Otto von Guericke University Magdeburg

Faculty of Computer Science

Master Thesis

Evolutionary Multi-Objective Optimization for Mixed-Model Assembly Line Balancing Problems

Author:

Iffat Jamil

October 5, 2022

Advisors:

Prof. Dr.-Ing. habil. Sanaz Mostaghim
Chair of Computational Intelligence
Otto von Guericke University Magdeburg

Jens Weise, M.Sc.
Computational Intelligence
Otto von Guericke University Magdeburg

Jamil, Iffat:
Evolutionary Multi-Objective Optimization for Mixed-Model Assembly Line Balancing Problems
Master Thesis, Otto von Guericke University Magdeburg, 2022.

Abstract

Assembly lines are a crucial aspect and a necessity in today's assembly and manufacturing processes. Due to the rising competition of today's industries and global market, businesses increase production versatility by lowering batch sizes and diversifying current products. Despite mixed or multi-model assembly lines being more prevalent in practice, the literature has far more research on single-model assembly lines. Still, there are a good number of research papers on mixed and multi-model assembly lines, but more work needs to be done in this direction. Moreover, the techniques to solve assembly line balancing problems are mostly limited to exact methods and heuristics as compared to the more recently focused meta-heuristic techniques, which show promising results. More research and experimentation is required in this area.

This thesis contributes to fill the mentioned gaps in the literature. A mixed-model assembly line with station restrictions is being considered, and evolutionary optimization techniques are used to balance the assembly line. In order to have high production rate while making the process time and cost-effective, multiple objectives have to be considered simultaneously. Here, four objectives have been taken into account: *cycle time*, *number of workstations*, *smooth task distribution* and *total idle time*. This thesis addresses two main challenges, modelling of the mixed-model assembly line into a single-model assembly line and optimization of the assembly process to make it cost and time efficient in the presence of workstation restrictions.

Option mix joint precedence graph is used to tackle the challenges of converting mixed-model assembly lines into single-model assembly lines. For optimization, the multi-objective evolutionary algorithms NSGA-II and NSGA-III are used. Several problems are taken into consideration for experimentation, including three famous benchmark problems (Bowman, Buxey and WeeMag) alongside two mixed-model assembly line problems. Experiments show that both algorithms can deliver promising results with the proposed methodology.

Acknowledgments

I would like to dedicate this work to my mother, Shafqat-un-Nisa, for her support and courage to stand against all the cultural norms for my education. I would also like to recognize the invaluable assistance and moral support of my supervisor, Jens Weise, and of Thomas Seidelmann.

Contents

List of Figures

List of Tables

List of Acronyms

ALBP	Assembly Line Balancing Problem
ALDP	Assembly Line Design Problems
DM	Decision Maker
DPGA	Distance-Based Pareto Genetic Algorithm
DR	Dominance Ratios
EA	Evolutionary Algorithm
EMO	Evolutionary Multi-Objective Optimization
GLABP	General Assembly Line Balancing Problems
HV	Hypervolume
IGD	Inverted Generational Distance
JIT	Just-In-Time
MCDM	Multi-criteria decision-making
MMALB	Mixed-Model Assembly Line Balancing
MOEA	Multi-Objective Evolutionary Algorithm
MOGA	Multiple Objective Genetic Algorithm
MOOP	Multi-Objective Optimization Problem
MuMALB	Multi-Model Assembly Line Balancing
NPGA	Niched Pareto Genetic Algorithm
NSGA	Non-dominated Sorting Genetic Algorithm
PAES	Pareto Archived Evolution Strategy
PALBP	Parallel Assembly Line Balancing Problem
SALBP	Simple Assembly Line Balancing Problems
SMALB	Single Model Assembly Line Balancing
SPEA	Strength Pareto Evolutionary Algorithm
VEGA	Vector Evaluation Genetic Algorithm

1. Introduction

"An assembly line is a manufacturing process in which parts are added to a product in a sequential manner using optimally planned logistics to create a finished product in the fastest possible way. It is a flow-oriented production system where the productive units performing the operations, referred to as stations, are aligned in a serial manner." Grzechca [2011a]

Grzechca elaborates the concept of assembly line manufacturing as a kind of production system that is particularly well suited to mass production. The manufacturing system operates at a very high production rate, and it is expected that there is room in the market to consume this throughput. The optimization and planning of these systems for efficient usage of the available resources is known as assembly line balancing and has several advantages, including enhanced productivity, low-cost manufacture of large quantities of standardized products, decreased work congestion, and less material handling.

There are various tasks that must be executed in order to successfully complete the production of a single product. On an assembly line, these tasks are carried out in the order specified by the workstations. Parts for the final product go from one workstation to the next on the assembly line. Due to the predetermined and directed flow, tasks must be allocated to successive workstations such that no part is returned to be reprocessed. The sequence in which the tasks must be accomplished is indicated by the precedence dependencies between the tasks. A task cannot be executed before other tasks that are located in front of it on the graph of precedence relationships. Assembly line balancing entails distributing tasks to workstations along the line in such a way that all precedence constraints are met and production proceeds in a directed flow. The time an assembly line takes to produce a single product is referred to as cycle time. This cycle time is a time constraint for each workstation that cannot be exceeded Grzechca [2011a].

Assembly lines are usually categorized into three types: single-model lines that are tailored to the production of a single specific product, multi-model assembly lines that yield two or more similar products in separate batches, and mixed-model assembly lines that produce two or more similar products simultaneously on a line with very small batch sizes, or even with one batch. Real-world problems are inherently complicated, and constraints further complicate the problems. When a problem has several and competing goals, solving it becomes more difficult. Assembly line balancing problems (ALBP), particularly those involving multiple/mixed-model assembly lines, are complicated in nature and can include more than one and conflicting performance targets. The number of deployed workstations, cycle duration, idle times, identical tasks between models, setup cost for shifting from one model's production to another, and so on are all factors that impact the assembly line balancing solutions Grzechca [2011a]. For multi/mixed-model assembly lines, the sequencing problem coexists with the balancing problem. Because the common tasks performed by sequential models vary throughout models, it becomes critical to find the optimal order. If this is the case, establishing the optimal order or sequencing of the models becomes another challenge without balancing the line for each model. The objective of this thesis is to convert a mixed-model assembly line to a single-model assembly line and to balance the assembly line with station restrictions. On the other hand, the challenge of optimally sequencing the models on the assembly line is not in the scope of this thesis. We want to address this issue as a continuation of our work in the future.

Due to the rising competition of today's industries and global market, businesses want to ramp up production versatility by lowering batch sizes and diversifying their current products. Because of this competitive nature, single-model manufacturing is less prevalent than multi/mixed-model manufacturing. According to our limited findings, despite mixed/multi-model assembly lines being more prevalent in practice, the literature has far more research on single-model assembly lines. Although there still exists a substantial amount of research regarding mixed and multi-model assembly lines, many research gaps exist, and more work should be done in this direction. This study makes a modest contribution to filling these gaps in the literature.

ABLPs have proven to be far more challenging in practise than in theory. It gets even more challenging to solve these problems when, to the best of our knowledge, there exist no benchmarks for mixed/multi-model assembly lines. Another complicating factor is the lack of published real-world problems with all the sales data of a company. Such data is nearly impossible to find because of confidentiality. This makes it harder to model the problem and prove the efficacy of the proposed solution. However, information gleaned from research publications is utilized to create testing case scenarios that are as near to reality as possible. These case problems are used to assess the proposed methodology. A case study (mixed-model smartphone manufacturing problem) has been fabricated as close to a real world scenario as possible, the proposed approach has also been tested on single-model assembly line benchmarks to check the efficiency of the optimization algorithms.

Gutjahr and Nemhauser [1964] demonstrated that the ALBP issue is NP-hard combinatorial optimization problem. This indicates that for problems of large magnitude, an optimal answer is not always guaranteed. As a result, heuristic approaches proven to be the most often used strategies for problem resolution Waldemar [2011]. Since the time of Henry Ford and the model-T, however, the demands of products and the preferences of manufacturing systems have changed significantly. Companies need to be able to make their products unique in order to meet the demands of a diverse range of consumers. For example, the German car company BMW alone has a list of optional features that could make 10^{32} different models Meyr [2009]. This drastic change in the global market toward mass manufacturing and customization makes ALBP even more difficult to solve and explains the rise in popularity of meta-heuristic techniques in recent years.

In this thesis, the questions under discussion are how to translate mixed-model assembly lines for use in a single-model framework and how to optimize the balancing of such assembly lines with workstation restrictions in a multi-objective way, for cost reduction and efficient use of processing times.This thesis is organized into seven sections. *Section* 2 is dedicated to provide the basic background information about the nature of the problem and explanation of the proposed methodology. *Section* 3 describes the concepts of Assembly Line Balancing and different approaches for resolving ALBPs from the literature. *Section* 4 discusses the mathematical modelling of the problem under consideration. *Section* 5 details the implementation of the proposed methodology. Since the ALBP is an NP-Hard combinatorial problem, two meta-heuristic approaches, NSGA-II and NSGA-III, are considered along with an option-based technique for obtaining joint precedence graphs Boysen et al. [2009]. *Section* 6 explains the experimental setup and analyses the obtained data in a detailed discussion. The methodology has proven to be effective for finding good solutions in moderate-to-large sized ALBPs. The same outcome, however, cannot be guaranteed for extremely large problems. Finally, the last section summarizes all the finding and future recommendations for the extension of this work.

2. Background

The ALBPs that are considered in this thesis are classified as multi-objective combinatorial optimization problems. In this section, the background on optimization methods, especially evolutionary multi-objective optimization, is highlighted for a thorough understanding of the problem's nature.

2.1 Optimization

The term "optimization" refers to the act of improving something. Real-life scenarios are riddled with optimization difficulties, many of which we face on a daily basis. Which route has the shortest distance to university? Which food is the best value for money while still providing the necessary nutrients? Optimization is the process of fine-tuning a process, function, or device's inputs in order to get the maximum or minimal output(s). The variables are the inputs, the objective function, cost function, or fitness value is the process or function, and the output(s) are fitness or cost Haupt and Haupt [2004]. This thesis addresses cost minimization. In functions where a maximum of cost is necessary, the output is minimized by appending a minus to the objective function. As a result, all the functions discussed here, are considered to be minimized. Some optimization problems involve only one objective function, and are hence use single-objective optimization. However, most real world problems need optimization of multiple parameters, and hence these problems are referred to as multi-objective optimization problems Amouzgar [2012].

Deb [2001] divided optimization techniques into two broad categories:

- Classical techniques
- Evolutionary strategies

Traditionally used approaches start with a single random solution, which is then modified in each iteration using a deterministic procedure in order to eventually obtain the optimal solution. There are two different categories for these methods: the first are direct methods, and use only objective functions and constraints to find the optimal solution. The second are gradient-based methods, which use derivatives of objective functions (or constraints) to determine a promising direction to steer the search Deb [2001].

Evolutionary strategies are later discussed in detail, after important concepts for it were introduced.

2.2 Multi-Criteria Decision Making

Multi-criteria decision-making (MCDM) problems are those that involve multiple conflicting criteria or objectives. Rather than a well-defined single optimal solution, such problems include several compromise alternatives, known as Pareto optimal solutions. In MCDM literature, it is common to assume that solving multistep optimization problems is done to aid a human decision maker (DM). The goals are to help the DMs to efficiently find a Pareto optimal solution which is most promising according to their preferences. In these cases the solution process generally requires the involvement of the DMs, and the final decision is governed by their preferences Miettinen and Hakanen [2017]. MCDM approaches are classified in different ways. According to Miettinen and Salminen [1999], There are four classifications based on the DM's role in the optimization procedure:

- **No Preference**

  If there is no DM and no preference information, non-preference approaches can be used to obtain a reasonable compromise without any further preference information.

- **A Priori Method**

  In an a priori method, the DM initially provides preference information, and the method then seeks a Pareto optimal solution that best meets the goals. This is a simple procedure, but challenging as the DM can be overly optimistic or pessimistic, and answers may not adhere to the expectations.

- **A Posteriori Method**

  The a posteriori method generates a representative set of Pareto optimal solutions, from which the DM can choose the most desired one. The DM gains an overview of the problem in this manner, but it may be difficult for the DM to analyse all alternatives thoroughly without being overwhelmed. This is especially problematic for problems with many objectives, as only two objectives can benefit from an impartial representation on a plane.

- **Interactive Approaches**

  A solution set is generated, and the process is repeated in an interactive way. The DM can define and modify preferred information between every iteration.

For this thesis, the a posteriori method has been implemented using an EMO strategy, which refers to evolutionary multi-objective optimization. The evolutionary method is a stochastic search and optimization algorithm that mimics natural evolution.

The interest in mimicking living beings has risen since the 1960s. Evolutionary methods can also outperform the classical techniques in many respects Gen and Cheng [1997]. EMO is an a posteriori method, meaning that the algorithm will generate a set of Pareto optimal solutions, and the DM will decide on what is the most favoured solution of all in the end Miettinen and Hakanen [2017]. Classical approaches solve these problems from different perspectives, mostly due to a lack of an efficient optimization tool for finding many optimal results. They frequently require repeated execution of an algorithm in order to uncover many Pareto optimal answers.

Usually, such approaches cannot actually guarantee the existence of a diverse Pareto optimal set in the end. This is contrasted by evolutionary algorithms (EAs), which use solution populations to identify diverse Pareto optimal sets efficiently and in a single simulation cycle Deb [2014]. This property has made EMO approaches popular for many applications and in research.

2.3 Multi-Objective Optimization

Multi-objective optimization deals with optimization problems where a multitude of objectives need to be satisfied at the same time. It can therefore be considered a subfield of multi-criteria decision-making Chang [2015].

As mentioned before, real world problems often contain multiple conflicting objectives. So, one extreme answer will generally not fulfil both objectives, and the ideal solution for one objective is not necessarily the perfect solution for another. Due to these trade-offs between objectives, a set of ideal solutions representing all objectives adequately is necessary Amouzgar [2012]. Multi-objective optimization has been used in a wide variety of research sectors, including engineering, where designers must choose between conflicting objectives such as vehicle performance, fuel consumption, and emissions. In these instances, a multi-objective optimization approach reflects the trade-offs between the objective functions well and should be used Chang [2015]. ALBP is another such problem and can benefit from multi-objective optimization. Often, the multi-objective nature of these problems is dealt with by scalarizing numerous objectives into a single objective, usually as a weighted sum. However, this is not ideal and the evolutionary approach solves the multi-objective optimization problem as-is Deb [2014].

The following is a general formulation of a multi-objective optimization problem (MOOP):

$$\min \setminus \max \ \ f_m(x), m = 1, 2, ..M; \tag{2.1}$$

$$subject\ to\ \ g_j(x) \leq 0, j = 1, 2, ..J; \tag{2.2}$$

$$h_k(x) = 0, k = 1, 2, ..K; \tag{2.3}$$

$$x_n^{(L)} \leq x_n \leq x_n^{(U)}, n = 1, 2, ..N. \tag{2.4}$$

Where,

f is the objective function to be maximized or minimized, and it's a function of x which denotes the vector of decision variables, M is the total number of objectives, $g_j \, and \, h_k$ represent constraints, $J \, and \, K$ show total number of constraints for $g_j \, and \, h_k$ respectively. Whereas, $x_n^{(L)} \, and \, x_n^{(U)}$ indicate the lower and upper bounds for each decision variable and N is the number of these variables.

According to Deb [2014] Multi-objective optimization possesses the following characteristics:

1. The optimal set's cardinality is typically greater than one.
2. There are two distinctive optimization goals rather than one.
3. There are two distinct search spaces.

In the majority of MOOPs, the Pareto optimal solutions share some similarities amongst the decision variables Deb and Goel [2003], but objective values differ. On the other hand, in multi-modal optimization problems (see section 2.4), there may be several solutions with identical objective values, but different decision variables. According to a study, which was conducted on a variety of engineering cases Deb and Goel [2003], The resulting trade-off solutions have revealed the following properties:

1. Certain decision variables take identical values across all Pareto optimal solutions. The decision variables possessing this feature indicates that the solution is optimal.
2. Other decision variables take on different values, resulting in a trade-off between the solutions' objective values.

Unlike in single-objective optimization, where the primary objective is just to identify the optimal solution, there are two distinct challenges for MOOPs:

1. Convergence to the Pareto optimal solutions.
2. Maintenance of diversity in a set of Pareto optimal solutions.

In some ways, these challenges are distinct from one another. An optimization algorithm must possess properties to respect both of these challenges adequately.

Another distinction between single- and multi-objective optimization is that the objective functions in multi-objective optimization form a multidimensional space next to the common decision variable space. The additional space is denoted by the letter Z, which stands for objective space. Every point x from the decision variable space maps to a specific point $f(x)$ in the objective space. Hence, each solution consisting of an array of decision variables has certain objective values. The fig. 2.1 depicts the two spaces and their mapping Deb [2014].

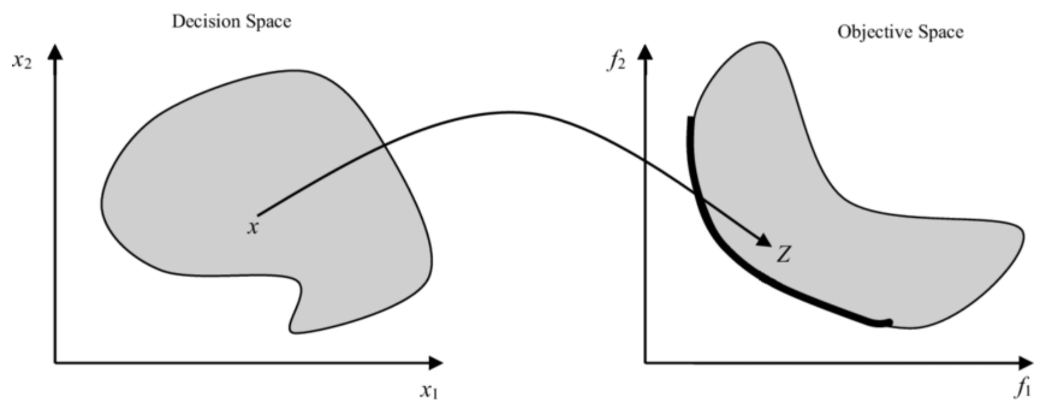


Figure 2.1: Decision Variable Space and its Corresponding Objective Space Nagaiah and Geiger [2019]

2.4 Multi-Modal Problems

A multi-modal function or problem is one that exhibits multiple "modes" or optimal states (e.g., valleys). Further, there also exist non-convex multi-modal functions. Rather than a single global optimal solution, this kind of problem has many global solutions or at least multiple local peaks of equal quality. Generally, in such multi-modal circumstances, there exist multiple different arrays of decision variables that nonetheless map to identical points in objective space Javadi and Mostaghim [2021]. Multi-modal multi-objective optimization problems are prominent in a wide variety of real-world problems, among which is ALBP as a non-convex multi-modal multi-objective problem.

2.5 Multi-Objective Optimization Terminologies

The following terminologies are defined to further assist the reader in comprehending the concept of multi-objective optimization.

Decision Space

The variable bounds impose a lower and upper limit on each decision variable, defining a space called decision variable space Amouzgar [2012].

Objective Space

When multiple objective functions are optimized, the values which the objective functions can assume define a space called objective space. This space has as many dimensions as there are objectives. Each solution from decision space has a corresponding point in the objective space Amouzgar [2012].

Linearity and Non-Linearity

A linear optimization problem is one in which both constraints and objectives are linear, in which case it's called a multi-objective linear problem. In comparison, when one or more of the objectives and/or constraints are non-linear, the problem will be called a multi-objective non-linear problem Deb [2001].

Convex and Non-Convex Problems

A problem is convex if all its objective functions and feasible regions are convex Deb [2001]. Convexity is critical in MOOPs, as solutions produced using preference-based approaches cannot cover non-convex trade-off solutions in the Pareto front for problems that have them. Additionally, many of the known algorithms are restricted to convex problems. While convexity can be defined in terms of the decision space, as well as the objective space, it is especially important in the objective space for the mentioned reasons Amouzgar [2012].

Dominated and Non-Dominated Solutions

Due to conflicting objectives, a solution optimized for one objective does not necessarily fare well in other objective(s). If the given problem is of minimization nature, then a solution A is better than a solution B if all its objective values are equal or better than those of B, and at least one is strictly better. In this case, B is a dominated solution and A is the dominating solution. To further explain the domination criterion, a general mathematical definition for both minimization and maximization in multi-objective problems can be made:

A feasible solution x_1 dominates another feasible solution x_2 ($x_1 \preceq x_2$), if and only if:

1. The solution x_1 is no worse than x_2 with respect to all objective values, hence $f_m(x_1) \leq f_m(x_2)\ \forall m = 1, 2, ..., M$
2. The solution x_1 is strictly better than x_2 in at least one objective value, hence $f_m(x_1) < f_m(x_2)$ for at least one $m \in \{1, 2, ..., M\}$

where, M is a number of objectives and $\preceq$ is the domination sign.

Therefore, solution x_1 dominates solution x_2, solution x_1 is non-dominated by solution x_2 and solution x_2 is dominated by solution x_1.

Pareto Optimal Set

Pareto optimal solutions are defined as those, which are non-dominated. Each Pareto optimal solution represents the best known answer for all objectives, and cannot be improved without compromising another objective. The Pareto optimal or non-dominated set is the collection of all possible solutions that are not dominated by any other solution. If the non-dominated set is contained entirely within the feasible search space, it is referred to as the globally Pareto optimal set Amouzgar [2012].

Pareto Optimal Front

Pareto front refers to the values of objective functions associated with each solution of a Pareto optimal set in objective space.

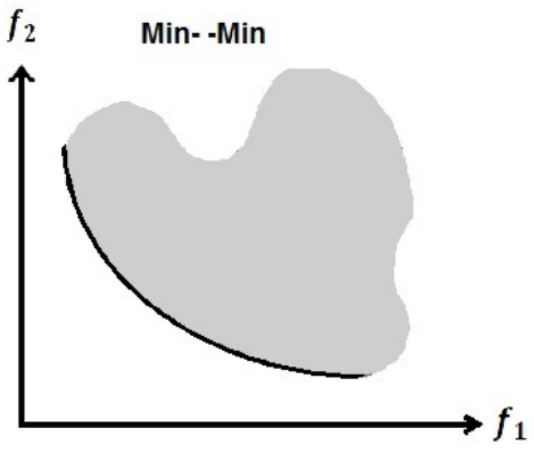


Figure 2.2: Example of a minimization Pareto front for two objectives Amouzgar [2012]

2.6 Multi-Objective Optimization Techniques

Although a lot of research work has been done in the area of multi-objective optimization, most of that research work circumvents the difficulties and challenges by converting multi-objective problems into single-objective problems Amouzgar [2012]. According to Deb [2001], this is in contrast to the preferable method of solving such problems as they are, and the approaches can thus be divided,

- *"Ideal multi-objective optimization, where a set of solutions in form of a trade-off curve is obtained, and the desired solution is selected according to some higher level information of the problem".*
- *"Preference based multi-objective optimization, which by using the higher level information of a preference vector transforms the multi-objective problem to a single-objective optimization. The optimal solution is obtained by solving the single-objective problem".*

The former approach has been used in this thesis, where the problem will be solved as it is without converting it into a single-objective problem, and no preference vector is required.

Using the ideal approach, without higher-level information, none of the Pareto optimal solutions is favoured over others. Thus, the primary goal of the ideal approach is converging to a set of solutions that is as close to the genuine Pareto optimal set as possible, which is the common goal of all optimization problems. However, the second aim specific to multi-objective problems is diversification in the derived Pareto optimal solutions Amouzgar [2012].

Classical approaches can also be used for solving multi-objective optimization problems. Multiple objectives can be converted to a parametric single-objective function using a parametric scalarizing strategy (including the weighted-sum approach, the epsilon-constraint approach, and others). Several Pareto optimal solutions can be obtained by simply adjusting the parameters (weight vector or epsilon-vector) and maximizing the scalarized function. In contrast, EMO attempts to find multiple Pareto optimal solutions in a single simulation by identifying numerous non-dominated solutions at once Deb [2001].

2.7 Evolutionary Algorithms

Evolutionary algorithms (EAs) are a broad class of algorithms that use a procedure inspired by the process of natural evolution to solve problems. This process involves first representing the solution space as a series of possible solutions called a population, then representing the fitness of each individual in the population by a fitness function. The next step is to select a set of individuals from the current population to be the parents for the next generation. The final step is to generate the offspring, which are the solutions in the next population, fon this section / chapter, the discussion will point for which the selected parents are used as a guide.

Typically, evolutionary algorithms are employed to provide good approximations to problems that could not be easily addressed using other techniques. This category encompasses a large number of optimization problems. Because it may be computationally extremely expensive to obtain an exact result, there are situations where near-optimal solutions suffice. Evolutionary approaches can be effective in solving NP-Hard Problems. Due to their inherent randomness, evolutionary algorithms cannot promise the ideal solution to any problem, however EAs typically find a decent solution if one exists. The key advantage of using EAs is therefore that they are well suited to large problems, where classical methods fail or take too long to compute. Among many examples is the field of protein folding, where using an evolutionary algorithm has been shown to complete the folding process in a timescale of minutes where using a classical method would have taken months. This is also evident in the field of human-assisted machine learning, where EAs are often used to find optimal training sets for machine learning algorithms where classical methods would require human knowledge and a lot of time to compute. Moreover, as there is little prior knowledge required for solving the problem, there is also a reduced vulnerability to shape (convex or non-convex problems) and continuity of the Pareto front. Ease of implementation, resilience, and parallelism are several further advantages of evolutionary algorithms as described in Abraham and Jain [2005] and Amouzgar [2012].

2.8 Multi-Objective Evolutionary Algorithms

Most of the literature indicates, that multi-objective evolutionary algorithms (MOEA) were invented by David Schaffer in the mid-1980s with his approach named VEGA (Vector Evaluation Genetic Algorithm), which was designed to solve optimization problems in machine learning. Apart from VEGA, there are various MOEAs identified in Deb [2001] and Coello et al. [2007] including; Niched Pareto Genetic Algorithm (NPGA, NPGA2), Non-dominated Sorting Genetic Algorithm (NSGA, NSGA-II, NSGA-III), Strength Pareto Evolutionary Algorithm (SPEA, SPEA2), Distance-Based Pareto Genetic Algorithm (DPGA), Pareto Archived Evolution Strategy (PAES), Multiple Objective Genetic Algorithm (MOGA) and many others. Given the existence of multiple MOEAs, the topic of which approach performs the best is frequently asked by scientists and researchers. To arrive at a conclusion, multiple test problems have been developed and a huge amount of analysis has been conducted. Deb's book discusses key research for the comparison of EAs Deb [2001]. In addition to that, Konak et al. [2006] presents a table outlining the pros and cons of the well-known EAs.

In the following, some of the most used and debated algorithms are listed, beginning with Non-dominated Sorting Genetic Algorithm (NSGA-II) Deb et al. [2002], and then the Strength Pareto Evolutionary Algorithm (SPEA, SPEA2) Zitzler and Thiele [1998] and Zitzler et al. [2001], the Pareto archived Evolutionary Strategy (PAES) Knowles and Corne [2000], and the Pareto Enveloped Based Selection. Extensive comparative research and numerical simulations on a variety of test problems demonstrate that NSGA-II and SPEA2 exhibit superior overall behaviour to other algorithms Amouzgar [2012].

NSGA-II and NSGA-III have been selected in this thesis to solve the multi-objective Assembly Line Balancing Problem and are explained in detail in Section 5.

2.9 Constraint Handling

One of the most important and difficult aspects of solving a multi-objective optimization problem is the handling of constraints. Since most of the real-world problems involve constraints, they must be handled carefully. Constraints can be soft or hard, and in the form of equality or inequality. A hard constraint is fixed and must not be violated. On the other hand, a soft constraint can be ignored in order to accept a solution Amouzgar [2012] Coello et al. [2007] and Knowles and Corne [2000]. In a constrained problem, the decision space is split into two distinct regions: feasible and infeasible. Whereby the feasible region is containing all solutions which meet all constraints, while the infeasible space contains all other solutions Amouzgar [2012].

Deb [2001] summarizes the five categories into which the majority of existing constraint handling methods fall:

- Preserving feasibility of solutions.
- Penalty functions.
- Biasing feasible over infeasible solutions.
- Methods based on decoders.
- Hybrid methods.

The penalty function is the most common and generally practised methodology for constraint handling. The penalty function approach transforms a constraint problem into an unconstrained one. In the thesis, the penalty function has been chosen to handle the constraints involved in the ALBP. In this case, the penalty of infeasible solutions is proportional to the degree to which they break the constraint. The penalty increases proportionately to the number of violated constraints.

The constraints involved for assembly line balancing problems have been mathematically explained and presented in detail in the problem formulation section.

3. Literature Review

In this section, the discussion will point to the thorough introduction of the assembly line balancing problem, along with the review of its current state in the literature.

3.1 Assembly Line

Henry Ford pioneered the factory assembly line in the early 1900s. It was created to be a highly productive and efficient method of manufacturing a certain product. The simplest assembly line consists of a series of workstations connected by a material handling equipment in a linear pattern. The assembly line's fundamental movement starts with a part being fed into the first workstation at a specified feed rate. Any spot on the assembly line at which an action is performed on a part is designated a workstation. These tasks or actions can be accomplished through the use of machines, human operators or robots. Once a part reaches a workstation, it is assigned a task and then transported to the next workstation Waldemar [2011].

According to Sury [1971], the time period required to perform a job or a task at each workstation is referred to as the process time of that task. Fonseca et al. [2005] and Baybars [1986] define the cycle time of an assembly line as a predetermined targeted output rate of production. This production rate is determined in such a way that the required supply of final product is manufactured within a specified time period. To keep the assembly line running at a constant rate, the total processing times at each workstation must not exceed the cycle time limitation of an assembly line Fonseca et al. [2005]. Erel and Sarin [1998] described idle time as the difference between the processing times on a given workstation and the cycle time. These terminologies are very crucial and are discussed in detail in the later section *problem formulation*.

3.2 Assembly Line Balancing Problem

An assembly line is composed of workstations that are configured along a conveyor belt or other mechanical material handling machinery. The parts to be assembled are launched down the line sequentially and are transported from workstation to workstation. Certain processes are repeated at each station in order to maintain the cycle time. According to Kao [1976], one of the primary concerns when developing an assembly line is the distribution of the tasks to be done, which is known as the assembly line balancing problem (ALBP). While this distribution is subjective to some extent, it must adhere to the underlying constraints imposed by the production sequence: To manufacture any product there are certain task sequences which should be followed. Line balancing is used in assembly processes and can also be used in production or manufacturing. A balanced line aims to reduce resource count while increasing resource use. Halgeson and Birnie [1961] were the first proposers of the ALBP, and Salveson [1955] published the ALBP in its mathematical formulation for the first time. However, According to Erel and Sarin [1998], throughout the first forty years of the assembly line's existence, lines were balanced solely via trial and error. Numerous approaches for solving the various variants of the ALBP have been proposed since then. Salveson [1955] presented the first mathematical solution to solve the ALBP as a linear program Waldemar [2011].

According to Baybars [1986], an assembly line is to be called perfectly balanced if the workload on each workstation is exactly equal. However, in practice, perfectly balancing a line is very difficult. The assembly line is then said to be balanced if all the workstations have total idle time as low as possible. Scholl and Becker [2006] defines ALBP in simple words as:

"The decision problem of optimally partitioning (balancing) the assembly work among the stations with respect to some objective is known as the assembly line balancing problem(ALBP)"

Generally, the assembly line designer's objective is to create a line that is more efficient, has less delay, is smoother to operate, has an optimal processing time, is cost-effective, has a higher overall work efficiency and produces just-in-time (JIT). *Just in time production* is the most often used philosophy in ALBP. It is critical that items are made precisely in response to market demand, which means that it is not preferable to produce goods ahead or behind the specified deadline/schedule, but precisely on time. This is crucial because if the manufacturer produces items after the deadline, production is delayed. If the manufacturer produces items ahead of schedule, it can result in storage problem, which is rather expensive and undesirable.

3.3 Classification of Assembly Line Balancing Problems

According to Uddin and Lastra [2011], ALBPs are classified mainly on the basis of their objective functions and structure. Due to the heterogeneity of objectives, many variants of ALBPs are proposed. The following is the classification of ALBP according to the objective functions:

Type 1

Given the cycle time, if the objective is to reduce the number of workstations, this kind of problem is known as type 1.

Type 2

When the number of workstation is given and the problem is concerned with reducing the cycle time, this kind of problem is known as type 2. Type 1 and 2 are the most popular ALBP variants in the literature.

Type 3, 4 and 5

These correlate to the optimization of workload smoothness, the maximizing of work relatedness and multiple objectives, respectively.

Type E

This type of problem is commonly acknowledged as the most general variant of ALBP. This variant is related to increasing line efficiency by reducing both cycle time and the number of stations simultaneously.

Type F

It is a feasibility problem in which the objective is to determine whether an assembly line balance is achievable given a certain configuration of workstations and cycle time.

3.3.1 Classification According to Scholl and Becker

According to the structure of an assembly line, Scholl and Scholl [1999] and Scholl and Becker [2006] have defined three main categories:

Single Model Assembly Line Balancing (SMALB)

The single assembly line balancing problem deals with the challenges of single model manufacturing (where only a single product is manufactured).

Multi-Model Assembly Line Balancing (MuMALB)

Multi-model assembly line balancing problems concern setups where more than one item or product is manufactured on the assembly line in different batches.

Mixed-Model Assembly Line Balancing (MMALB)

Mixed-model assembly line balancing problems concern setups where various versions or models of a generic product are manufactured on the assembly line in an intermixed situation, and preferably in a single batch.

3.3.2 Classification According to Baybars

Baybars [1986] classified ALBP into two broad categories, namely simple assembly line balancing problems (SALBP) and general assembly line balancing problems (GLABP).

SALBP

The simple assembly line balancing problem is the simplest type of balancing problem, in which the main objective is to effectively reduce the cycle time for a set number of workstations or vise-versa. In a very recent survey on ALBP, Boysen et al. [2021] explains SALBP as:

"To explain the importance of the SALBP to an operations' researcher who has not yet heard of this optimization problem, it can be SALBP is the pendant for the production domain what the traveling salesman problem is for the transportation area".

Most variants of assembly line balancing problems in the existing literature are based on a number of limiting assumptions. These assumptions are given by Baybars [1986] as follows:

1. The manufacturing system is considered to be tailored for a single, unique product.
2. Every task needs to be processed.
3. Every task has a fixed processing time.
4. Processing of the tasks is subjected to the precedence constraints and cannot be done in an arbitrary fashion.
5. There are no assignment restrictions for tasks apart from precedence restrictions.
6. Each of the workstations is equally equipped with respect to machines or workers, meaning, every worker or machine can process all sorts of task.
7. A task or a job can only be assigned to one and only workstation.
8. Any type of task can be processed or assigned to any workstation, meaning there are no layout, zoning or positional restrictions.
9. An assembly line is considered to be serial, when there are no parallel workstations or feeder sub-assembly lines.
10. The pacing of an assembly line is set according to the market demand with a predefined cycle time, meaning for a fixed amount of time number of workstations needs to be reduced, which is SALBP of type-1. On the other hand, if a number of workstations is given, and the optimization is trying to find the minimum cycle time, the problem is called SALBP and is of type-2.

Several versions of SALBP have been introduced based on the objective in consideration. A SALBP can be of type SALBP-1, SALBP-2, SALBP-E or SALBP-F.

- SALBP-1 indicates the objective is to minimize the number of workstations given a fixed cycle time.
- SALBP-2 indicates the objective is to minimize cycle time for a given number of workstations.
- SALBP-E deals with the objective of increasing line efficiency by minimizing both cycle time and number of workstations.

- SALBP-F is a feasibility problem and checks the feasibility of an assembly line for a given number of workstations and cycle time.

Even in the simplest form, ALBP is an NP-hard combinatorial problem. Though real-world problems are not as simple as SALBP, SALBP is the most studied problem in the literature and provides a solid base to understand this vast problem.

GALBP

Alongside SALBP, which simplifies work distribution to its most fundamental level, a research area called general assembly line balancing problem (GALBP) has emerged. The GALBP includes real-world complexities that are excluded in SALBP, therefore focusing on assembly line problems closer to real world scenarios. These complexities include, mixed-model assembly line balancing, parallel workstations, U-shaped assembly line, two-sided assembly lines, stochastic nature of task processing times etc.

Baybars [1986] has introduced the differentiation between SALBP and GALBP, where GALBP has been defined in a number of different ways by relaxing one or more of the presumptions from 1 to 9 (given above), except for assumption number 5. There is no explicit concern in GALBP for the variable cost of operating the workstation and fixed cost for a workstation, so, assumption number 5 is true for GALBP and also for SALBP. In the instances where the assumption number 5 is not true, those problems will be considered as assembly line design problems (ALDP). The main distinction between GALBP and ALDP is that, ALDP includes technology or labor Baybars [1986]. Methodologies and models related to ALDP are out of the scope of this thesis and will not be discussed here.

3.4 Assembly Line Layouts

According to Grzechca and Foulds [2015], in practice, there can be different layouts for an assembly line, including parallel lines, serial lines, two-sided lines, u-lines, etc. Overall, the assembly line's layout also contributes in dictating the type of ALBP. The basic scenario is simply task assignment to workstations on a serial or straight, single-model assembly line. Additionally, multi/mixed model lines necessitate model scheduling and batch sizing, U-shaped assembly lines demand assigning multiple jobs to employees or workstations, and parallel assembly lines entail deciding how many assembly lines to generate. In this study, the most popular *straight assembly line layout* has been considered. However, for the sake of completeness, several different popular layouts are discussed in the following.

Straight Assembly Line

The serial or straight assembly line was the very first assembly line layout described, for the layout, see fig. 3.1. A typical assembly line sets workstations and the tasks associated with them in a sequential fashion along a straight line. In recent times, many products are constructed not solely of basic components, but often of complicated pieces (assembled earlier). These complicated pieces can then be easily assembled with a limited number of steps. As a result, serial assembly line layout is still widely used in final product assembly.

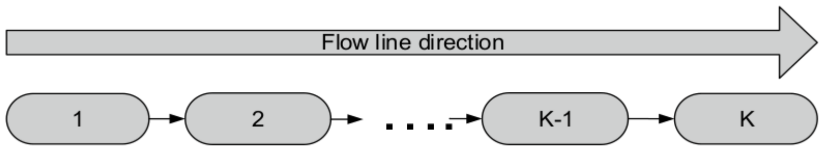


Figure 3.1: Layout of a Straight Assembly Line

U-shaped Assembly Line

The U-line layout was initially introduced in 1994. Workstations are organized in a U-line pattern around a U-shaped assembly line, see fig. 3.2. Operators or workers work on a task inside this U-line. According to Nakade and Nishiwaki [2008], U-shaped assembly lines enhance worker's communication skills and visibility, minimize worker requirements and increase performance, among other benefits. The layout gives operators the ability to complete many tasks located at various workstations along the assembly line efficiently, and thus the U-shaped assembly line has become a viable choice for assembly production systems. Additionally, since the U-shaped layout provides for more flexibility in assigning tasks to workstations, the number of workstations required for the U-shaped line configuration is never higher than the number of workstations required for the standard straight assembly line layout.

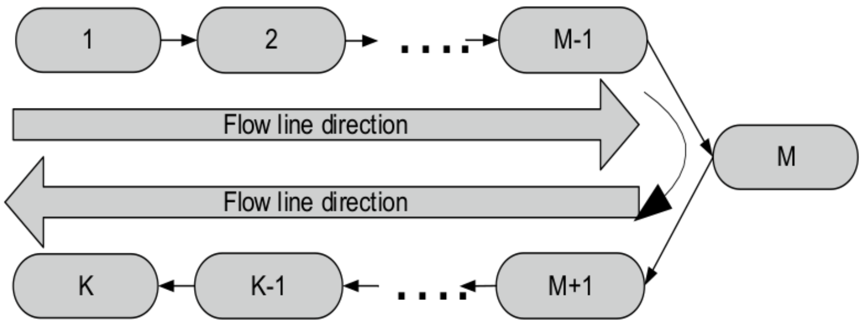


Figure 3.2: Layout of a U-Shaped Assembly Line

In the conventional assembly line balancing problem, the validity of a set of assignable tasks depends on the tasks whose predecessors already have been delegated to workstations. However, in the U-line balancing problem, the set of assignable tasks is decided by all tasks whose successors and predecessors have been already assigned Miltenburg [1998] and Miltenburg [2001]. One of the distinctive properties of U-shaped assembly lines in comparison with straight assembly lines is that the U-shaped assembly line has a single entering and exiting point Avikal et al. [2013].

Parallel Assembly Lines

Simultaneously, with the emergence of the U-shaped layout, the topic of balancing parallel assembly lines (fig. 3.3) was posed. The parallel assembly line balancing problem (PALBP) is composed of two related sub-problems: work assignment to parallel assembly lines and line balancing. To identify the range of assembly lines that could be operated with the least amount of overall workforce, Gökçen et al. [2006] and Ismail et al. [2011] have investigated different assembly line layouts for products with single and multiple models.

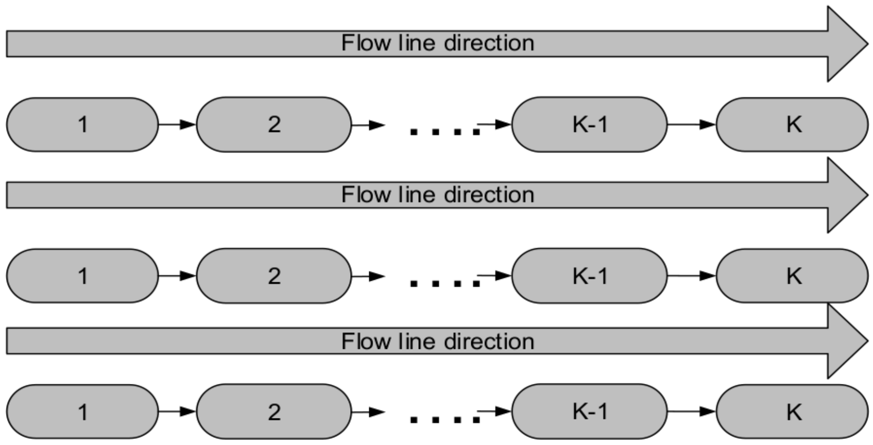


Figure 3.3: Layout of a Parallel Assembly Lines

Numerous manufacturing firms employ one or even more assembly lines. When market demand is sufficiently great, it is not unusual for the entire manufacturing line to be duplicated. This has the benefit of allowing for the achievement of production rates within a certain time period. Parallel assembly lines also provide an advantage during workstation breakdowns. If a workstation's equipment fails, other lines might continue to operate.

Parallel Workstations

Parallel stations are used to alleviate bottlenecks on serial lines, see fig. 3.4. Parallel stations execute the same tasks, hence increasing throughput Bukchin and Rubinovitz [2003].

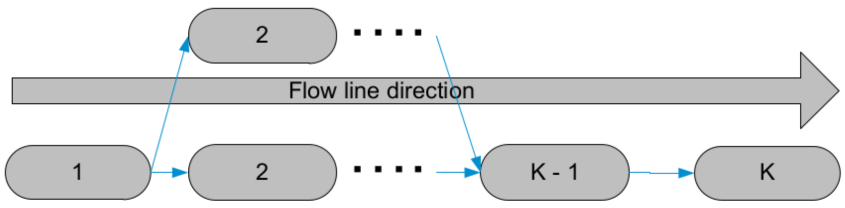


Figure 3.4: Layout of an Assembly Line with Parallel Workstation

Two-sided Assembly Line

Two-sided assembly lines are mostly utilized for heavier parts because operating on both sides of a heavy part is more convenient than rotating it. Rather than using a single workstation, pairs of immediately opposite workstations, such as 1 and 2 in fig. 3.5 are used. This technique greatly increases the line's flexibility, as the heavy part can be accessed from bothside. According to Lee et al. [2001], comparing with serial assembly lines, two-sided assembly lines have the potential to reduce the size of the assembly line and obviate the need for additional labour caused by handling the work pieces.

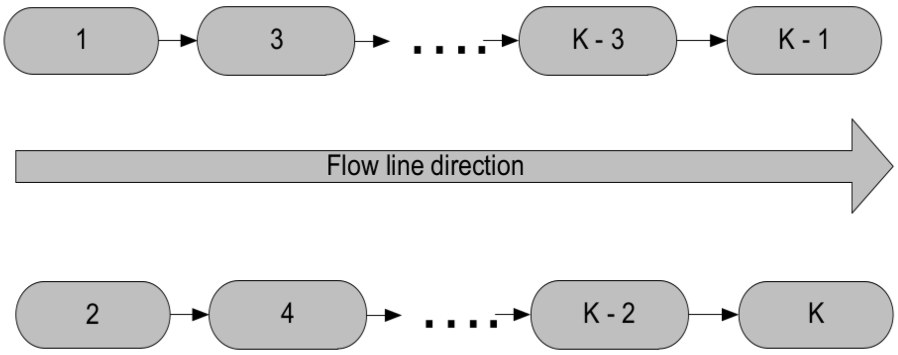


Figure 3.5: Layout of a Two-Sided Assembly Line

3.5 Precedence Graph

As described in Scholl and Becker [2006], assembly line manufacturing requires division of the total workload into a set of smaller, elementary operations, which are called tasks $V = 1, ..., n$. A task j requires a task time t_j to complete a job and a certain kind of machine and/or skills of workers. Precedence constraints between tasks must be adhered to due to technological and organizational constraints. These elements are captured and illustrated by a precedence graph. This type of graph consists of a node for each task, node weights that indicate task processing times, and arcs represents precedence constraints. The fig. 3.6, shows a precedence graph with total number of tasks, $n = 10$. The precedence constraints for, e.g., task 5 dictate that its processing requires task 3 (direct predecessor) and task 1 & 2 (indirect predecessor) to be completed.

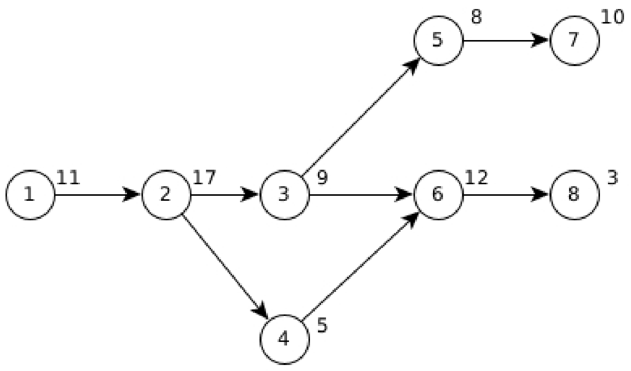


Figure 3.6: Bowman's Benchmark Precedence Graph

3.6 Joint Precedence Graph

A joint precedence graph is the combination of all the precedence graphs for every existing model in the production line. How to combine these precedence graphs is heavily influenced by customer demand Boysen et al. [2009]. Through the use of a joint precedence graph, the mixed-model ALBP is regularly transformed into a single-model problem to enable algorithmic solutions and reflect the usual need for common tasks to be assigned to the same workstation for several models.

The former approach for balancing mixed-model assembly lines relies on detailed forecasts of each model's demand and is known as the *model mix* method. This method is based on the estimated model mix, and from this a joint precedence graph is constructed which represents a single, average, virtual model. In this way, the mixed-model balancing problem is transformed into a single-model balancing problem and any single-model balancing method can be used on it Boysen et al. [2009].

However, individual model forecasting is frequently hampered by today's ever-increasing product variety. With mass customization, the variety of products increases along with the uniqueness of each model. Additionally, it becomes more difficult to predict which model will be released next. Therefore, only forecasts of the estimated occurrences for each product feature or option (e.g., the percentage of cars equipped with a navigation system) are possible. Boysen et al. [2009] demonstrated how a joint precedence graph can be generated in a way that accounts for this difference in the available information. They claim to present the first tractable approach to balance mixed-model assembly lines with such high product variance. The proposed approach is an option based joint precedence graph or option mix approach. The option mix joint precedence graph, unlike the model mix joint precedence graph, which depends on the forecast for each model, depends on the forecast for each option. Thus, reliable estimations can be deduced only for the frequency of option occurrences, regardless of which specific model they would appear on (option mix).

3.7 State-of-the-Art Methods

The following are well known and the most repeated approaches in the literature to solve ALBP.

3.7.1 Exact

Much of operations research has been devoted to coming up with the best ways to solve SALBP-1 specifically. This resulted in approximately two dozen techniques, which can be classified as branch and bound (B&B) procedures or dynamic programming (DP) approaches. These approaches include lower bounds, bin packing bounds, one-machine scheduling bounds, and destructive improvement bounds. While there are numerous exact solution procedures for SALBP-1, only a handful have been devised for SALBP-2. The majority of the research has been dedicated to examining search strategies focused on repeatedly solving SALBP-1. For SALBP-2, exact approaches include lower bound search and binary search. Scholl and Becker [2006]

3.7.2 Heuristic Approaches

Over the last few decades, a vast number of heuristic approaches to various variations of SALBP were proposed. Constructive procedures for producing one or more feasible solutions were created until the mid-1990s. Most of the proposed constructive procedures for SALBP-1 are based on priority principles, and others are restricted enumerative techniques. Constructive procedures include ranked positional weight technique, immediate-update-first, general-first-fit method. On the other hand, enumerative techniques include heuristic of Hoffman and truncated enumeration. A detailed comparison between state-of-the-art exact and heuristics for ALBP has given by Scholl and Becker [2006]

3.7.3 Meta-Heuristics Approaches

Since the tremendous shift in the global market and the assembly processes (especially in the automobile industry), scientists have focused their efforts recently on improving procedures using meta-strategies. Some famous and most used meta-heuristic approaches include ant-colony optimization, simulated annealing, tabu search and genetic algorithms. Since ALBP is an NP-hard problem, different meta-heurstics have been used to solve this problem. However, there is no apparent evidence to know which kind of algorithms are better than the others. A study, proposed by Nourmohammadi et al. [2019], discusses the selection procedure for meta-heuristic methods by conducting landscape (search space) analysis of SALBP and also, statistically validates the efficiency of the population based algorithms.

Mass customization has made the ALBP a lot more difficult to solve. It becomes highly inconvenient and impractical to deploy exact methods on problems having more than 60 tasks, because of high complexity and the computation is immensely expensive. However, heuristics have the drawback to get stuck in the local optima, which is not ideal for many problems.

In the last two decades, especially, last 12 years, the scientific work has noticeably shifted towards Meta-heuristics, which have shown promising results to overcome the above-mentioned challenges. However, more research and experimentation is required in the direction of general assembly line balancing problems using meta-heuristics.

4. Problem Formulation

As previously stated, there are several variants of ALBP. However, in this chapter, the discussion will point to the variant of mixed-model assembly line balancing problem (MMALBP) which is under consideration in the thesis, as well as its straightforward, yet descriptive mathematical modelling. Since a joint precedence graph approach has been implemented, the mathematical formulation is that of a single model assembly line balancing problem (SMALB) of type 5 with a station-restriction constraint.

4.1 Important Terminologies in ALBP

To facilitate comprehension, it is essential that the reader understands the fundamental ALBP terminologies.

- **Task:** An assembly line breaks the manufacturing or assembling process of a product down into several steps. The term "task" refers to each individual step on the assembly line.
- **Workstation:** According to Grzechca and Foulds [2015], a workstation is any location or point on an assembly line where a task is being executed.
- **Processing Time:** Grzechca and Foulds define processing time as the time which a task requires to be performed. In other words, the amount of time that a workstation needs to execute a task.
- **Cycle Time:** From Grzechca [2011b]: "Cycle time is a period when tasks can be assigned to the workstation." Therefore, cycle time is the interval between finished products leaving the assembly line ready for dispatching.
- **Takt Time vs Cycle Time:** In assembly line balancing, takt time is critical. Takt time can sometimes be misinterpreted as cycle time, even though they are not synonymous. Takt time is determined by market demand, whereas cycle time is determined by the production process. In other words, takt time is what the manufacturing or assembly line *has to do* in order to meet the market demand.

Whereas, cycle time is what the manufacturing or assembly line is *able to do* depending on the resources available and their usage.

$$Takt\,Time = \frac{availale\,time\,for\,production}{no.\,of\,products\,required} \tag{4.1}$$

$$Cyle\,Time = \frac{net\,production\,time}{no.\,of\,products\,produced} \tag{4.2}$$

Ideally, takt time should be equal to cycle time according to the philosophy of just in time production. Why is it not considered to be efficient if the manufacturing process produces products in less than the takt time? To answer this question, consider the following three scenarios:

1. $cycle\,time > takt\,time$, will result in a delay of production compared to market demand, which means loss of sales.
2. $cycle\,time < takt\,time$, will result in overproduction and necessitates storage of goods, which can become very expensive.
3. $cycle\,time = takt\,time$, will be JIT. This is a win-win situation because in this case, neither delay will occur nor storage is required.

4.2 Presumptions

ALBP is a very wide topic. While the primary objective stays the same (to increase assembly line efficiency), ALBP has shown itself in a variety of ways in the literature. It is vital to specify which assumptions are taken into account in the thesis.

1. Straight assembly line layout with no feeder lines or parallel stations.
2. The manufacturing system is considered to be tailored for mass-production of mixed-model products.
3. Every task needs to be processed.
4. Every task has a fixed processing time.
5. Processing of the tasks is subjected to the precedence constraints and cannot be done arbitrarily.
6. There are no assignment restrictions of tasks, except for the precedence restrictions and station restrictions. Not all stations are able to perform all the existing tasks.
7. A task cannot be split among two or more workstations.
8. Both, cycle time and number of workstations are variable and optimized for to thoroughly explore the best possible solutions for a given scenario without predefined preferences.

The above-given assumptions describe the ALBP variant under consideration. In summary, the considered assembly line layout is straight, and the structure dictates the mixed-model production system. By relaxing or augmenting these assumptions, an entirely new variant of the ALBP can be defined. It is very important to understand the given problem and to formulate it correctly before starting with the optimization steps.

4.3 The Decision Variables

The decision variables dictate the task allocation to workstations, including their distribution, for a given cycle time. In the simplest case, a binary modelling can be employed:

X_{ij} *represents the allocation of tasks* i *to the workstation* j

$X_{ij} = 1,$ *if the task* i *is allocated to the workstation* j*, otherwise* $X_{ij} = 0$

Where, $i = 1...m$ and $m = max\ number\ of\ tasks$

for example, a problem with 8 tasks and 6 workstations will result in 48 decision variables using the binary based mathematical modelling above. This way of mathematical modelling has been chosen to describe the problem because it's simple and easy to follow. However, it should be noted that a task sequence representation was used for the actual implementation of decision variables (described later), rather than the binary matrix itself. Given a cycle time, a binary matrix like presented can then be constructed.

4.4 Constraints

What makes ALBP even more challenging is the number of constraints that it has to deal with. The two major constraints this thesis deals with are precedence constraints and station restrictions. Before discussing these major constraints in depth, the general constraints are presented as the following:

- Each task should go to only one station, meaning a single task cannot be allocated to two different workstations,

$$\sum_{j=1}^{n} X_{ij} = 1 \quad \forall\, i \tag{4.3}$$

- The sum of all the processing times for all the tasks allocated to a workstation, should not exceed the cycle time,

$$\sum_{i=1}^{m} t_i \cdot X_{ij} \leq C \quad \forall\, j \tag{4.4}$$

Where, t_i *is the processing time for task* i*, and* C *is cycle time*

4.5 Precedence Constraints

The layout of a straight assembly line dictates the movement of the product in forward direction only, this means that no sub-part of the final product on the assembly line can go to a workstation with lower number than the workstation at which it is currently. Hence, the sub-part of the product cannot go in reverse direction on a straight assembly line and the following operations can only be performed by the workstations having higher number than its latest workstation.

As an example, if there is a precedence relation between task 1 and task 2, the workstation to which task 2 is assigned should be the same as the one for task 1, or a workstation further down the line. Following are the mathematical expressions for the precedence constraints between task 1 and task 2 given that the eq. (4.3) holds true, and six workstations are assumed:

$$X_{1,1} \leq X_{2,1} + X_{2,2} + X_{2,3} + X_{2,4} + X_{2,5} + X_{2,6} \tag{4.5}$$
$$X_{1,2} \leq X_{2,2} + X_{2,3} + X_{2,4} + X_{2,5} + X_{2,6} \tag{4.6}$$
$$X_{1,3} \leq X_{2,3} + X_{2,4} + X_{2,5} + X_{2,6} \tag{4.7}$$
$$X_{1,4} \leq X_{2,4} + X_{2,5} + X_{2,6} \tag{4.8}$$
$$X_{1,5} \leq X_{2,5} + X_{2,6} \tag{4.9}$$
$$X_{1,6} \leq X_{2,6} \tag{4.10}$$

X_{ij} is the task allocation variable, where i is the task number and j is the number of the workstation. All of the above equations must hold true for a given task allocation on six workstations with the given precedence relation between task 1 and task 2. Otherwise there is a constraint violation.

4.6 Station Restrictions

Station restrictions are a constraint that is often excluded in the literature. For the sake of simplicity, it is often assumed that there exist no station restrictions while allocating the tasks. Therefore, it is assumed that each workstation is capable of carrying out all sorts of operations. Unfortunately, this is not the case with most real-world problems. Not every workstation, whether it is a machine, robot or a human being, is capable of performing all kinds of operations. Hence, station restrictions have been included in this study, making the approach more practical.

Tasks and workstations are divided into groups based on their compatibility, e.g. if there are 6 workstations and 8 tasks divided into two groups, A and B, then the task that has been categorized as group A cannot be performed by a workstation that has been categorized as group B, as shown by the table below:

Table 4.1: Grouping Tasks and workstations according to their compatibility

Group	Workstation Number	Task Number
A	1,2,3	1,3,4,5
B	4,5,6	2,6,7,8

Considering the example shown by Table 4.1, task 1 and task 2 belong to different groups, hence task 1 cannot be allocated to workstations 4, 5 and 6, see eq. (4.11) and task 2 cannot be allocated to workstations 1, 2 and 3, see eq. (4.12).

$$X_{1,4} + X_{1,5} + X_{1,6} = 0 \tag{4.11}$$

$$X_{2,1} + X_{2,2} + X_{2,3} = 0 \tag{4.12}$$

$$X_{3,4} + X_{3,5} + X_{3,6} = 0 \tag{4.13}$$

$$X_{4,4} + X_{4,5} + X_{4,6} = 0 \tag{4.14}$$

$$X_{5,4} + X_{5,5} + X_{5,6} = 0 \tag{4.15}$$

$$X_{6,1} + X_{6,2} + X_{6,3} = 0 \tag{4.16}$$

$$X_{7,1} + X_{7,2} + X_{7,3} = 0 \tag{4.17}$$

$$X_{8,1} + X_{8,2} + X_{8,3} = 0 \tag{4.18}$$

The above equations describe all the station restrictions for the selected example. To tackle these constraints, a simple but effective heuristic has been introduced in this thesis, which is discussed in details under Section 5.

4.7 Objective Functions

The goal of assembly line balancing is mainly to improve the efficiency of the production process. Although there are many variants of ALBP, the main goal remains the same. An increase of productivity and the maximum utilization of the deployed resources can be achieved by varying the number of workstations (and/or adjusting the cycle time limit), while the smoothness of the line can be increased by making sure the tasks are well distributed amongst the workstations. There can be more objectives than mentioned here, but these are the most fundamental and important ones. In this thesis, four objectives are chosen, including minimization of cycle time (eq. (4.20)), number of workstations (eq. (4.19)), smoothness index (eq. (4.22)) and total Idle time (eq. (4.24)).

Cycle Time and Number of Workstations: One of the main objectives of ALBPs is to reduce the required resources, which are mainly cost and time. The second main objective in ALBPs is to make sure that the deployed resources are being utilized to 100% of their capacity, ideally. However, in reality, as close to 100% utilization as possible. These objectives can be achieved by means of minimization of the number of stations or by lowering the cycle time.

$$\min N = \sum_{j=1}^{n} S_j \tag{4.19}$$

where N is the total number of workstations, $S_j = 1$ if workstation j was allocated (had any tasks given), and n is the maximum number of workstations.

$$\min C \tag{4.20}$$

where C is the cycle time

Another very important and often used performance indicator is the line efficiency (eq. (4.21)). It computes the ratio of minimum possible production time for one product (the sum of task processing times) to total available processing time in the assembly line. The result is a percentage of resource utilization, or the efficiency of the assembly line, respectively. Hence, line efficiency gives important information about the utilization of the whole assembly line. However, since it is a function of cycle time and total number of workstations, it has not been included as an individual objective function.

$$\max Line\ Efficiency = \frac{\sum_{i=1}^{m} t_i}{N \cdot C} \cdot 100 \tag{4.21}$$

where t_i is the processing time of task i

Smoothness Index: Smoothness index indicates how well the tasks are distributed among the workstations. It simultaneously penalizes the total idle time of the system, as well as variances in the idle times among the workstations. It can be mathematically represented as follows:

$$\min Smoothness\ Index = \sqrt{\sum_{j=1}^{n}(C - \sum_{i=1}^{m} t_i \cdot X_{ij})^2} \tag{4.22}$$

When there are two solutions with an equal number of workstations, one might be "better balanced" than the other. For example, consider the fig. 4.1 and fig. 4.2 of two different assembly lines taken from Waldemar [2011], both assembly lines have the same cycle time, total idle time and number of workstations. However, the assembly line fig. 4.2 is considered to be superior or better balanced compared to assembly line fig. 4.1 in terms of task distribution. In other words, the total idle time of the assembly line fig. 4.2 is more evenly distributed over all the workstations as compared to the assembly line fig. 4.1.

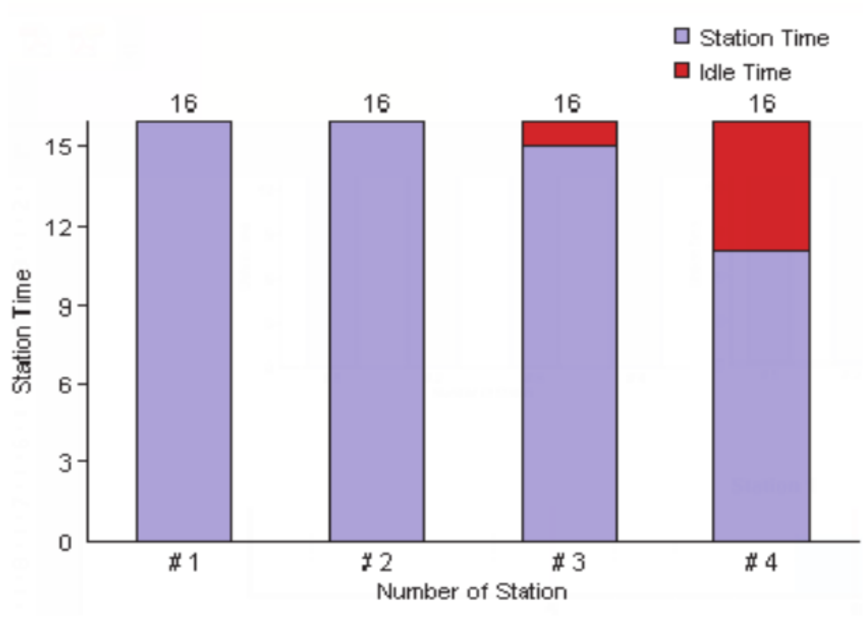


Figure 4.1: Final Solution of Assembly Line Balancing Problem, Using RPW Heuristic

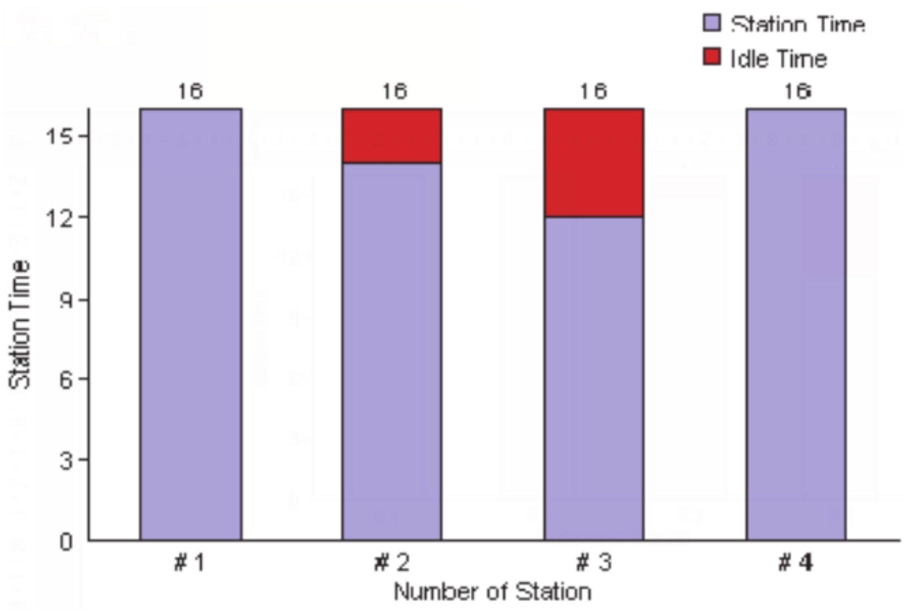


Figure 4.2: Final Solution of Assembly Line Balancing Problem, Using IUFF-WET Heuristic

Importantly, in most of the early literature there is a prominent assumption that minimizing smoothness index can be used interchangeably with minimizing number of workstations. That is because they correlate significantly. However, Fathi et al. [2018] proves that minimizing the smoothness index is not necessarily equivalent to minimizing the number of workstations and hence these two objectives are not substitutes for each other and must be considered as two different objectives. The importance of the smoothness index has been discussed much in the literature (Nourmohammadi and Zandieh [2011], Kellegöz [2016], Mozdgir et al. [2013]), although the nuances of this discussion are outside the scope of this thesis.

Variance in Task Distribution: Although the smoothness index is an important, often used objective function, it is a relative term. Meaning that problems with different sizes cannot be compared fairly based on their smoothness indexes. This is because the smoothness index correlates strongly with the total idle time, which naturally tends to increase for scenarios with more workstations. To counteract these issues, a new objective function (Variance in Task Distribution) has been formulated in this thesis, which is not a relative term and purely indicates how evenly the tasks are distributed among the workstation without correlating to the total idle time. The smaller the value of variance in task distribution, the better the distribution is. This allows adding the total idle time as the fourth objective. The mathematical representation of this objective function is as follows:

$$\min Var\,Task\,Distribution = Var\{\frac{\sum_{i=1}^{m} t_i \cdot X_{ij}}{C}\} \;\; \forall\, j \tag{4.23}$$

Total Idle Time: The last objective function is the total idle time, which shows the collective idle time of all the workstations. Minimizing idle time maximizes the utilization of the deployed resources, and it can be represented mathematically as:

$$\min Total\,Idle\,Time = \sum_{j=1}^{n} (C - \sum t_i \cdot X_{ij}) \tag{4.24}$$

5. Proposed Methodology

In this section, the discussion centres on the proposed methodology to face the challenges of multi-objective mixed-model assembly line balancing problems. The section is divided into three main parts: First, details of converting mixed-model assembly lines into single-model assembly lines, which addresses the implementation of the joint precedence graph using the option mix method. Second, the optimization process to balance the assembly line in an optimal way is discussed, along with the evolutionary algorithms that were implemented. Third, the station restrictions and a simple, yet effective heuristic to tackle the challenge is discussed.

5.1 Option-based Joint Precedence Graph

To get an algorithmic solution and to deal with the typical requirement of assigning tasks shared by many models to the very same station, the mixed-model ALBP is often converted into a single-model ALBP by using a joint precedence graph Thomopoulos [1970]. As a first step, the precedence graphs of different models have been combined into one joint precedence graph using the option-based technique proposed by Boysen et al. [2009].

According to Breginski et al. [2013], the most important and crucial condition for making a joint graph is, that there should not be any conflicts of precedence between the models. Although Boysen et al. [2009] explains how to remedy the problem of conflicting precedence relations between models, it leads to cycles in the joint precedence graph. For this thesis, it has been considered that there are no precedence conflicts amongst the models.

For the sake of completeness and better comprehension, before discussing the implementation of the option-based technique, the well-known model based technique is discussed briefly: The model-mix method is based on the usage of each model's probability (these probabilities are driven from the sales data of the manufacturing firm, which shows how many times in the past a particular model has been sold or manufactured) instead of considering the individual probabilities for each available option. Thus, the mixed-model method results in an average model for all the existing models, see the fig. 5.1 below.

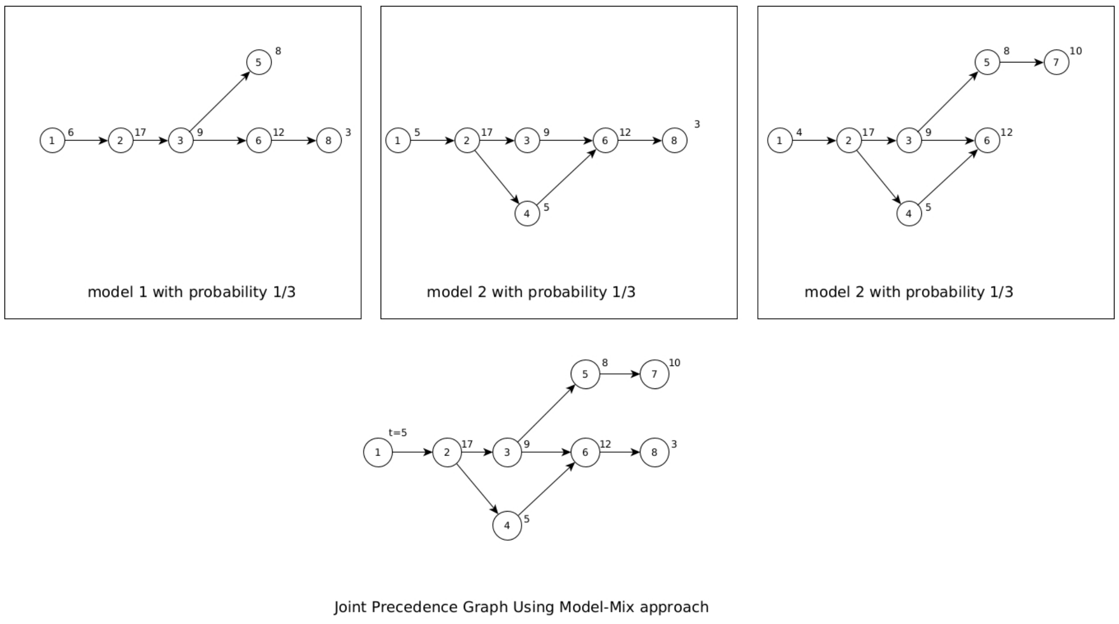

model 1 with probability 1/3
model 2 with probability 1/3
model 2 with probability 1/3
t=5
Joint Precedence Graph Using Model-Mix approach

Figure 5.1: The joint precedence graph is the average of the three models with respect to their Probabilities, Since all the Tasks have the same processing time, except task 1, only the joint processing time for task 1 has been calculated using eq. (5.2)

However, with the increase in customization, where there can be thousands of options to choose from, resulting in millions of models, the mixed-model method fails, because it is not possible to predict any specific model's demand reliably. Hence, the option-mix method has been introduced, which predicts the selected options instead of predicting the whole model. The mathematical definition of a joint precedence graph $G = (V, E, \bar{t})$ based on the model-based approach is given by Macaskill [1972] and Van Zante-de Fokkert and de Kok [1997] as:

$$V = \bigcup_{m \in M} V_m \tag{5.1}$$

$$\bar{t}_i = \sum_{m \in M} P_m \cdot t_{im} \quad \forall\, i \in V \tag{5.2}$$

$$E = \bigcup_{m \in M} E_m \,/\{RedundantArcs\} \tag{5.3}$$

Using the option mix approach, the joint precedence graph will be constructed according to the general procedure and structure of the traditional approach, as shown above. In order to generate the joint node set V in eq. (5.1), tasks that are common to different models, despite requiring different processing times, are given a consistent node number across all models. Thereby, the assignment of the same task to different workstations is prevented. Processing time, referred to as node weight, is zero for tasks that are not required by a model. In this way, the average processing time $t_{[}i]$ can simply be obtained from the model-specific task time $t_{[}im]$, but weighted according to the demand probability P_m of the model from eq. (5.2). The joint precedence constraints are determined by joining model-specific arc sets in eq. (5.3). This process can lead to arc-redundancy. Since the arc is redundant, it can easily be removed without any information loss Boysen et al. [2009]. However, to make things a little simpler, for this study, only problems that will not result in redundancy of the arcs have been selected.

An acyclic graph is created by transferring all tasks and their respective precedence constraints to the option mix joint precedence graph. In contrast to the model mix approach, the main difference is in how the joint task times are calculated from the estimated probabilities that products will contain specific options, rather than from the estimation of which exact models are going to be produced how many times. For each task i belonging to V, the joint processing times $t_{[}i]$ are computed individually. The following four steps are given by Boysen et al. [2009] based on the set of options O to obtain the values:

1: List the options in a set O_i which requires the task i to be carried out, set O_i will be the subset of the set O.

2: Determine all combinations in the form of set $VO_i \subset P(O_i)$, which can appear in the same model feasibly, where $P(O_i)$ is the power set of O_i.

"Each feasible combination defines a virtual option. In order to get a set of mutually exclusive options, the original set O_i is temporarily replaced by the set VO_i of all virtual options" Boysen et al. [2009]

3: Determine the respective task time $t_i(v)$ and the probability $p(v)$ for each virtual option v belonging to VO_i.

Where $p(v)$ represents the probability of the case that all options o belonging to v are chosen and all the remaining options o belong to O_i.
If a task i is not required for a virtual option v belonging to VO_i, its task time $t_i(v)$ will be set to zero.

4: The joint task times can be computed by the following eq. (5.4):

$$\bar{t}_i = \sum_{v \in VO_i} P(v) \cdot t_i(v) \quad \forall\, i \tag{5.4}$$

Boysen et al. [2009] then split the set V into three disjoint subsets $V_A \cup V_B \cup V_C = V$ to which a task i is assigned with regard to the number and interaction of options in the set O_i as follows:

Common tasks: The tasks that belong to V_A have to be performed on any model independent of the required options, i.e., $O_i = O$, including the event that no option is chosen at all. Moreover, these tasks have similar task times t_i for all option combinations. Then, the joint task times are, $\bar{t}_i = t_i$ thus eliminating Steps 2 to 4 for all tasks i belonging to V_A.

Single-option tasks: V_B consists of all the tasks which can only be assigned to a single option occurrence. This encompasses all the tasks with a single binary option ($|O_i| = 1$). An example of a binary option, whether there will be a navigation system in the car or not, it's either a yes or a no. A task i can also be interpreted as a single-option task even with more than one option $O[i]$, provided that these options o must be mutually exclusive, meaning these options can not be present simultaneously on the same model. In this case of a single-option task i, the joint processing time $/bar[t]_[i]$ reflects the weighted average of the processing time needed for all the option-specific tasks $t[io]$ in proportion to the probability of occurrence $P[o]$ of the respective option. Thus, Steps given above from 2 to 4 will be replaced by directly computing the following eq. (5.5):

$$\bar{t}_i = \sum_{o \epsilon O_{oi}} P_o \cdot t_{io} \quad \forall\, i \,\in\, V^B \tag{5.5}$$

Multiple-option tasks: The subset V_C consists of all tasks i, which have a set of options ($|O_i| > 1$), where at least two of these options are able to occur independently. Further, the task processing times may differ between the various possible combinations of options. In these cases, Steps 2 to 4 must be implemented for all the tasks i belonging to V_C. For an in-depth comprehension, the reader can refer to the paper Boysen et al. [2009] for a detailed example.

Option-mix approach, despite being a very good remedy for the highly productive mixed-model facilities, also has some limitations because of the stochastic nature of this method. These limitations are explained by Boysen et al. [2009] as follows:

The balancing of the assembly line, when using this approach, is relying on estimated processing times. To balance an assembly line in this way cannot ensure its operation without capacity violations. The reason is that both task times and the resulting workstation processing times are of probabilistic nature, which still holds even in cases where the processing times themselves are deterministic. Secondly, the prediction of the sales, which is based on existing sales data, can be problematic due to forecasting errors. Since the joint processing graph is heavily dependent on the sales data, these forecasting errors can lead to an inefficient assembly line.

To tackle the probabilistic nature of these problems, it has to be noted that the combined probabilities of a joint precedence graph necessarily result in an optimization for a line balance that is also probabilistic in nature. Thus, this technique does not prevent cycle time violations, although a constraint can be added, which can only guarantee to a certain degree of likelihood that workstation processing times will not exceed the cycle time by a certain percentage. It can not be guaranteed 100%.

For a high diversity in products, the option mix approach is nevertheless a good alternative for the traditional model mix approach. Therefore, it can be considered as a trade-off between the reduction of planning efforts and precision in the solutions, and is befitting for practical scenarios.

5.2 Multi-Objective Evolutionary Algorithms

The second part of this section consists of optimization, and the two multi-objective evolutionary algorithms chosen for this study are NSGA-II and NSGA-III. The fundamentals of evolutionary algorithms have been discussed in the background section.

5.2.1 NSGA-II

The elitist non-dominated sorting genetic algorithm II (NSGA-II), given by Deb et al. [2002], is a widely used EMO procedure that seeks multiple Pareto optimal solutions to a multi-objective optimization problem. It possesses the following three characteristics:

1. It is based on an elitist premise.
2. It incorporates an explicit mechanism for preserving diversity.
3. It places an emphasis on non-dominated solutions.

Amouzgar [2012] explains the NSGA-II as follows: At any generation t, the offspring population Q_t is initially constructed using the parent population P_t and standard genetic operators (crossover and mutation). Afterwards, the two populations are joined to generate a new population (R_t) with a size of $2N$ (where, N is the size of the initial population). The individuals of the population R_t are then categorized into various non-dominated fronts.

Following that, a new population of size N is gradually populated by choosing solution out of Rt. The re-population process begins with the first non-dominant front and progresses towardsthe individuals in the second non-dominant front, and so on. Due to the fact that $R_t's$ total population size is $2N$, not all fronts can be accommodated within this population of size N For this reason, all extra fronts are eventually eliminated. When the final permissible front is evaluated, it is possible that the front contains more solutions or individuals than the available slots in the new population, see fig. 5.2. Rather than randomly selecting some individuals from this last permissible front, the individuals that contribute to the diversity are selected.

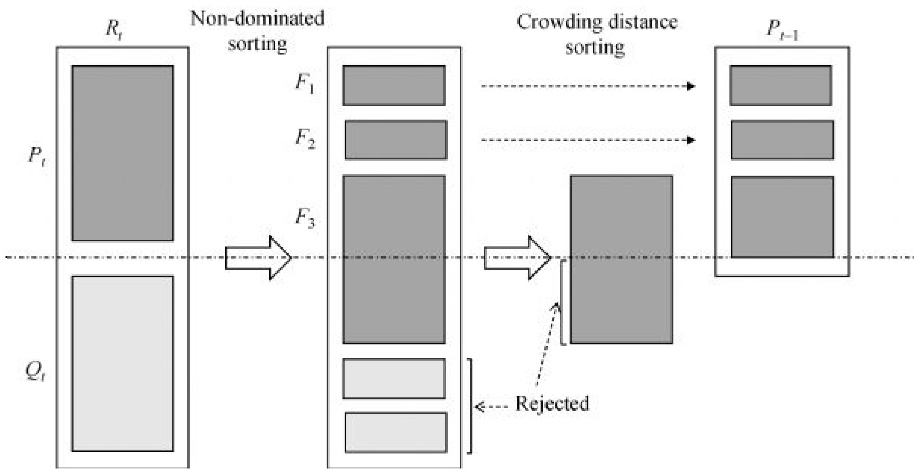


Figure 5.2: Shows the whole Procedure of NSGA-II with Crowding Distance Method Ren et al. [2018]

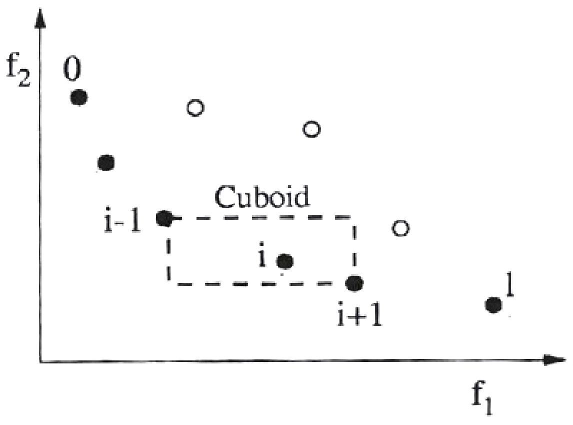


Figure 5.3: Shows the Crowding Distance Method to Preserve Diversity among the Solutions Deb et al. [2002]

For the individuals of the last front, which cannot all be included fully, a crowded-sorting method is used to prioritize the solutions. For every solution in the objective space a crowding distance is calculated, and then the solutions with the greatest crowding distances are included first, until no more solutions can be added. The crowding distance d_i, for a point i, is obtained by measuring the size of the objective space around i, which is not covered by any other solution. For faster calculation, the nearest neighbours of i create a cuboid, whose perimeter can be used as an approximation, which is the method that is used here, see fig. 5.3.

5.2.2 NSGA-III

The non-dominated sorting genetic algorithm III (NSGA-III) is a reference-point-based evolutionary algorithm for many-objective problems, which is based on the framework of NSGA-II and proposed by Deb and Jain [2013]. NSGA-III focuses on the individuals in the population that are non-dominated and closer to the set of given reference points. Details for the implementation of this algorithm can be found in Blank et al. [2019].

The non-dominating sorting as shown in the fig. 5.4 is done in similar fashion as in NSGA-II.

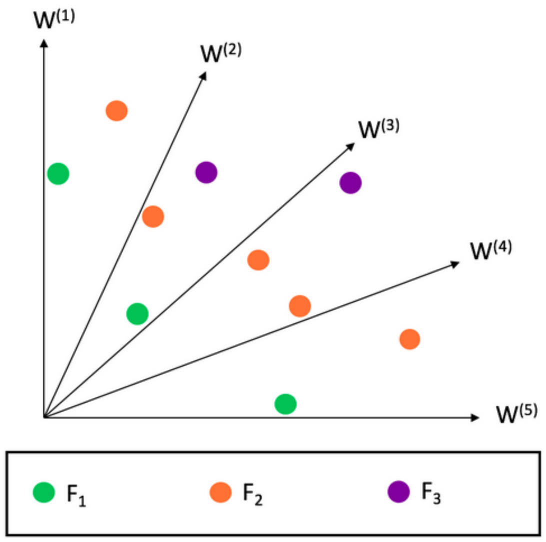


Figure 5.4: Shows the non-dominating sorting process, Blank and Deb [2020]

The next step is to select the solutions from these different fronts, while some solutions need to be discarded. NSGA-III does that by selecting the underrepresented reference directions first. In case there is no solution assigned to the reference direction, The solution with the smallest perpendicular distance in the normalized objective space will be selected. If another solution for the same reference direction has been added, then the assignment of these solutions will be decided randomly Blank and Deb [2020].

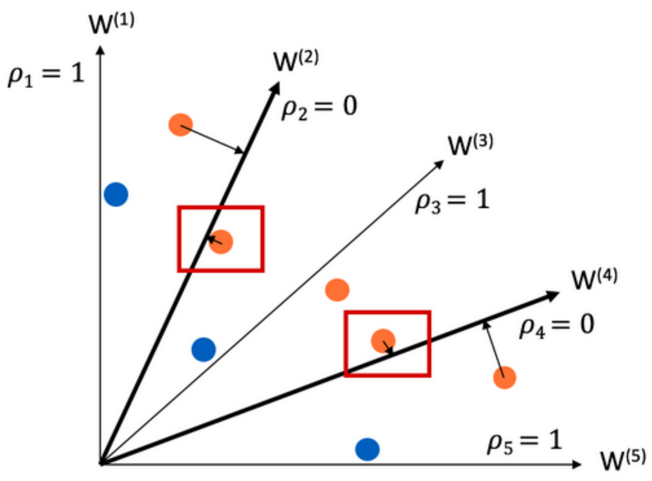


Figure 5.5: Selecting solutions closer to reference directions, Blank and Deb [2020]

According to Deb and Jain [2013], NSGA-III has been proven to successfully converge and found diverse sets of solutions for many-objective problems. Thus, this algorithm has been selected for this study along with NSGA-II. The detailed comparison between NSGA-II and NSGA-III is presented in the results section.

5.2.3 Components of the Algorithms

Following is the description of the crucial components used in NSGA-II and NSGA-III, and the explanation on how they are used in this study.

Representation of the Decision Variables

The decision variables are given by a sequence of integers. Each task is denoted by a number that is added to a vector (i.e., chromosome) with a length equal to the total number of tasks. The tasks are arranged on the chromosome in the sequence in which they are processed and ideally without violating the precedence constraints (in which case a penalty would be applied, as discussed further down). The tasks are then assigned to workstations in such a way that the total time spent on tasks at each workstation does not violate the cycle time constraint Sabuncuoglu et al. [2000].

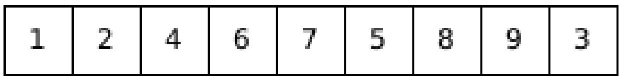


Figure 5.6: Shows a chromosome or a potential solution for a problem with 9 tasks

Initial Population

The initial population is generated randomly, and may or may not include feasible solutions. Importantly, the size of the population should be adjusted according to the size of the chromosome, otherwise it may result in very high computational time or low overall diversity. The population for problems with a large chromosome size should therefore be larger than the population for problems with a smaller chromosome size. To maintain good diversity, the choice of an appropriate size of population is crucial.

Crossover:

The crossover, also known as recombination, has been defined as follows: Two parents will be selected for the crossover and will produce a single offspring. The first parent will be cut at two points. These two points are positioned in such a way that they divide the first parent into three sections, and the middle section is one-third of the size of the parent. The offspring takes the same genes encapsulated by the points at the same location as the first parent, and the genes on the outside of the cut-points are taken from the second parent in the same order they appear. This procedure is demonstrated in fig. 5.7. Moreover, whether to perform a crossover or not depends on a crossover probability, and this probability has been set to 0.9.

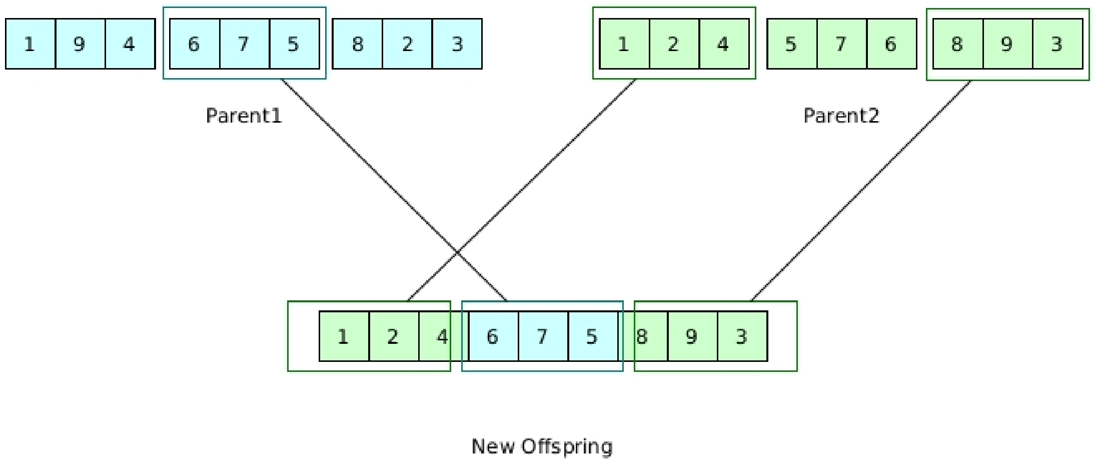


Figure 5.7: Shows the crossover for two parents with two cutting points each, resulting in a single child. In this case the problem has 9 tasks.

Mutation

The process of mutation involves modification of the current configuration of a chromosome. These modifications can be of different sorts, but in this study the mutation has been defined as follows: An offspring or a child will be selected for the mutation and will be cut at two random points. These two cutting points will divide the child or the selected chromosome into three sections. Then, the sequence of the middle section will be reversed, while the other two sections will remain unchanged. Let's assume that the random cutting points result in three equal parts, then the mutated child might look like in fig. 5.8. The probability of mutation has been set to 0.3.

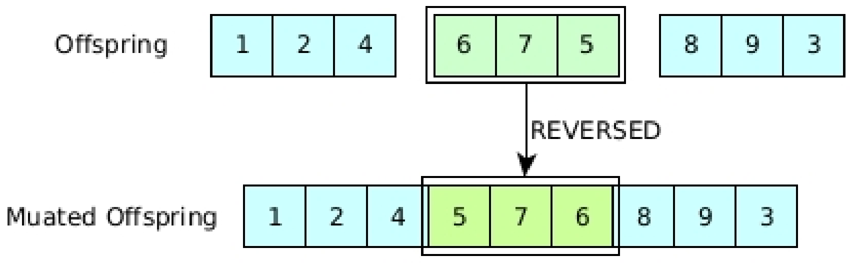


Figure 5.8: Shows the mutation procedure, resulting from a single offspring with two random cutting points, for a problem with 9 tasks.

Selection Procedure

Tournament selection has been chosen as the selection mechanism. Individuals or parents will be selected at random from the population according to the tournament size. Here, the tournament size is set to be three; hence, three individuals will be selected each time. These selected individuals will then be compared against each other, and the individual with the weakest fitness will not be selected. This method of selection makes sure that the worst individual in the population is never chosen.

Stopping Criteria

The stopping criteria is given by “the total number of generations”. This number is dependent on the size of the problem. Otherwise, the algorithm may fail to uncover promising solutions and converge pre-maturely or may take an unnecessary time to compute. In this study, five problems are being selected with different instances and the number of generations selected for these problems ranges from 50 to 1500 (these numbers were selected on the basis of trial and error).

Constraint Handling Method

In order to deal with infeasible solutions, a penalty value is used. Solutions that violate the precedence constraints will be subjected to severe penalties (they will be assigned the worst objective values). However, these solutions will remain in the population to maintain the diversity and because of the possibility to evolve into promising solutions in later generations.

The penalized value is set proportional to the deviation of an infeasible solution from a feasible one. In other words, an infeasible solution that violates only one constraint will be penalized less severely than a solution that violates two constraints. In the light of experiments, this allowed the algorithm to be biased towards relatively better solutions during the selection phase and reduced the computational time significantly, as compared to the scenario where all infeasible solutions were penalized equally. It should be noted that the penalty values were chosen such that the worst feasible solution is still always better than the best infeasible solution.

Station Restrictions

In reality, not every workstation or operator can perform all the tasks, and this fact has been taken into consideration in this thesis. Station restrictions inform us whether a specific task can be done on the selected workstation or not. To tackle this challenge, a simple heuristic has been introduced, which comprises two main steps.

The idea behind this heuristic is very simple: first to categorize tasks according to the workstations on which they can be performed. The second step is the allocation of these tasks according to these categories. When assigning tasks to workstations from the decision variables, the proposed method will check the category of the previous task. If the present and previous tasks belong to the same category, only then will the present task be allocated to the same workstation (assuming that doesn’t violate the cycle time constraint), otherwise the next workstation in the assembly line will be used. This makes sure that no task is given to a workstation not compatible with the allocated task.

In addition, if there are no conflicting precedence relations and no violations of the cycle time constraint, then the tasks from the same group will always be allocated to the same workstation, otherwise a next workstation in the assembly line has to be used.

6. Results and Analysis

Following are the five problems considered for the evaluation of the selected methodology,

1. Bowman's Problem
2. Buxey's Problem
3. WeeMag's Problem
4. Mixed-model Camera Problem
5. Mixed-model SmartPhone Problem

The first three problems are taken from Scholl [1993] where they are presented as a benchmark for simple assembly line problems. Although this study is dealing with multi-objective mixed-model assembly line balancing problems, instead of simple assembly line balancing problems, these benchmarks can be used to evaluate the second part (optimization with NSGA-II and NSGA-III) of the methodology. The chosen benchmarks are of three different sizes, the smallest problem with 8 tasks (Bowman), the medium problem with 29 tasks (Buxey) and one of the largest problems with 75 tasks (WeeMag).

The last two problems are related to mixed-model assembly lines, including the mixed-model camera problem taken from Uddin and Lastra [2011], and a mixed-model smartphone problem, which was based on previous research published in cooperation with the supervisors of this thesis, Seidelmann et al. [2021].

Only the smartphone problem encapsulates the aspects of mixed-model assembly lines, stations restrictions and multi-objective optimization fully. To the best of our knowledge, there are no benchmarks for multi-objective mixed-model assembly lines. Hence, we had to create a scenario that is as close to the real world problems as possible. The existing benchmarks in the literature are not sufficient and need to be reviewed, a lot of work needs to be done in this direction, but this topic is out of the scope of this study.

Each problem has been tested in a multi-objective formulation (cycle time, number of workstations, and smoothness index) and in a many-objective formulation (cycle time, number of workstations, variance in task distribution, and total idle time). NSGA-II and NSGA-III have been run for 31 times for each problem with multi- and many-objectives. A crossover probability of 0.9 and a mutation probability of 0.3 were chosen. A very detailed comparison between these two algorithms is given below using the following performance indicators,

1. Hypervolume (HV)
2. Inverted Generational Distance (IGD)
3. Dominance Ratios (DR)

For the calculation of these performance indicators, all solutions found by each algorithm and each run are archived in one place. In order to have a reference, a combined Pareto front for each problem is created from both the algorithms and all their runs and solutions for that problem. Using this reference front, the final generations of each run are then compared.

6.1 The Bowman Benchmark Problem

The Bowman's benchmark is one of the smallest and the simplest problems in the literature, with 8 tasks. The fig. 6.1 shows the precedence graph of Bowman's problem with the corresponding processing time for each task.

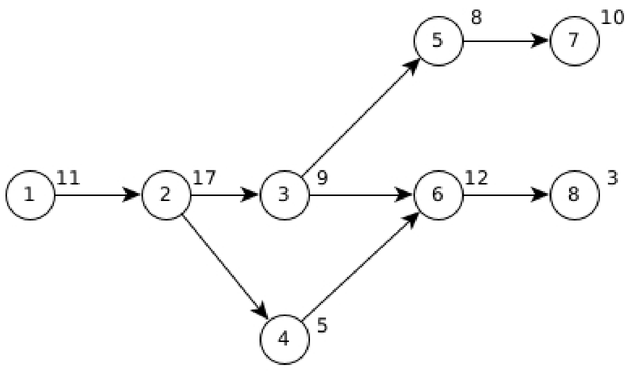


Figure 6.1: Bowman's precedence graph

6.1.1 Results for the Multi-Objectives Case

The algorithms NSGA-II and NSGA-III were run on the multi-objective scenario for minimization of cycle time, number of workstations, and smoothness index. The population size was set to 100 and there were 10,000 evaluations per run.

The fig. 6.2 shows the set of non-dominated solutions from all solutions evaluated over the 31 runs for NSGA-II and NSGA-III. Since the problem is of simplistic nature, it is very apparent that both algorithms were able to converge to the optimal solutions. There is no difference between the two Pareto fronts, however, fig. 6.3 shows that NSGA-II converges faster than NSGA-III with a small margin, although NSGA-III eventually finds better solutions on average. Additionally, the colors of the Pareto fronts show that smoothness index is not a perfect substitute for the number of workstations. This observation also holds true for several other experiments.

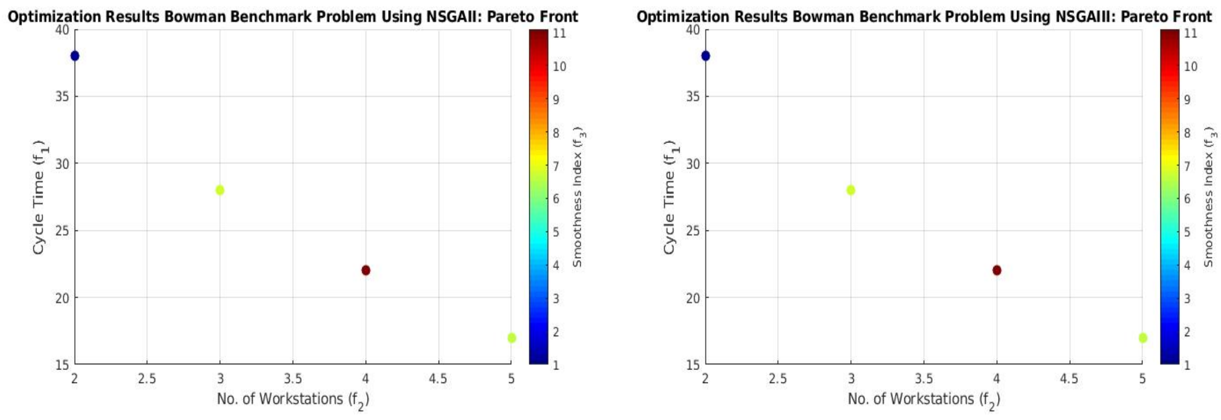


Figure 6.2: Pareto front for the multi-objective Bowman problem.

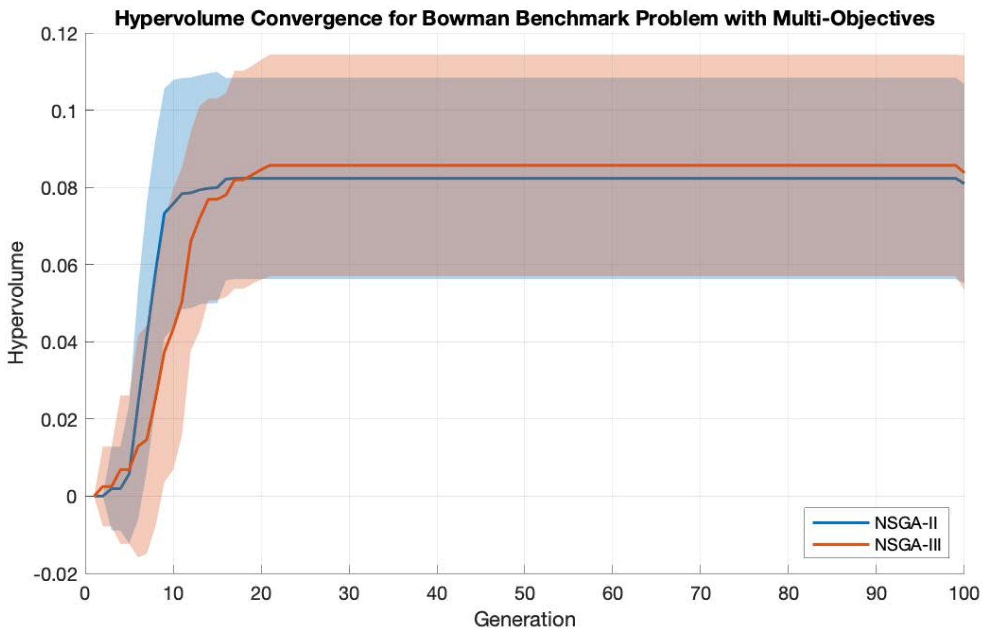


Figure 6.3: Hypervolume convergence of Bowman's problem for both algorithms across all the generations and runs. Standard deviation is represented by the shaded area.

The table 6.1 shows a test for statistical significance between the results of NSGA-II and NSGA-III in terms of HV, IGD and DR. The high *p values* indicate that both algorithms have performed significantly similar and no statistical difference can be derived.

P-Values for Bowman' Problem		
P_HV	P_IGD	P_DR
0.4049	0.7390	0.4052

Table 6.1: Shows statistical significance between final generations of NSGA-II and NSGA-III for Bowman's problem (multi-objective case).

Average and median values for HV, IGD and DR are shown in the table 6.2. The performance of both algorithms is almost identical, however, NSGA-II has performed better with regard to DR by a small margin and NSGA-III has performed a little better with regard to HV and IGD.

Mean Performance Indicators			
Algorithm	Hypervolume	Inverted Generational Distance	Dominance Ratios
NSGA-II	0.0809	3.6735	0.3134
NSGA-III	0.0837	3.6293	0.2451
Median Performance Indicators			
NSGA-II	0.0773	3.9292	0.3333
NSGA-III	0.1020	2.8640	0.25

Table 6.2: Median and average values of performance indicators of the last generations for multi-objective Bowman's problem.

6.1.2 Results for the Many-Objectives Case

The algorithms NSGA-II and NSGA-III were run on the many-objective scenario for minimization of cycle time, number of workstations, variance in task distribution, and total idle time. The population size was set to 220 and there were 11,000 evaluations per run.

The fig. 6.4 shows the set of non-dominated solutions from all solutions evaluated over the 31 runs for NSGA-II and NSGA-III. Both algorithms converged to the same solutions. While there is no difference between the two Pareto fronts, fig. 6.5 shows that NSGA-II converged faster than NSGA-III with a small margin, although NSGA-III had less variation in hypervolume values.

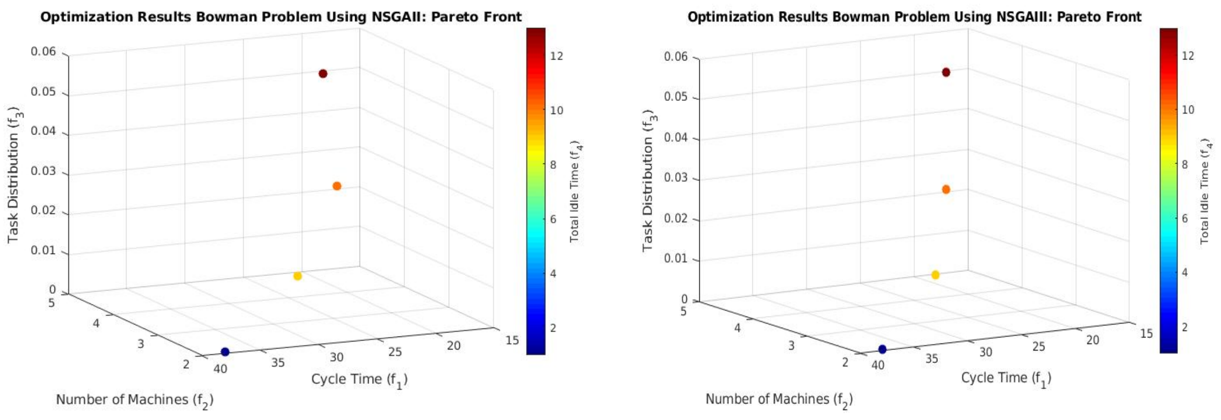


Figure 6.4: Pareto front for the many-objective Bowman problem.

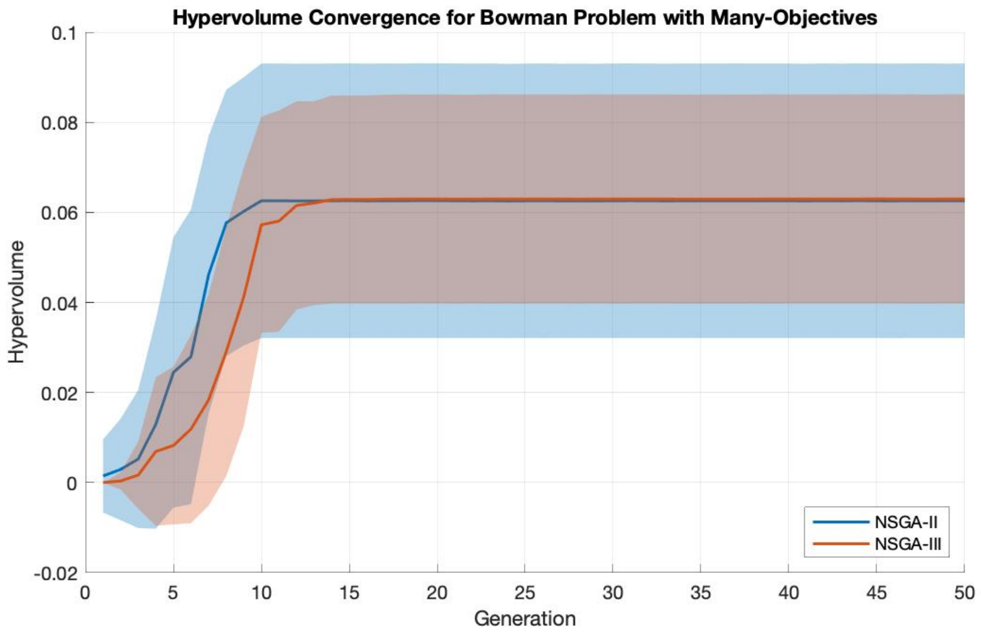


Figure 6.5: Hypervolume convergence of Bowman's problem for both algorithms across all the generations and runs. Standard deviation is represented by the shaded area.

The table 6.3 shows a test for statistical significance between the results of NSGA-II and NSGA-III in terms of HV, IGD and DR. Once again, the high *p values* indicate that both algorithms have performed significantly similar and no statistical difference can be derived.

P-Values for Bowman' Problem

P_HV	P_IGD	P_DR
0.8908	0.4907	0.8004

Table 6.3: Shows statistical significance between final generations of NSGA-II and NSGA-III for Bowman's problem (many-objective case).

Average and median values for HV, IGD and DR are shown in the table 6.4. The performance of both algorithms is almost identical.

Mean Performance Indicators			
Algorithm	Hypervolume	Inverted Generational Distance	Dominance Ratios
NSGA-II	0.0625	3.5984	0.2822
NSGA-III	0.0629	3.9155	0.2688
Median Performance Indicators			
NSGA-II	0.0831	3.7151	0.25
NSGA-III	0.0542	3.7151	0.25

Table 6.4: Median and average values of performance indicators of the last generations for many-objective Bowman's problem.

6.2 The Buxey Benchmark Problem

The Buxey benchmark problem is of medium size and one of the comparatively simpler problems in the literature. It contains 29 tasks. The fig. 6.6 shows the precedence graph of Buxey's problem with the corresponding processing time for each task.

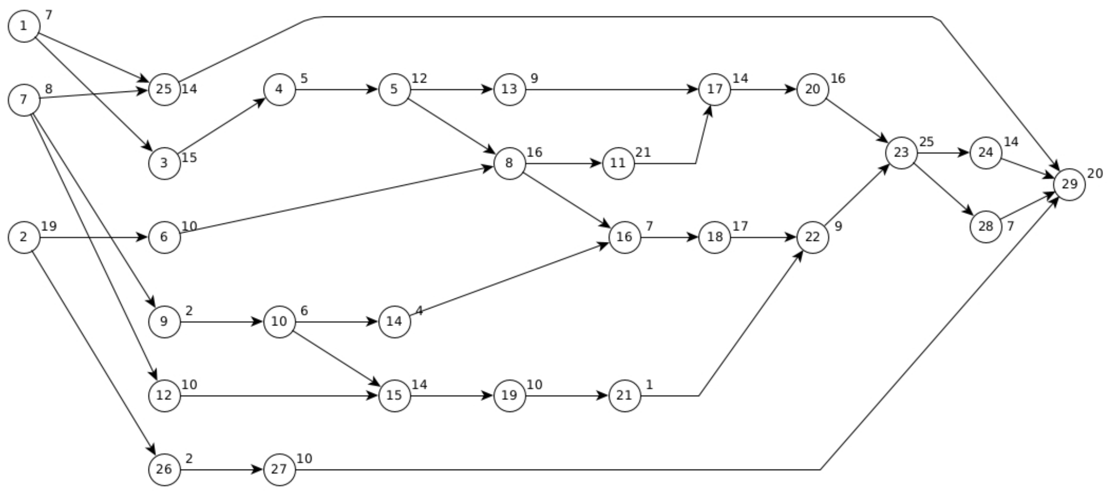


Figure 6.6: Buxey's precedence graph

6.2.1 Results for the Multi-Objectives Case

The algorithms NSGA-II and NSGA-III were run for the multi-objective scenario for three objectives, cycle time, number of workstations and smoothness index. The population size was set to 100 and there were 100,000 evaluations in total. The fig. 6.8 shows the set of non-dominated solutions from all solutions evaluated over the 31 runs for NSGA-II and NSGA-III. Both algorithms have performed well, with only slight differences. The fig. 6.8 shows that NSGA-II, again, converges faster than NSGA-III. However, in this scenario NSGA-III obtained noticeably better HV values, even in the final generations. It can also be seen that both algorithms kept improving even towards the end. Therefore, they might have benefited from more computational time.

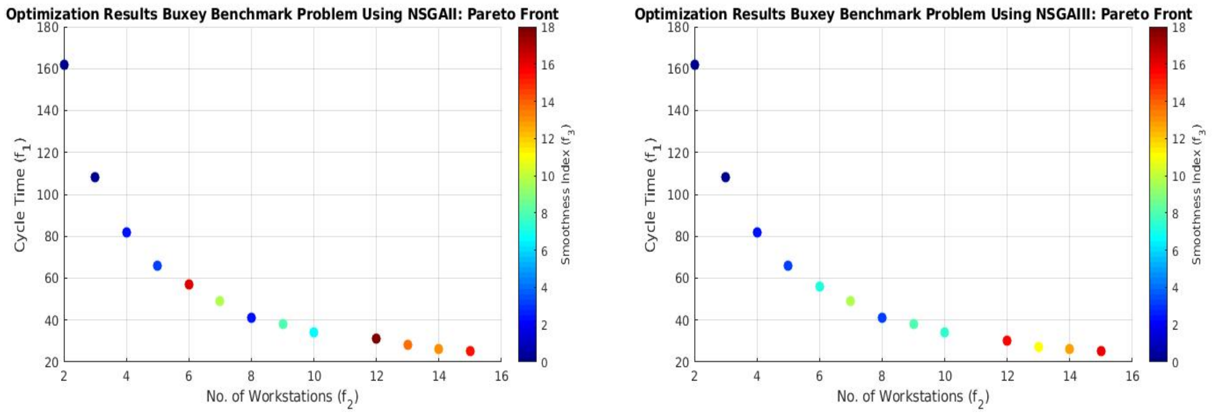


Figure 6.7: Pareto front for the multi-objective Buxey problem.

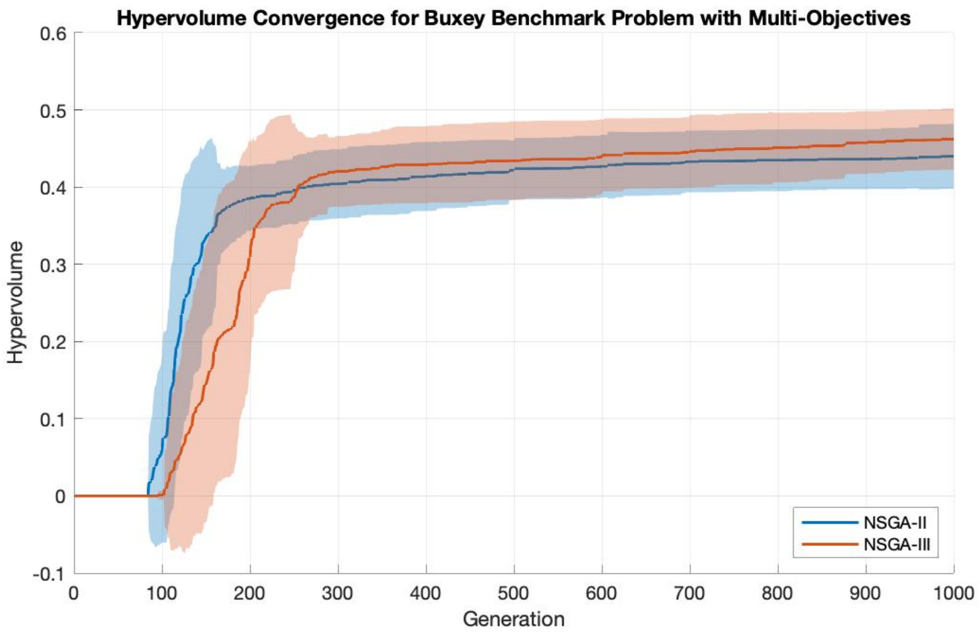


Figure 6.8: Hypervolume convergence of Buxey's problem for both algorithms across all the generations and runs. Standard deviation is represented by the shaded area.

The table 6.5 shows a statistical significant difference between NSGA-II and NSGA-III in terms of achieved HV. The value for IGD is also low, but doesn't indicate a strong difference. Finally, the value or DR shows that there is a significant overlap among the final solutions produced by both algorithms in this regard.

P-Values for Buxey's Problem

P_HV	P_IGD	P_DR
0.0599	0.1263	0.3036

Table 6.5: Shows statistical significance between final generations of NSGA-II and NSGA-III for Buxey's problem (multi-objective case).

Average and median values for HV, IGD and DR are shown in the table 6.6. The performance of both algorithms is close, however, NSGA-III has performed better for the median values and seems to have an edge in HV values.

Mean Performance Indicators			
Algorithm	Hypervolume	Inverted Generational Distance	Dominance Ratios
NSGA-II	0.4398	4.3947	0.4533
NSGA-III	0.46211	4.0925	0.4976
Median Performance Indicators			
NSGA-II	0.4380	4.3097	0.4285
NSGA-III	0.5479	3.9048	0.5

Table 6.6: Median and average values of performance indicators of the last generations for multi-objective Buxey's problem.

6.2.2 Results for the Many-Objectives Case

The algorithms NSGA-II and NSGA-III were run on the many-objective scenario for four objectives, cycle time, number of workstations, variance in task distribution and total idle time. The population size was set to 120 and there were 72,000 evaluations in total. The fig. 6.9 shows the set of non-dominated solutions from all solutions evaluated over the 31 runs for NSGA-II and NSGA-III. Both algorithms converged to mostly the same solutions, with slight differences. The fig. 6.10 shows that NSGA-II was able to find good solutions faster than NSGA-III, as before. In this case, NSGA-III did not eventually take the lead from NSGA-II, though.

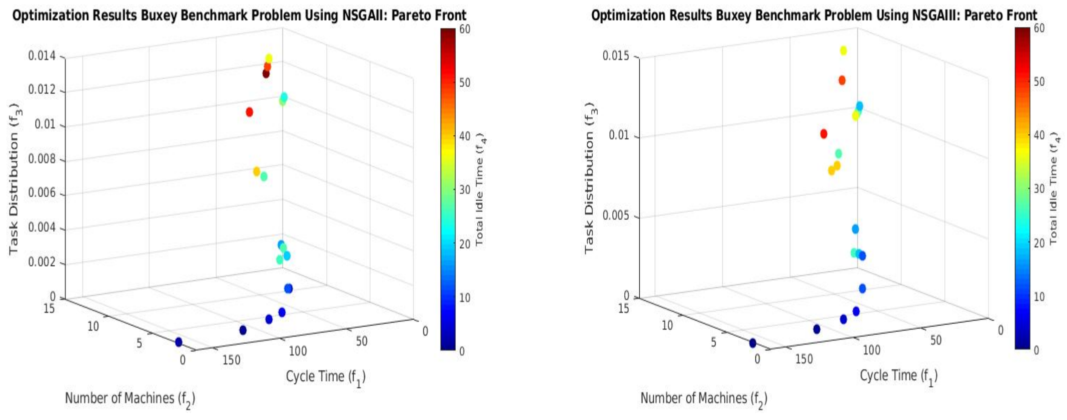


Figure 6.9: Pareto front for the many-objective Buxey problem.

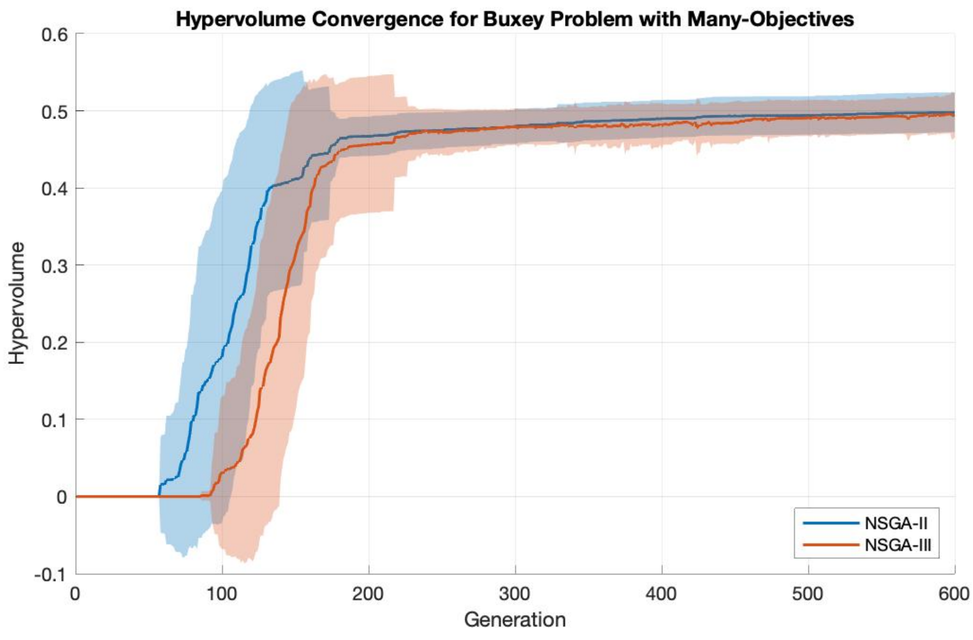


Figure 6.10: Hypervolume convergence of Buxey's problem for both algorithms across all the generations and runs. Standard deviation is represented by the shaded area.

As the table 6.7 shows, there is no tractable statistical difference between the results of NSGA-II and NSGA-III for any of the performance indicators regarding their last generations. The same was also indicated by fig. 6.10.

P-Values for Buxey's Problem

P_HV	P_IGD	P_DR
0.7989	0.5435	0.6189

Table 6.7: Shows statistical significance between final generations of NSGA-II and NSGA-III for Buxey's problem (many-objective case)

Average and median values for HV, IGD and DR are shown in the table 6.8. The performance of both algorithms is almost identical in all aspects.

Mean Performance Indicators			
Algorithm	Hypervolume	Inverted Generational Distance	Dominance Ratios
NSGA-II	0.4979	7.239	0.4646
NSGA-III	0.4943	7.0411	0.4470
Median Performance Indicators			
NSGA-II	0.4967	7.3085	0.4666
NSGA-III	0.5028	7.10005	0.4738

Table 6.8: Median and average values of performance indicators of the last generations for many-objective Buxey's problem.

6.3 The WeeMag Benchmark Problem

The WeeMag benchmark problem is one of the largest among all problems in the literature, and contains 75 tasks. The fig. 6.11 shows the precedence graph of WeeMag's problem with the corresponding processing time for each task.

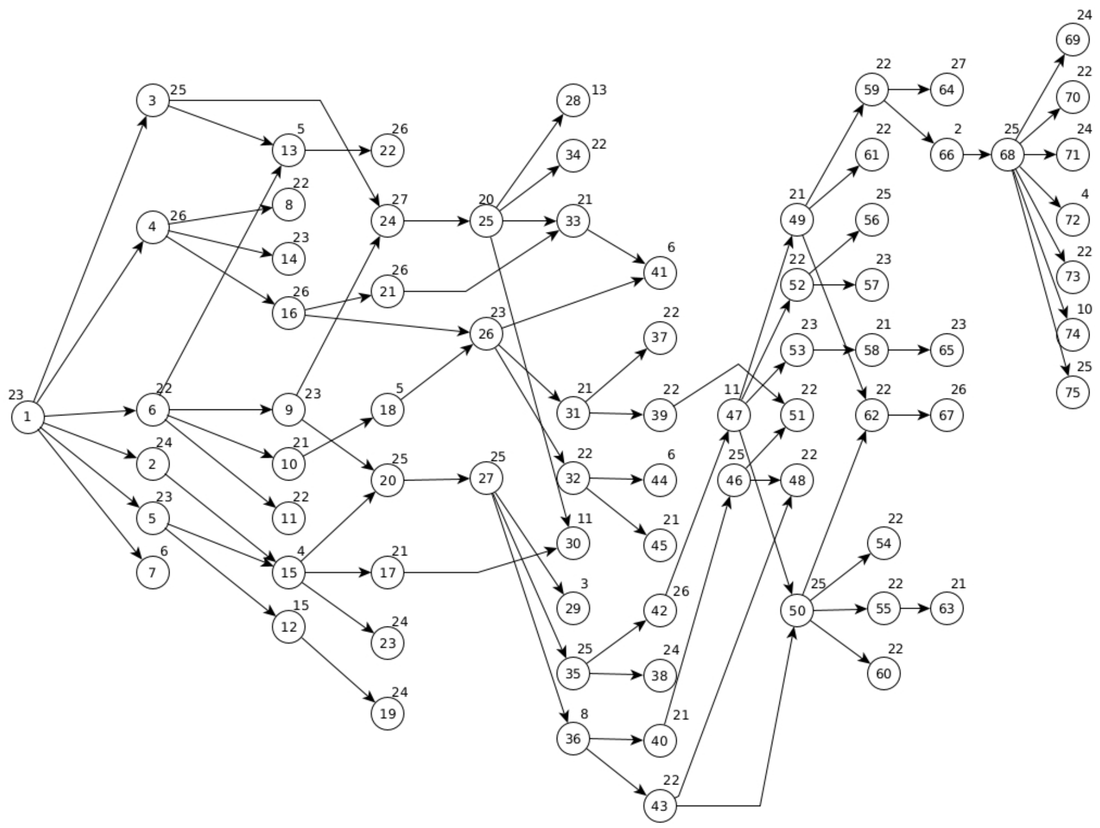


Figure 6.11: WeeMag's precedence graph

6.3.1 Results for Multi-Objectives Case

The algorithms NSGA-II and NSGA-III were run for the multi-objective scenario for three objectives, cycle time, number of workstations and smoothness index. The population size was set to 100 and 150,000 evaluations in total. The fig. 6.12 shows the set of non-dominated solutions from all solutions evaluated over the 31 runs for NSGA-II and NSGA-III. Both algorithms agree on the Pareto fronts, with only very slight differences. The fig. 6.13 shows that NSGA-II found the final solution set faster than NSGA-III with a small margin, upholding the previously established trend.

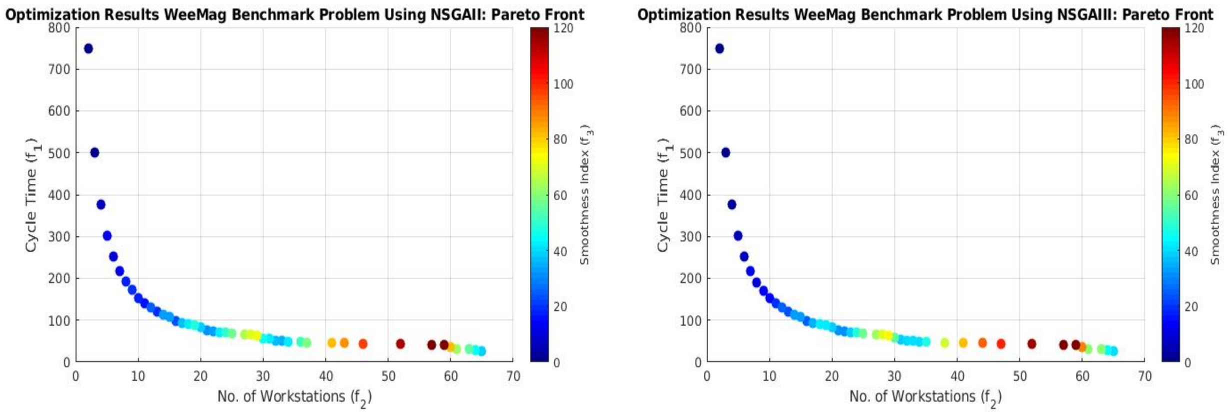


Figure 6.12: Pareto front for the multi-objective WeeMag problem.

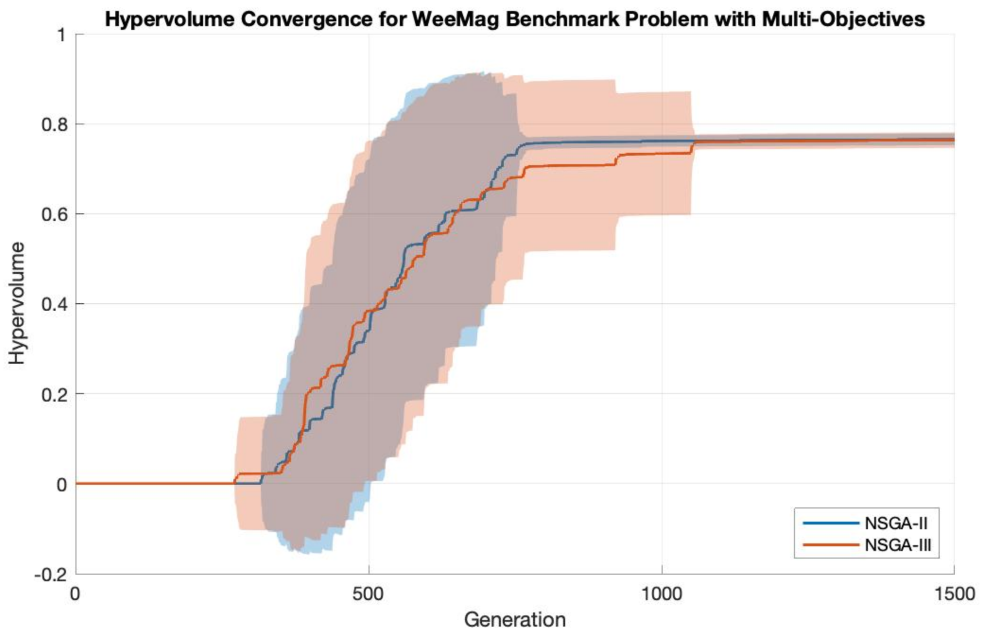


Figure 6.13: Hypervolume convergence of WeeMag's problem for both algorithms across all the generations and runs. Standard deviation is represented by the shaded area.

Once again, there is little statistical difference between the performance of NSGA-II and NSGA-III as indicated by the table 6.9.

P-Values for WeeMag's Problem		
P_HV	P_IGD	P_DR
0.5052	0.5052	0.9375

Table 6.9: Shows statistical significance between the last generations of NSGA-II and NSGA-III for WeeMag's problem (multi-objective case).

Average and median values for HV, IGD and DR are shown in the table 6.10. The performance of both algorithms is almost identical with respect to the final generations.

Mean Performance Indicators			
Algorithm	Hypervolume	Inverted Generational Distance	Dominance Ratios
NSGA-II	0.7652	8.0162	0.49100
NSGA-III	0.7632	8.0946	0.4971
Median Performance Indicators			
NSGA-II	0.7688	7.8107	0.5106
NSGA-III	0.7655	8.031	0.4897

Table 6.10: Median and average values of performance indicators of the last generations for multi-objective WeeMag's problem.

6.3.2 Results for Many-Objectives Case

The algorithms NSGA-II and NSGA-III were run for the many-objective scenario for four objectives, cycle time, number of workstations, variance in task distribution and total idle time. The population size was set to 120 and there were 120,000 evaluations for each run. The fig. 6.14 shows the set of non-dominated solutions from all solutions evaluated over the 31 runs for NSGA-II and NSGA-III. Again, both algorithms were able to find almost identical Pareto fronts. From fig. 6.15, however, it is visible that NSGA-II had a very significant and consistent lead in convergence towards the final solution sets, while also having slightly less variance in the process. Interestingly, the final generations show very little variance for both algorithms.

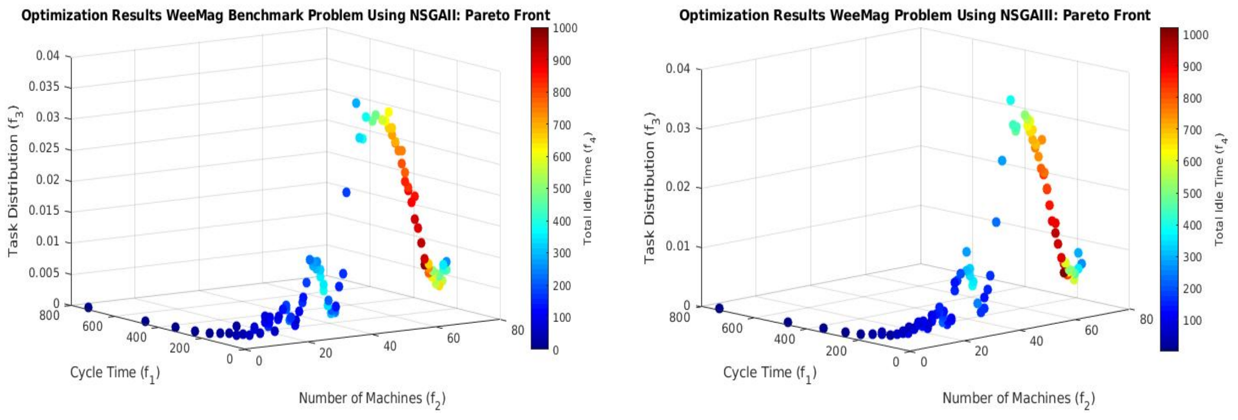


Figure 6.14: Pareto front for the many-objective WeeMag problem.

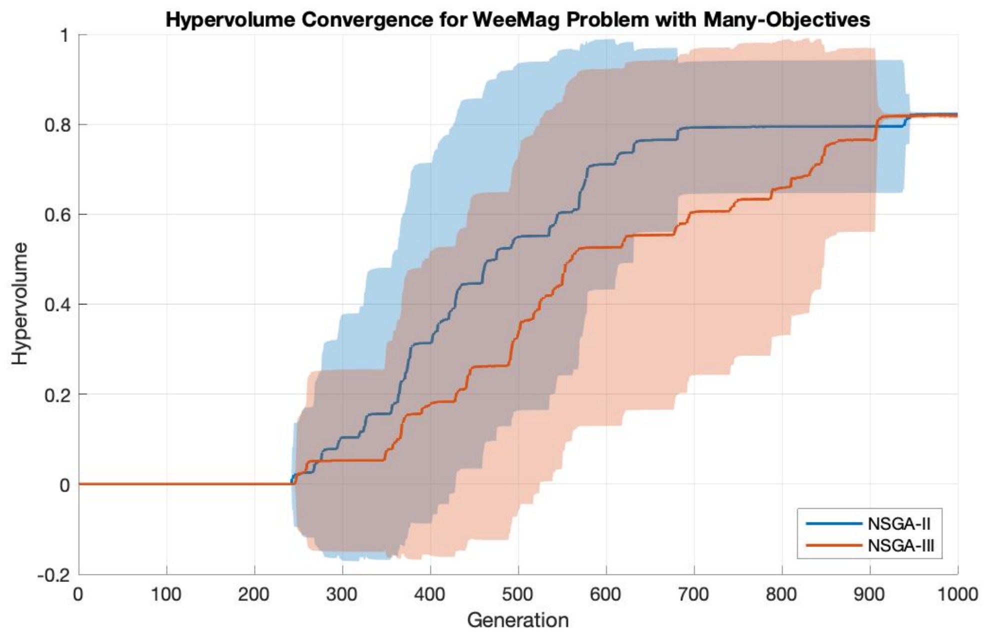


Figure 6.15: Hypervolume convergence of WeeMag's problem for both algorithms across all the generations and runs. Standard deviation is represented by the shaded area.

The table 6.11 shows that there is a statistical significant difference between the final generations found by NSGA-II and NSGA-III for all performance metrics, especially IGD.

P-Values for WeeMag's Problem

P_HV	P_IGD	P_DR
0.0599	0.00006	0.0072

Table 6.11: Shows statistical significance between the last generations of NSGA-II and NSGA-III for WeeMag's problem (many-objective case).

Average and median values for HV, IGD and DR are shown in the table 6.12. The performance of both algorithms is close. However, it is evident that NSGA-II performed noticeably better for every metric, especially in regards to IGD and DR.

Mean Performance Indicators			
Algorithm	Hypervolume	Inverted Generational Distance	Dominance Ratios
NSGA-II	0.8219	12.7018	0.4773
NSGA-III	0.8189	14.782	0.3736
Median Performance Indicators			
NSGA-II	0.82186	12.6898	0.4696
NSGA-III	0.8187	14.2789	0.3636

Table 6.12: Median and average values of performance indicators of the last generations for many-objective WeeMag's problem.

6.4 The Mixed-Model Camera Manufacturing Problem

The mixed-model camera problem is one of the smaller sized problems, and it is discussed in the paper Uddin and Lastra [2011]. This problem has just 10 tasks. The fig. 6.16 shows the precedence graph for the camera problem with the corresponding processing times for each task.

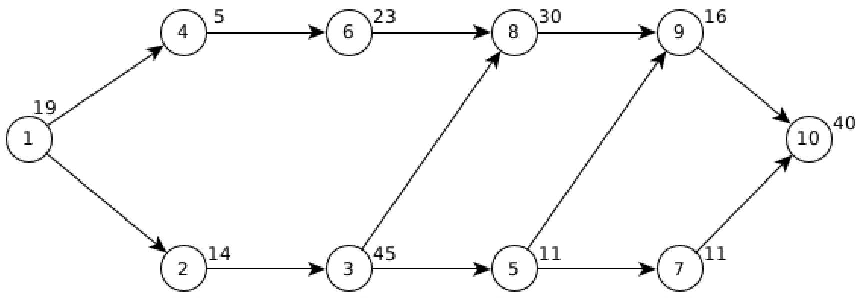
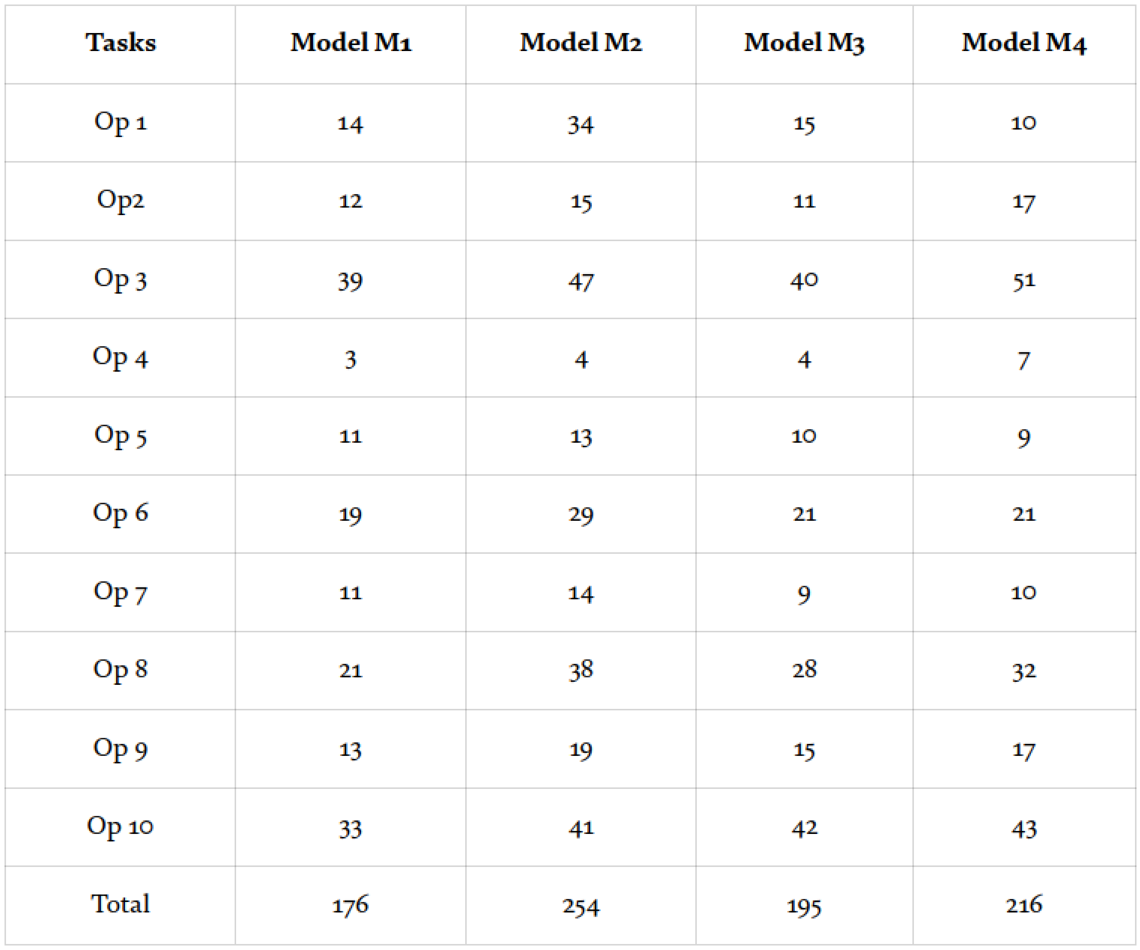


Figure 6.16: Camera's precedence graph

The information for the different models of the cameras is given by the following fig. 6.17, based on the table given by Uddin and Lastra [2011].



Tasks	Model M1	Model M2	Model M3	Model M4
Op 1	14	34	15	10
Op2	12	15	11	17
Op 3	39	47	40	51
Op 4	3	4	4	7
Op 5	11	13	10	9
Op 6	19	29	21	21
Op 7	11	14	9	10
Op 8	21	38	28	32
Op 9	13	19	15	17
Op 10	33	41	42	43
Total	176	254	195	216

Figure 6.17: Information about processing times for all 4 models of the camera.

This problem has been modified to be a multi-objective problem unlike given in the paper Uddin and Lastra [2011]. The results here are therefore not directly comparable to theirs. However, before the modification of this problem, the algorithms have been tested using the single-objective version of the problem. It was observed that both algorithms were able to find the optimal results that were also given by the original paper.

6.4.1 Results for the Multi-Objectives Case

The algorithms NSGA-II and NSGA-III were run on the multi-objective scenario for three objectives, cycle time, number of workstations and smoothness index. The population size was set to 100 and there were 10,000 evaluations for each run. The fig. 6.18 shows the set of non-dominated solutions from all solutions evaluated over the 31 runs for NSGA-II and NSGA-III. In this case, both algorithms have performed equally well. However, fig. 6.19 shows that NSGA-II converges significantly faster than NSGA-III, as seen before.

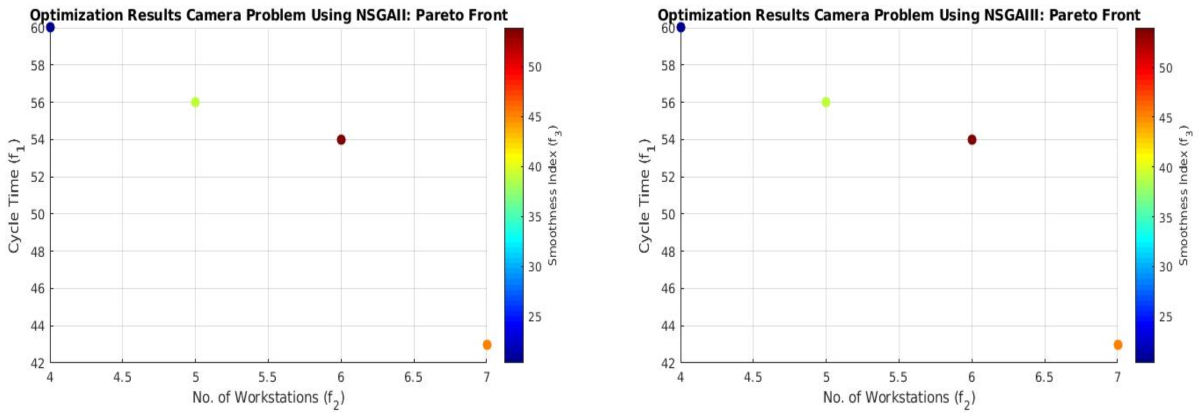


Figure 6.18: Pareto front for the multi-objective camera problem.

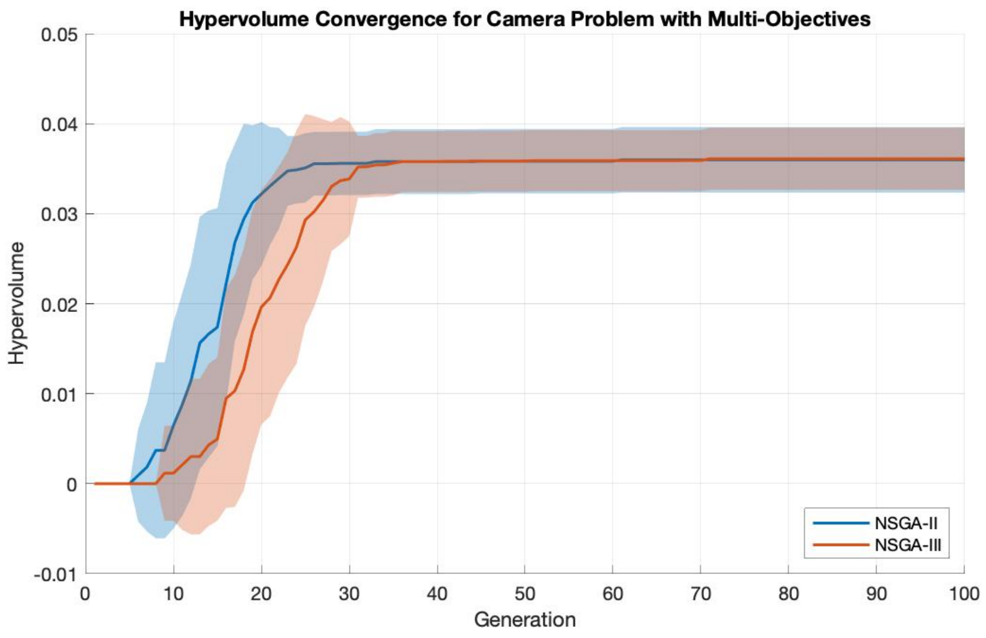


Figure 6.19: Hypervolume convergence of the camera problem for both algorithms across all the generations and runs. Standard deviation is represented by the shaded area.

The table 6.13 shows a lack of statistical significant differences between the results of NSGA-II and NSGA-III for all performance metrics regarding the final generations.

P-Values for Mixed-Model Camera Problem

P_HV	P_IGD	P_DR
0.9753	0.8936	0.2162

Table 6.13: Shows statistical significance between final generations of NSGA-II and NSGA-III for the camera problem (multi-objective case).

Average and median values for HV, IGD and DR are shown in the table 6.14. Evidently, the performance of both algorithms is almost the same.

Mean Performance Indicators			
Algorithm	Hypervolume	Inverted Generational Distance	Dominance Ratios
NSGA-II	0.0359	4.3759	0.1747
NSGA-III	0.0360	4.2602	0.2607
Median Performance Indicators			
NSGA-II	0.0351	3.8061	0.25
NSGA-III	0.03425	5.5295	0.25

Table 6.14: Median and average values of performance indicators of the last generations for the multi-objective Camera problem.

6.4.2 Results for the Many-Objectives Case

The algorithms NSGA-II and NSGA-III were run for the many-objective scenario for four objectives, cycle time, number of workstations, variance in task distribution and total idle time. The population size was set to 100 and there were 50,000 evaluations for each run. The fig. 6.20 shows the set of non-dominated solutions from all solutions evaluated over the 31 runs for NSGA-II and NSGA-III. It can be seen that both algorithms converged to the same solutions. The fig. 6.21 shows that NSGA-II was able to find better solutions faster than NSGA-III by a small margin, although NSGA-III tended to converge to better solutions in the end. Both algorithms were able to solve this problem faster than expected and didn't need nearly as many evaluations as were used, giving the graph a compressed look on the x-axis.

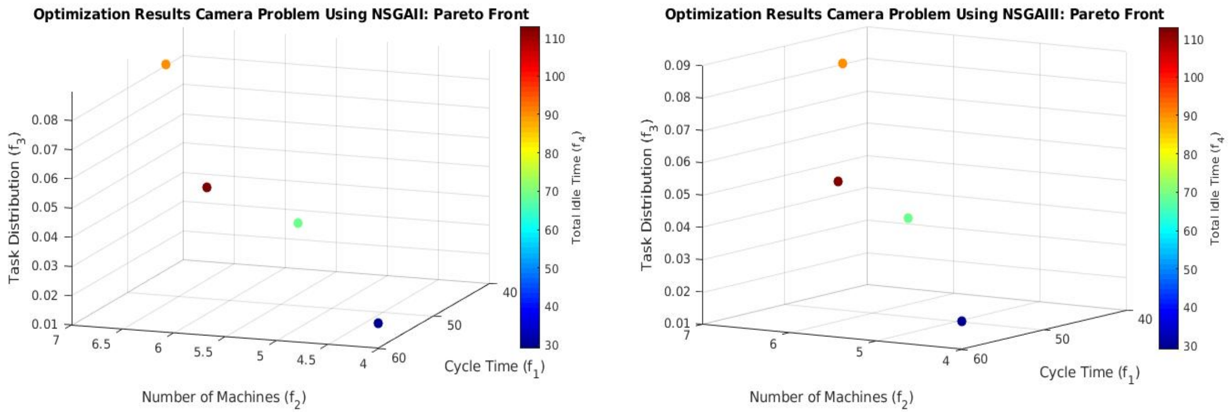


Figure 6.20: Pareto front for the many-objective camera problem.

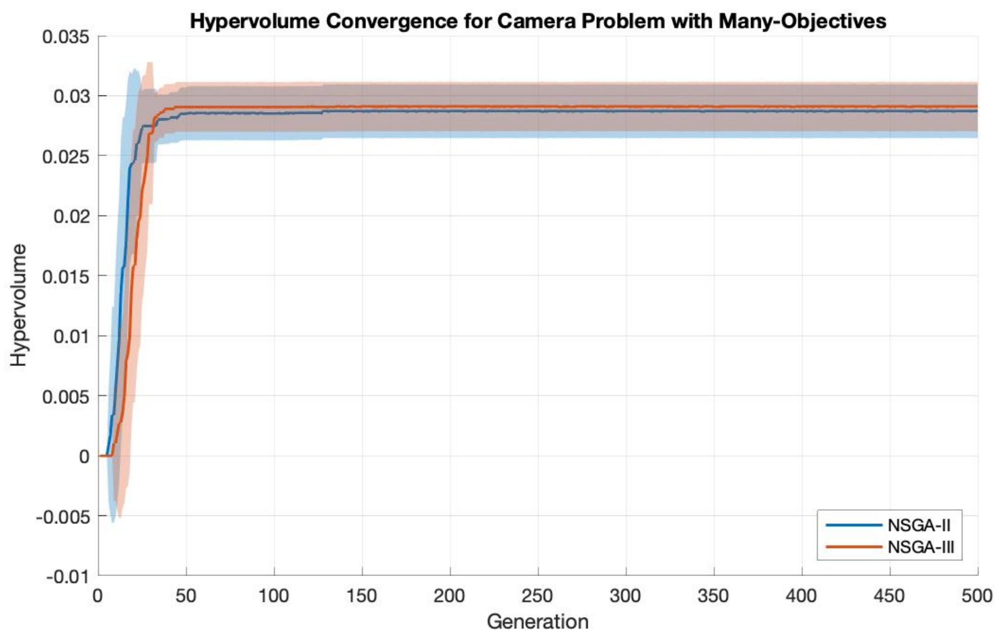


Figure 6.21: Hypervolume convergence of the camera problem for both algorithms across all the generations and runs. Standard deviation is represented by the shaded area.

The table 6.15 shows that there was no statistical significant difference between the final generations of NSGA-II and NSGA-III for any performance metric.

P-Values for Mixed-Model Camera Problem

P_HV	P_IGD	P_DR
0.6583	0.9297	0.4176

Table 6.15: Shows statistical significance between final generations of NSGA-II and NSGA-III for the camera problem (many-objective case).

Average and median values for HV, IGD and DR are shown in the table 6.16. The performance of both algorithms is almost the same, however, NSGA-III has performed slightly better than NSGA-II overall.

Mean Performance Indicators			
Algorithm	Hypervolume	Inverted Generational Distance	Dominance Ratios
NSGA-II	0.0287	3.8735	0.2123
NSGA-III	0.0291	3.6742	0.2634
Median Performance Indicators			
NSGA-II	0.0270	6.1949	0.2
NSGA-III	0.0300	6.2097	0.25

Table 6.16: Median and average values of performance indicators of the last generations for the many-objective camera problem.

6.5 The Mixed-Model SmartPhone Problem

The mixed-model smartphone problem is one of the medium to large sized problems. It has been fabricated in a way that it has very close resemblance to the real world mixed-model problems and contains 44 tasks. The fig. A.16 in the *appendix*, shows the precedence graph of the smartphone problem consisting all the possible options. The fig. 6.22 shows the joint precedence graph after taking all the probabilities for each option into consideration.

Group	Task Number
A	1
B	2
C	3
D	4
E	5
F	6,8,19
G	9
H	10
I	11
J	12,13,14,16
K	15, 17, 21
L	18, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42, 44
M	20
N	41
O	43

Table 6.17: Shows grouping of the tasks that are of the same nature and can be performed by the same workstations.

The mixed-model smartphone manufacturing process provides 9 tasks that come with different options to choose from. Table 6.18 and Table 6.19 show all the details of the smartphone manufacturing process along with the description for each task and its options. The table 6.17 shows categorization of all the tasks into similar categories that can be performed by the same kind of workstation. Considering all option possibilities, 15,552 different models of the smartphone can be manufactured. For the sake of visualization, 3 possible models out of the 15,552 models are shown in the *appendix*: fig. A.17, fig. A.18, and fig. A.19.

Tasks	Options	Nature of the Tasks	Precedence
1. Die Group Case Molding	no options	Common task (Mandatory)	-
2. Cutting Molding Excess	no options	Common task (Mandatory)	1. Die Group Case Molding
3. Case Anodizing	no options	Common task (Mandatory)	2. Cutting Molding Excess
4. Antenna Interface Cutting	no options	Common task (Mandatory)	3. Case Anodizing
5. Creating Plastic Speaker Cover	no options	Common task (Mandatory)	4. Antenna Interface Cutting
6. Case Drilling	no options	Common task (Mandatory)	5. Creating Plastic Speaker Cover
7. Cutting Antenna Groove	no options	Common task (Mandatory)	5. Creating Plastic Speaker Cover
8. Case Outline Precision Cutting	no options	Common task (Mandatory)	5. Creating Plastic Speaker Cover
9. Case Sand Blasting	no options	Common task (Mandatory)	6. Case Drilling 7. Cutting Antenna Groove 8. Case Outline Precision Cutting
10. Case Plating	no options	Common task (Mandatory)	9. Case Sand Blasting
11. Anti-Oxidant Coating	no options	Common task (Mandatory)	10. Nickel Case Plating
12. Cut Glass into Shape	no options	Common task (Mandatory)	-
13. Attach Protective Film to Glass	no options	Common task (Mandatory)	12. Cut Glass into Shape
14. Mount Glass on Cutting Surface	no options	Common task (Mandatory)	13. Attach Protective Film to Glass
15. Press Bond Protective Film to Glass	no options	Common task (Mandatory)	14. Mount Glass on Cutting Surface
16. Cut Excess of Protective Film	no options	Common task (Mandatory)	15. Press Bond Protective Film to Glass
17. Join Glass and LCD Display	no options	Common task (Mandatory)	16. Cut Excess of Protective Film
18. Check LCD Functionality	no options	Common task (Mandatory)	17. Join Glass and LCD Display
19. Case Inside Precision Cut	no options	Common task (Mandatory)	5. Creating Plastic Speaker Cover
20. Injection Molding Back Shell	no options	Common task (Mandatory)	-
21. Joining Case and Back Shell	no options	Common task (Mandatory)	11. Anti-Oxidant Coating 19. Case Inside Precision Cut 20. Injection Molding Back Shell
22. Split PCB	no options	Common task (Mandatory)	-
23. Assemble Camera Module	[Single Camera, Double Camera, Triple Camera]	XOR (mutually exclusive tasks)	22. Split PCB
24. Attach Camera Cover	no options	Common task (Mandatory)	23. Assemble Camera Module
25. Test Camera Module	no options	Common task (Mandatory)	23. Assemble Camera Module
26. Install CPU on Motherboard	[Quad Core, Octo Core]	XOR (mutually exclusive tasks)	22. Split PCB
27. Install RAM on Motherboard	[4 GB, 8 GB, 12 GB]	XOR (mutually exclusive tasks)	22. Split PCB
28. Install Mandatory Communication Chips	no options	Common task (Mandatory)	22. Split PCB
29. Install 5G Chip	no options	Optional	28. Install Mandatory Communication Chips
30. Install Mandatory Sensor Chips	no options	Common task (Mandatory)	22. Split PCB
31. Install Optional Sensor Chips	[Thermometer, Barometer, Humidity Sensor]	Multi-optional (Can be selected arbitrarily)	30. Install Mandatory Sensor Chips
32. Install SIM Card Reader	[Single SIM, Dual SIM]	XOR (mutually exclusive tasks)	22. Split PCB
33. Connect SSD to Motherboard	no options	Common task (Mandatory)	22. Split PCB
34. Bolt Motherboard to Case	no options	Common task (Mandatory)	21. Joining Case and Back Shell 24. Attach Camera Cover 25. Test Camera Module 26. Install CPU on Motherboard 29. Install 5G Chip 31. Install Optional Sensor Chips 32. Install SIM Card Reader 33. Connect SSD to Motherboard
35. Wire Motherboard and Other PCBs	no options	Common task (Mandatory)	34. Bolt Motherboard to Case
36. Install Battery	no options	Common task (Mandatory)	35. Wire Motherboard and Other PCBs
37. Screw in Plastic PCB Covers	no options	Common task (Mandatory)	36. Install Battery
38. Put Warning and Guarantee labels	no options	Common task (Mandatory)	37. Screw in Plastic PCB Covers
39. Attach Glass with LCD Display	no options	Common task (Mandatory)	37. Screw in Plastic PCB Covers
40. Install Software Packages	[Package 1, Package 2, Package 3]	XOR (mutually exclusive tasks)	39. Attach Glass with LCD Display
41. Engrave Back Shell	[Nothing, Name Engraving, Pattern Engraving]	XOR (mutually exclusive tasks)	21. Joining Case and Back Shell
42. Extensive Functionality Test	no options	Common task (Mandatory)	40. Install Software Packages
43. Print Gift Card	no options	Optional	-
44. Package Smartphone	no options	Common task (Mandatory)	41. Engrave Back Shell 42. Extensive Functionality Test 43. Print Gift Card 18. check LCD Functionality

Table 6.18: Shows information about all the available options to create a smartphone model along with the nature of the options and full description of each task corresponding to its number.

Combinations of options according to their nature	Probabilities for each option combinations	Processing time for each combination
only one	1	4
only one	1	1
only one	1	20
only one	1	2
only one	1	2
only one	1	3
only one	1	3
only one	1	4
only one	1	5
only one	1	10
only one	1	5
only one	1	6
only one	1	1
only one	1	2
only one	1	3
only one	1	5
only one	1	3
only one	1	4
only one	1	4
only one	1	4
only one	1	4
only one	1	2
3 mutually exclusive ways	[0.3, 0.2, 0.5]	[4, 5, 6]
only one	1	1
only one	1	7
2 mutually exclusive ways	[0.4, 0.6]	[3, 3]
3 mutually exclusive ways	[0.2, 0.3, 0.5]	[2, 3, 4]
only one	1	5
2 ways: either selected, or not selected	[0.4, 0.6]	[3, 0]
only one	1	7
8 ways: none, or any possible combination	[0.3, 0.2, 0.05, 0.05, 0.05, 0.05, 0.05, 0.25]	[0, 3, 3, 3, 5, 5, 5, 7]
2 mutually exclusive ways	[0.4, 0.6]	[3, 4]
only one	1	2
only one	1	5
only one	1	9
only one	1	3
only one	1	4
only one	1	2
only one	1	5
3 mutually exclusive ways	[0.6, 0.3, 0.1]	[6, 10, 14]
3 mutually exclusive ways	[0.4, 0.3, 0.3]	[0, 5, 7]
only one	1	15
2 ways: either selected, or not selected	[0.05, 0.95]	[0,5]
only one	1	3

Table 6.19: Shows information about all the possible combinations for each option, along with their probabilities and processing times for each combination of the option.

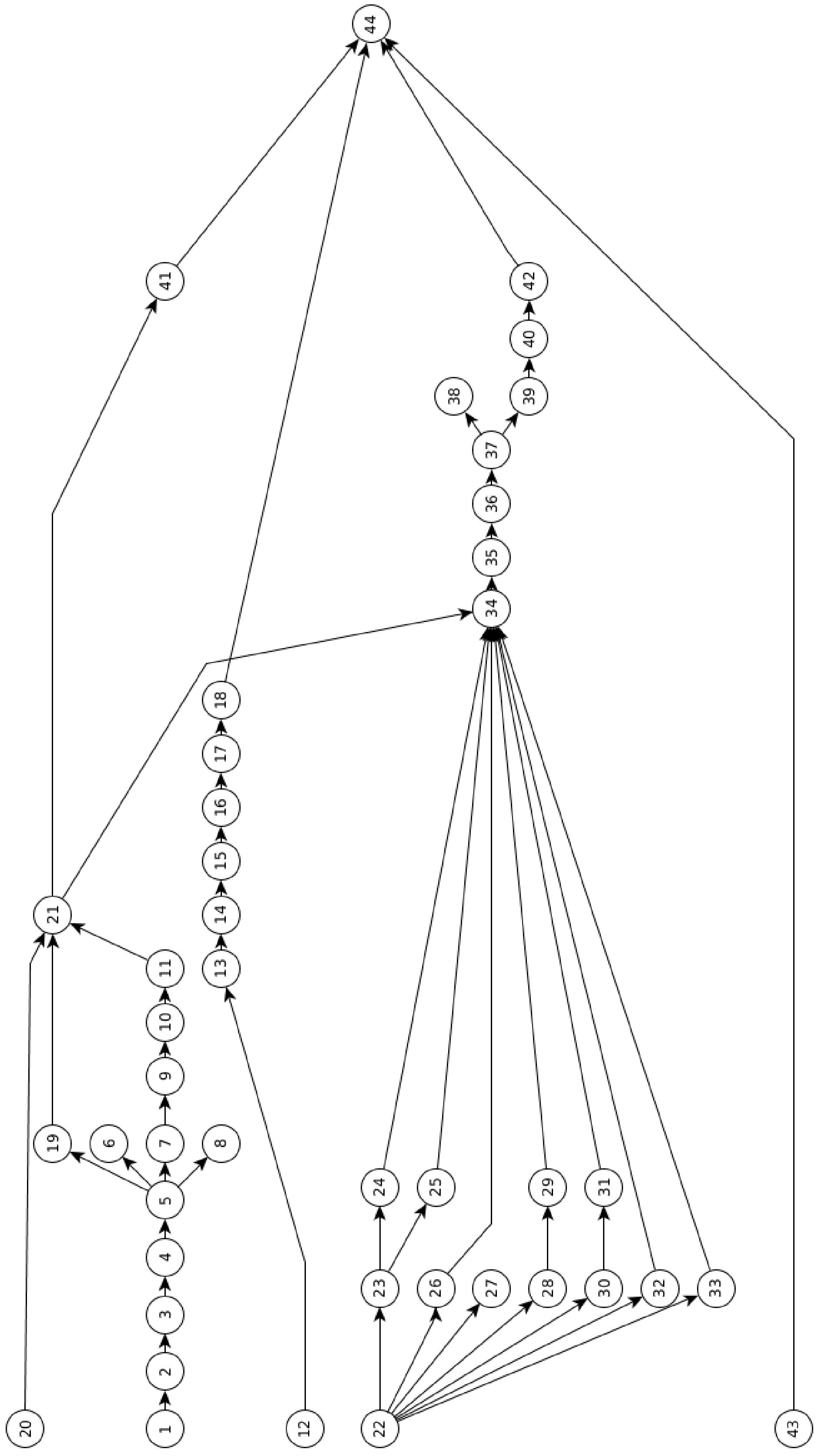


Figure 6.22: SmartPhone joint precedence graph

6.5.1 Results for Multi-Objectives Case

The algorithms NSGA-II and NSGA-III were run for the multi-objective scenario for three objectives, cycle time, number of workstations and smoothness index. The population size was set to 100 and there were 80,000 evaluations for each run. The fig. 6.23 shows the set of non-dominated solutions from all solutions evaluated over the 31 runs for NSGA-II and NSGA-III. For the first time, the two algorithms are found to disagree on the Pareto front substantially. As indicated by fig. 6.24 it is evident that NSGA-II has performed significantly better than NSGA-III in terms of HV, although it appears that neither algorithm was able to fully converge in the given computational time. While NSGA-II was able to find better solutions, NSGA-III found more non-dominated solutions.

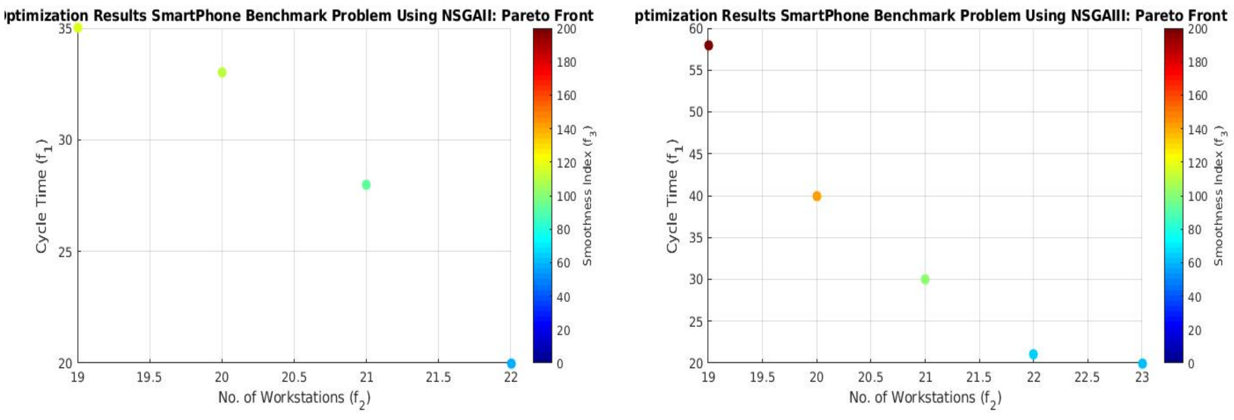


Figure 6.23: Pareto front for the multi-objective SmartPhone problem.

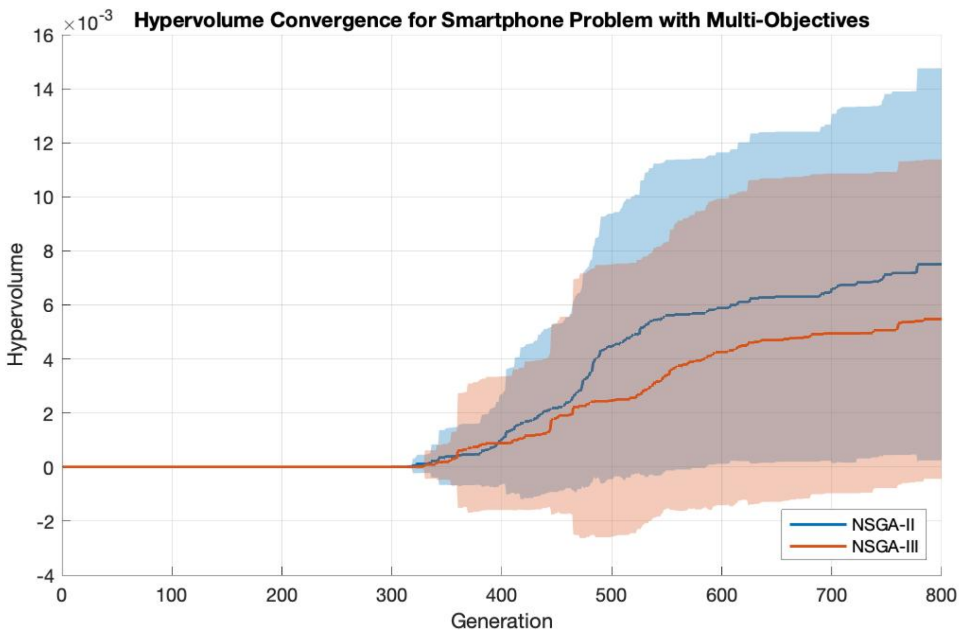


Figure 6.24: Hypervolume convergence of the SmartPhone problem for both algorithms across all the generations and runs. Standard deviation is represented by the shaded area.

The table 6.20 shows that there is a moderate statistical difference between the final generations of NSGA-II and NSGA-III in terms of HV, IGD. Only the value for DR shows a strong significance. Despite the HV graph showing a big difference between NSGA-II and NSGA-III, the statistical significance seems comparatively low. This is explained by the very high variance present, which also shows in the HV plot. The algorithms had difficulties escaping from infeasible solutions and even in later generations frequently contained many infeasible ones, making the algorithms perform more similarly than the plot seems to suggest.

P-Values for SmartPhone Manufacturing Problem

P_HV	P_IGD	P_DR
0.1215	0.1763	0.0390

Table 6.20: Shows statistical significance between final generations of NSGA-II and NSGA-III for the SmartPhone problem (multi-objective case).

Average and median values for HV, IGD and DR are shown in the table 6.21. It is very apparent that, on average, NSGA-II has outperformed NSGA-III. Especially because solutions found by NSGA-II have frequently dominated the best solutions found by NSGA-III.

Mean Performance Indicators			
Algorithm	Hypervolume	Inverted Generational Distance	Dominance Ratios
NSGA-II	0.0075	10.322	0.6596
NSGA-III	0.0054	10.7469	0.3252
Median Performance Indicators			
NSGA-II	0.005	10.3388	1.0
NSGA-III	0.004	11.1904	0.0

Table 6.21: Median and average values of performance indicators of the last generations for the multi-objective SmartPhone problem.

6.5.2 Results for the Many-Objectives Case

The algorithms NSGA-II and NSGA-III were run for the many-objective scenario for four objectives, cycle time, number of workstations, variance in task distribution and total idle time. The population size was set to 120 and there were 120,000 evaluations for each run. The fig. 6.25 shows the set of non-dominated solutions from all solutions evaluated over the 31 runs for NSGA-II and NSGA-III. In this case, both algorithms have performed more similar than before, and many more solutions were found. The fig. 6.26 shows that NSGA-II, as usual, was able to converge to the good solutions faster than NSGA-III, although in this case the final generations showed little difference. Notably, there is much less variance in HV values overall compared to the multi-objective case.

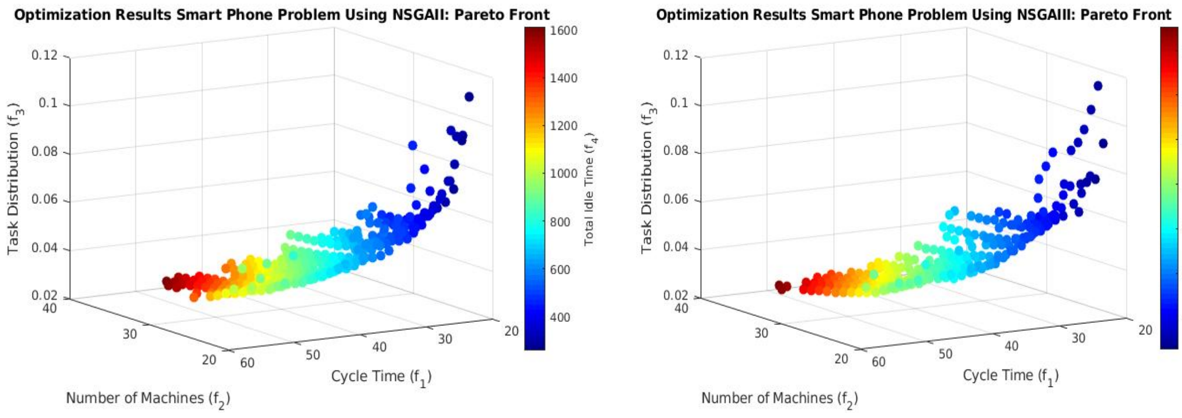


Figure 6.25: Pareto front for the many-objective SmartPhone problem.

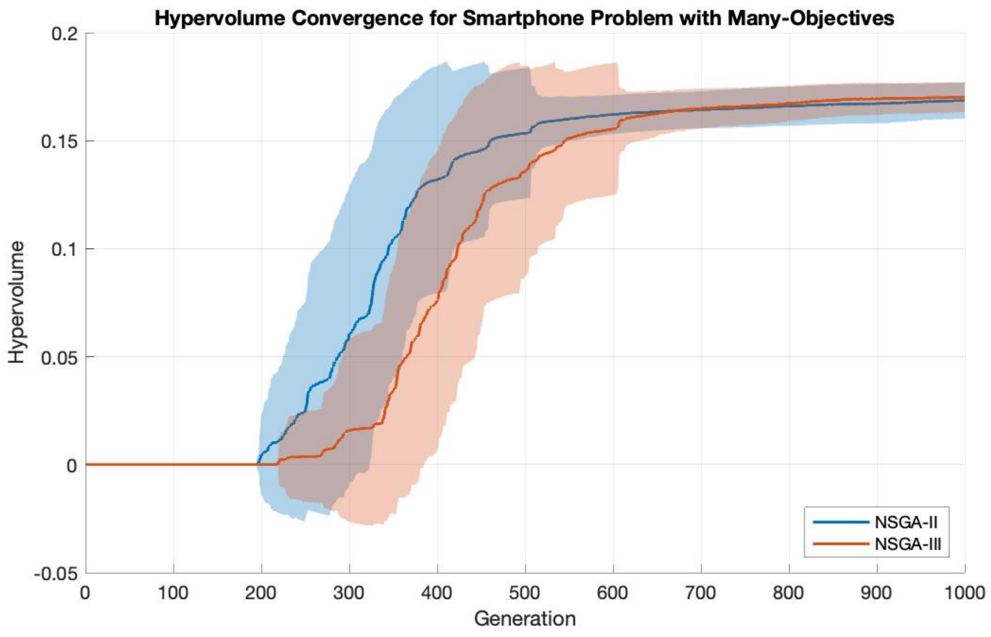


Figure 6.26: Hypervolume convergence of the SmartPhone problem for both algorithms across all the generations and runs. Standard deviation is represented by the shaded area.

The table 6.22 shows that there was a statistical significant difference between the results of NSGA-II and NSGA-III in terms of IGD and DR. However, the very high value for HV suggests that both algorithms performed very similar with respect to this metric.

P-Values for SmartPhone Manufacturing Problem		
P_HV	P_IGD	P_DR
0.5306	0.000001	0.0343

Table 6.22: Shows statistical significance between final generations of NSGA-II and NSGA-III for the SmartPhone problem (many-objective case).

Average and median values for HV, IGD and DR are shown in the table 6.23. On average, NSGA-II has outperformed NSGA-III significantly, especially in terms of IGD. But also the achieved DR is very superior to NSGA-III.

Mean Performance Indicators			
Algorithm	Hypervolume	Inverted Generational Distance	Dominance Ratios
NSGA-II	0.1685	8.8476	0.3544
NSGA-III	0.1701	26.0437	0.2127
Median Performance Indicators			
NSGA-II	0.1690	7.2906	0.3653
NSGA-III	0.1699	26.1232	0.1842

Table 6.23: Median and average values of performance indicators of the last generations for the many-objective SmartPhone problem.

6.5.3 Overall Results

The following table 6.24 and table 6.25 summarize overall performance of both algorithms across all the problems for their final generations, based on the achieved mean values of the performance indicators and the statistical significance of these values. Grey colours indicate that there was no statistical significant difference, green indicates that NSGA-II performed better, blue indicates that NSGA-III performed better. The table 6.24 shows, that there was mostly no statistically significant difference between the algorithms for multi-objective problems. In the cases where there was a difference, each algorithm is represented equally. However, these tables don't capture the speed of convergence, in which case it was found that NSGA-II consistently outperformed NSGA-III, making NSGA-II favoured overall.

Problem	HV	IGD	DR
Bowman	⬜	⬜	⬜
Buxey	🟦	⬜	⬜
WeeMag	⬜	⬜	⬜
Mixed-Model Camera	⬜	⬜	⬜
Mixed-Model Smatphone	⬜	⬜	🟩

Table 6.24: Summarizing the results for multi-objective instances. Showing which algorithm, on average, performed better using mean values. *NSGA-II has been denoted by green colour and NSGA-III has been denoted by blue color. Results without statistical significance are grey.

The table 6.25 displays similar information, but for the many-objective problems. In these cases it can be clearly seen that NSGA-II has performed better overall than NSGA-III, although for many problems the two still performed very similarly. Notably, the differences are especially relevant for the harder to solve and larger problems. Even though NSGA-III is designed for many-objective problems Deb and Jain [2013], it was not superior to NSGA-II for these problems. Like in the multi-objective case, this table doesn't capture the convergence speed, which was consistently superior for NSGA-II, making it the clearly superior choice for these many-objective problems.

Problem	HV	IGD	DR
Bowman	⬜	⬜	⬜
Buxey	⬜	⬜	⬜
WeeMag	🟩	🟩	🟩
Mixed-Model Camera	⬜	⬜	⬜
Mixed-Model Smartphone	⬜	🟩	🟩

Table 6.25: Summarizing the results for many-objective instances. Showing which algorithm on average performed better using mean values. *NSGA-II has been denoted by green colour and NSGA-III has been denoted by blue colour. Results without statistical significance are grey.

The solutions best demonstrating the diversity for each problem have been selected and explained in form of bar graphs. For the visualization of the found solutions and to understand what each objective really means, the reader can refer to the appendix.

The findings and analysis of results for the considered assembly line problems show the multi-modal nature of the multi-objective mixed-model assembly line problem. See table A.1 in the appendix, where several unique decision variables resulted in the same objective values.

7. Conclusion

Assembly lines are a common manufacturing structure, a crucial aspect and a necessity in most of the industries. It is often used to produce products fast and efficiently. Due to the rising competition of today's market, businesses want to ramp up production versatility by lowering batch sizes and diversifying their products. This increase in customization and production rate resulted in multi-objective mixed-model assembly line balancing problems. Assembly line balancing problems are under discussion for several decades and there is plenty of research that has been done. A lot of heuristics and exact methods have been developed to solve the challenges of assembly line balancing. However, most of the literature and work is based on simple assembly lines that are not a true depiction of real world problems. Moreover, the complex forms of assembly line problems that exist today because of vast production, cannot be effectively addressed by traditional exact and heuristic methods. In recent years, there has been a noticeable advancement in the direction of meta-heuristics and general assembly line problems. Still, a lot of work needs to be done. Since, to the best of our knowledge, there exists no standard benchmark for general assembly line problems and most of the research papers are unique in their considerations of the type of assembly line problem, challenges and chosen assembly line structure, comparison among the used methodologies is not possible. Hence, it is hard to make decisions on which meta-heuristics are best suited for these problems. It is to be expected that the future research on this topic will more be focused on the use of meta-heuristics and general assembly line problems.

This study makes a contribution to filling this gap in the literature. Mixed-model assembly lines with station restrictions have been considered, and evolutionary optimization techniques have been used to balance these assembly lines. In order to have a high production rate, while making the process time and cost-effective, multiple objectives have to be considered simultaneously. In this thesis, four objectives have been considered, including the minimization of *cycle time*, *number of workstations*, *smooth task distribution* and *total idle time*. This study addresses two main challenges, modelling of the mixed-model assembly line as a single-model assembly line and optimization of the assembly process to make it cost and time efficient, while respecting station restrictions.

The option mix approach has been used to tackle the challenge of converting mixed-model assembly lines into single-model assembly lines with the help of a joint precedence graph. For optimization, multi-objective evolutionary algorithms, NSGA-II and NSGA-III have been selected. Several problems are taken into consideration for experimentation, including three famous benchmark problems (Bowman, Buxey and WeeMag) alongside two mixed-model assembly line problems. Experiments have shown that the proposed methodology and both algorithms can produce promising results. Both algorithms were able to solve the benchmarks successfully. However, NSGA-II has outperformed NSGA-III overall, especially with respect to speed of convergence. Regarding final generation results, larger problems and many-objective instances were better handled by NSGA-II, while smaller problem instances had very little statistical significant differences between the algorithms.

The biggest challenge faced in this study was the lack of available benchmarks in the literature for multi-objective multi-model assembly line problems. Since this study is alone in regard to the combination of station restrictions, option mix joint precedence graph, and evolutionary algorithms, it cannot be compared directly against any other research paper, and neither have the proposed problems been implemented in any real world scenario.

The findings and analysis of results for the considered assembly line problems, show a multi-modal nature of the multi-objective mixed-model assembly line problem. Several unique decision variable vectors resulted in the same objective values. For this reason, other algorithms suitable for multi-modal problems can be expected to perform well and might be used to improve upon this work.

The option mix approach reduces the investigation to the options, which highly affects assembly line planning and puts less emphasis on the options that have less impact on the planning. However, because of its stochastic nature, the drawback of this technique includes overestimating or underestimating the task processing times. This can lead to waste of resources or violation of the cycle time constraints, respectively. Since the estimation of the joint precedence graph is highly dependent on sales data, any inconsistencies in the provided data can lead to inefficient assembly lines, which is a drawback to keep in mind. For high product variety, the option mix approach is nevertheless a good alternative to the traditional model mix approach. Although the presented work might be improved upon by finding a better solution to this problem.

Another future recommendation for this work is to consider model sequencing after the line balancing. Sequencing of the models requires a dispatching system that decides, based on several objectives, which model should be produced first. It's a way to prioritize the customer orders while using the assembly line efficiently. It is another critical problem that needs to be solved in order to make multi-model production processes efficient.

Appendix

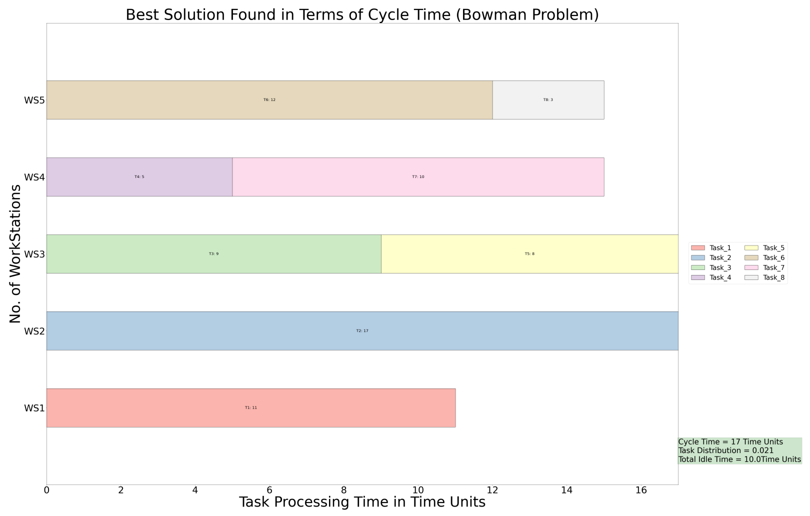


Figure A.1: Best Found Value of Cycle Time = 17.0 Time Units for 5 Workstations with Task Distribution and Total Idle time of 0.021 and 10.0 Time Units respectively.

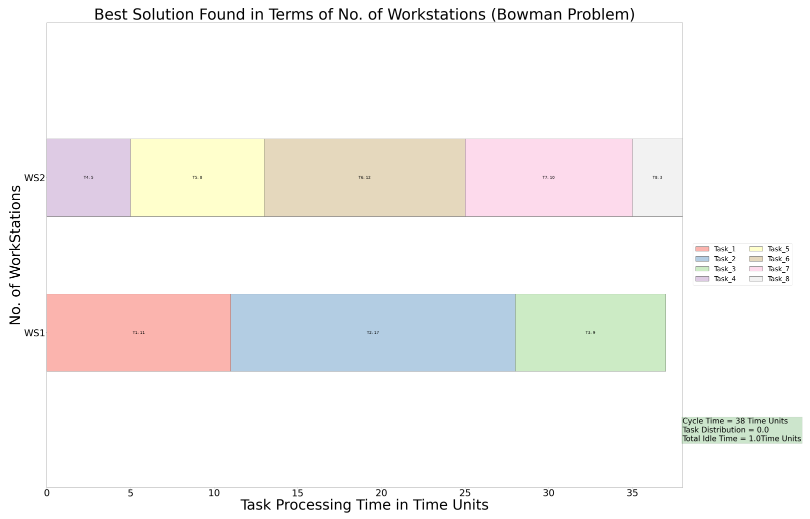


Figure A.2: Best Found Number of Workstations = 2 for Cycle Time = 38.0 Time Units with Task Distribution and Total Idle time of 0.0 and 1.0 Time Units respectively.

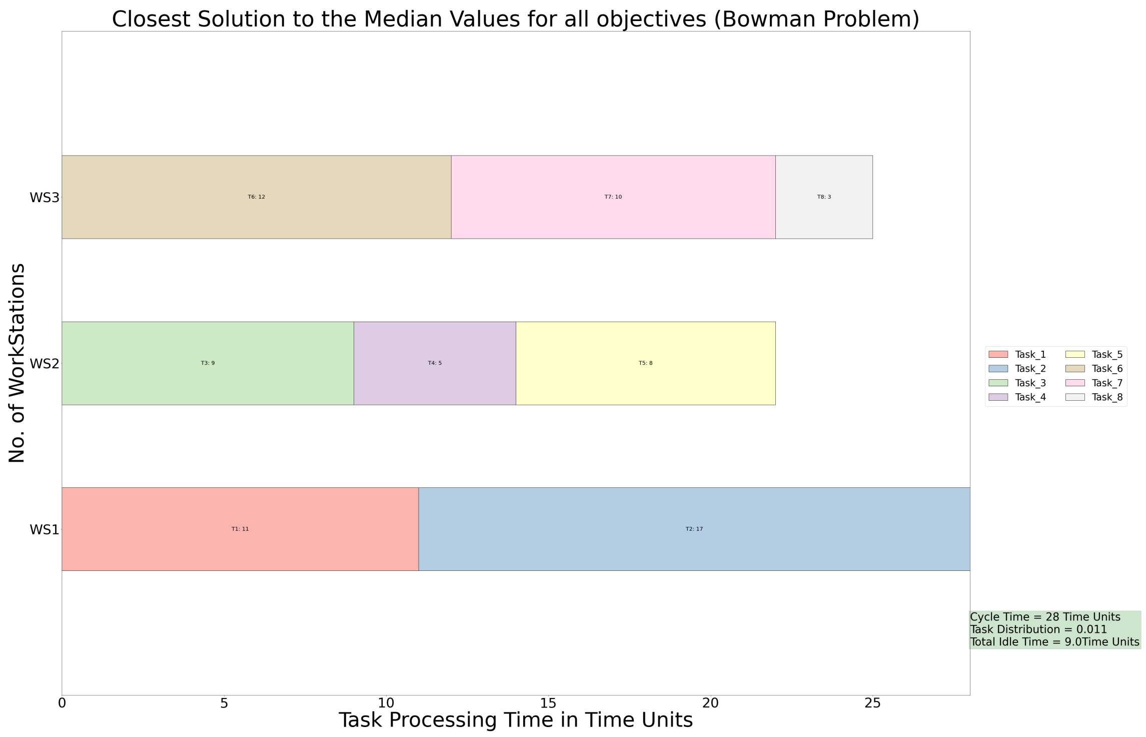


Figure A.3: These Objective Values are the Closest to the Median Values, show the best compromised values for all Objectives.

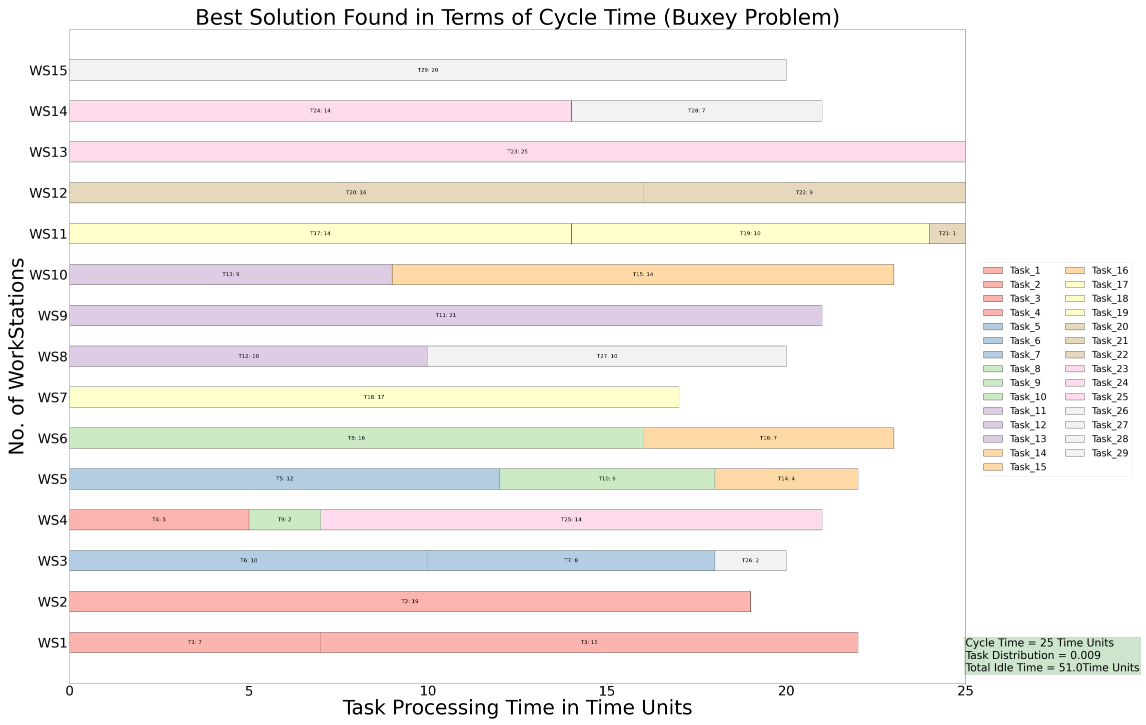


Figure A.4: Best Found Value of Cycle Time = 25.0 Time Units for 15 Workstations with Task Distribution and Total Idle time of 0.009 and 51.0 Time Units respectively.

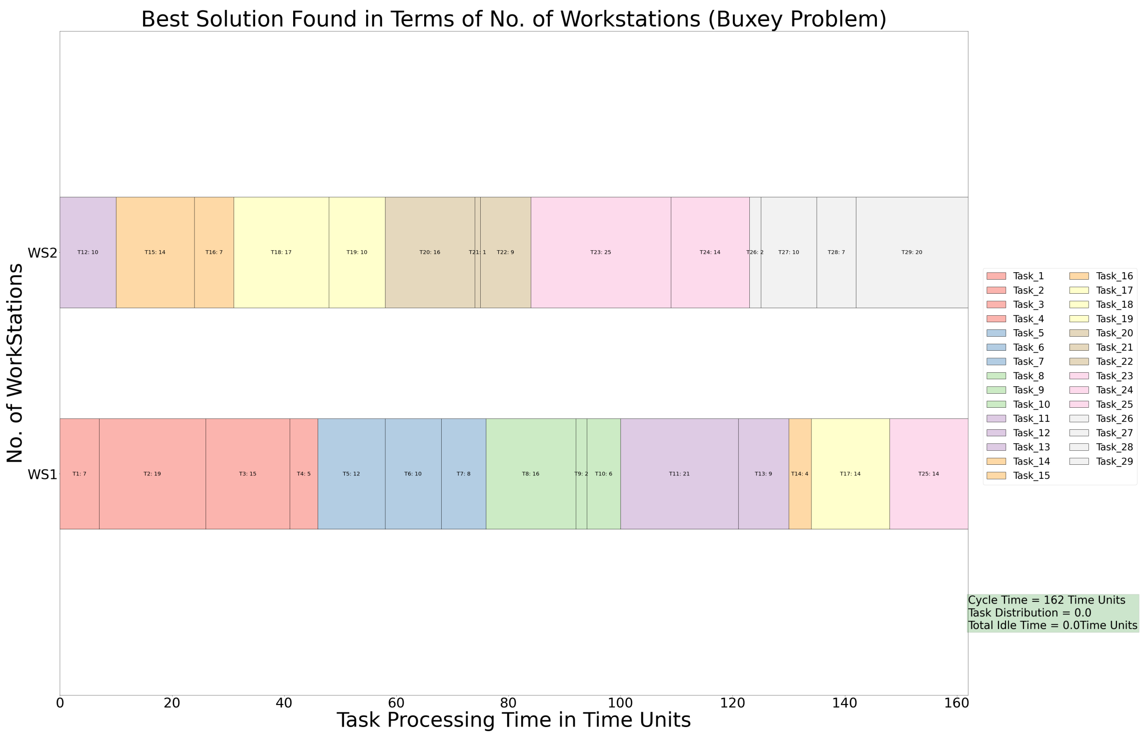


Figure A.5: Best Found Number of Workstations = 2 for Cycle Time = 162.0 Time Units with Task Distribution and Total Idle time of 0.0 and 0.0 Time Units respectively.

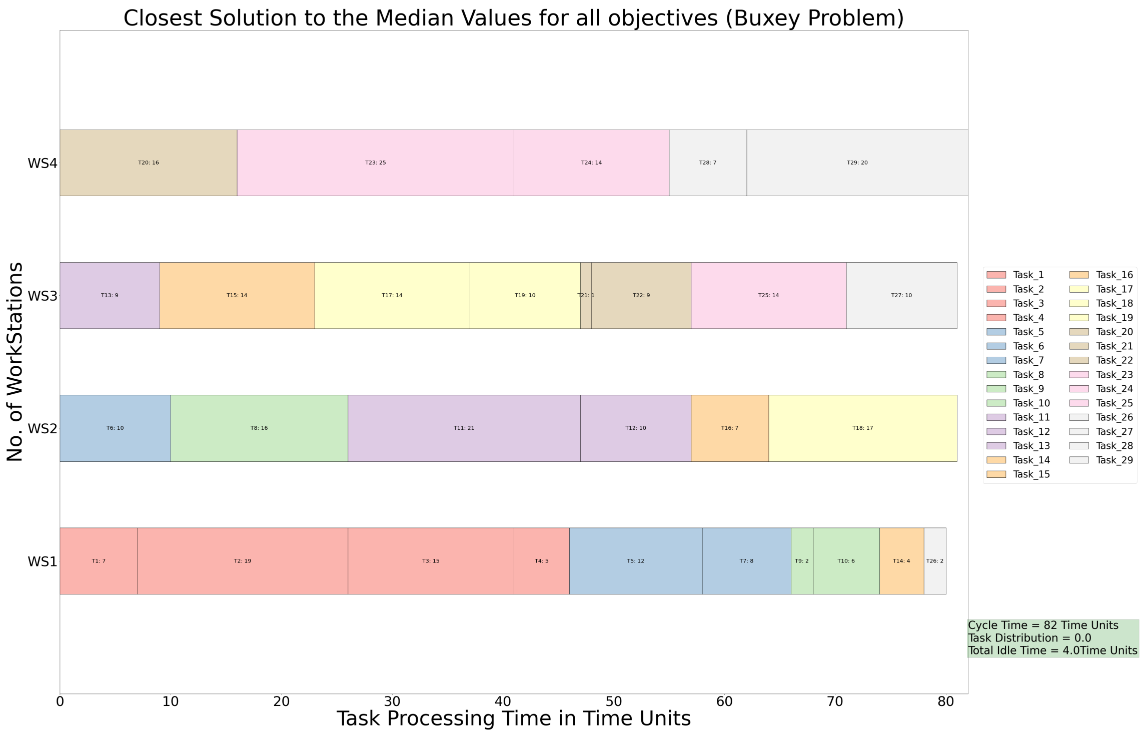


Figure A.6: Objective Values, Closest to the Median Values, shows the best compromised values for all Objectives.

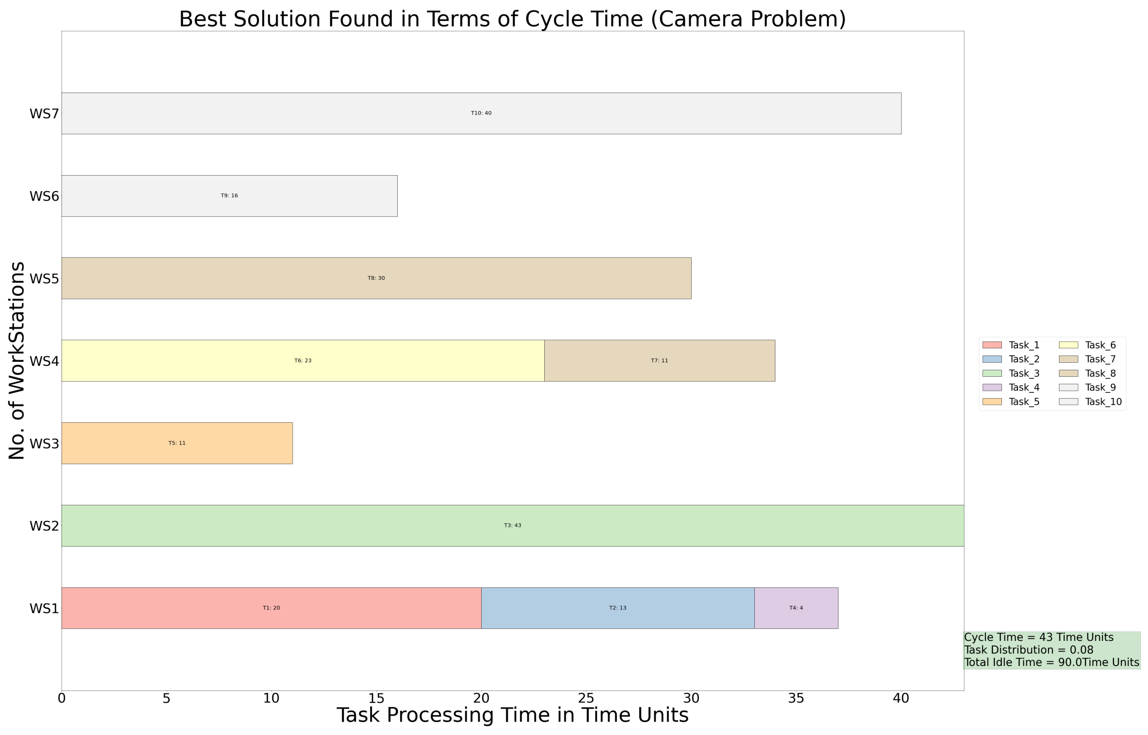


Figure A.7: Best Found Value of Cycle Time = 43.0 Time Units for 7 Workstations with Task Distribution and Total Idle time of 0.08 and 90.0 Time Units respectively.

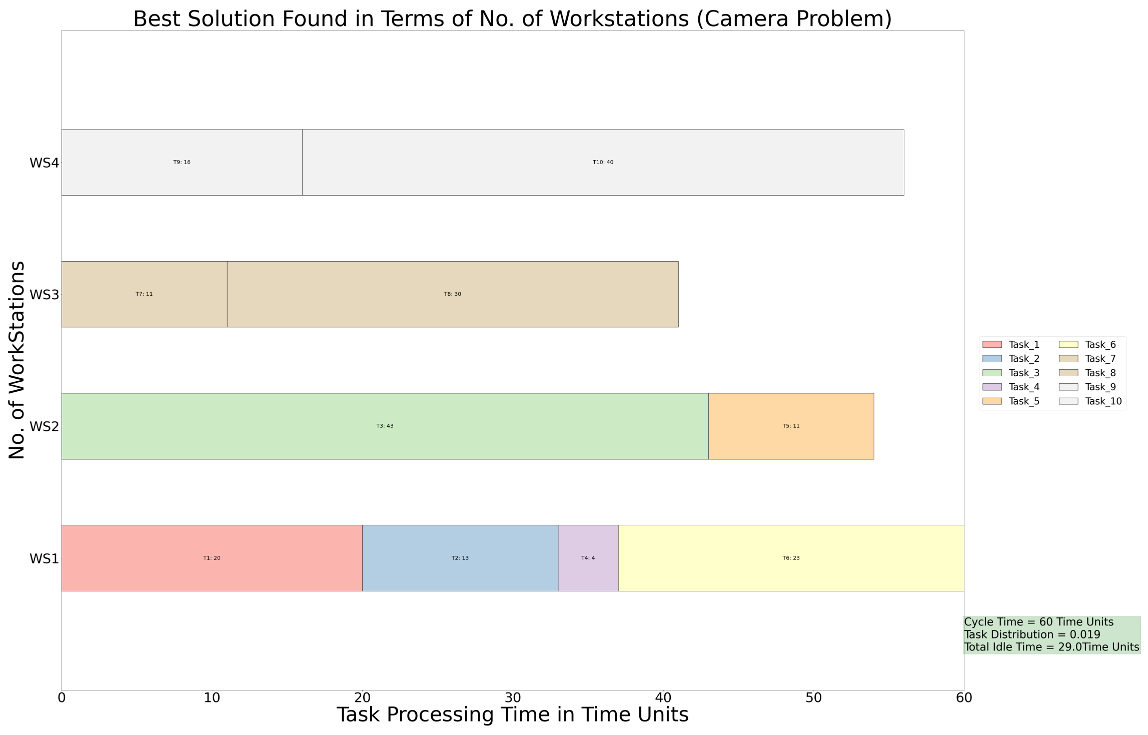


Figure A.8: Objective Values, Closest to the Median Values, shows the best compromised values for all Objectives.

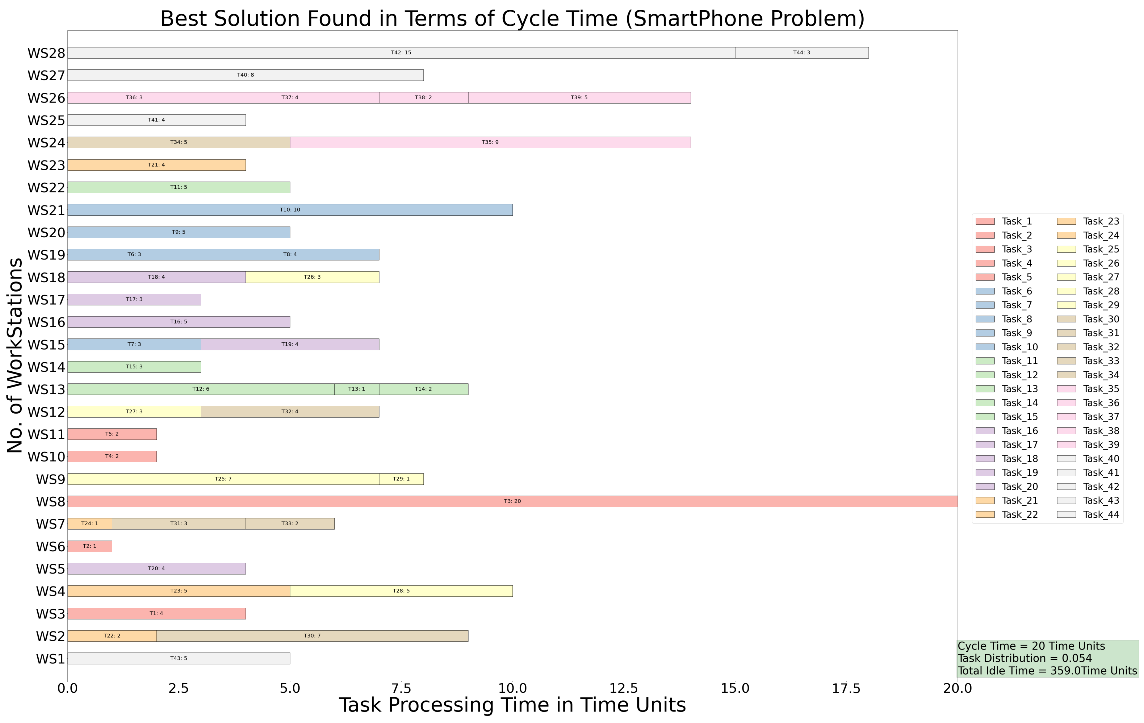


Figure A.9: Best Found Value of Cycle Time = 20.0 Time Units for 28 Workstations with Task Distribution and Total Idle time of 0.054 and 359.0 Time Units respectively.

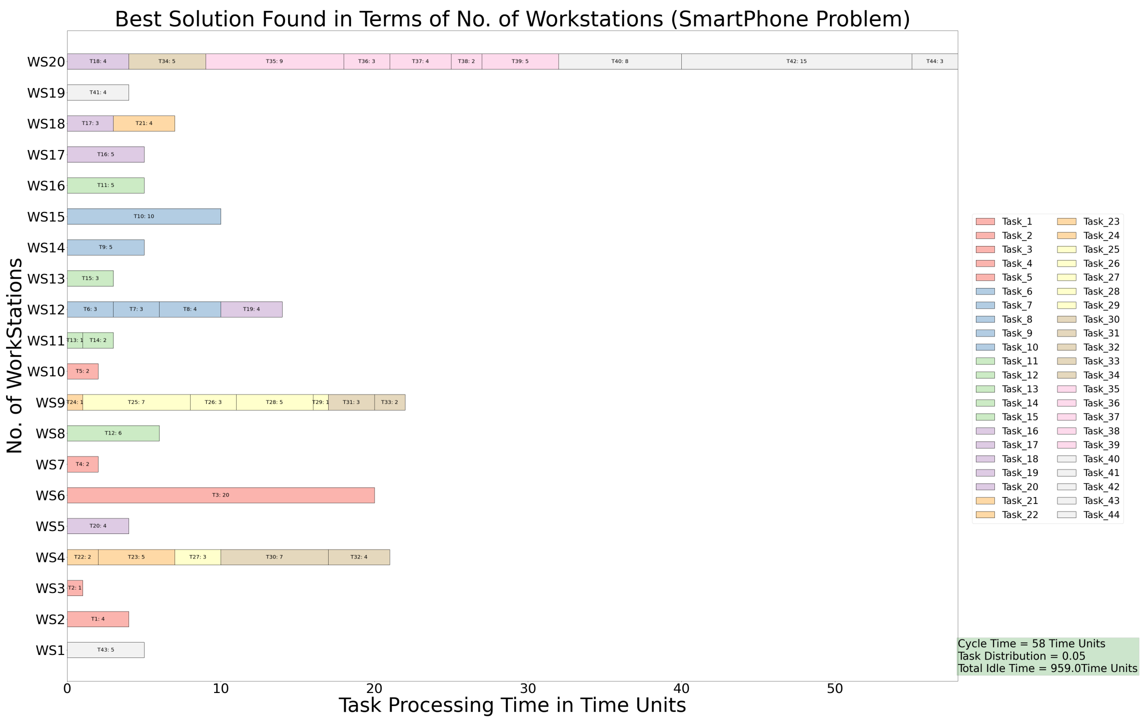


Figure A.10: Best Found Number of Workstations = 20 for Cycle Time = 58.0 Time Units with Task Distribution and Total Idle time of 0.005 and 959.0 Time Units respectively.

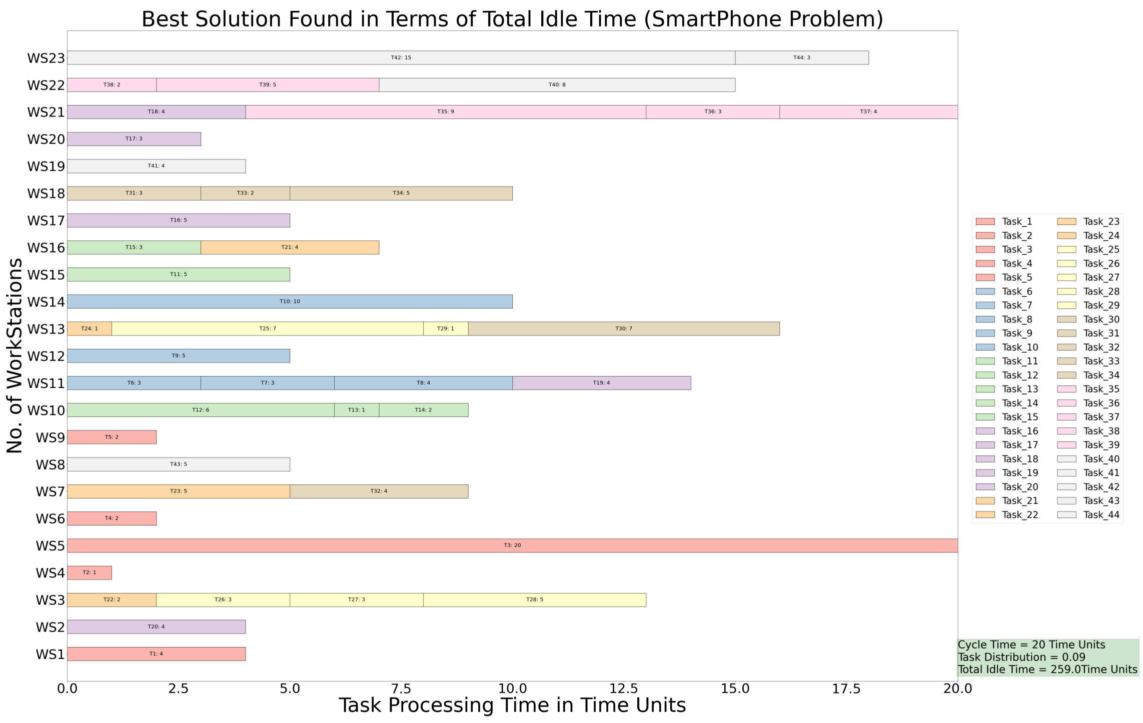


Figure A.11: Best Found Value for Total Idle Time = 259.0 Time Units for Cycle Time = 58.0 Time Units, Number of Workstations = 23 with Task Distribution of 0.09.

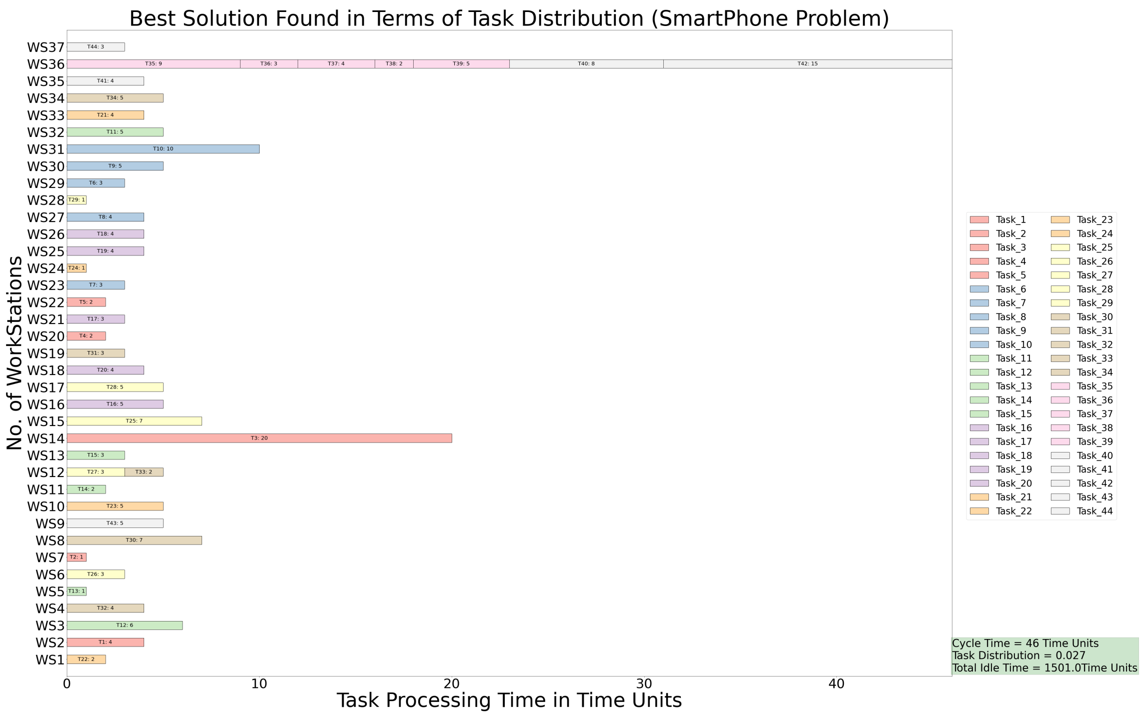


Figure A.12: Best Found Value for Variance in Task Distribution = 0.027 Time Units for Cycle Time = 46.0 Time Units, Number of Workstations = 37 with Total Idle Time of 1501.0 Time Units.

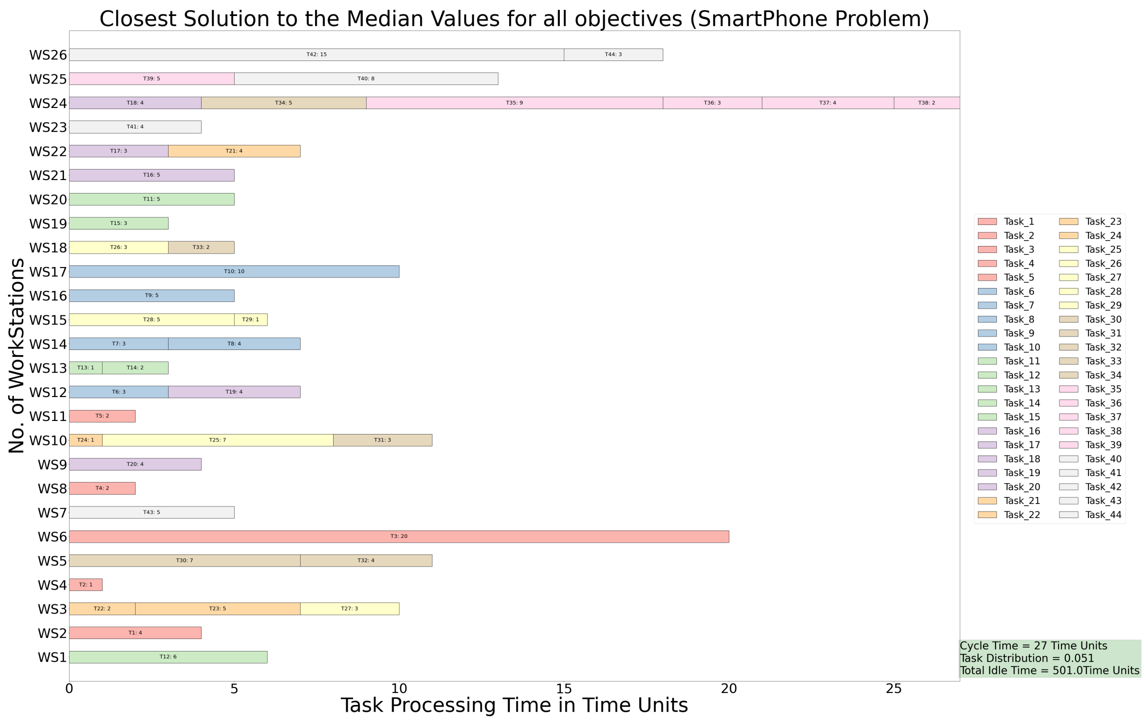


Figure A.13: Objective Values, Closest to the Median Values, shows the best compromised values for all Objectives.

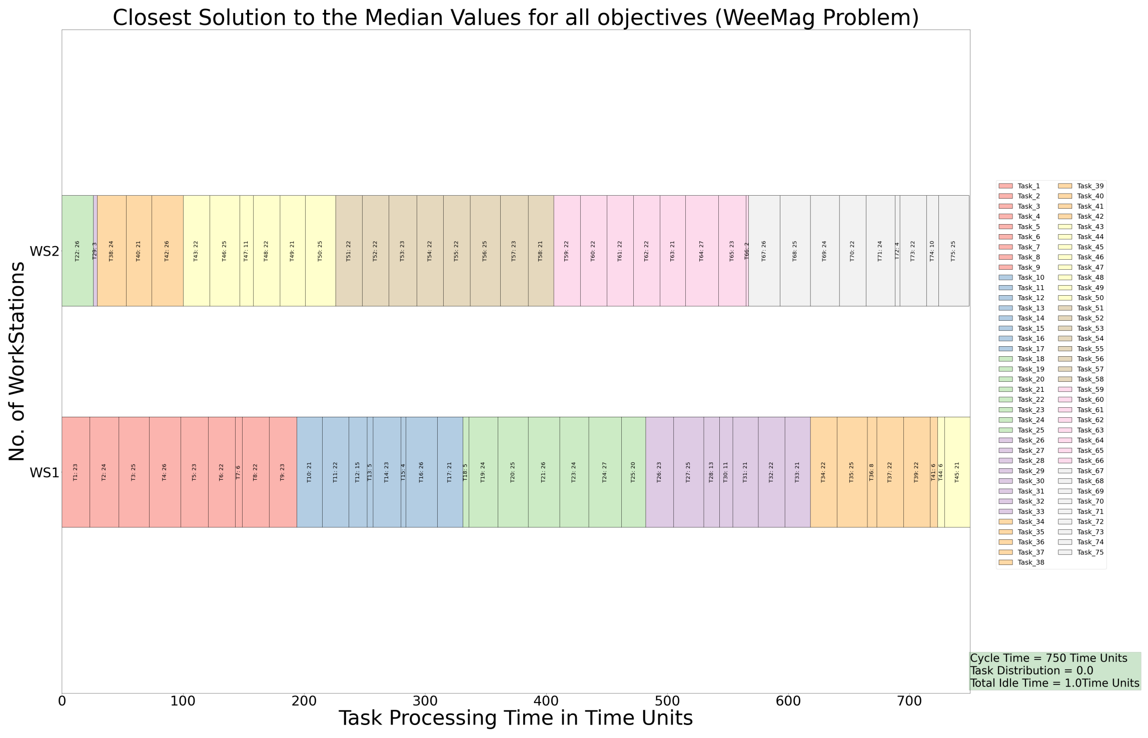


Figure A.14: Objective Values, Closest to the Median Values, shows the best compromised values for all Objectives.

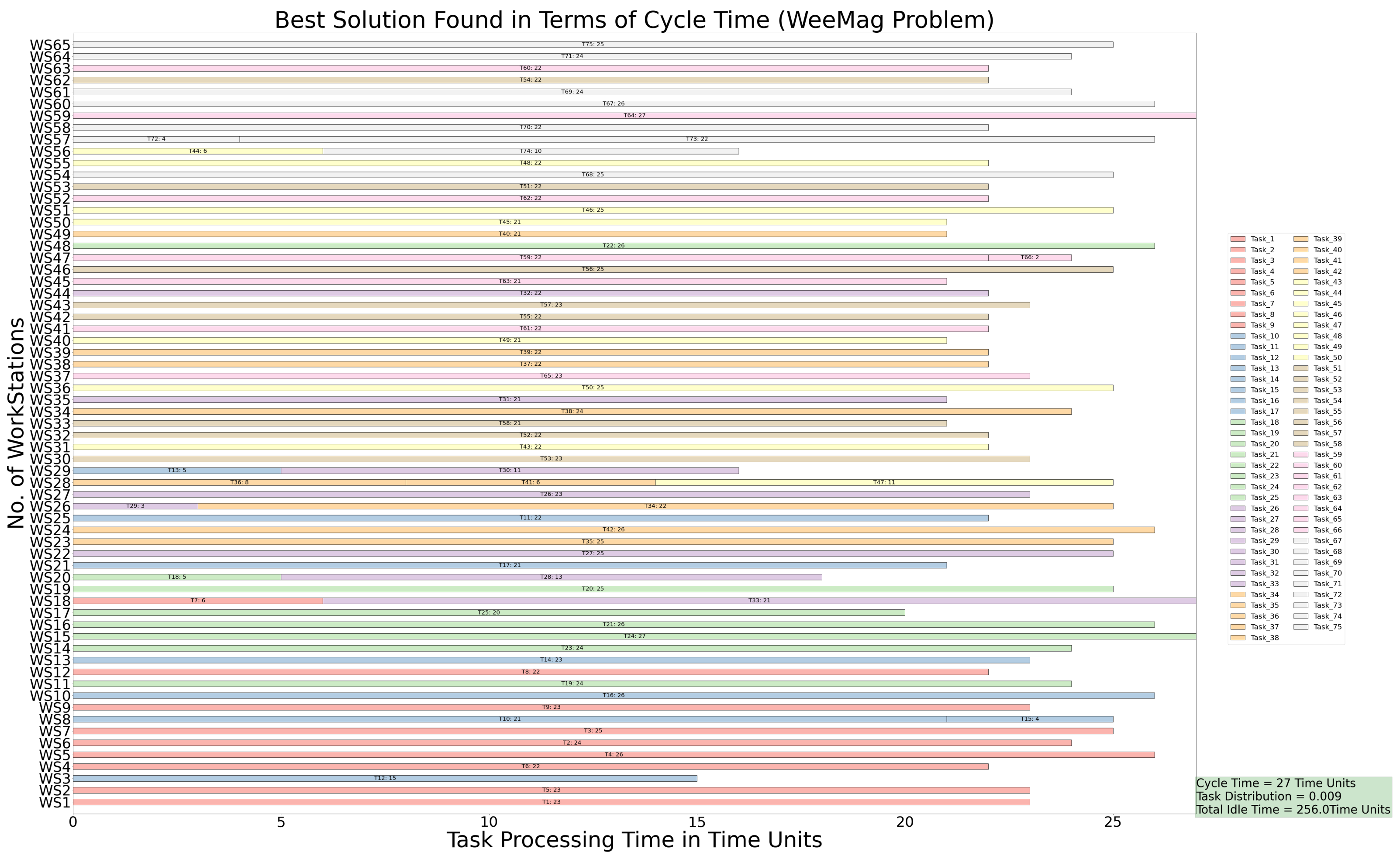


Figure A.15: Best Found Value of Cycle Time = 27.0 Time Units for 65 Workstations with Task Distribution and Total Idle time of 0.009 and 256.0 Time Units respectively.

Cycle Time	No. of Machines	Variance in Task Distribution	Total Idle Time	Corresponding Task Sequence
28	3	0.0114796	9	[1 2 4 3 5 6 7 8]
20	5	0.03125	25	[1 2 3 4 6 5 8 7]
38	2	0.0003463	1	[1 2 3 5 4 7 6 8]
28	3	0.0114796	9	[1 2 3 5 4 6 8 7]
26	4	0.0713757	29	[1 2 3 5 7 4 6 8]
22	4	0.0556129	13	[1 2 4 3 6 5 7 8]
38	2	0.0003463	1	[1 2 3 4 6 8 5 7]
28	3	0.0114796	9	[1 2 3 4 5 7 6 8]
38	2	0.0003463	1	[1 2 3 5 4 6 8 7]

Table A.1: Shows the Multi-Modal nature of the Multi-Objective Multi-Model Assembly Line Problem, the same Coloured Decision Variables Corresponds to the Same Objective Values.

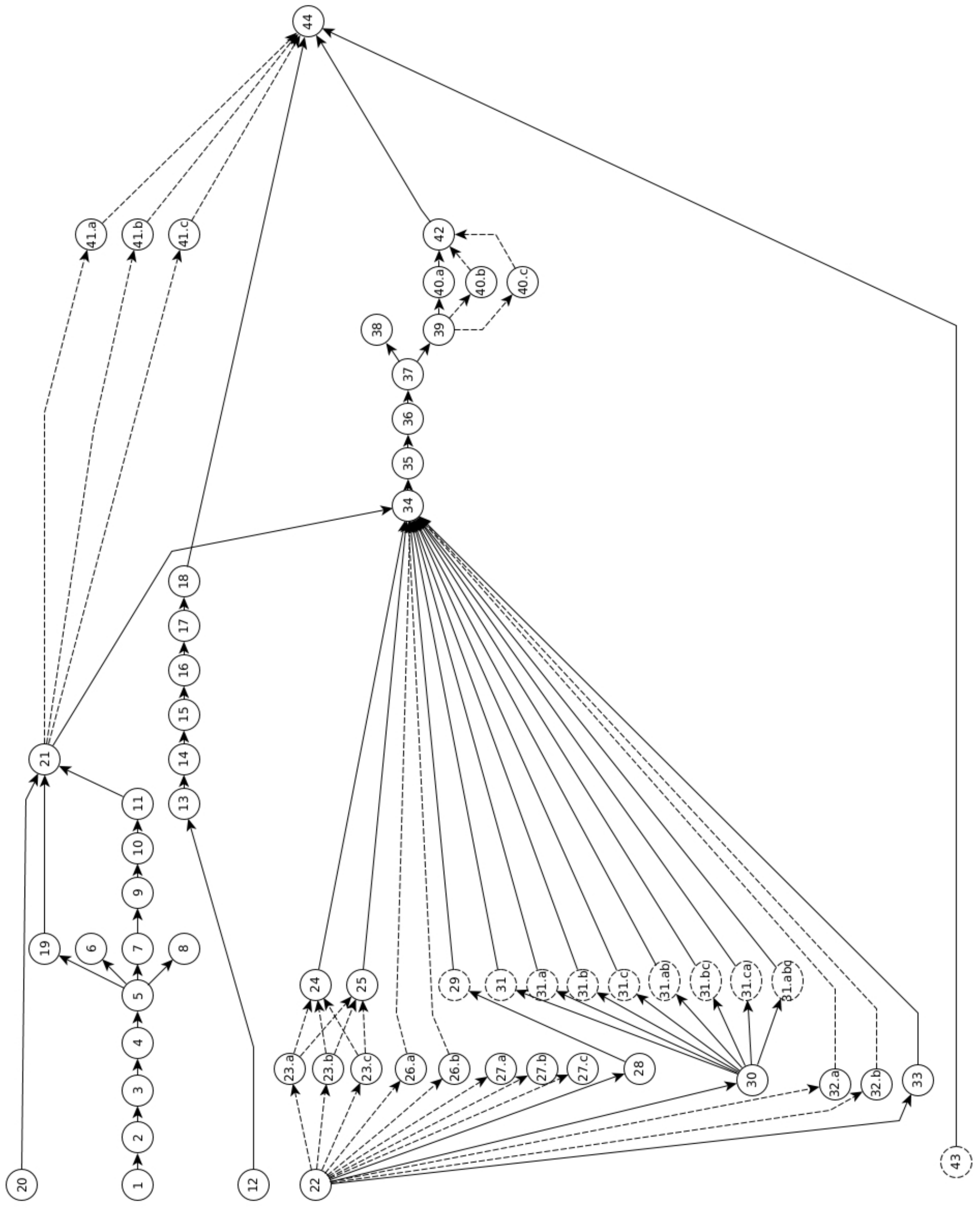


Figure A.16: SmartPhone Precedence Graph for all Possible Options

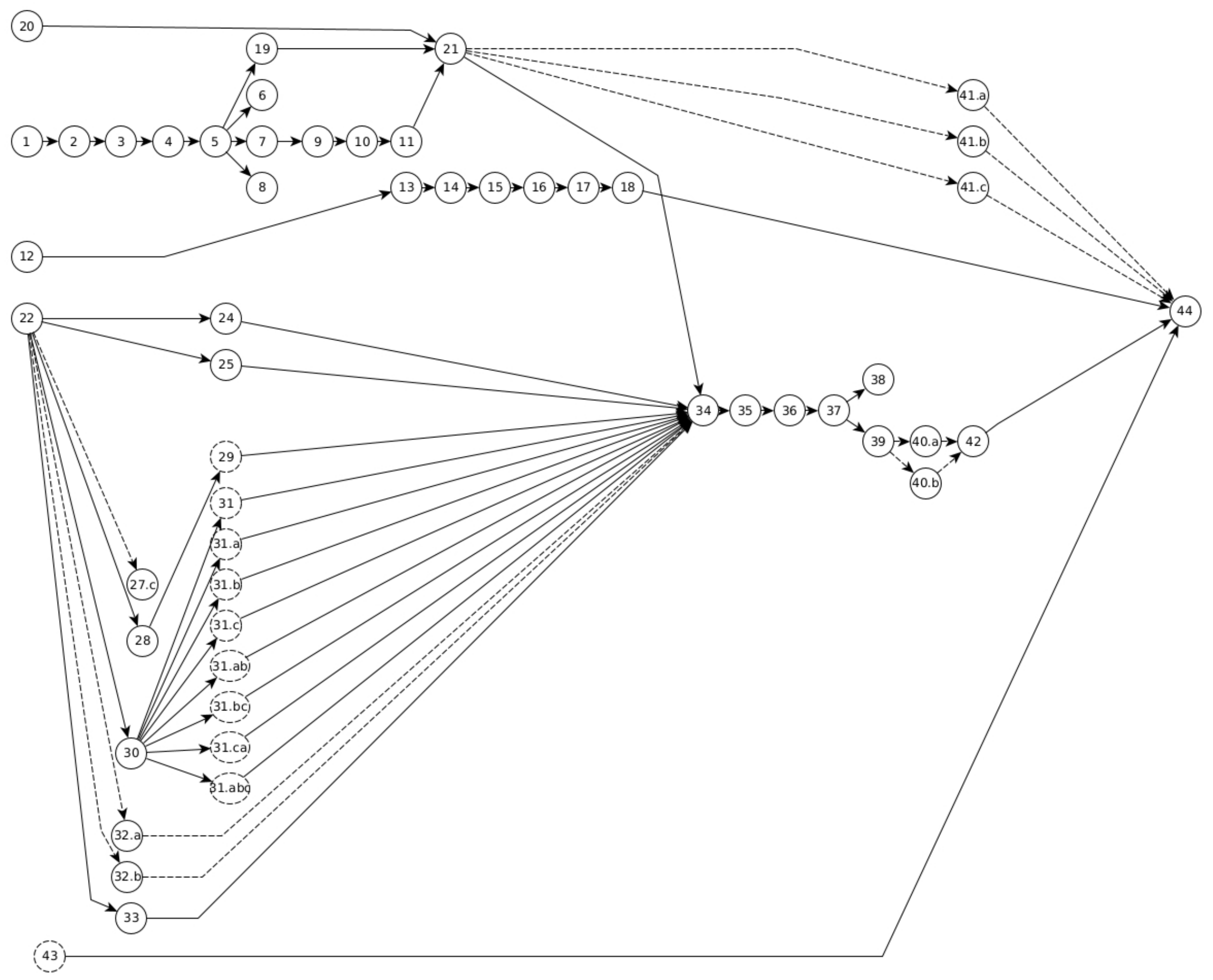


Figure A.17: SmartPhone Model1 Precedence Graph

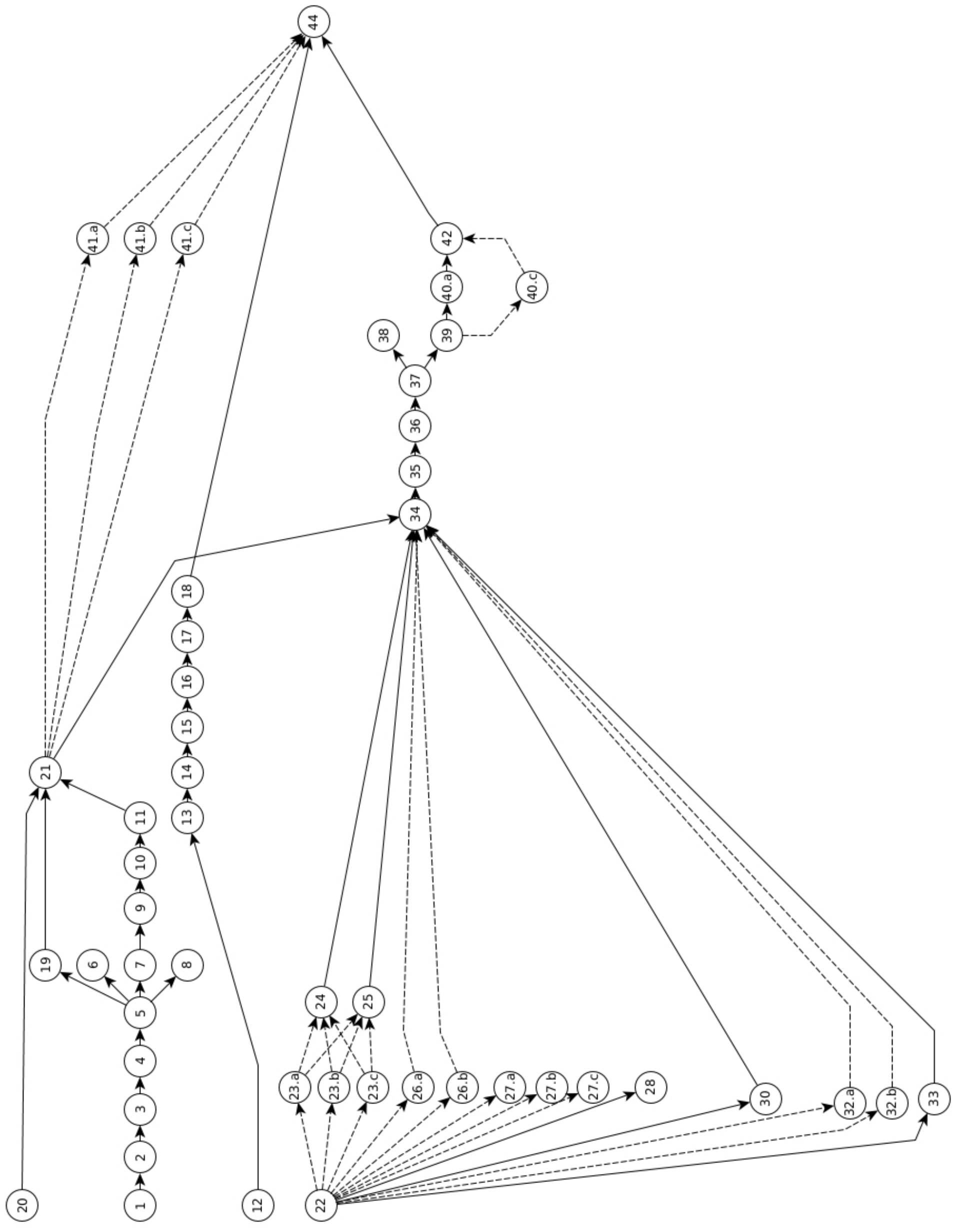


Figure A.18: SmartPhone Model2 Precedence Graph

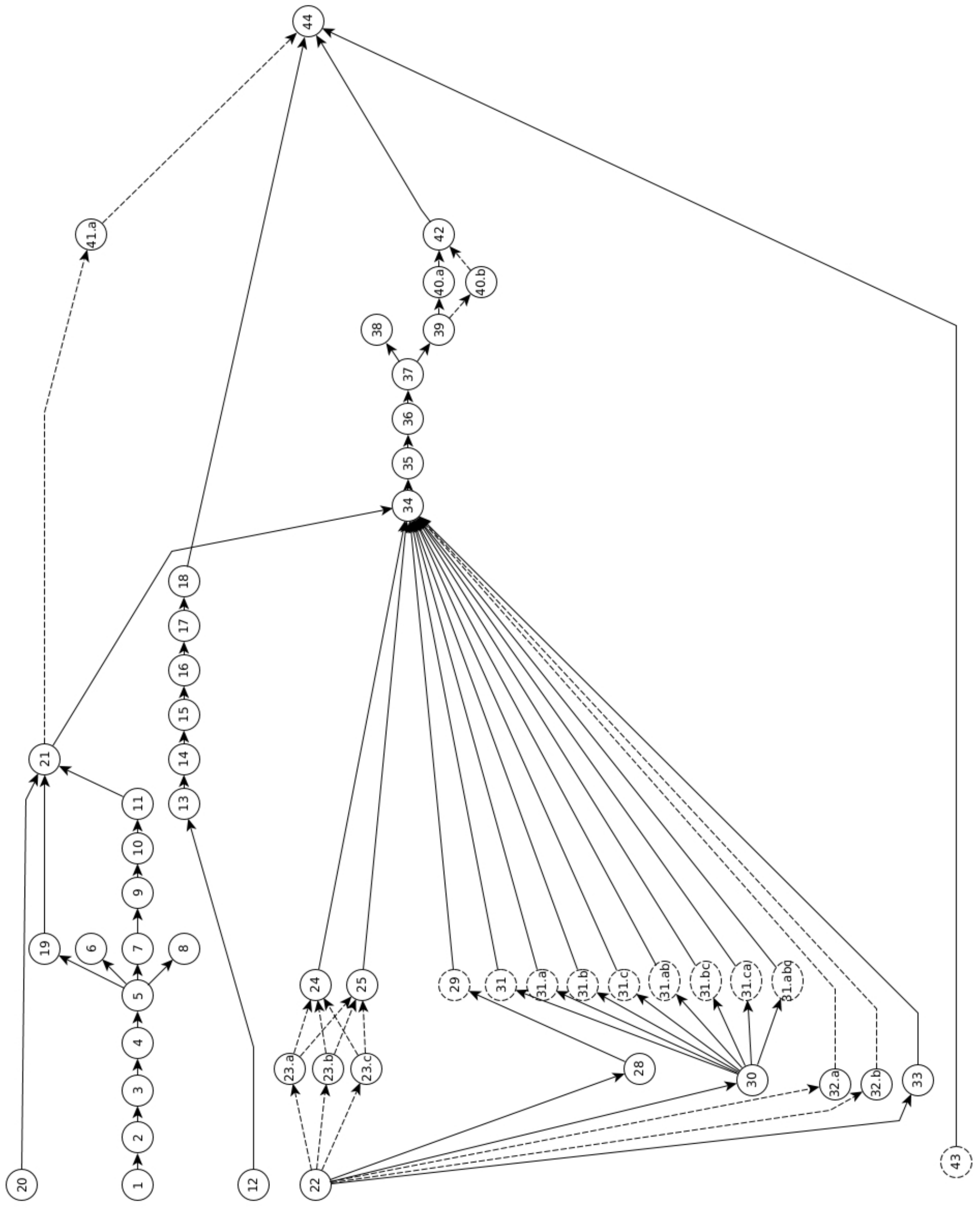


Figure A.19: SmartPhone Model3 Precedence Graph

Bibliography

Ajith Abraham and Lakhmi Jain. Evolutionary multiobjective optimization. In *Evolutionary multiobjective optimization*, pages 1–6. Springer, 2005. (cited on Page 12)

Kaveh Amouzgar. Multi-objective optimization using genetic algorithms, 2012. (cited on Page 5, 7, 9, 10, 11, 12, 13, and 37)

Shwetank Avikal, Rajeev Jain, PK Mishra, and HC Yadav. A heuristic approach for u-shaped assembly line balancing to improve labor productivity. *Computers & Industrial Engineering*, 64(4):895–901, 2013. (cited on Page 20)

Ilker Baybars. A survey of exact algorithms for the simple assembly line balancing problem. *Management science*, 32(8):909–932, 1986. (cited on Page 15, 16, 17, 18, and 19)

J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020. (cited on Page 39)

Julian Blank, Kalyanmoy Deb, and Proteek Chandan Roy. Investigating the normalization procedure of nsga-iii. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 229–240. Springer, 2019. (cited on Page 39)

Nils Boysen, Malte Fliedner, and Armin Scholl. Assembly line balancing: Joint precedence graphs under high product variety. *Iie Transactions*, 41(3):183–193, 2009. (cited on Page 3, 23, 33, 35, and 36)

Nils Boysen, Philipp Schulze, and Armin Scholl. Assembly line balancing: What happened in the last fifteen years? *European Journal of Operational Research*, 2021. (cited on Page 18)

RB Breginski, M Cleto, and JS Junior. Assembly line balancing using eight heuristics. In *22nd International Conference on Production Research*, 2013. (cited on Page 33)

Joseph Bukchin and Jacob Rubinovitz. A weighted approach for assembly line design with station paralleling and equipment selection. *IIE transactions*, 35(1):73–85, 2003. (cited on Page 21)

Kuang-Hua Chang. Chapter 19 - multiobjective optimization and advanced topics. In Kuang-Hua Chang, editor, *e-Design*, pages 1105–1173. Academic Press, Boston, 2015. ISBN 978-0-12-382038-9. doi: https://doi.org/10.1016/B978-0-12-382038-9.00019-3. URL https://www.sciencedirect.com/science/article/pii/B9780123820389000193. (cited on Page 7)

Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007. (cited on Page 12 and 13)

Kalyanmoy Deb. Multi-objective optimization using evolutionary algorithms. john wiley& sons. *Inc., New York, NY*, 2001. (cited on Page 5, 6, 9, 10, 11, 12, and 13)

Kalyanmoy Deb. *Multi-objective Optimization*, pages 403–449. Springer US, Boston, MA, 2014. ISBN 978-1-4614-6940-7. doi: 10.1007/978-1-4614-6940-7_15. URL https://doi.org/10.1007/978-1-4614-6940-7_15. (cited on Page 7 and 8)

Kalyanmoy Deb and Tushar Goel. Multi-objective evolutionary algorithms for engineering shape design. In *Evolutionary optimization*, pages 147–175. Springer, 2003. (cited on Page 8)

Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013. (cited on Page 38, 40, and 69)

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002. (cited on Page 13, 37, and 38)

Erdal Erel and Subhash C Sarin. A survey of the assembly line balancing procedures. *Production Planning & Control*, 9(5):414–434, 1998. (cited on Page 15 and 16)

Masood Fathi, Dalila Benedita Machado Martins Fontes, Matias Urenda Moris, and Morteza Ghobakhloo. Assembly line balancing problem: A comparative evaluation of heuristics and a computational assessment of objectives. *Journal of Modelling in Management*, 2018. (cited on Page 31)

Daniel J Fonseca, Matthew Elam, Charles L Karr, and CL Guest. A fuzzy logic approach to assembly line. *Mathware & soft computing, 2005, vol. 12, núm. 1*, 2005. (cited on Page 15)

Mitsuo Gen and Runwei Cheng. *Genetic Algoritms for Control and Engineering Design*. John Wiley & Sons. Inc., 1997. (cited on Page 7)

Hadi Gökçen, Kürşad Ağpak, and Recep Benzer. Balancing of parallel assembly lines. *International Journal of Production Economics*, 103(2):600–609, 2006. (cited on Page 21)

W Grzechca and LR Foulds. The assembly line balancing problem with task splitting: A case study. *IFAC-PapersOnLine*, 48(3):2002–2008, 2015. (cited on Page 19 and 25)

Waldemar Grzechca. *Assembly Line: Theory and Practice*. BoD–Books on Demand, 2011a. (cited on Page 1 and 2)

Waldemar Grzechca. Cycle time in assembly line balancing problem. In *2011 21st International Conference on Systems Engineering*, pages 171–174. IEEE, 2011b. (cited on Page 25)

Allan L Gutjahr and George L Nemhauser. An algorithm for the line balancing problem. *Management science*, 11(2):308–315, 1964. (cited on Page 2)

WB Halgeson and DP Birnie. Assembly line balancing using the ranked positional weighting technique. *Journal of Industrial Engineering*, 12(6):394–398, 1961. (cited on Page 16)

Randy L Haupt and Sue Ellen Haupt. *Practical genetic algorithms*. John Wiley & Sons, 2004. (cited on Page 5)

N Ismail, GR Esmaeilian, M Hamedi, and S Sulaiman. Balancing of parallel assembly lines with mixed-model product. In *International Conference on Management and Artificial Intelligence IPEDR*, volume 6, pages 120–124, 2011. (cited on Page 21)

Mahrokh Javadi and Sanaz Mostaghim. Using neighborhood-based density measures for multimodal multi-objective optimization. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 335–345. Springer, 2021. (cited on Page 9)

Edward PC Kao. A preference order dynamic program for stochastic assembly line balancing. *Management Science*, 22(10):1097–1104, 1976. (cited on Page 16)

Talip Kellegöz. Balancing lexicographic multi-objective assembly lines with multi-manned stations. *Mathematical problems in Engineering*, 2016, 2016. (cited on Page 32)

Joshua D Knowles and David W Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary computation*, 8(2): 149–172, 2000. (cited on Page 13)

Abdullah Konak, David W Coit, and Alice E Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability engineering & system safety*, 91 (9):992–1007, 2006. (cited on Page 13)

Tae Ok Lee, Yeongho Kim, and Yeo Keun Kim. Two-sided assembly line balancing to maximize work relatedness and slackness. *Computers & Industrial Engineering*, 40(3):273–292, 2001. (cited on Page 22)

JLC Macaskill. Production-line balances for mixed-model lines. *Management Science*, 19(4-part-1):423–434, 1972. (cited on Page 34)

Herbert Meyr. Supply chain planning in the german automotive industry. In *Supply Chain Planning*, pages 343–365. Springer, 2009. (cited on Page 3)

Kaisa Miettinen and Jussi Hakanen. Why use interactive multi-objective optimization in chemical process design? In *MULTI-OBJECTIVE OPTIMIZATION: Techniques and Application in Chemical Engineering*, pages 157–197. World Scientific, 2017. (cited on Page 6 and 7)

Kaisa Miettinen and Pekka Salminen. Decision-aid for discrete multiple criteria decision making problems with imprecise data. *European Journal of Operational Research*, 119(1):50–60, 1999. (cited on Page 6)

John Miltenburg. Balancing u-lines in a multiple u-line facility. *European journal of operational research*, 109(1):1–23, 1998. (cited on Page 20)

John Miltenburg. U-shaped production lines: A review of theory and practice. *International Journal of Production Economics*, 70(3):201–214, 2001. (cited on Page 20)

Ashkan Mozdgir, Iraj Mahdavi, Iman Seyedi Badeleh, and Maghsud Solimanpur. Using the taguchi method to optimize the differential evolution algorithm parameters for minimizing the workload smoothness index in simple assembly line balancing. *Mathematical and Computer Modelling*, 57(1-2):137–151, 2013. (cited on Page 32)

Narasimha R Nagaiah and Christopher D Geiger. Application of evolutionary algorithms to optimize cooling channels. *International Journal for Simulation and Multidisciplinary Design Optimization*, 10:A4, 2019. (cited on Page 9)

Koichi Nakade and Rei Nishiwaki. Optimal allocation of heterogeneous workers in a u-shaped production line. *Computers & Industrial Engineering*, 54(3):432–440, 2008. (cited on Page 20)

A Nourmohammadi and M Zandieh. Assembly line balancing by a new multi-objective differential evolution algorithm based on topsis. *International Journal of Production Research*, 49(10):2833–2855, 2011. (cited on Page 32)

Amir Nourmohammadi, Masood Fathi, and Amos HC Ng. Choosing efficient meta-heuristics to solve the assembly line balancing problem: A landscape analysis approach. *Procedia CIRP*, 81:1248–1253, 2019. (cited on Page 24)

Hongbo Ren, Yinlong Lu, Qiong Wu, Xiu Yang, and Aolin Zhou. Multi-objective optimization of a hybrid distributed energy system using nsga-ii algorithm. *Frontiers in Energy*, 12(4):518–528, 2018. (cited on Page 38)

Ihsan Sabuncuoglu, Erdal Erel, and M Tanyer. Assembly line balancing using genetic algorithms. *Journal of intelligent manufacturing*, 11(3):295–310, 2000. (cited on Page 40)

Melvin E Salveson. The assembly line balancing problem. *The Journal of Industrial Engineering*, pages 18–25, 1955. (cited on Page 16)

A Scholl. Data of assembly line balancing problems, schriften zur quantitativen betriebswirtschaftslehre 16/93. *Copyright of Applied Mechanics & Materials is the property of Trans Tech Publications, Ltd and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use*, 1993. (cited on Page 43)

Armin Scholl and Christian Becker. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168(3):666–693, 2006. (cited on Page 16, 17, 22, 23, and 24)

Armin Scholl and Armin Scholl. *Balancing and sequencing of assembly lines*. Springer, 1999. (cited on Page 17)

Thomas Seidelmann, Jens Weise, and Sanaz Mostaghim. Meeting demands for mass customization: A hybrid organic computing approach. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2021. doi: 10.1109/SSCI50451.2021.9659946. (cited on Page 43)

RJ Sury. Aspects of assembly line balancing. *International Journal of Production Research*, 9(4):501–512, 1971. (cited on Page 15)

Nick T Thomopoulos. Mixed model line balancing with smoothed station assignments. *Management science*, 16(9):593–603, 1970. (cited on Page 33)

Mohammad Kamal Uddin and Jose Luis Martinez Lastra. Assembly line balancing and sequencing. *Assembly Line–Theory and Practice*, pages 13–36, 2011. (cited on Page 16, 43, 56, and 57)

Jannet I Van Zante-de Fokkert and Ton G de Kok. The mixed and multi model line balancing problem: a comparison. *European Journal of Operational Research*, 100 (3):399–412, 1997. (cited on Page 34)

Grzechca Waldemar. Final results of assembly line balancing problem. *Assembly Line–Theory and Practice*, 2011. (cited on Page 2, 15, 16, and 30)

Eckart Zitzler and Lothar Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. *TIK-report*, 43, 1998. (cited on Page 13)

Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001. (cited on Page 13)

I herewith assure that I wrote the present thesis independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or to their meaning.

[illegible]

Magdeburg, 5th October 2022