*GAMMA[m,j,s] <= 0 for m in MODELS for j in STATIONS for s in SIDES)

# Equation (4.14)
opt_model.addConstrs(grb.quicksum(GAMMA[m,j,s] for m in MODELS) - len(MODELS)*U[j,s] <= ↵
↳0 for j in STATIONS for s in SIDES)

# Equation (4.15)
opt_model.addConstrs(grb.quicksum(U[j,s] for s in SIDES) - 2*F[j] - G[j] == 0 for j in ↵
↳STATIONS)
```

C:\Users\yusuf\Desktop\28CIRPLCL-master\gurobi_opt\main.py

```
#-------------------------
# Model Setup
# -------------------------

opt_model.ModelSense = grb.GRB.MINIMIZE
opt_model.setObjective(objective)
opt_model.update()
opt_model.optimize()

print(opt_model)

print("\nSolution Results\n")
print("Time = ", time.time() - time1, "second")
print("Total number of stations opened from both sides\t\t:\t", sum([F[j].X for j in ↵
↳STATIONS]))
print("Total number of stations opened from only one side\t:\t", sum([G[j].X for j in ↵
↳STATIONS]))
print("Total number of stations opened\t\t\t\t:\t", sum([U[j,s].X for j in STATIONS for ↵
↳s in SIDES]))
for m in MODELS:
  print("#### MODEL-",m," ####")
  print("(m, i)\t\t (j,s)\t\t Processing Time\t Starting Time\t Ending Time")
  for i in range(1, nN[m]+1):
    if TF[m,i].X != 0.0:
      if i < 10:
        print((m,i), " :\t", [(j,s) for j in STATIONS for s in SIDES if X[m,i,j,s].X == ↵
↳       1.00], "\t", t[m,i], "\t\t\t", TF[m,i].X - t[m,i], "\t\t", TF[m,i].X)
      else:
        print((m,i), ":\t", [(j,s) for j in STATIONS for s in SIDES if X[m,i,j,s].X == ↵
↳       1.00], "\t", t[m,i], "\t\t\t", TF[m,i].X - t[m,i], "\t\t", TF[m,i].X)
```

APPENDIX – 03B. Inputs Python Code of MOMA for MTDLB Problem

```
C:\Users\yusuf\Desktop\masaüstü\Dosyalar\Projeler\28th CIRP Conference on Life Cycle
Engineering\gurobi_opt\inputs.py

def MTDLBInput(mod1, mod2, mod3, mod4, mod5, mod6, mod7, mod8):

    nA_all = {
        ("m1"): 8,
        ("m2"): 20,
        ("m3"): 44,
        ("m4"): 14,
        ("m5"): 8,
        ("m6"): 10,
        ("m7"): 25,
        ("m8"): 47
    }
    nN_all = {
        ("m1"): 10,
        ("m2"): 30,
        ("m3"): 97,
        ("m4"): 20,
        ("m5"): 8,
        ("m6"): 10,
        ("m7"): 25,
        ("m8"): 47
    }
    PRE_all = {
        ("m1", 0): [],
        ("m1", 1): [1],
        ("m1", 2): [2],
        ("m1", 3): [2, 3],
        ("m1", 4): [3, 4],
        ("m1", 5): [5],
        ("m1", 6): [1, 4, 8],
        ("m1", 7): [9],
        ("m2", 0): [],
        ("m2", 1): [1],
        ("m2", 2): [2, 3],
        ("m2", 3): [3, 10],
        ("m2", 4): [2],
        ("m2", 5): [2],
        ("m2", 6): [11],
        ("m2", 7): [14],
        ("m2", 8): [19],
        ("m2", 9): [18, 20],
        ("m2", 10): [13],
        ("m2", 11): [4, 12],
        ("m2", 12): [15, 21],
        ("m2", 13): [23, 24, 25],
        ("m2", 14): [5],
        ("m2", 15): [16, 26],
        ("m2", 16): [6, 22],

Page 1, last modified 01 Sep 2021 21:11:04
```

```
        ("m2", 17): [17, 27, 28],
        ("m2", 18): [7],
        ("m2", 19): [8, 29],
        ("m3", 0): [],
        ("m3", 1): [1],
        ("m3", 2): [2],
        ("m3", 3): [3],
        ("m3", 4): [4, 7],
        ("m3", 5): [5],
        ("m3", 6): [6],
        ("m3", 7): [8],
        ("m3", 8): [9, 12],
        ("m3", 9): [10, 16],
        ("m3", 10): [11, 20],
        ("m3", 11): [13, 23],
        ("m3", 12): [14, 17],
        ("m3", 13): [15, 21],
        ("m3", 14): [18],
        ("m3", 15): [19, 22],
        ("m3", 16): [1, 7, 23],
        ("m3", 17): [24, 32],
        ("m3", 18): [25, 27, 35],
        ("m3", 19): [26, 30, 39],
        ("m3", 20): [28, 42],
        ("m3", 21): [29, 31, 45],
        ("m3", 22): [33, 36],
        ("m3", 23): [34, 40],
        ("m3", 24): [37, 43],
        ("m3", 25): [38, 41, 46],
        ("m3", 26): [44, 47],
        ("m3", 27): [49],
        ("m3", 28): [50, 52, 61],
        ("m3", 29): [51, 55, 64],
        ("m3", 30): [53, 57, 66],
        ("m3", 31): [54, 56, 59, 69],
        ("m3", 32): [58, 60, 72],
        ("m3", 33): [62, 67],
        ("m3", 34): [63, 65, 70],
        ("m3", 35): [68, 71, 73],
        ("m3", 36): [48],
        ("m3", 37): [75, 76, 83],
        ("m3", 38): [76, 77, 85],
        ("m3", 39): [79, 81, 82, 87],
        ("m3", 40): [84, 86, 88],
        ("m3", 41): [89],
        ("m3", 42): [74, 90],
        ("m3", 43): [48, 74],
        ("m4", 0): [],
```

```
        ("m4", 1): [1],
        ("m4", 2): [2],
        ("m4", 3): [4, 5],
        ("m4", 4): [1, 5],
        ("m4", 5): [3],
        ("m4", 6): [3, 6],
        ("m4", 7): [7],
        ("m4", 8): [9],
        ("m4", 9): [8, 12],
        ("m4", 10): [8, 13],
        ("m4", 11): [9, 14, 16],
        ("m4", 12): [14, 15],
        ("m4", 13): [18],
        ("m5", 1): [1],
        ("m5", 2): [1],
        ("m5", 3): [1],
        ("m5", 4): [2, 3],
        ("m5", 5): [5, 6],
        ("m5", 6): [8],
        ("m5", 7): [5, 7],
        ("m6", 0): [],
        ("m6", 1): [],
        ("m6", 2): [],
        ("m6", 3): [],
        ("m6", 4): [],
        ("m6", 5): [],
        ("m6", 6): [5, 6],
        ("m6", 7): [4, 7],
        ("m6", 8): [1, 8, 9, 10],
        ("m6", 9): [1, 8, 9, 10],
        ("m7", 0): [],
        ("m7", 1): [],
        ("m7", 2): [],
        ("m7", 3): [],
        ("m7", 4): [4, 5],
        ("m7", 5): [10],
        ("m7", 6): [11],
        ("m7", 7): [2],
        ("m7", 8): [2],
        ("m7", 9): [2],
        ("m7", 10): [1, 2],
        ("m7", 11): [3],
        ("m7", 12): [6, 7, 8, 9],
        ("m7", 13): [6, 7, 8, 9],
        ("m7", 14): [6, 7, 8, 9],
        ("m7", 15): [6, 7, 8, 9],
        ("m7", 16): [15],
        ("m7", 17): [13, 14],
```

```
        ("m7", 18): [13, 14, 16, 18],
        ("m7", 19): [17],
        ("m7", 20): [17],
        ("m7", 21): [21],
        ("m7", 22): [21],
        ("m7", 23): [16, 22],
        ("m7", 24): [19, 23],
        ("m8", 0): [],
        ("m8", 1): [1],
        ("m8", 2): [2],
        ("m8", 3): [2],
        ("m8", 4): [],
        ("m8", 5): [5],
        ("m8", 6): [],
        ("m8", 7): [],
        ("m8", 8): [8],
        ("m8", 9): [8],
        ("m8", 10): [9, 10],
        ("m8", 11): [10],
        ("m8", 12): [],
        ("m8", 13): [13],
        ("m8", 14): [],
        ("m8", 15): [15],
        ("m8", 16): [16],
        ("m8", 17): [17],
        ("m8", 18): [14],
        ("m8", 19): [18],
        ("m8", 20): [20],
        ("m8", 21): [18],
        ("m8", 22): [22],
        ("m8", 23): [22],
        ("m8", 24): [23, 24],
        ("m8", 25): [19],
        ("m8", 26): [26],
        ("m8", 27): [27],
        ("m8", 28): [28],
        ("m8", 29): [29],
        ("m8", 30): [30],
        ("m8", 31): [1, 5, 9, 18],
        ("m8", 32): [32],
        ("m8", 33): [33],
        ("m8", 34): [33],
        ("m8", 35): [34, 35],
        ("m8", 36): [36],
        ("m8", 37): [37],
        ("m8", 38): [38],
        ("m8", 39): [39],
        ("m8", 40): [40],
```

```
        ("m8", 41): [41],
        ("m8", 42): [36],
        ("m8", 43): [43],
        ("m8", 44): [44],
        ("m8", 45): [45],
        ("m8", 46): [18]
    }
    SUC_all = {
        ("m1", 0): [1, 2],
        ("m1", 1): [3],
        ("m1", 2): [4, 5],
        ("m1", 3): [7],
        ("m1", 4): [6],
        ("m1", 5): [8],
        ("m1", 6): [9],
        ("m1", 7): [10],
        ("m2", 0): [1, 2],
        ("m2", 1): [3],
        ("m2", 2): [30],
        ("m2", 3): [4],
        ("m2", 4): [10],
        ("m2", 5): [11],
        ("m2", 6): [12, 13, 14],
        ("m2", 7): [19, 20],
        ("m2", 8): [25, 26],
        ("m2", 9): [24],
        ("m2", 10): [18, 21],
        ("m2", 11): [5, 15],
        ("m2", 12): [22, 23],
        ("m2", 13): [27],
        ("m2", 14): [6, 16],
        ("m2", 15): [28],
        ("m2", 16): [7, 17],
        ("m2", 17): [29],
        ("m2", 18): [8],
        ("m2", 19): [9],
        ("m3", 0): [1, 2],
        ("m3", 1): [3, 4, 5, 6],
        ("m3", 2): [7, 8],
        ("m3", 3): [9, 10, 11],
        ("m3", 4): [12, 13, 14, 15],
        ("m3", 5): [16, 17, 18, 19],
        ("m3", 6): [20, 21, 22],
        ("m3", 7): [23],
        ("m3", 8): [24, 25, 26],
        ("m3", 9): [27, 28, 29],
        ("m3", 10): [30, 31],
        ("m3", 11): [32, 33, 34],
```

```
        ("m3", 12): [35, 36, 37, 38],
        ("m3", 13): [39, 40, 41],
        ("m3", 14): [42, 43, 44],
        ("m3", 15): [45, 46, 47],
        ("m3", 16): [48, 49],
        ("m3", 17): [50, 51],
        ("m3", 18): [52, 53, 54],
        ("m3", 19): [55, 56],
        ("m3", 20): [57, 58],
        ("m3", 21): [59, 60],
        ("m3", 22): [61, 62, 63],
        ("m3", 23): [64, 65],
        ("m3", 24): [66, 67, 68],
        ("m3", 25): [69, 70, 71],
        ("m3", 26): [72, 73],
        ("m3", 27): [74],
        ("m3", 28): [75, 76],
        ("m3", 29): [77],
        ("m3", 30): [76, 79],
        ("m3", 31): [80, 81],
        ("m3", 32): [82],
        ("m3", 33): [83, 84],
        ("m3", 34): [85, 86],
        ("m3", 35): [87, 88],
        ("m3", 36): [89, 90],
        ("m3", 37): [91],
        ("m3", 38): [92],
        ("m3", 39): [93],
        ("m3", 40): [94],
        ("m3", 41): [95],
        ("m3", 42): [96],
        ("m3", 43): [97],
        ("m4", 0): [1, 2],
        ("m4", 1): [3, 4],
        ("m4", 2): [5],
        ("m4", 3): [6],
        ("m4", 4): [7, 8, 9],
        ("m4", 5): [10],
        ("m4", 6): [11],
        ("m4", 7): [12, 14],
        ("m4", 8): [13, 15],
        ("m4", 9): [16],
        ("m4", 10): [17],
        ("m4", 11): [18],
        ("m4", 12): [19],
        ("m4", 13): [20],
        ("m5", 0): [1],
        ("m5", 1): [5],
```

```
        ("m5", 2): [3],
        ("m5", 3): [2],
        ("m5", 4): [6],
        ("m5", 5): [8],
        ("m5", 6): [7],
        ("m5", 7): [4],
        ("m6", 0): [1],
        ("m6", 1): [4],
        ("m6", 2): [5],
        ("m6", 3): [6],
        ("m6", 4): [9],
        ("m6", 5): [10],
        ("m6", 6): [7],
        ("m6", 7): [8],
        ("m6", 8): [2],
        ("m6", 9): [3],
        ("m7", 0): [5],
        ("m7", 1): [4],
        ("m7", 2): [2],
        ("m7", 3): [1],
        ("m7", 4): [10],
        ("m7", 5): [11],
        ("m7", 6): [12],
        ("m7", 7): [8],
        ("m7", 8): [7],
        ("m7", 9): [6],
        ("m7", 10): [3],
        ("m7", 11): [9],
        ("m7", 12): [16],
        ("m7", 13): [15],
        ("m7", 14): [14],
        ("m7", 15): [13],
        ("m7", 16): [18],
        ("m7", 17): [17],
        ("m7", 18): [19],
        ("m7", 19): [21],
        ("m7", 20): [20],
        ("m7", 21): [22],
        ("m7", 22): [25],
        ("m7", 23): [23],
        ("m7", 24): [24],
        ("m8", 0): [1],
        ("m8", 1): [2],
        ("m8", 2): [3],
        ("m8", 3): [4],
        ("m8", 4): [5],
        ("m8", 5): [6],
        ("m8", 6): [7],
```

```
        ("m8", 7): [8],
        ("m8", 8): [9],
        ("m8", 9): [10],
        ("m8", 10): [11],
        ("m8", 11): [12],
        ("m8", 12): [13],
        ("m8", 13): [14],
        ("m8", 14): [15],
        ("m8", 15): [16],
        ("m8", 16): [17],
        ("m8", 17): [18],
        ("m8", 18): [19],
        ("m8", 19): [20],
        ("m8", 20): [21],
        ("m8", 21): [22],
        ("m8", 22): [23],
        ("m8", 23): [24],
        ("m8", 24): [25],
        ("m8", 25): [26],
        ("m8", 26): [27],
        ("m8", 27): [28],
        ("m8", 28): [29],
        ("m8", 29): [30],
        ("m8", 30): [31],
        ("m8", 31): [32],
        ("m8", 32): [33],
        ("m8", 33): [34],
        ("m8", 34): [35],
        ("m8", 35): [36],
        ("m8", 36): [37],
        ("m8", 37): [38],
        ("m8", 38): [39],
        ("m8", 39): [40],
        ("m8", 40): [41],
        ("m8", 41): [42],
        ("m8", 42): [43],
        ("m8", 43): [44],
        ("m8", 44): [45],
        ("m8", 45): [46],
        ("m8", 46): [47]
    }
    THETA_all = {
        ("m1", 1): [1],
        ("m1", 2): [2],
        ("m1", 3): [2],
        ("m1", 4): [1, 2],
        ("m1", 5): [2],
        ("m1", 6): [1],
```

```
        ("m1", 7): [1, 2],
        ("m1", 8): [1, 2],
        ("m1", 9): [2],
        ("m1", 10): [1, 2],
        ("m2", 1): [2],
        ("m2", 2): [1],
        ("m2", 3): [1],
        ("m2", 4): [1, 2],
        ("m2", 5): [2],
        ("m2", 6): [2],
        ("m2", 7): [2],
        ("m2", 8): [1],
        ("m2", 9): [1],
        ("m2", 10): [1, 2],
        ("m2", 11): [1, 2],
        ("m2", 12): [1, 2],
        ("m2", 13): [1, 2],
        ("m2", 14): [1],
        ("m2", 15): [1],
        ("m2", 16): [1, 2],
        ("m2", 17): [2],
        ("m2", 18): [2],
        ("m2", 19): [1, 2],
        ("m2", 20): [1],
        ("m2", 21): [2],
        ("m2", 22): [1],
        ("m2", 23): [1],
        ("m2", 24): [2],
        ("m2", 25): [2],
        ("m2", 26): [1, 2],
        ("m2", 27): [1],
        ("m2", 28): [1],
        ("m2", 29): [1, 2],
        ("m2", 30): [1, 2],
        ("m3", 1): [1],
        ("m3", 2): [1],
        ("m3", 3): [1, 2],
        ("m3", 4): [1, 2],
        ("m3", 5): [1, 2],
        ("m3", 6): [1],
        ("m3", 7): [1, 2],
        ("m3", 8): [2],
        ("m3", 9): [1, 2],
        ("m3", 10): [2],
        ("m3", 11): [1],
        ("m3", 12): [1, 2],
        ("m3", 13): [1],
        ("m3", 14): [1],
```

```
        ("m3", 15): [2],
        ("m3", 16): [1, 2],
        ("m3", 17): [2],
        ("m3", 18): [2],
        ("m3", 19): [1, 2],
        ("m3", 20): [1, 2],
        ("m3", 21): [1, 2],
        ("m3", 22): [1],
        ("m3", 23): [2],
        ("m3", 24): [2],
        ("m3", 25): [2],
        ("m3", 26): [1],
        ("m3", 27): [2],
        ("m3", 28): [1, 2],
        ("m3", 29): [2],
        ("m3", 30): [1, 2],
        ("m3", 31): [1],
        ("m3", 32): [1, 2],
        ("m3", 33): [1],
        ("m3", 34): [1],
        ("m3", 35): [1, 2],
        ("m3", 36): [1, 2],
        ("m3", 37): [2],
        ("m3", 38): [2],
        ("m3", 39): [2],
        ("m3", 40): [1, 2],
        ("m3", 41): [1],
        ("m3", 42): [2],
        ("m3", 43): [1],
        ("m3", 44): [2],
        ("m3", 45): [2],
        ("m3", 46): [2],
        ("m3", 47): [1],
        ("m3", 48): [1, 2],
        ("m3", 49): [1, 2],
        ("m3", 50): [2],
        ("m3", 51): [1, 2],
        ("m3", 52): [1],
        ("m3", 53): [2],
        ("m3", 54): [1],
        ("m3", 55): [1],
        ("m3", 56): [2],
        ("m3", 57): [1, 2],
        ("m3", 58): [1, 2],
        ("m3", 59): [1],
        ("m3", 60): [1, 2],
        ("m3", 61): [2],
        ("m3", 62): [1],
```

```
        ("m3", 63): [2],
        ("m3", 64): [2],
        ("m3", 65): [1],
        ("m3", 66): [2],
        ("m3", 67): [1],
        ("m3", 68): [2],
        ("m3", 69): [1],
        ("m3", 70): [2],
        ("m3", 71): [1],
        ("m3", 72): [1, 2],
        ("m3", 73): [2],
        ("m3", 74): [1],
        ("m3", 75): [2],
        ("m3", 76): [1, 2],
        ("m3", 77): [1],
        ("m3", 78): [2],
        ("m3", 79): [1, 2],
        ("m3", 80): [1],
        ("m3", 81): [1],
        ("m3", 82): [2],
        ("m3", 83): [1, 2],
        ("m3", 84): [1, 2],
        ("m3", 85): [1, 2],
        ("m3", 86): [2],
        ("m3", 87): [1, 2],
        ("m3", 88): [1, 2],
        ("m3", 89): [1],
        ("m3", 90): [2],
        ("m3", 91): [2],
        ("m3", 92): [1, 2],
        ("m3", 93): [2],
        ("m3", 94): [1],
        ("m3", 95): [1, 2],
        ("m3", 96): [2],
        ("m3", 97): [1],
        ("m4", 1): [1],
        ("m4", 2): [1, 2],
        ("m4", 3): [2],
        ("m4", 4): [2],
        ("m4", 5): [1],
        ("m4", 6): [1, 2],
        ("m4", 7): [1, 2],
        ("m4", 8): [1, 2],
        ("m4", 9): [2],
        ("m4", 10): [2],
        ("m4", 11): [1, 2],
        ("m4", 12): [1],
        ("m4", 13): [1],
```

```
        ("m4", 14): [1, 2],
        ("m4", 15): [1],
        ("m4", 16): [1],
        ("m4", 17): [2],
        ("m4", 18): [2],
        ("m4", 19): [1, 2],
        ("m4", 20): [1, 2],
        ("m5", 1): [1, 2],
        ("m5", 2): [2],
        ("m5", 3): [2],
        ("m5", 4): [1, 2],
        ("m5", 5): [1],
        ("m5", 6): [1, 2],
        ("m5", 7): [1],
        ("m5", 8): [1, 2],
        ("m6", 1): [1, 2],
        ("m6", 2): [1, 2],
        ("m6", 3): [1, 2],
        ("m6", 4): [2],
        ("m6", 5): [1],
        ("m6", 6): [1, 2],
        ("m6", 7): [1, 2],
        ("m6", 8): [1],
        ("m6", 9): [2],
        ("m6", 10): [1, 2],
        ("m7", 1): [2],
        ("m7", 2): [1, 2],
        ("m7", 3): [1, 2],
        ("m7", 4): [1, 2],
        ("m7", 5): [1],
        ("m7", 6): [1, 2],
        ("m7", 7): [1, 2],
        ("m7", 8): [1],
        ("m7", 9): [2],
        ("m7", 10): [1, 2],
        ("m7", 11): [2],
        ("m7", 12): [1, 2],
        ("m7", 13): [1, 2],
        ("m7", 14): [1, 2],
        ("m7", 15): [2],
        ("m7", 16): [1, 2],
        ("m7", 17): [1, 2],
        ("m7", 18): [1, 2],
        ("m7", 19): [1, 2],
        ("m7", 20): [2],
        ("m7", 21): [1],
        ("m7", 22): [1, 2],
        ("m7", 23): [1, 2],
```

```
        ("m7", 24): [1, 2],
        ("m7", 25): [1],
        ("m8", 1): [1, 2],
        ("m8", 2): [1],
        ("m8", 3): [1, 2],
        ("m8", 4): [2],
        ("m8", 5): [1],
        ("m8", 6): [2],
        ("m8", 7): [1, 2],
        ("m8", 8): [1, 2],
        ("m8", 9): [1, 2],
        ("m8", 10): [1, 2],
        ("m8", 11): [1],
        ("m8", 12): [1, 2],
        ("m8", 13): [1],
        ("m8", 14): [1, 2],
        ("m8", 15): [1, 2],
        ("m8", 16): [2],
        ("m8", 17): [1, 2],
        ("m8", 18): [1],
        ("m8", 19): [1, 2],
        ("m8", 20): [2],
        ("m8", 21): [1, 2],
        ("m8", 22): [1, 2],
        ("m8", 23): [2],
        ("m8", 24): [1, 2],
        ("m8", 25): [1],
        ("m8", 26): [1, 2],
        ("m8", 27): [2],
        ("m8", 28): [1, 2],
        ("m8", 29): [1, 2],
        ("m8", 30): [2],
        ("m8", 31): [1],
        ("m8", 32): [1, 2],
        ("m8", 33): [2],
        ("m8", 34): [1, 2],
        ("m8", 35): [1, 2],
        ("m8", 36): [1, 2],
        ("m8", 37): [1],
        ("m8", 38): [1, 2],
        ("m8", 39): [1, 2],
        ("m8", 40): [1, 2],
        ("m8", 41): [2],
        ("m8", 42): [1, 2],
        ("m8", 43): [2],
        ("m8", 44): [1, 2],
        ("m8", 45): [1, 2],
        ("m8", 46): [1, 2],
```

```
        ("m8", 47): [1, 2]
    }
    t_all = {
        ("m1",1): 30,
        ("m1",2): 28,
        ("m1",3): 12,
        ("m1",4): 34,
        ("m1",5): 13,
        ("m1",6): 21,
        ("m1",7): 6,
        ("m1",8): 18,
        ("m1",9): 25,
        ("m1",10): 10,
        ("m2",1): 11,
        ("m2",2): 20,
        ("m2",3): 20,
        ("m2",4): 14,
        ("m2",5): 19,
        ("m2",6): 1,
        ("m2",7): 7,
        ("m2",8): 6,
        ("m2",9): 6,
        ("m2",10): 7,
        ("m2",11): 19,
        ("m2",12): 11,
        ("m2",13): 18,
        ("m2",14): 13,
        ("m2",15): 5,
        ("m2",16): 11,
        ("m2",17): 6,
        ("m2",18): 4,
        ("m2",19): 6,
        ("m2",20): 8,
        ("m2",21): 18,
        ("m2",22): 15,
        ("m2",23): 15,
        ("m2",24): 6,
        ("m2",25): 10,
        ("m2",26): 20,
        ("m2",27): 13,
        ("m2",28): 1,
        ("m2",29): 4,
        ("m2",30): 5,
        ("m3",1): 37,
        ("m3",2): 3,
        ("m3",3): 5,
        ("m3",4): 3,
        ("m3",5): 10,
```

C:\Users\yusuf\Desktop\masaüstü\Dosyalar\Projeler\28th CIRP Conference on Life Cycle Engineering\gurobi_opt\inputs.py

```
        ("m3",6): 6,
        ("m3",7): 37,
        ("m3",8): 2,
        ("m3",9): 3,
        ("m3",10): 10,
        ("m3",11): 6,
        ("m3",12): 5,
        ("m3",13): 2,
        ("m3",14): 10,
        ("m3",15): 6,
        ("m3",16): 5,
        ("m3",17): 3,
        ("m3",18): 19,
        ("m3",19): 6,
        ("m3",20): 5,
        ("m3",21): 3,
        ("m3",22): 10,
        ("m3",23): 37,
        ("m3",24): 2,
        ("m3",25): 10,
        ("m3",26): 6,
        ("m3",27): 3,
        ("m3",28): 19,
        ("m3",29): 6,
        ("m3",30): 3,
        ("m3",31): 10,
        ("m3",32): 5,
        ("m3",33): 10,
        ("m3",34): 6,
        ("m3",35): 5,
        ("m3",36): 2,
        ("m3",37): 19,
        ("m3",38): 6,
        ("m3",39): 5,
        ("m3",40): 2,
        ("m3",41): 10,
        ("m3",42): 5,
        ("m3",43): 3,
        ("m3",44): 6,
        ("m3",45): 5,
        ("m3",46): 3,
        ("m3",47): 19,
        ("m3",48): 11,
        ("m3",49): 34,
        ("m3",50): 10,
        ("m3",51): 6,
        ("m3",52): 2,
        ("m3",53): 19,
```

C:\Users\yusuf\Desktop\masaüstü\Dosyalar\Projeler\28th CIRP Conference on Life Cycle Engineering\gurobi_opt\inputs.py

```
        ("m3",54): 6,
        ("m3",55): 2,
        ("m3",56): 10,
        ("m3",57): 3,
        ("m3",58): 6,
        ("m3",59): 3,
        ("m3",60): 19,
        ("m3",61): 5,
        ("m3",62): 19,
        ("m3",63): 6,
        ("m3",64): 5,
        ("m3",65): 10,
        ("m3",66): 5,
        ("m3",67): 2,
        ("m3",68): 6,
        ("m3",69): 5,
        ("m3",70): 2,
        ("m3",71): 19,
        ("m3",72): 5,
        ("m3",73): 3,
        ("m3",74): 11,
        ("m3",75): 19,
        ("m3",76): 6,
        ("m3",77): 10,
        ("m3",78): 2,
        ("m3",79): 6,
        ("m3",80): 2,
        ("m3",81): 19,
        ("m3",82): 3,
        ("m3",83): 5,
        ("m3",84): 6,
        ("m3",85): 5,
        ("m3",86): 19,
        ("m3",87): 5,
        ("m3",88): 2,
        ("m3",89): 29,
        ("m3",90): 34,
        ("m3",91): 6,
        ("m3",92): 19,
        ("m3",93): 2,
        ("m3",94): 5,
        ("m3",95): 34,
        ("m3",96): 29,
        ("m3",97): 14,
        ("m4", 1): 5,
        ("m4", 2): 28,
        ("m4", 3): 32,
        ("m4", 4): 7,
```

Page 16, last modified 01 Sep 2021 21:11:04

C:\Users\yusuf\Desktop\masaüstü\Dosyalar\Projeler\28th CIRP Conference on Life Cycle Engineering\gurobi_opt\inputs.py

```
        ("m4", 5): 19,
        ("m4", 6): 11,
        ("m4", 7): 38,
        ("m4", 8): 29,
        ("m4", 9): 16,
        ("m4", 10): 17,
        ("m4", 11): 33,
        ("m4", 12): 16,
        ("m4", 13): 6,
        ("m4", 14): 10,
        ("m4", 15): 32,
        ("m4", 16): 18,
        ("m4", 17): 16,
        ("m4", 18): 32,
        ("m4", 19): 35,
        ("m4", 20): 17,
        ("m5", 1): 14,
        ("m5", 2): 10,
        ("m5", 3): 12,
        ("m5", 4): 18,
        ("m5", 5): 23,
        ("m5", 6): 16,
        ("m5", 7): 20,
        ("m5", 8): 36,
        ("m6", 1): 14,
        ("m6", 2): 10,
        ("m6", 3): 12,
        ("m6", 4): 17,
        ("m6", 5): 23,
        ("m6", 6): 14,
        ("m6", 7): 19,
        ("m6", 8): 36,
        ("m6", 9): 14,
        ("m6", 10): 10,
        ("m7", 1): 3,
        ("m7", 2): 2,
        ("m7", 3): 3,
        ("m7", 4): 10,
        ("m7", 5): 10,
        ("m7", 6): 15,
        ("m7", 7): 15,
        ("m7", 8): 15,
        ("m7", 9): 15,
        ("m7", 10): 2,
        ("m7", 11): 2,
        ("m7", 12): 2,
        ("m7", 13): 2,
        ("m7", 14): 2,
```

```
        ("m7", 15): 2,
        ("m7", 16): 2,
        ("m7", 17): 2,
        ("m7", 18): 3,
        ("m7", 19): 18,
        ("m7", 20): 5,
        ("m7", 21): 1,
        ("m7", 22): 5,
        ("m7", 23): 15,
        ("m7", 24): 2,
        ("m7", 25): 2,
        ("m8", 1): 14,
        ("m8", 2): 28,
        ("m8", 3): 3,
        ("m8", 4): 2,
        ("m8", 5): 3,
        ("m8", 6): 4,
        ("m8", 7): 8,
        ("m8", 8): 12,
        ("m8", 9): 4,
        ("m8", 10): 28,
        ("m8", 11): 3,
        ("m8", 12): 4,
        ("m8", 13): 6,
        ("m8", 14): 1,
        ("m8", 15): 20,
        ("m8", 16): 5,
        ("m8", 17): 28,
        ("m8", 18): 4,
        ("m8", 19): 3,
        ("m8", 20): 12,
        ("m8", 21): 3,
        ("m8", 22): 3,
        ("m8", 23): 28,
        ("m8", 24): 28,
        ("m8", 25): 12,
        ("m8", 26): 76,
        ("m8", 27): 6,
        ("m8", 28): 28,
        ("m8", 29): 3,
        ("m8", 30): 6,
        ("m8", 31): 3,
        ("m8", 32): 98,
        ("m8", 33): 14,
        ("m8", 34): 2,
        ("m8", 35): 6,
        ("m8", 36): 7,
        ("m8", 37): 60,
```

```
        ("m8", 38): 6,
        ("m8", 39): 60,
        ("m8", 40): 8,
        ("m8", 41): 12,
        ("m8", 42): 3,
        ("m8", 43): 12,
        ("m8", 44): 3,
        ("m8", 45): 28,
        ("m8", 46): 2,
        ("m8", 47): 3
    }

    MODELS = []
    nA = {}
    nN = {}
    PRE = {}
    SUC = {}
    THETA = {}
    C = 36
    t = {}
    nJ = 0

    if mod1 == 1:
        MODELS.append("m1")
        nA["m1"] = nA_all["m1"]
        nN["m1"] = nN_all["m1"]
        for k in range(0, nA["m1"]):
            if ("m1", k) in SUC_all:
                SUC[("m1", k)] = SUC_all[("m1", k)]
            if ("m1", k) in PRE_all:
                PRE[("m1", k)] = PRE_all[("m1", k)]
        for i in range(1, nN["m1"] + 1):
            THETA[("m1", i)] = THETA_all[("m1", i)]
            t[("m1", i)] = t_all[("m1", i)]
        if nJ < 5:
            nJ = 5
    if mod2 == 1:
        MODELS.append("m2")
        nA["m2"] = nA_all["m2"]
        nN["m2"] = nN_all["m2"]
        for k in range(0, nA["m2"]):
            if ("m2", k) in SUC_all:
                SUC[("m2", k)] = SUC_all[("m2", k)]
            if ("m2", k) in PRE_all:
                PRE[("m2", k)] = PRE_all[("m2", k)]
        for i in range(1, nN["m2"] + 1):
            THETA[("m2", i)] = THETA_all[("m2", i)]
            t[("m2", i)] = t_all[("m2", i)]
```

```
        if nJ < 8:
            nJ = 8
    if mod3 == 1:
        MODELS.append("m3")
        nA["m3"] = nA_all["m3"]
        nN["m3"] = nN_all["m3"]
        for k in range(0, nA["m3"]):
            if ("m3", k) in SUC_all:
                SUC[("m3", k)] = SUC_all[("m3", k)]
            if ("m3", k) in PRE_all:
                PRE[("m3", k)] = PRE_all[("m3", k)]
        for i in range(1, nN["m3"] + 1):
            THETA[("m3", i)] = THETA_all[("m3", i)]
            t[("m3", i)] = t_all[("m3", i)]
        if nJ < 8:
            nJ = 8
    if mod4 == 1:
        MODELS.append("m4")
        nA["m4"] = nA_all["m4"]
        nN["m4"] = nN_all["m4"]
        for k in range(0, nA["m4"]):
            if ("m4", k) in SUC_all:
                SUC[("m4", k)] = SUC_all[("m4", k)]
            if ("m4", k) in PRE_all:
                PRE[("m4", k)] = PRE_all[("m4", k)]
        for i in range(1, nN["m4"] + 1):
            THETA[("m4", i)] = THETA_all[("m4", i)]
            t[("m4", i)] = t_all[("m4", i)]
        if nJ < 9:
            nJ = 9
    if mod5 == 1:
        MODELS.append("m5")
        nA["m5"] = nA_all["m5"]
        nN["m5"] = nN_all["m5"]
        for k in range(0, nA["m5"]):
            if ("m5", k) in SUC_all:
                SUC[("m5", k)] = SUC_all[("m5", k)]
            if ("m5", k) in PRE_all:
                PRE[("m5", k)] = PRE_all[("m5", k)]
        for i in range(1, nN["m5"] + 1):
            THETA[("m5", i)] = THETA_all[("m5", i)]
            t[("m5", i)] = t_all[("m5", i)]
        if nJ < 5:
            nJ = 5
    if mod6 == 1:
        MODELS.append("m6")
        nA["m6"] = nA_all["m6"]
        nN["m6"] = nN_all["m6"]
```

```
        for k in range(0, nA["m6"]):
            if ("m6", k) in SUC_all:
                SUC[("m6", k)] = SUC_all[("m6", k)]
            if ("m6", k) in PRE_all:
                PRE[("m6", k)] = PRE_all[("m6", k)]
        for i in range(1, nN["m6"] + 1):
            THETA[("m6", i)] = THETA_all[("m6", i)]
            t[("m6", i)] = t_all[("m6", i)]
        if nJ < 5:
            nJ = 5
    if mod7 == 1:
        MODELS.append("m7")
        nA["m7"] = nA_all["m7"]
        nN["m7"] = nN_all["m7"]
        for k in range(0, nA["m7"]):
            if ("m7", k) in SUC_all:
                SUC[("m7", k)] = SUC_all[("m7", k)]
            if ("m7", k) in PRE_all:
                PRE[("m7", k)] = PRE_all[("m7", k)]
        for i in range(1, nN["m7"] + 1):
            THETA[("m7", i)] = THETA_all[("m7", i)]
            t[("m7", i)] = t_all[("m7", i)]
        if nJ < 6:
            nJ = 6
    if mod8 == 1:
        MODELS.append("m8")
        nA["m8"] = nA_all["m8"]
        nN["m8"] = nN_all["m8"]
        for k in range(0, nA["m8"]):
            if ("m8", k) in SUC_all:
                SUC[("m8", k)] = SUC_all[("m8", k)]
            if ("m8", k) in PRE_all:
                PRE[("m8", k)] = PRE_all[("m8", k)]
        for i in range(1, nN["m8"] + 1):
            THETA[("m8", i)] = THETA_all[("m8", i)]
            t[("m8", i)] = t_all[("m8", i)]
        if nJ < 6:
            nJ = 6

    STATIONS = {(j) for j in range(nJ)}
    SIDES = {(1), (2)}
    MONO = {(m,i) for m in MODELS for i in range(1, nN[m]+1)}
    MONONO = {(m,i,i2) for m in MODELS for i in range(1, nN[m]+1) for i2 in range(1, ↵
↳   nN[m]+1) if i2 < i}

    return nA, nN, MODELS, STATIONS, SIDES, MONO, MONONO, PRE, SUC, THETA, C, t
```

APPENDIX – 03C. Initial Solutions Python Code of MOMA for MTDLB Problem

C:\Users\yusuf\Desktop\masaüstü\Dosyalar\Projeler\28th CIRP Conference on Life Cycle Engineering\initial.py

```
import random as rnd
import numpy as np

def randoninit(MODELS, nN, nORSuc, maxselnode):
    sol = [[[rnd.randint(0,1) for _ in range(7)] for _ in range(nORSuc[MODELS[m]])] for m in range(len(MODELS))]
    for m in range(len(MODELS)):
        sol[m].append([rnd.randint(0,1) for _ in range(maxselnode[MODELS[m]])])
        sol[m].append(list(np.random.permutation(nN[MODELS[m]])))
    dec_sol = [[sum([(2**(6-p))*sol[m][n][p] for p in range(7)])/128 for n in range(nORSuc[MODELS[m]])] for m in range(len(MODELS))]
    return sol, dec_sol
```

APPENDIX – 03D. Other Algorithms Python Code of MOMA for MTDLB Problem

```
C:\Users\yusuf\Desktop\masaüstü\Dosyalar\Projeler\28th CIRP Conference on Life Cycle
Engineering\algorithms.py

import math

def AlgoCDS(MODELS, nA, suc, pre, dec_sol):
    selected = [list() for _ in range(len(MODELS))]
    for m in range(len(MODELS)):
        ## Set Initial Values
        cA = [k for k in range(nA[MODELS[m]]) if pre[MODELS[m],k] == []]
        count_ORsuc = 0
        while cA != []:
            for k in cA:
                if (len(suc[MODELS[m], k]) > 1):
                    for i in range(len(suc[MODELS[m], k])):
                        if suc[MODELS[m], k][i] in selected[m]:
                            break
                        elif i == len(suc[MODELS[m], k]) - 1:
                            selected[m].append(suc[MODELS[m], ↵
↳                           k][math.ceil((len(suc[MODELS[m], ↵
↳                           k]))*(dec_sol[m][count_ORsuc]))-1])
                            count_ORsuc += 1
                            break
                elif (len(suc[MODELS[m], k]) == 1) and (suc[MODELS[m], k][0] not in ↵
↳               selected[m]):
                    selected[m].append(suc[MODELS[m], k][0])
            ## Update cA
            cA = [k for k in range(nA[MODELS[m]]) for i in range(len(selected[m])) if ↵
↳           (MODELS[m], k) in pre if selected[m][i] in pre[MODELS[m], k]]
            removed = list()
            for k in cA:
                for i in suc[MODELS[m], k]:
                    if i in selected[m]:
                        removed.append(k)
                        break
            for r in removed:
                cA.pop(cA.index(r))
    return selected

def AlgoADL(sol, selected, nA, nORSuc, MODELS, STATIONS, SIDES, SUC, PRE, THETA, t, C):
    X = [[[[0 for _ in range(len(SIDES))] for _ in range(len(STATIONS))] for _ in ↵
↳   range(len(selected[m])))] for m in range(len(MODELS))]
    U = [[0 for _ in range(len(SIDES))] for _ in range(len(STATIONS))]
    tf = [[0 for _ in range(len(selected[m])))] for m in range(len(MODELS))]
    task_pre, task_theta = setpretheta(MODELS, selected, nA, PRE, SUC, THETA, sol, nORSuc)
    for m in range(len(MODELS)):
        j = 0
        assignable_task = [i for i in selected[m] if task_pre[m][selected[m].index(i)] ↵
↳       == []]
        assigned_task = list()
        while len(assigned_task) != len(selected[m]):

Page 1, last modified 27 Nov 2020 11:03:12
```

```
            if len(assignable_task) == 1:
                r = 0
                j, X, tf, assigned_task, assignable_task, U = assign_rule(m, r, j, ↵
↳               MODELS, C, U, selected, assignable_task, assigned_task, t, task_pre, ↵
↳               task_theta, X, tf)
            elif len(assignable_task) >= 2:
                r = 0
                for o in range(1, len(assignable_task)):
                    if sol[m][nORSuc[MODELS[m]]+1].index(assignable_task[o]-1) < ↵
↳                   sol[m][nORSuc[MODELS[m]]+1].index(assignable_task[r]-1):
                        r = o
                j, X, tf, assigned_task, assignable_task, U = assign_rule(m, r, j, ↵
↳               MODELS, C, U, selected, assignable_task, assigned_task, t, task_pre, ↵
↳               task_theta, X, tf)
            for i in selected[m]:
                if (i not in assigned_task) and (i not in assignable_task):
                    for i2 in task_pre[m][selected[m].index(i)]:
                        if i2 not in assigned_task:
                            break
                        elif i2 == ↵
↳                       task_pre[m][selected[m].index(i)][len(task_pre[m][selected[m].ind↵
↳                       ex(i)]) - 1]:
                            assignable_task.append(i)
    return X, tf, U

def assign_rule(m, r, j, MODELS, C, U, selected, assignable_task, assigned_task, t, ↵
↳task_pre, task_theta, X, tf):
    if assigned_task == []:
        assigned_task.append(assignable_task[r])
        ↵
↳       X[m][selected[m].index(assignable_task[r])][j][task_theta[m][selected[m].index(as↵
↳       signable_task[r])][0]-1] = 1
        tf[m][selected[m].index(assignable_task[r])] = t[MODELS[m], assignable_task[r]]
        U[j][task_theta[m][selected[m].index(assignable_task[r])][0]-1] = 1
        assignable_task.pop(r)
    else:
        ftime = 0
        for i in range(len(task_pre[m][selected[m].index(assignable_task[r])])):
            if ↵
↳           (X[m][selected[m].index(task_pre[m][selected[m].index(assignable_task[r])][i]↵
↳           )][j][0] == 1) or ↵
↳           (X[m][selected[m].index(task_pre[m][selected[m].index(assignable_task[r])][i]↵
↳           )][j][1] == 1):
                ftime = max(ftime, ↵
↳               tf[m][selected[m].index(task_pre[m][selected[m].index(assignable_task[r])↵
↳               ][i])])
            if i == len(task_pre[m][selected[m].index(assignable_task[r])]) - 1:
                if ftime + t[MODELS[m], assignable_task[r]] > C:
```

```
                    j += 1
                    assigned_task.append(assignable_task[r])
                    X[m][selected[m].index(assignable_task[r])][j][task_theta[m][selected[m].index(assignable_task[r])][0]-1] = 1
                    tf[m][selected[m].index(assignable_task[r])] = t[MODELS[m], assignable_task[r]]
                    U[j][task_theta[m][selected[m].index(assignable_task[r])][0]-1] = 1
                    assignable_task.pop(r)
                    break
                else:
                    aux = 1
                    for tim in range(ftime, C-tf[m][selected[m].index(assignable_task[r])]):
                        aux = 1
                        for i2 in assigned_task:
                            if X[m][selected[m].index(i2)][j][task_theta[m][selected[m].index(assignable_task[r])][0]-1] == 1:
                                if (tf[m][selected[m].index(i2)] - t[MODELS[m], i2] <= tim) and (tf[m][selected[m].index(i2)] > tim):
                                    aux = 0
                                    break
                        if (aux == 1) and (i2 == assigned_task[len(assigned_task)-1]) and (tim + t[MODELS[m], assignable_task[r]] <= C):
                            assigned_task.append(assignable_task[r])
                            X[m][selected[m].index(assignable_task[r])][j][task_theta[m][selected[m].index(assignable_task[r])][0]-1] = 1
                            tf[m][selected[m].index(assignable_task[r])] = tim + t[MODELS[m], assignable_task[r]]
                            U[j][task_theta[m][selected[m].index(assignable_task[r])][0]-1] = 1
                            assignable_task.pop(r)
                            break
                        elif (aux == 1) and (i2 == assigned_task[len(assigned_task)-1]):
                            j += 1
                            assigned_task.append(assignable_task[r])

                            X[m][selected[m].index(assignable_task[r])][j][task_theta[m][selected[m].index(assignable_task[r])][0]-1] = 1
                            tf[m][selected[m].index(assignable_task[r])] = t[MODELS[m], assignable_task[r]]
                            U[j][task_theta[m][selected[m].index(assignable_task[r])][0]-
```

```
                            1] = 1
                            assignable_task.pop(r)
                            break
                    if aux == 0:
                        j += 1
                        assigned_task.append(assignable_task[r])
                        X[m][selected[m].index(assignable_task[r])][j][task_theta[m][selected[m].index(assignable_task[r])][0]-1] = 1
                        tf[m][selected[m].index(assignable_task[r])] = t[MODELS[m], assignable_task[r]]
                        U[j][task_theta[m][selected[m].index(assignable_task[r])][0]-1] = 1
                        assignable_task.pop(r)
                        break
    return j, X, tf, assigned_task, assignable_task, U

def setpretheta(MODELS, selected, nA, PRE, SUC, THETA, sol, nORSuc):
    task_pre = [[list() for _ in range(len(selected[m]))] for m in range(len(MODELS))]
    task_theta = [[list() for _ in range(len(selected[m]))] for m in range(len(MODELS))]
    for m in range(len(MODELS)):
        for i in selected[m]:
            for k in range(nA[MODELS[m]]):
                if ((MODELS[m],k) in PRE) and (((MODELS[m],k) in SUC)):
                    if (i in SUC[MODELS[m],k]):
                        for i2 in PRE[MODELS[m],k]:
                            if i2 in selected[m]:
                                task_pre[m][selected[m].index(i)].append(i2)
            if len(THETA[MODELS[m], i]) == 1:
                task_theta[m][selected[m].index(i)].append(THETA[MODELS[m], i][0])
            else:
                task_theta[m][selected[m].index(i)].append(sol[m][nORSuc[MODELS[m]]][selected[m].index(i)] + 1)
    return task_pre, task_theta
```

APPENDIX – 03E. Calculations Python Code of MOMA for MTDLB Problem

C:\Users\yusuf\Desktop\masaüstü\Dosyalar\Projeler\28th CIRP Conference on Life Cycle Engineering\calculation.py

```
import time

def cost(U, selected, MODELS, STATIONS, EPSILON, t):
    DLBcost = 0
    DLBcost2 = 0
    DLBcost3 = 0
    F = [0 for _ in STATIONS]
    G = [0 for _ in STATIONS]
    for j in STATIONS:
        if (U[j][0] == 1) and (U[j][1] == 1):
            F[j] = 1
            DLBcost += 1
            DLBcost += (j+1)*2*EPSILON
            DLBcost2 += 2
        elif (U[j][0] == 1) or (U[j][1] == 1):
            G[j] = 1
            DLBcost += 1
            DLBcost += (j+1)*EPSILON
            DLBcost2 += 2
    for m in range(len(MODELS)):
        for i in selected[m]:
            DLBcost3 += t[MODELS[m], i]
    return F, G, DLBcost, DLBcost2, DLBcost3

def costupdate(BestCost1, BestCost2, BestCost3, DLBcost, DLBcost2, DLBcost3, ParetoF):
    if len(BestCost1) == 0:
        for i in ParetoF[0]:
            BestCost1.append(DLBcost[i])
            BestCost2.append(DLBcost2[i])
            BestCost3.append(DLBcost3[i])
    else:
        BestCost11 = []
        BestCost21 = []
        BestCost31 = []
        for p in ParetoF[0]:
            BestCost1.append(DLBcost[p])
            BestCost2.append(DLBcost2[p])
            BestCost3.append(DLBcost3[p])
        F, degree = FastNonDominatedSort(len(BestCost1), BestCost1, BestCost2, BestCost3)
        for p in F[0]:
            BestCost11.append(BestCost1[p])
            BestCost21.append(BestCost2[p])
            BestCost31.append(BestCost3[p])
        BestCost1 = BestCost11
        BestCost2 = BestCost21
        BestCost3 = BestCost31
        while True:
            again = 0
```

Page 1, last modified 01 Sep 2021 22:57:28

```
            for p1 in range(len(BestCost1)):
                for p2 in range(len(BestCost1)):
                    if p1 != p2:
                        if (BestCost1[p1] == BestCost1[p2]) and (BestCost2[p1] == ↵
↳                       BestCost2[p2]) and (BestCost3[p1] == BestCost3[p2]):
                            BestCost1.pop(p1)
                            BestCost2.pop(p1)
                            BestCost3.pop(p1)
                            again = 1
                            break
                if again == 1:
                    break
            if again == 0:
                break


    return BestCost1, BestCost2, BestCost3


def FastNonDominatedSort(NoP, cost1, cost2, cost3):
    S = [[] for _ in range(NoP)]
    n = [0 for _ in range(NoP)]
    degree = [0 for _ in range(NoP)]
    F = [[]]
    for p in range(NoP):
        for q in range(NoP):
            if ((cost1[p] < cost1[q]) and (cost2[p] <= cost2[q]) and (cost3[p] <= ↵
↳           cost3[q])) or ((cost1[p] <= cost1[q]) and (cost2[p] < cost2[q]) and ↵
↳           (cost3[p] <= cost3[q])) or ((cost1[p] <= cost1[q]) and (cost2[p] <= ↵
↳           cost2[q]) and (cost3[p] < cost3[q])):
                S[p].append(q)
            elif ((cost1[q] < cost1[p]) and (cost2[q] <= cost2[p]) and (cost3[q] <= ↵
↳           cost3[p])) or ((cost1[q] <= cost1[p]) and (cost2[q] < cost2[p]) and ↵
↳           (cost3[q] <= cost3[p])) or ((cost1[q] <= cost1[p]) and (cost2[q] <= ↵
↳           cost2[p]) and (cost3[q] < cost3[p])):
                n[p] += 1

        if n[p] == 0:
            degree[p] = 1
            F[0].append(p)
    i = 0
    while F[i] != []:
        Q = []
        for p in F[i]:
            for q in S[p]:
                n[q] -= 1
                if n[q] == 0:
                    degree[q] = i + 1
```

```
                    Q.append(q)
        i += 1
        F.append(Q)
    return F, degree
```

APPENDIX – 03F. Genetic Operators Python Code of MOMA for MTDLB Problem

```
C:\Users\yusuf\Desktop\masaüstü\Dosyalar\Projeler\28th CIRP Conference on Life Cycle
Engineering\genetic_operators.py

import random as rnd

def roulette_wheel(nP, DLBcost):
    selection_points = [0 for _ in range(nP)]
    for p in range(nP):
        if p == 0:
            selection_points[p] = round((1/DLBcost[p])/(sum([1/DLBcost[p2] for p2 in ↵
            range(nP)])), 5)
        else:
            selection_points[p] = round(selection_points[p-1] + ↵
            (1/DLBcost[p])/(sum([1/DLBcost[p2] for p2 in range(nP)])), 5)
    selection_pp = [[0, 0] for _ in range(nP)]
    for nep in range(nP):
        while selection_pp[nep][0] == selection_pp[nep][1]:
            rnd_num1 = round(rnd.uniform(0,1), 5)
            rnd_num2 = round(rnd.uniform(0,1), 5)
            for p in range(nP):
                if rnd_num1 <= selection_points[p]:
                    selection_pp[nep][0] = p
                    break
            for p in range(nP):
                if rnd_num2 <= selection_points[p]:
                    selection_pp[nep][1] = p
                    break
    return selection_pp

def crossover(nP, nN, MODELS, nORSuc, maxselnode, sol, selection_pp):
    new_sol = [[[] for m in range(len(MODELS))] for p in range(nP)]
    # -- two point
    for nep in range(nP):
        for m in range(len(MODELS)):
            for nop in range(nORSuc[MODELS[m]]):
                new_sol[nep][m].append(sol[selection_pp[nep][0]][m][nop][0:2] + ↵
                sol[selection_pp[nep][1]][m][nop][2:5] + ↵
                sol[selection_pp[nep][0]][m][nop][5:7])
                ↵
            new_sol[nep][m].append(sol[selection_pp[nep][0]][m][nORSuc[MODELS[m]]][0:int(↵
            round((maxselnode[MODELS[m]])/4, 0))] + ↵
            sol[selection_pp[nep][1]][m][nORSuc[MODELS[m]]][int(round(maxselnode[MODELS[m↵
            ]]/4, 0)):int(round(3*maxselnode[MODELS[m]]/4, 0))] + ↵
            sol[selection_pp[nep][0]][m][nORSuc[MODELS[m]]][int(round(3*maxselnode[MODELS↵
            [m]]/4, 0)):int(round(maxselnode[MODELS[m]], 0))])

            # -- OX'1
            appending = []
            app1 = 0
            app2 = 0
            for _ in range(0, int(round((nN[MODELS[m]])/4, 0))):

Page 1, last modified 19 Sep 2020 15:55:56
```

```
                while True:
                    if sol[selection_pp[nep][0]][m][nORSuc[MODELS[m]]+1][app1] not in ↵
↳                   sol[selection_pp[nep][1]][m][nORSuc[MODELS[m]]+1][int(round(nN[MODELS↵
↳                   [m]]/4, 0)):int(round(3*nN[MODELS[m]]/4, 0))+1]:
                        ↵
↳                       appending.append(sol[selection_pp[nep][0]][m][nORSuc[MODELS[m]]+1↵
↳                       ][app1])
                        app1 += 1
                        break
                    else:
                        app1 += 1
            for f in range(int(round((nN[MODELS[m]])/4, 0)), ↵
↳           int(round(3*(nN[MODELS[m]])/4, 0))):
                appending.append(sol[selection_pp[nep][1]][m][nORSuc[MODELS[m]]+1][f])
            for _ in range(0, int(round((nN[MODELS[m]])/4, 0))):
                while True:
                    if sol[selection_pp[nep][0]][m][nORSuc[MODELS[m]]+1][app2] not in ↵
↳                   appending:
                        ↵
↳                       appending.append(sol[selection_pp[nep][0]][m][nORSuc[MODELS[m]]+1↵
↳                       ][app2])
                        app2 += 1
                        break
                    else:
                        app2 += 1
            new_sol[nep][m].append(appending)
    sol = []
    sol = new_sol
    dec_sol = [[] for nep in range(nP)]
    for nep in range(nP):
        dec_sol[nep] = [[sum([(2**(6-p))*sol[nep][m][n][p] for p in range(7)])/128 for ↵
↳       n in range(nORSuc[MODELS[m]])] for m in range(len(MODELS))]
    return sol, dec_sol
```

APPENDIX – 03G. Local Search Algorithm Python Code of MOMA for MTDLB Problem

```
C:\Users\yusuf\Desktop\masaüstü\Dosyalar\Projeler\28th CIRP Conference on Life Cycle
Engineering\local.py

import random as rnd

import algorithms
import calculation

def ils(t1, nP, maxils_iter, sol, dec_sol, MODELS, nORSuc, nN, nA, STATIONS, SIDES, ↵
↳SUC, PRE, THETA, t, C, selected, X, U, F, G, DLBcost, tf, bestcost, EPSILON):
    for mup in range(nP):
        for ils_iter in range(maxils_iter):
            ils_sol = sol[mup]
            for m in range(len(MODELS)):
                for i in range(nORSuc[MODELS[m]]+1):
                    if i == nORSuc[MODELS[m]]:
                        rand_point = rnd.randint(0, nORSuc[MODELS[m]])
                        if ils_sol[m][i][rand_point] == 0:
                            ils_sol[m][i][rand_point] = 1
                        else:
                            ils_sol[m][i][rand_point] = 1
                    else:
                        rand_point = rnd.randint(0, 6)
                        if ils_sol[m][i][rand_point] == 0:
                            ils_sol[m][i][rand_point] = 1
                        else:
                            ils_sol[m][i][rand_point] = 1
                while True:
                    point1 = rnd.randint(0, nN[MODELS[m]]-1)
                    point2 = rnd.randint(0, nN[MODELS[m]]-1)
                    if point1 != point2:
                        break
                ils_sol[m][nORSuc[MODELS[m]]+1][point1], ↵
↳               ils_sol[m][nORSuc[MODELS[m]]+1][point2] = ↵
↳               ils_sol[m][nORSuc[MODELS[m]]+1][point2], ↵
↳               ils_sol[m][nORSuc[MODELS[m]]+1][point1]
            dec_ils_sol = [[sum([(2**(6-p))*ils_sol[m][n][p] for p in range(7)])/128 ↵
↳           for n in range(nORSuc[MODELS[m]])] for m in range(len(MODELS))]
            select_ils = algorithms.AlgoCDS(MODELS, nA, SUC, PRE, dec_ils_sol)
            X_ils, tf_ils, U_ils = algorithms.AlgoADL(ils_sol, select_ils, nA, nORSuc, ↵
↳           MODELS, STATIONS, SIDES, SUC, PRE, THETA, t, C)
            F_ils, G_ils, DLBcost_ils = calculation.cost(U_ils, select_ils, MODELS, ↵
↳           STATIONS, EPSILON, t)
            if DLBcost_ils < DLBcost[mup]:
                DLBcost[mup] = DLBcost_ils
                sol[mup] = ils_sol
                dec_sol[mup] = dec_ils_sol
                selected[mup] = select_ils
                X[mup] = X_ils
                tf[mup] = tf_ils
                U[mup] = U_ils

Page 1, last modified 19 Sep 2020 16:55:58
```

```
                F[mup] = F_ils
                G[mup] = G_ils
                bestcost = calculation.costupdate(bestcost, DLBcost_ils, t1, MODELS, ↵
↳               STATIONS, SIDES, select_ils, X_ils, tf_ils, t)
    return DLBcost, sol, dec_sol, selected, X, tf, U, F, G, bestcost
```

CURRICULUM VITAE

ORCID ID:

I am Serkan Mutlu. I specialize in Operations Research, Life Cycle Engineering and Supply Chain Management. Brief information about myself:

Education:

Master 2018 – 2021 Eskişehir Technical University, Department of Industrial Engineering

Bachelor 2014 – 2018 Dokuz Eylül University, Department of Industrial Engineering

Publication:

- Mutlu, S., & Güner, B. (2021). A memetic algorithm for mixed-model two-sided disassembly line balancing problem. *Procedia CIRP*, 98, 67–72. https://doi.org/10.1016/j.procir.2021.01.007
- Mutlu, S., Ünal, H., & Güner, B. (2021). Paralel İstasyonları Dikkate Alan Montaj Hattı Dengeleme Problemi İçin Bir Doğrusal Tamsayılı Programlama Modeli Ve Bir İşletmede Uygulama. *Yönetim Araştırmaları / Mühendislik Uygulamaları Sempozyumu 2021 (YÖNAR/MU'2021) Bildiri Özetleri Kitabı*, 14–15.
- Mutlu, S., & Güner, B. (2021). Sürekli Kısıtsız Optimizasyon Problemlerinde Parçacık Sürü Optimizasyonu Algoritması Parametrelerinin Tam Faktöriyel Deney Tasarımı İle Belirlenmesi. *Yönetim Araştırmaları / Mühendislik Uygulamaları Sempozyumu 2021 (YÖNAR/MU'2021) Bildiri Özetleri Kitabı*, 123–124.
- Mutlu, S., Bilgen, B., & Ghallali, M. (2019). Comparison of particle swarm optimization and teaching-learning based optimization algorithms for dynamic berth allocation problem with port structure constraints. *Proceedings of the International Conference on Data Science, Machine Learning and Statistics (DMS-2019)*, 224-227.

Achievements

- 2019 Competition “Simulation Arena” 1st Place in Turkey
- 2018 High Honour of Dokuz Eylül University Faculty of Engineering
- 2018 1st Degree Graduation in Dokuz Eylül University Department of Industrial Engineering
- 2018 Competition “Industrial Engineering Undergraduate Project Competition” 1st Place in Eagean/Turkey
- 2018 Competition “Industrial Engineering Undergraduate Project Competition” 1st Place in Dokuz Eylül University
- 2017 Competition “Hugo BOSS Hack the Boss vol 2.0” 1st Place