Robust design and reconfiguration planning of mixed-model assembly lines under uncertain evolutions of product family

Yosra Mezghani, S. Ehsan Hashemi-Petroodi, Simon Thevenin, Alexandre Dolgui

Robust design and reconfiguration planning of mixed-model assembly lines under uncertain evolutions of product family

Yosra Mezghani[a], S. Ehsan Hashemi-Petroodi[b,*], Simon Thevenin[a], Alexandre Dolgui[a]

[a]*IMT Atlantique, LS2N - CNRS, 4 rue Alfred Kastler, Nantes, France*
[b]*KEDGE Business School Campus Bordeaux, 33405 Talence, France*
(e-mail: yosra.mezghani@etudiant-enit.utm.tn - seyyed-ehsan.hashemi-petroodi@kedgebs.com - simon.thevenin@imt-atlantique.fr - alexandre.dolgui@imt-atlantique.fr)

Abstract

Assembly lines commonly run for dozens of years before being decommissioned. As product families may evolve several times per year by following the needs of sales and marketing, process engineers reconfigure the lines several dozens of times throughout their life cycle. If the line is not flexible enough, these reconfigurations may be costly, and they can lead to poor efficiency. The present work investigates the possibility of designing a line while accounting for product evolution throughout the life cycle of the line. The evolution of the product family is unknown and we consider a robust optimization approach. We study a mixed-model assembly line, where each station contains a worker/robot and its equipment. The line produces different product models from the same family, and a reconfiguration occurs when a new product model replaces one of the current variants in the product family. Reconfiguration re-arranges resources and equipment pieces, and it can re-assign some tasks. In this study, we formulate a novel Mixed-Integer Linear Programming ($MILP$) that minimizes the total cost of the initial design and future reconfigurations of the line over some future product family evolution for the worst case. We consider the worst-case among different scenarios

*Corresponding author
Email address: seyyed-ehsan.hashemi-petroodi@imt-atlantique.fr (S. Ehsan Hashemi-Petroodi)

that represent possible production requirements of the new product model. An adversarial approach is also developed to solve large-size instances. We perform computational experiments on the benchmark data from the literature. The results show the proposed adversarial approach performs well, and the proposed robust model significantly reduces the design and reconfiguration costs when compared to the classical approach that designs and reconfigures by accounting only for the current product family.
Keywords: Robust optimization, Mixed model assembly line, Reconfigurability, Product family evolutions, Adversarial approach

1. Introduction

Nowadays, manufacturing companies are facing an increasing demand for product customization, and high-frequency market changes (Schuh et al., 2017; Rahman, 2020). As a result, assembly lines evolve frequently throughout their life cycle to adapt to changes in customer requirements, product requirements, and processing technologies (AlGeddawy and ElMaraghy, 2012). In this context, companies must create assembly lines that are easy to upgrade, where the integration of new technologies or functions is seamless (Molina et al., 2005). In particular, such manufacturing systems must handle quick changes caused by the frequent introduction of new product variants.

This work is aligned to the ASSISTANT (LeArning and robuSt deciSIon SupporT systems for agile mANufacTuring environments) project funded by the European Commission which aims to create an Artificial Intelligence (AI)-based software by developing digital twins for management and decision aid of the production planning and control of adaptive manufacturing environments (Castañé et al., 2022). The project involves several academic and industrial partners, such as producers of automotive and industrial equipment. Discussions with our industrial partners in the project highlighted the importance of our current work in practice and that changes in product variants are a challenge for production line design-

ers. In such a sense, a small change in a product may have a large impact on the structure and efficiency of an assembly line. In this paper, we investigate the possibility of foreseeing changes in product variants in the design of an assembly, and we evaluate the impact of prescriptive actions to smooth the reconfigurations. The objective is to plan the evolution of the manufacturing systems to deal with the changing requirements. For instance, if the manufacturer foresees that the workload to assemble a specific part may increase, it might be worth leaving an empty station in the line. This empty station can perform additional tasks associated with the new product variant without disturbing other stations.

Our work also falls in the line of research related to Reconfigurable Manufacturing Systems (RMS). Koren et al. (1999) introduced the RMS paradigm to efficiently cope with market changes, customized products, and volatile demand (Singh et al., 2017; Khettabi et al., 2021; Wang and Koren, 2012). One of the properties of RMS is their ability to evolve with the product families cost-effectively and efficiently (Koren et al., 1999). However, the design of an RMS able to reconfigure to face all product variants in a family remains a challenge (e.g., Altemeier et al., 2010). As optimization models for the design of manufacturing and assembly lines are similar, the present work is also applicable in the context of RMS.

We focus on the reconfiguration planning of a mixed-model assembly line (MMAL) for several production generations ahead. Each production generation corresponds to product evolutions, which occur every 6-12 months in today's unstable manufacturing environment. In each period, process engineers configure the line to produce a given product family, and reconfigure it at the beginning of each generation to account for new product models in the family. As mentioned, the problem arises from a real-world automotive manufacturing company where an assembly line is organized using human workers and robots at stations. As mentioned, this company is one of the partners in our common project ASSISTANT.

Our work seeks to answer the following research questions:

1. Does modeling uncertain product evolution in an assembly line design optimization model yield more robust and resilient assembly lines?

2. What is the impact of the unknown product evolution on the design and reconfiguration costs?

We consider robust optimization rather than stochastic optimization because such methods do not require a precise description of the probabilistic evolution of the product family. Computing such a probabilistic forecast of the evolution of the products would be very difficult. In addition, developing an optimization approach to optimize such reconfiguration planning costs within the lifetime of the assembly line is challenging, and robust optimization approaches are usually less demanding in computing resources than stochastic optimization methods.

Our main objective is to design a line to produce a given product family while accounting for future evolutions of the product family. To account for future generations, we define several scenarios that give the product variants in each production generation over the life cycle of the line. The contribution of this work is threefold. First, we formulate a novel robust optimization problem that minimizes the total design and reconfiguration costs of the line for the worst possible scenario. The total costs correspond to the reconfiguration effort of the line. These costs include buying and (un-)installing (re-assigning) the resources and equipment, the cost of hiring workers, and the profits of selling the resources and equipment. Following the robust optimization paradigm, we optimize these costs for the worst-case product family evolution scenario. We formulate mixed-integer linear programming ($MILP$) for this problem. Second, we propose an adversarial approach to find a more robust solution faster. Finally, we perform computational experiments in a simulation framework. These numerical results show that our robust model provides significantly lower design and reconfiguration costs for the worst-case scenario when compared to the classical model where the

design and reconfiguration of the line are optimized at each period without foreseeing the next generations. The robust solution requires a larger initial investment, but the reconfiguration costs of a flexible line designed with the robust approach do not vary significantly when product families change a lot in different generations.

The rest of our paper is organized as follows. Section 2 presents the literature review of the topic at hand. Section 3 successively introduces the problem description, a generic approach to generate the scenario tree of product evolutions, an illustrative example, and the new Mixed Integer Linear Programming ($MILP$). Section 4 describes the developed adversarial approach. Section 5 presents the computational results, and it gives a discussion on several managerial insights. Finally, Section 6 concludes the paper and suggests some future work directions.

2. Literature review

This section reviews the literature related to the main concepts studied in this paper, namely the mixed-model assembly line, reconfigurability, uncertainty in assembly lines, and commonly developed optimization approaches, respectively. We also highlight the major contributions of this work compared to the literature.

In terms of product variety, assembly lines are categorized into three types: single-, mixed-model, and multi-model lines (Kucukkoc and Zhang, 2014). A single line produces a single type of product, while mixed/multi-model lines produce multiple types of product models (Sivasankaran and Shahabudeen, 2017). The main difference between mixed-model and multi-model lines is the ordering of product models entering the line. In multi-model lines, product models enter in separate lots, and there is a noticeable setup time between models. In mixed-model lines, products enter in an arbitrary order where the setup time is negligible compared to the processing and cycle times. A mixed-model line offers a very high level of flexibility (Johansen, 1990) and can benefit from the reconfigurability concept.

To do so, our research focuses on improving the reconfiguration of such lines considering the shortening product lifecycles, product evolutions, and frequent market changes is getting growing attractions.

The studied line is reconfigurable since the resources and equipment can be removed, moved, or added to stations for a new product family producing at the line. Reconfigurable manufacturing systems are designed to rapidly adapt functionality and production capacity in response to new changes by re-arranging/changing production components (Koren et al., 1999). On the one hand, changes in product demand affect the takt time and the input sequences, and they might require re-balancing of the line. On the other hand, changes or additions to a product variant may require new tools or skills (Mehrabi et al., 2000). These changes happen after the design of the line, and they can not be predicted accurately. There is a need to develop methods that can provide an initial assembly line design, while also accounting for future reconfigurations of mixed model (Şeker et al., 2013; Manzini et al., 2018).

As the market is no longer satisfied with a mass-producing uniform product (Singh et al., 2017), manufacturing companies must offer a variety of customized products to the market (Alptekinoğlu and Corbett, 2008). The explosion of product variety is particularly evident in the automobile and computer industries (Pil and Holweg, 2004). However, product variety dramatically increases the complexity of the tasks of process design and production management (Benkamoun et al., 2013). In addition, an MMAL adapts to face changes in product variety and changes in demand during its life cycle (Hu et al., 2008; Zhang et al., 2023). With the shortening of product life cycles and unpredictable demand, these changes become more frequent. Some papers studied product family changes from a product design point of view, but not in terms of the assembly line design and reconfiguration (Johansen, 1990; Pirmoradi et al., 2014; Gembarski et al., 2021). Creating a product family has a significant impact on the understanding of the design and manufacturing characteristics during the product's life

cycle (Zhang et al., 2020; Stief et al., 2023). Note that product development in the future corresponds to uncertainty factors (Wei et al., 2017), either in terms of product structure design (product features) or market demand. For example, Wei et al. (2017) mentioned that customer requirements create uncertainty for future product evolutions, and they developed a flexible design method for product family development considering such uncertainty. Moreover, Biswas et al. (2023) mention that product family evolution is seen as the foundation of product development where new varieties of product models gradually enter the market to adapt to changing environments. In Biswas et al. (2023), authors propose a methodology to address uncertainty in the evolution of the dynamic modeling system of product family evolution. The authors consider three driving elements, market demand, customer requirements, and technological requirements. To the best of our knowledge, we are the first to consider the assembly line design and balancing problem under uncertain product family evolution. Our study aims to account for such uncertainty in the reconfiguration planning of a mixed-model assembly line. Such a concept was raised also from the real automotive industry as we had an interview with a company in France.

In terms of solution technique, several researchers developed methods to plan manufacturing system design changes over several generations of production (AlGeddawy and ElMaraghy, 2012; Bryan et al., 2013; Abbas and ElMaraghy, 2018; Koren et al., 2018). For instance, Bryan et al. (2013) studied the reconfiguration planning of a line that produces a family of products. The authors show that the planning can be improved by considering the future customization of products and variable demands for different product models. Our work studies a robust optimization model for the design and reconfiguration planning of an MMAL with the uncertainty caused by future evolutions of the product family. Robust optimization is one of the fundamental optimization approaches that model uncertainty and its effects (Liu et al., 2020; Pereira, 2018), and it focuses on modeling techniques to make plans that are insensitive to uncertainty. Robust optimization techniques optimize the per-

formance of the system for the worst case. Many parameters are subject to uncertainty in assembly lines, and several works rely on robust optimization models to design lines are can face, as some given publications in the recent literature (Wu et al., 2022; Liu et al., 2021; Breckle et al., 2021). The reformulation per constraint and dualization and the adversarial approaches are commonly used to tackle robust models. The dualization approach does not apply to our problem, since the associated sub-problem is non-convex. Adversarial techniques (Yanıkoğlu et al., 2019) refer to the approaches that do not rely on a tractable reformulation of the robust counterpart. These techniques usually perform by iteratively solving a restricted robust model that accounts for a set of scenarios to provide a possible robust solution. In each iteration, adversarial approaches find the worst possible scenario for the current solution. If an adversarial approach finds this scenario, it inserts the scenario into the restricted robust model. The algorithm stops when a possible robust solution is guaranteed to be feasible. As an example, Bienstock and Özbay (2008) proposed an adversarial approach using some decomposition techniques to iteratively restrict the space of realization of an uncertain parameter. The results show that the approach provides robust solutions even for large instances. A specificity of our problem is that the line is re-configured (re-optimized) in each generation. This corresponds to a multi-stage stochastic problem, and we adapt the adversarial approach to this context.

The main contributions of this work are given below. First, we deal with a novel mixed-model assembly reconfiguration planning problem ($MALRP$), where the line is reconfigured in each product generation. Second, we formulate a new scenario-based $MILP$ for the robust optimization model which aims to minimize total design and reconfiguration costs. The model optimizes for the worst-case sequence of product generation when the model variants in the future generation are unknown. Third, this study develops a generic technique to generate the scenario tree for the product family evolutions. Fourth, we propose an adversarial technique to solve larger instances. Fifth, we perform a simulation to compare the proposed

robust model with the classical approach, and this simulation leads to several managerial insights. The results demonstrate the better performance of the developed adversarial approach compared to the proposed $MILP$, giving a good quality solution in less computational time. Note that this work is an extended version of our recent proceeding (Hashemi-Petroodi et al., 2022). Hashemi-Petroodi et al. (2022) developed a mathematical model for the robust optimization of the design and reconfiguration planning of an MMAL with the uncertainty of the future product family evolution. The present work contains several major contributions compared to our previous study. First, the current mathematical model is slightly improved by considering more practical assumptions. Second, we develop an approach to deal with the product family evolution and construct the new product model requirements (e.g. the precedence graph), while in our previous conference paper (Hashemi-Petroodi et al., 2022), we assumed that the new product requirements are given. Third, an adversarial approach with corresponding simulation tests has been proposed to solve the problem for larger instances more efficiently. Fourth, more computational experiments and insights are provided in this work compared to Hashemi-Petroodi et al. (2022).

3. Problem description and formulation

This section successively describes the studied mixed-model assembly line reconfiguration planning problem ($MALRP$), an approach to generate the scenario tree for future evolutions of the product family based on user opinions, and a novel scenario-based mixed-integer linear programming ($MILP$) model. This section also provides a simple example for further clarification of the problem.

3.1. Description of MALRP

The studied problem is motivated by the situation encountered by an automotive producer in France. Our discussion with an automotive manufacturer highlighted the difficulty

they are facing in reconfiguring the production line every few months (e.g., 6/12 months). Such reconfiguration is challenging and costly for the company and they usually perform it during weekends or public holidays. The reason behind such a reconfiguration is the market changes when the company adds a new product variant to the current product family. Moreover, the changes in the demand for existing product models may cause reconfiguration. Often, the new product added to the line replaces other product variants whose production is discontinued. Therefore, accommodating the new product variant(s) under development implies the reallocation of tasks to the stations and reconfiguration which corresponds to removing/adding/moving the equipment pieces, human workforce, and mobile robots at stations. The company aims to optimize the design and reconfiguration planning of the line in terms of total costs of eliminating/purchasing, (un-)installing/re-allocating the resources and equipment. This is expected because the assembly lines are usually designed for a long lifetime with high investment and following amortized costs (Boysen et al., 2009).

The line produces a part family of products in the context of a mixed-model assembly line. The products enter the line in arbitrary orders with negligible setup time compared to the processing times. The set of product models is given as $\mathcal{I} = \{1 \dots I\}$. The line has several sequential stations $\mathcal{S} = \{1 \dots S\}$, and the product items enter one by one, and they pass through all stations. The items move to the next station at the same time step, called takt time C, which is the same for all stations as defined in a paced line. Each station has a single resource (either a human worker or a robot) with several equipment pieces. Each task can be performed either by a worker or by a robot. We denote by $\mathcal{R}$ the set of resources and by $\mathcal{E}$ the set of equipment as well.

At each takt, there is only a single product unit at each station. Each product model $i \in \mathcal{I}$ requires a set of $\mathcal{O}_i$ tasks. Each task o of product model i requires a processing time pt_{oei} if it is performed by equipment e. Note that the process duration for a task o may change with the product models. Each product model $i \in \mathcal{I}$ has a set A_i of precedence relationships

(o, o') between tasks, where task o must be performed before task o' for this product model.

The compatibilities among tasks, resources, and equipment are represented by two sets. CE_o contains the types of equipment capable to perform task o, and CR_e contains the resources certified to use the equipment e. Note that only one resource (a worker or a robot) and several pieces of equipment can be located at each station.

The line must be reconfigured in each production generation where the set of model variants in the product family changes. We distinguish four types of changes in the product family: (1) the new product family may include a new product model; (2) some model variants in the new product family may be discontinued; (3) the set of tasks for each model in the family can change; (4) the demand for each variant can change. Therefore, because of such changes, the duration of the tasks and their precedence relationships can change. In each generation g in the set of generation $\mathcal{G} = \{0 \dots G\}$, the line may assemble a set $\mathcal{PS}_g = \{1 \dots PS_g\}$ of product variants. Note that $g = 0$ refers to the current generation and the set of product models PS_0 is known ($|PS_0| = 1$). Each product family p at generation g contains the set $\mathcal{I}_{pg} = \{1 \dots I_{pg}\}$ of product models. We denote by m^g_{ip}, the market demand volume of the existing product model i in product family p at generation g. On the contrary, each of the future generations includes different scenarios that represent different evolutions of the product families. We explain the generation of these scenarios later in this section.

The objective is to design a reconfigurable assembly line ready to evolve for different product generations. The reconfiguration is performed by re-arranging the resources and equipment. A resource or/and the equipment piece(s) might be removed from one station, a new resource or/and the equipment piece(s) might be added to a station, or a resource or/and the equipment piece(s) might be removed from one station and installed in the other station. Each of these reconfigurations is associated with a cost: α_{eg} and α'_{rg} denote the purchasing cost of equipment e and robot r (or hiring cost of worker r) in production generation g, respectively. β_{eg} and β'_{rg} denote the selling price of equipment e and robot r in production

generation g, respectively. λ_{eg} and λ'_{rg} are the installation cost of equipment e and robot r in production generation g, respectively. Finally, γ_{eg} and γ'_{rg} are the un-installation cost of equipment e and robot r in production generation g, respectively.

The objective is to minimize the design and reconfiguration cost for the worst-case evolution path of the line, but this evolution path must take value in a well-defined uncertainty set. For simplicity, in this work, we consider that only one new product model is released in each generation, and this model replaces one of the ancient ones in the family. In addition, we let the user define some restrictions on what can happen in the future because designing a line that can handle all possible changes (a new non-restricted product model) would be too expensive and is not logical. More precisely, the user must intervals for the number of additional required tasks, removal of tasks, the processing time of those tasks, and the precedence relationship between tasks.

3.2. New product requirements

The considered problem is a multi-stage robust optimization problem, but the well-known affine rules cannot be used because the recourse problems are combinatorial. Therefore, we approximate the problem by sampling a scenario tree that represents the future evolution of the product. This section explains how the scenarios are generated in a tree using an illustrative example. Each scenario gives a possible definition of the product in each product family of each generation. The features of the product models correspond to the product design, and they consist of the number of required tasks, the processing time of tasks, and the precedence relationships.

Figure 1 illustrates a simple example of the scenario tree for two future production generations of a mixed-model line which is initially designed for two parts A and B with equal demand volumes 50%. Based on the market situation, the marketing and product development department decided to upgrade one of these product models. Therefore, two scenarios

are possible in the next generation ($g = 1$) where product A or product B evolve to C_1 or C_2, respectively. Then, the same can happen for the following generation ($g = 2$) that one of the product models is replaced by another new model (e.g., given in Figure 1, products B or C_1 be evolved to D_2 or D_1, respectively). Many scenarios can be generated into this tree based on the features of the new product model (e.g., set of tasks, processing time, precedence graph) and the demand volume of product models.

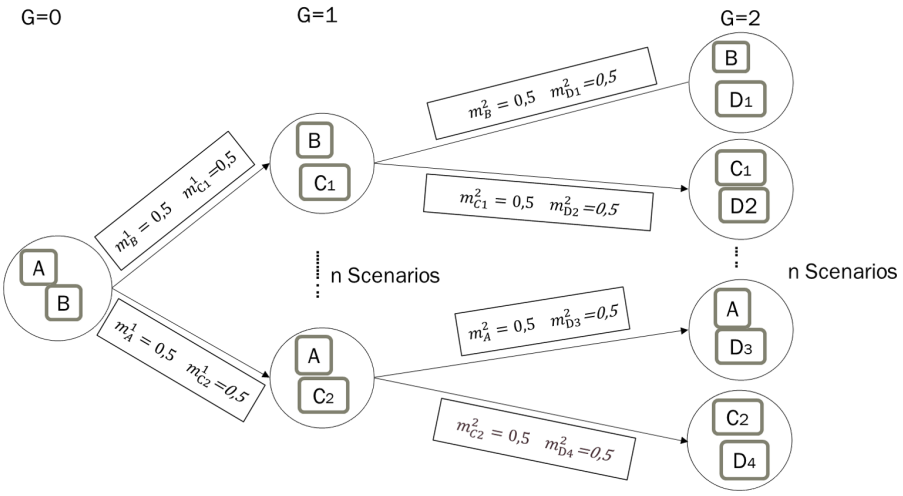

Figure 1: The product families in two production generations (scenario tree).

[Alt-Text:]The Figure shows the proposed scenario tree for this study where several evolved product families are generated for some future production generations.

We denote the set of scenarios by $SC = \{1 \dots N\}$, where each scenario $n \in SC$ is a succession of pairs (p, p') of product families such that production moves from family $p \in PS_{g-1}$ to family $p' \in PS_g$ in the next generation. Note that in the proposed approach for the scenario tree generations, the probability of all scenarios is equally likely. The reason is the lack of information on the future evolution of products and the nature uncertainty of the new product requirements. In this study, we are using the generated scenario tree to solve our robust optimization problem. As we optimize against the worst-case, the model does not

account for the probability of the scenarios.

The proposed $MALRP$ aims to study a robust optimization model aiming to minimize the total cost of design and future reconfigurations of the line for the worst scenario in the tree. The total cost consists of the cost of buying/hiring new resources and equipment, (un)installing the resources and equipment from the stations, and removing/firing the useless resources and equipment. We give below the main steps of the approach to provide the scenario tree, along with an illustrative example for two generations (current product family $p = \{A, B\}$ in the design stage $g = 0$ and the new family $p = \{B, C_1\}$ in generation $g = 1$ as given in Figure 1):

Step 1: Create the joint precedence graph for the current product family, e.g. $p = \{A, B\}$ at the current design stage $g = 0$.

This paper considers the product family representation proposed by Bryan et al. (2013). This approach is appealing because, if the line includes buffers to handle fluctuations in the time required in each station, methods designed for the simple assembly line balancing problem can solve the resulting problem (Becker and Scholl, 2006). The approach introduced by Thomopoulos (1967) creates the product family precedence graph A'_p, which is the integrated graph of all product models in family p. O'_p represents the set of all tasks needed for all product models in a product family where $O'_p = \cup_{i \in p} O_i$, where $\mathcal{O}_i$ is the set of tasks needed for product model i. Bryan et al. (2013) uses the weighted average times of tasks to determine the processing time of tasks. In our work, we create a common precedence graph for all variants of a given product family. We denote by pt^g_{oep} the average time of each task o executing by equipment e in the joint graph of the product family p at generation g. Moreover, pt^g_{oeip} is the processing time of each task o of the certain product model i when it is executed by equipment e in product family p at generation g. As shown in Figure 2, pt^g_{oep} is calculated with the ratio of the market demand of the product models m^g_{ip} as follows:

$$pt^g_{oep} = \sum_{i \in \mathcal{I}_{pg}} m^g_{ip}\ pt^g_{oeip} \qquad g \in \mathcal{G},\ p \in PS_g,\ o \in \mathcal{O}'_p,\ e \in \mathcal{E}$$

Figure 2 demonstrates the joint precedence graph for a product family $p \in PS_g$ with two product models A, B of the production generation $g = 0$. Figure 2 gives the precedence graph of model A and model B as well as the joint precedence graph. We remind that m^g_{ip} represents the volume of the market demand. For example, the volume of product models A and B is $m^0_{A1} = 30\%$ and $m^0_{B1} = 70\%$ for the first product family in the current design stage, respectively.

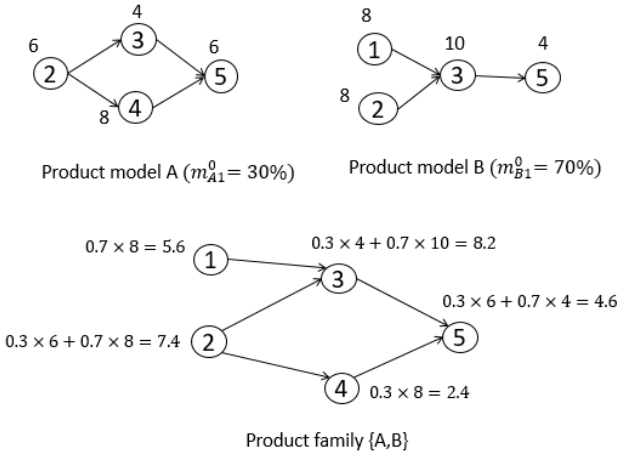

Figure 2: The joint precedence graph for a product family including two product models $\{A, B\}$.

[Alt-Text:]The Figure shows how to create a joint precedence graph of a product family including several product models (e.g. $\{A, B\}$).

Step 2: Construct a new graph for the new product family p' (e.g., $p' = \{B, C_1\}$ in Figure 1) of the next following generation (e.g., $g = 1$ in Figure 1).

This graph is built from the graph of the previous stage, by successively, removing $\delta\%$ of the tasks, adding $\delta\%$ of tasks, and modifying the process duration to the task in the interval $[pt^{g-1}_{op} - \sigma_1.pt^{g-1}_{op}, pt^{g-1}_{op} + \sigma_2.pt^{g-1}_{op}]$. The parameters δ, σ_1, and σ_2 are given, and they control

the stability of the product family. Note that the product family will not change to something completely different. Even if the product is evolving, it will remain with a lot of similarities with the initial version. In most cases, we keep almost a similar number of tasks, the process duration does not change so much, and some precedence constraints never change.

For example, Figure 3 shows how the joint precedence graph obtained in the example given in Figure 2 as step 1 converts to the new joint graph in a new generation. In the current product family, two products A and B are produced where a joint graph is given. By step 2 the joint graph for the new product family $p' = \{B, C_1\}$ in the following generation $g = 1$ is provided. We select a random value 6 for the number of required tasks in the range of $[5, 1.5 * 5] = [5, 8]$ with $\delta = 50\%$. For the processing time, we select a random value in the range of 40% below and 60% above the processing time of each existing task ($\sigma_1 = 40\%$ and $\sigma_2 = 60\%$ here). For instance, for task 1 a random value from the interval $[3.3s, 8.9s]$ is taken which is $5.6s$ in the new graph. For the newly added task 6 we choose a random value in the range of 40% below minimum and 60% above the average processing time ($5.6s$) which is equal to $[3.3s, 9s]$. Therefore, the processing time of task 6 is given $8s$.

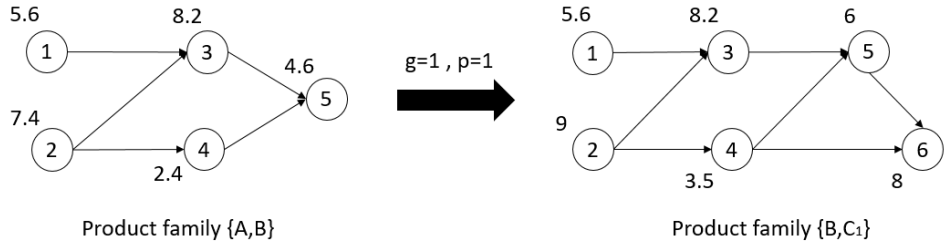

Figure 3: Creating the joint precedence graph of a new evolved product family from a joint graph of an ancient product family (Step 1 and Step 2).

[Alt-Text:]The Figure shows how a joint precedence graph is created for the newly evolved product family (e.g. $\{B, C_1\}$) from a joint graph of the current product family (e.g. $\{A, B\}$).

Step 3: In this step, we create the graph of the new product model from the obtained joint precedence graph of the new family product. We propose the following formulation to calculate the processing time of each task o of the new product model i' in a new family p'.

$$pt^{g}_{oei'p'} = \left[pt^{g}_{oep'} - \sum_{i \in \mathcal{I}_{pg-i'}} m^{g}_{ip'}.pt^{g}_{oeip'}\right]/m^{g}_{i'p'} \qquad g \in \mathcal{G} - 0,\ p' \in PS_g,\ o \in \mathcal{O'}_p$$

Figure 4 shows how the precedence graph of the new product C_1 is extracted from the joint graph of the first product family of the generation $g = 1$ given in Figure 1. Wa assume a demand volume of product models of $m^{g=1}_{Bp=1} = 30\%$ and $m^{g=1}_{C_np=1} = 70\%$. For example for task 2, we obtain $pt^{1}_{2C_11} = [9 - (0.3.8)]/0.7 = 9.4$.

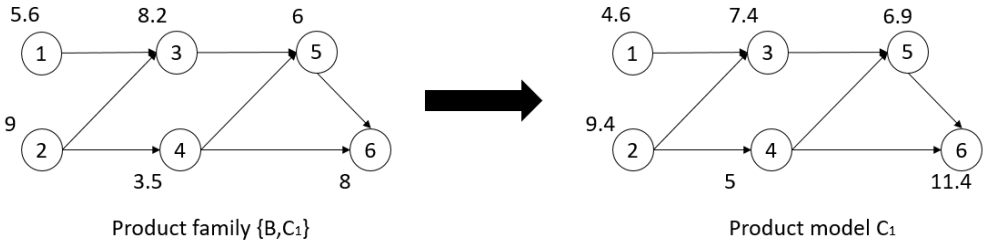

Figure 4: Obtaining the precedence graph of a single product model from the joint graph of a certain product family (step 3).

[Alt-Text:]The Figure shows how to create the precedence graph of a certain product model (e.g. C_1) in a product family from the joint graph of the family (e.g. $\{B, C_1\}$).

Figure 5 shows two other examples of scenarios that correspond to different demand volumes and different product models in the family.

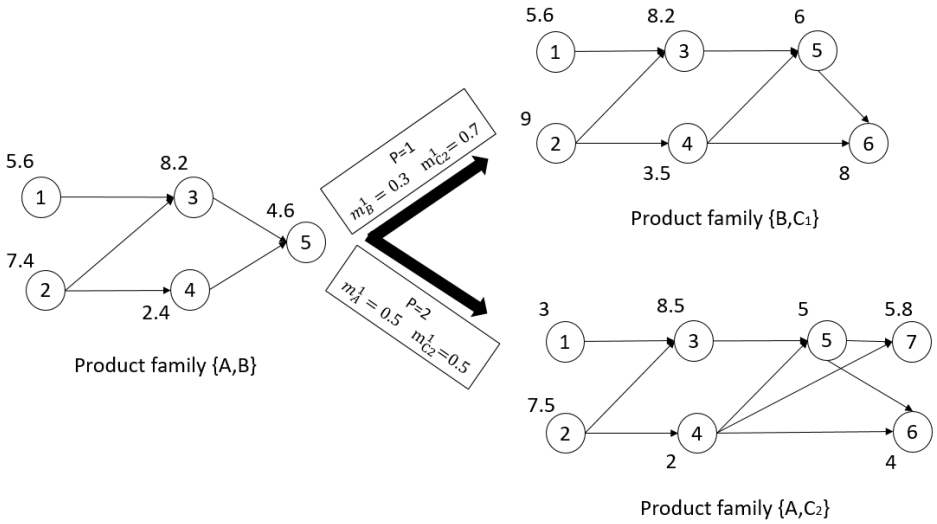

Figure 5: Precedence graph for two possible scenarios in a single generation.

[Alt-Text:]The procedure of generating the joint precedence graph of the newly evolved product families (e.g. $\{B, C_1\}$ and $\{A, C_2\}$) in the new production generation, based on the graph of the current product family (e.g. $\{A, B\}$), is demonstrated.

3.3. Mathematical model

A new mixed-integer linear programming ($MILP$) formulation is provided for the $MALRP$. A list of all parameters and variables used in this study is available in Appendix C. The model aims to minimize the total reconfiguration effort over the future production generation for the worst-case scenario generated in the scenario tree. The effort includes the total design and reconfiguration costs which consist of (un-)installing cost of resources and equipment pieces in case of removing and/or adding them in the line or moving them between stations. The decision variables are defined as follows:

- w_{srp}^g is equal to 1 if resource r is assigned to station s for producing product family p in generation g, and 0 otherwise.

- x^g_{sop} is equal to 1 if task o performed on product family p is performed at station s during production generation g, and 0 otherwise.
- b^g_{sep} is equal to 1 if equipment e is installed at station s for producing product family p in generation g, and 0 otherwise.
- b^g_{soep} is equal to 1 if equipment e is installed at station s to perform task o on product family p in generation g, and 0 otherwise.

The continuous variable Y gives the worst (maximum) value of the total design and reconfiguration cost. Precisely, Y represents the cost of purchasing equipment/robots, hiring workers, the cost of installing and uninstalling robots/equipment, and the cost of removing robots/equipment and firing the workers for the worst scenario among all $n \in SC$. Consequently, the objective function (1) aims to minimize the total design and reconfiguration cost of the worst scenario.

$$\min Y \tag{1}$$

Furthermore, other continuous decision variables are defined: Q_n is the purchase/selling cost of the equipment, Q'_n is the purchase/selling cost of the robots, and the cost of hiring/firing the workers, Z_n is the installing/uninstalling cost of the equipment, and Z'_n shows is the installing/uninstalling cost of the robots and workers. The value of variable Y is calculated by constraints (2) which is the total cost of a single scenario. Also, Q_n, Q'_n, Z_n and Z'_n are calculated by equations (3), (4), (5), and (6), respectively.

$$Y \geq Q_n + Q'_n + Z_n + Z'_n \quad n \in SC \tag{2}$$

$$Q_n = \sum_{g\in\mathcal{G}-0}\sum_{(p,p')\in SC_n}\left[\sum_{e\in\mathcal{E}}\left[\alpha_{e0}\left[\sum_{s\in\mathcal{S}}\ b^0_{se0}\right] + \alpha_{eg}\left[\sum_{s\in\mathcal{S}}\ b^g_{sep} - \sum_{s\in\mathcal{S}}\ b^{g-1}_{sep'}\right]^+ + \beta_{eg}\left[\sum_{s\in\mathcal{S}}\ b^g_{sep} - \sum_{s\in\mathcal{S}}\ b^{g-1}_{sep'}\right]^-\right]\right] \quad n \in SC \tag{3}$$

$$Q'_n = \sum_{g\in\mathcal{G}-0}\sum_{(p,p')\in SC_n}\left[\sum_{r\in\mathcal{R}}\left[\alpha'_{r0}\left[\sum_{s\in\mathcal{S}}\ w^0_{sr0}\right] + \alpha'_{rg}\left[\sum_{s\in\mathcal{S}}\ w^g_{srp} - \sum_{s\in\mathcal{S}}\ w^{g-1}_{srp'}\right]^+ + \beta'_{rg}\left[\sum_{s\in\mathcal{S}}\ w^g_{srp} - \sum_{s\in\mathcal{S}}\ w^{g-1}_{srp'}\right]^-\right]\right] \quad n \in SC \tag{4}$$

$$Z_n = \sum_{e\in\mathcal{E}}\gamma_{e0}\left[\sum_{s\in\mathcal{S}}\ b^0_{se0}\right] + \sum_{g\in\mathcal{G}-0}\sum_{(p,p')\in SC_n}\ \sum_{s\in\mathcal{S}}\left[\sum_{e\in\mathcal{E}}\left[\lambda_{eg}\left[b^g_{sep} - b^{g-1}_{sep'}\right]^+ - \gamma_{eg}\left[b^g_{sep} - b^{g-1}_{sep'}\right]^-\right]\right] \quad n \in SC \tag{5}$$

$$Z'_n = \sum_{r\in\mathcal{R}}\gamma'_{e0}\left[\sum_{s\in\mathcal{S}}\ w^0_{sr0}\right] + \sum_{g\in\mathcal{G}-0}\sum_{(p,p')\in SC_n}\ \sum_{s\in\mathcal{S}}\left[\sum_{r\in\mathcal{R}}\left[\lambda'_{rg}\left[w^g_{srp} - w^{g-1}_{srp'}\right]^+ - \gamma'_{rg}\left[w^g_{srp} - w^{g-1}_{srp'}\right]^-\right]\right] \quad n \in SC \tag{6}$$

Constraints (3) - (6) are non-linear. To linearize these constraints, several binary variables must be defined.

We define some binary variables related to equipment reconfiguration as $b^{+g}_{epp'} := \left[\sum_{s\in\mathcal{S}}\ b^g_{sep} - \sum_{s\in\mathcal{S}}\ b^{g-1}_{sep'}\right]^+$, $b^{-g}_{epp'} := -\left[\sum_{s\in\mathcal{S}}\ b^g_{sep} - \sum_{s\in\mathcal{S}}\ b^{g-1}_{sep'}\right]^-$, $b'^{+g}_{sepp'} := \left[b^g_{sep} - b^{g-1}_{sep'}\right]^+$, $b'^{-g}_{sepp'} :=$

$- \left[b^{g}_{sep} - b^{g-1}_{sep'}\right]^{-}$. These variables determine the re-arrangements of equipment at stations when the line moves from one generation to the next.

Similarly, some binary variables for resource reconfiguration are defined as $w^{+g}_{rpp'} := \left[\sum_{s\in\mathcal{S}} w^{g}_{srp} - \sum_{s\in\mathcal{S}} w^{g-1}_{srp'}\right]^{+}$, $w^{-g}_{rpp'} := -\left[\sum_{s\in\mathcal{S}} w^{g}_{srp} - \sum_{s\in\mathcal{S}} w^{g-1}_{srp'}\right]^{-}$, $w'^{+g}_{srpp'} := \left[w^{g}_{srp} - w^{g-1}_{srp'}\right]^{+}$, $w'^{-g}_{srpp'} := -\left[w^{g}_{srp} - w^{g-1}_{srp'}\right]^{-}$. These variables also determine the re-arrangements of resources at stations when the line switches from one generation to the next.

New constraints (7) - (14) must be added to the model for linearization. These constraints consider two consecutive generations $g-1$ and g, and they compute the value of the newly defined binary variables with a $+/-$. For example, in Constraints (7) and (8), if equipment e is needed in a station at generation g ($\sum_{s\in\mathcal{S}} b^{g}_{sep} = 1$) and was not already in the line during the previous generation $g-1$ ($\sum_{s\in\mathcal{S}} b^{g-1}_{sep} = 0$), then Constraint (7) results in $b^{+g}_{epp'} = 1$. Therefore, in Constraint (8), $b^{-g}_{epp'} = 0$, the equipment must be bought for the line, and the buying cost of equipment α_{eg} is calculated in the function (3). In contrast, if the equipment is used in the line within generation $g-1$ ($\sum_{s\in\mathcal{S}} b^{g-1}_{sep} = 1$) and not needed anymore in generation g ($\sum_{s\in\mathcal{S}} b^{g}_{sep} = 0$), then $b^{+g}_{epp'} = 0$ through Constraints (7) and $b^{-g}_{epp'} = 1$ by Constraint (8) that means $\left[\sum_{s\in\mathcal{S}} b^{g}_{sep} - \sum_{s\in\mathcal{S}} b^{g-1}_{sep'}\right]^{-} = -1$ in the function (3). When a certain equipment piece is either used or not used in two consecutive generations, these two variables are equal to 0. Constraints (9) to (14) are similar. Constraints (7) - (14) ensure that these binary variables get value 1 or 0, and their use in constraints (3) - (6) lead to the positive/negative value.

$$\sum_{s\in\mathcal{S}} b^{g}_{sep} - \sum_{s\in\mathcal{S}} b^{g-1}_{sep'} \;\leq\; b^{+g}_{epp'} \qquad g\in\mathcal{G}, n\in SC,\ (p,p')\in SC_n,\ e\in\mathcal{E} \tag{7}$$

$$\sum_{s\in\mathcal{S}} b^{g-1}_{sep'} - \sum_{s\in\mathcal{S}} b^{g}_{sep} \;\leq\; b^{-g}_{epp'} \qquad g\in\mathcal{G}, n\in SC,\ (p,p')\in SC_n,\ e\in\mathcal{E} \tag{8}$$

$$b^{g}_{sep} - b^{g-1}_{sep'} \;\leq\; b'^{+g}_{sepp'} \qquad g\in\mathcal{G}, n\in SC,\ (p,p')\in SC_n,\ s\in\mathcal{S},\ e\in\mathcal{E} \tag{9}$$

$$b^{g-1}_{sep'} - b^{g}_{sep} \leq b'^{-g}_{sepp'} \quad g \in \mathcal{G}, n \in SC,\ (p,p') \in SC_n,\ s \in \mathcal{S},\ e \in \mathcal{E} \tag{10}$$

$$\sum_{s\in\mathcal{S}} w^{g}_{srp} - \sum_{s\in\mathcal{S}} w^{g-1}_{srp'} \leq w^{+g}_{rpp'} \quad g \in \mathcal{G}, n \in SC,\ (p,p') \in SC_n,\ r \in \mathcal{R} \tag{11}$$

$$\sum_{s\in\mathcal{S}} w^{g-1}_{srp'} - \sum_{s\in\mathcal{S}} w^{g}_{srp} \leq w^{-g}_{rpp'} \quad g \in \mathcal{G}, n \in SC,\ (p,p') \in SC_n,\ r \in \mathcal{R} \tag{12}$$

$$w^{g}_{srp} - w^{g-1}_{srp'} \leq w'^{+g}_{srpp'} \quad g \in \mathcal{G}, n \in SC,\ (p,p') \in SC_n,\ s \in \mathcal{S},\ r \in \mathcal{R} \tag{13}$$

$$w^{g-1}_{srp'} - w^{g}_{srp} \leq w'^{-g}_{srpp'} \quad g \in \mathcal{G}, n \in SC,\ (p,p') \in SC_n,\ s \in \mathcal{S},\ r \in \mathcal{R} \tag{14}$$

The rest of the constraints are given as follows:

$$\sum_{s\in\mathcal{S}} x^{g}_{sop} = 1 \quad g \in \mathcal{G},\ p \in PS_g,\ o \in \mathcal{O}'_p \tag{15}$$

$$\sum_{s\in\mathcal{S}} b^{g}_{sep} \leq 1 \quad g \in \mathcal{G},\ p \in PS_g,\ e \in \mathcal{E} \tag{16}$$

$$\sum_{s\in\mathcal{S}} w^{g}_{srp} \leq 1 \quad g \in \mathcal{G},\ p \in PS_g,\ r \in \mathcal{R} \tag{17}$$

$$\sum_{r\in\mathcal{R}} w^{g}_{srp} \leq 1 \quad g \in \mathcal{G},\ p \in PS_g,\ s \in \mathcal{S} \tag{18}$$

$$x^{g}_{sop} \leq \sum_{e\in CE_o} b^{g}_{soep} \quad g \in \mathcal{G},\ p \in PS_g,\ s \in \mathcal{S},\ o \in \mathcal{O}'_p \tag{19}$$

$$b^{g}_{soep} \leq \sum_{r\in CR_e} w^{g}_{srp} \quad g \in \mathcal{G},\ p \in PS_g,\ o \in \mathcal{O}'_p,\ e \in CE_o,\ s \in \mathcal{S} \tag{20}$$

$$b^{g}_{soep} \leq b^{g}_{sep} \quad g \in \mathcal{G},\ p \in PS_g,\ o \in \mathcal{O}'_p,\ e \in CE_o,\ s \in \mathcal{S} \tag{21}$$

$$\sum_{o\in\mathcal{O}'_p} \sum_{e\in CE_o} pt^{g}_{oep} b^{g}_{soep} \leq C \quad g \in \mathcal{G},\ p \in PS_g,\ s \in \mathcal{S} \tag{22}$$

$$\sum_{s\in\mathcal{S}} s\ x^{g}_{sop} \leq \sum_{s'\in\mathcal{S}} s'\ x^{g}_{s'o'p} \quad g \in \mathcal{G},\ p \in PS_g,\ (o,o') \in \mathcal{A}'_p \tag{23}$$

Constraints (15) ensure that each task o performed on product family p is performed in only one station s at generation g. Constraints (16) and (17) ensure that equipment e/

resource r is assigned at only one station s for producing product family p in generation g. Constraints (18) assign only one worker/robot at each station s. Constraints (19) ensure equipment is assigned to the station when performing a task there. Constraints (20) identify a compatible resource (worker/robot) at a station when the task is performed with the required equipment at that station. Constraints (21) determine the value of b^g_{sep} according to the value of b^g_{seop}. Constraints (22) and (23) are respectively the classical takt time and precedence relationship constraints. Note that the model does not forbid keeping the possible stations free in the line to enhance the line's flexibility as needed. Finally, Constraints (24) - (33) give the bounds on the decision variables.

$$b^{+g}_{epp'}, b^{-g}_{epp'} \in [0,1] \qquad n \in SC,\ (p,p') \in SC_n,\ e \in \mathcal{E} \tag{24}$$

$$b'^{+g}_{sepp'}, b'^{-g}_{sepp'} \in [0,1] \qquad n \in SC,\ (p,p') \in SC_n,\ s \in \mathcal{S},\ e \in \mathcal{E} \tag{25}$$

$$w^{+g}_{rpp'}, w^{-g}_{rpp'} \in [0,1] \qquad n \in SC,\ (p,p') \in SC_n,\ r \in \mathcal{R} \tag{26}$$

$$w'^{+g}_{srpp'}, w'^{-g}_{srpp'} \in [0,1] \qquad n \in SC,\ (p,p') \in SC_n,\ s \in \mathcal{S},\ r \in \mathcal{R} \tag{27}$$

$$x^g_{sop} \in \{0,1\} \qquad g \in \mathcal{G},\ p \in PS_g,\ s \in \mathcal{S},\ o \in \mathcal{O}'_p \tag{28}$$

$$b^g_{sep} \in \{0,1\} \qquad g \in \mathcal{G},\ p \in PS_g,\ s \in \mathcal{S},\ e \in \mathcal{E} \tag{29}$$

$$w^g_{srp} \in \{0,1\} \qquad g \in \mathcal{G},\ p \in PS_g,\ s \in \mathcal{S},\ r \in \mathcal{R} \tag{30}$$

$$b^g_{soep} \in \{0,1\} \qquad g \in \mathcal{G},\ p \in PS_g,\ s \in \mathcal{S},\ o \in \mathcal{O}'_p,\ e \in \mathcal{E} \tag{31}$$

$$Q_n, Q'_n, Z_n, Z'_n \geq 0 \qquad n \in SC \tag{32}$$

$$Y \geq 0 \tag{33}$$

3.4. Illustrative example

For clarity, this section illustrates the studied $MALRP$ with a simple example. All the details of input used parameters in the example are given in Appendix A.

This illustrative example aims to further clarify the studied $MALRP$ considering a line with two sequential stations which is producing two product models. For each product family evolution in each period when a new product model is added to the family, the line should be reconfigured. At the initial stage of the line ($g = 0$), a product family $p = 0$ requires 5 tasks. In $g = 1$, the product family can evolve in different scenarios. In this example, we generate five scenarios to obtain the worst-case. However, only two scenarios are illustrated, namely, the case of product families $p = 0$ requiring 5 tasks, and $p = 1$ requiring 7 tasks. The takt time is equal to $600s$. The precedence graphs of the new product families are generated as explained in Section 3.2.

Figure 6 shows the initial design ($p = 0$ and $g = 0$) and the configuration of the line for two possible product families in the new generation $g = 1$. Among these two scenarios, the second one ($p = 1$ in $g = 1$) is worse (total cost of $73,247$ compared to $70,478$) and also the worst one among all 5 generated scenarios. The detailed calculation of cost functions are given in Appendix B.

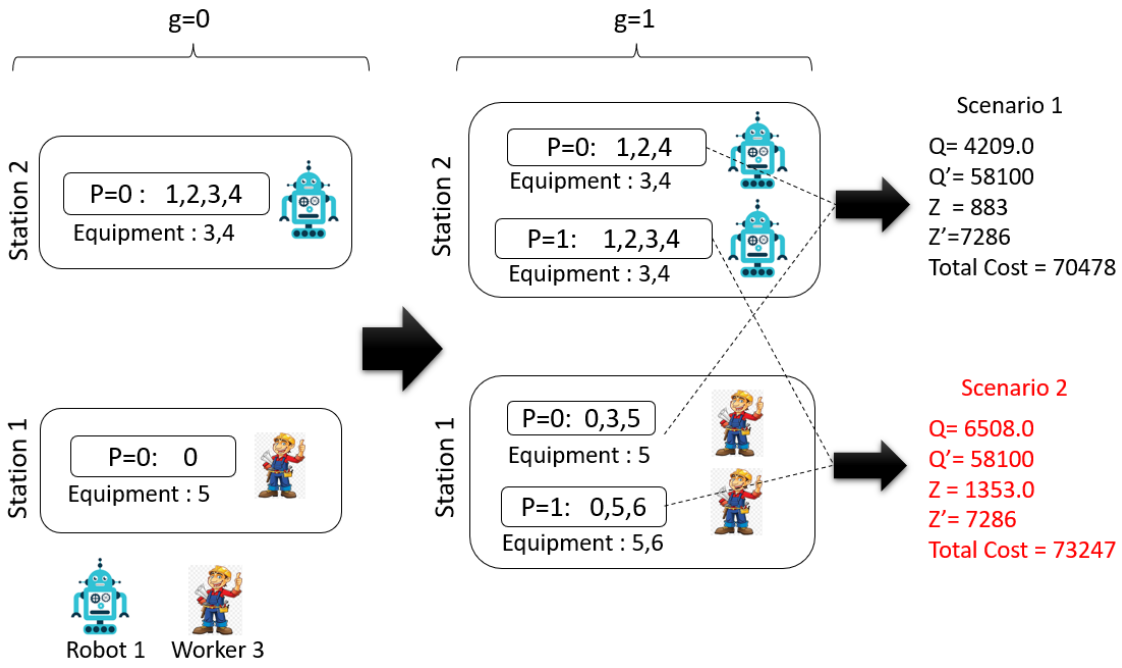

Figure 6: The solution of the illustrative example with one generation and two products family

[Alt-Text:]The solution of the illustrative example (including the required resources, equipment, task assignments, and cost values) for two possible scenarios of the current design and one future production generation.

4. Adversarial approach

This section introduces a heuristic based on the adversarial approach to robust optimization problems. The advantage of this method is that it does not require the random generation of the scenario tree. Instead, the worst-case scenarios are generated iteratively during the resolution. However, the proposed approach generates the worst-case scenarios with a heuristic. As a result, the solution may be sub-optimal. On the contrary, the MILP based on the sampled scenario tree converges toward the optimal solution as the size of the tree increases. The advantage of the heuristic based on the adversarial approach is its computational effi-

ciency. Solving the MILP for a large scenario tree would take too much computational effort. Our results show that the adversarial approach outperforms the scenario-tree approach in terms of speed and solution quality. In particular, the adversarial approach scales better to large-size instances than the MILP based on the sampled scenario tree.

An adversarial approach is an effective approach to solve robust optimization problems (Yanıkoğlu et al., 2019). This approach is also known as a two-step robust optimization approach since it consists of two stages, called the **master problem** and the **sub-problem**. The adversarial approach starts with a finite set of scenarios over the uncertain parameter. The first stage master problem makes the optimal decisions for the current set of scenarios. Then, the sub-problem finds a scenario for the uncertain parameter that makes the last found solution from the master problem infeasible, e.g., we can search for the scenario that maximizes the infeasibility. We add this scenario to the set of scenarios in the master problem and solve the resulting robust optimization problem. The approach improves the robustness of the solution, iteratively.

In this study, we develop an adversarial approach to accelerate finding the worst scenario of product variants for each production generation in the future. To adapt the approach to the problem at hand, we define the master problem and sub-problem(s). The **master problem** solves the proposed $MILP$ considering a smaller number of scenarios that are generated similarly as in $MILP$. The master problem results in the initial design of the line and reconfiguration plans considering the corresponding scenarios. Then, the **sub-problem** solves the $MILP_{sub}$ to find the worst product variant (the worst scenario) for each time period.

Master problem: We use the approach described in Section 3.2 to generate the initial scenario tree, and we solve the proposed MILP (1) - (33). The master problem solves the $MILP$ (1) - (33) with a smaller scenario tree compared to the main MILP model, called the set SC' (with fewer scenarios in the set SC_n). The set of scenarios is iteratively enriched

by solving the sub-problem.After getting the solution to the master problem, we keep only the initial design of the line including the resource and equipment assignment to stations at $g = 0$. More precisely, we fix the decision variables w^0_{srp}, b^0_{sep} and b^0_{soep} and rename them as w^{*0}_{sr}, b^{*0}_{se} and b^{*0}_{soe}, respectively. Then, the sub-problem is solved for each production generation, separately, to find a new scenario.

Sub-problem: We create a sub-problem for each generation. The sub-problem solves $MALRP_{sub}$ which is similar to the $MILP$, but the initial design of the line is fixed, it only optimizes one generation ahead and finds the worst product model adding to the line (the worst scenario). The first sub-problem solves $MALRP_{sub}$ for only the first generation ahead ($g = 1$) and the second one solves $MALRP_{sub}$ for the next and last generation ($g = 2$). Each sub-problem searches only one scenario $n = 1$ as the worst product family $|PS_g| = 1$ for each new generation ($|G| = 1$). Indeed, the design of the line from the master problem is given by w^{*0}_{sr}, b^{*0}_{se} and b^{*0}_{soe}, and the sub-problem optimizes the reconfiguration of the first generation and fix them as w^{*1}_{sr}, b^{*1}_{se} and b^{*1}_{soe}, then it optimizes the reconfiguration of the second generation. After generating the whole scenario (for all generations), we add this scenario to set SC', and the master problem is resolved with the new scenario tree. Therefore, the variables in the subproblem define the precedence graph, the processing times of the tasks, and the number of tasks required for the new product family.

Note that, for the number of required tasks, the worst case happens when the new product family requires the maximum number of possible tasks which is the upper bound of the proposed interval $\lceil \delta.|O_i| \rceil$.We define O', the set of tasks of the current product family, and O'' as the set of new tasks added to the joint precedence graph of the new product family in the new generation. To determine the processing time and precedence relationships of tasks, we define the following decision variables:

- $M_{oo'}$ is equal to 1 if task $o \in O' \cup O''$ preceeds task $o' \in O''$, and 0 otherwise.

- ${pt'_{oe}}^{g} \geq 0$ is the processing time of task $o \in O' \cup O''$ in the single scenario (product family) generated at the generation g.

These variables are involved in the takt time and precedence constraints in the $MILP_{sub}$. The approach seeks a product family that results in the worst-case, and this corresponds to the largest number of tasks with high process time and a complex precedence graph. After solving the master problem, the sub-problem aims to find the worst-case of the product family for each generation that optimizes the product process time pt'_{oe} and precedence relationships $M_{oo'}$ by maximizing the product variant complexity which is the same as maximizing the cost. On the other hand, the sub-problem aims to balance and design the line for each generation with less cost. That means that it optimizes the task, resource, and equipment assignments/reconfigurations (variable sets x^g_{so}, w^g_{sr}, b^g_{se} and b^g_{soe}, respectively) aiming to minimize the total costs. Therefore, the objective function of the $MILP_{sub}$ is given as (34), where $F(x, w, b, pt', M)$ is a function of existing variables in the model.

$$\max_{pt', M} \ \Big(\min \ F(x, w, b, pt', M) \Big) \tag{34}$$

The resulting objective function is difficult to integrate into the sub-problem. For example, a local-search algorithm could be developed for the Max-part of the model to provide several product variant scenarios, then the scenarios can be evaluated using $MILP_{sub}$ optimizing the inner Min-part of the model. However, solving the sub-problem with a combined approach of local search and mathematical programming is not efficient in the adversarial approach. Thus, we simplify the function to a Max-Max one instead of the Max-Min. The main idea behind this conversion is to provide an approximated solution, efficiently. Indeed, the Max-Min model discovers the robust solution for the real worst-case, where the process designers re-design the line after observing the new product family. However, the Max-Max function provides a solution corresponding to the worst scenario of the product variants, but it is not that good at balancing costs. This approximation has a minor impact on our results since it

concerns the sub-problem, and the subproblem only provides the worst scenario for the set of existing scenarios in the tree. Afterward, the master problem optimizes the total cost for the worst case.

Therefore, the function (34) is transformed to (35). The mathematical model $MILP_{sub}$ is finally given as (D.1) - (D.32) in Appendix D. Note that, $MILP_{sub}$ is for the single generation with a given design for the previous generation. The objective function D.1 aims to find the most costly product family.

$$\max_{pt',M} \ \Big(\max \ F(x, w, b, pt', M) \Big) \tag{35}$$

The $MILP_{sub}$ is adapted from the proposed $MILP$ by modifying Constraints (D.21) and (D.22). Constraint (D.21) is quadratic which needs to be linearized. So for that, we give a finite upper bound for ${pt'_{oe}}^{g}$ called M with $o \in \mathcal{O}' \cup \mathcal{O}'', e \in CE, g \in \mathcal{G}, |G| = 1$. Then this constraint is linearized by using the so-called big M method. We introduce a new variable $\pi^g_{soe} = {pt'_{oe}}^{g} b^g_{soe}$ with $s \in \mathcal{S}, o \in \mathcal{O}' \cup \mathcal{O}'', e \in CE, g \in \mathcal{G}, |G| = 1$. Note that the product that we model by π^g_{soe} equals zero if $b^g_{soe} = 0$ but π^g_{soe} can take any value in the range of 0 and M if $b^g_{soe} = 1$. This can be modeled using $\pi^g_{soe} \leq b^g_{soe} M$. Next, the product is always positive and smaller than ${pt'_{oe}}^{g}$, thus $\pi^g_{soe} \geq 0$ and $\pi^g_{soe} \leq {pt'_{oe}}^{g}$ with $s \in \mathcal{S}, o \in \mathcal{O}' \cup \mathcal{O}'', e \in CE, g \in \mathcal{G}, |G| = 1$. It is left to force π^g_{soe} to equal ${pt'_{oe}}^{g}$ in case $b^g_{soe} = 1$ which we obtain with constraint (36).

$$\pi^g_{soe} \geq {pt'_{oe}}^{g} - (1 - b^g_{soe}) M \qquad s \in \mathcal{S}, o \in \mathcal{O}', e \in CE, g \in \mathcal{G}, |G| = 1 \tag{36}$$

Constraints (D.21) are linearized as given in equation (36). Moreover, Constraints (D.22) are activated when a precedence relationship is taken into account between tasks o and o', otherwise they are deactivated. The details of the proposed adversarial approach are given in Algorithm 1. Moreover, in Appendix E, Figure E.9 illustrates the framework of the proposed adversarial technique with a precise description, as well.

Algorithm 1 Adversarial approach (AA)

Required: Generate a scenario tree with fewer scenarios (from section 3.2), compared to the $MILP$. We call the set of scenarios SC'.

Step 1: Solve the $MILP$ with set SC' of scenarios. Get the solution "sol". Start iteration numbering ($Iter := 1$).

Step 2: Fix the decision variables w^0_{sr0}, b^0_{se0} and b^0_{soe0} to become w^{*0}_{sr}, b^{*0}_{se} and b^{*0}_{soe}. The rest of these variables are optimized in the sub-problem for the generation $g = 1$.

Step 3: Solve the $MILP_{sub}$ searching for a single worst product family $|PS_g| = 1$ observing only one new generation ahead ($|G| = 1$).

Step 4: Similarly to Step 2, fix variables w^1_{sr0}, b^1_{se0} and b^1_{soe0} as w^{*1}_{sr}, b^{*1}_{se} and b^{*1}_{soe}, and re-solve $MILP_{sub}$ for $g = 2$ ($|G| = 1$).

Step 5: A single scenario for both generations is found as n'. Re-solve the master problem considering the single scenario n'. Get the new solution "sol_{new}".

Step 6: If "sol_{new}" is better than solution "sol", then set $sol := sol_{new}$ and add n' to the set of scenario ($SG' := SG' \cup n'$). Re-start the iterations ($Iter := 0$) with the new number of scenarios $|SC|$, and re-do the steps 1 to 5. Otherwise, if "sol_{new}" is not better than "sol" repeat steps 1 to 5 going to the next iteration ($Iter : += 1$).

Step 7: The algorithm stops when the number of iterations reaches N ($Iter := N$).

5. Computational experiments and results

This section successively reports the instances generation procedure, the results of computational experiments that analyze the performance of the proposed approach in terms of solution quality and CPU time, and several managerial insights. Within this section, AA stands for the adversarial approach. The models are solved with IBM ILOG CPLEX Optimization Studio V12.10. The experiments were run on an Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz 2.11 GHz processor with 32 GB of RAM in MS Windows 10 Pro (64 bit)

operational system.

5.1. Instance generation

The data generator proposed by Otto et al. (2013) is extended to the specificity of the current problem. Each of the instances in this study merges I random instances of Otto et al. (2013). As an example, each instance in this study contains the data from two random instances of Otto et al. (2013) corresponding to $I = 2$ product models with different precedence graphs and processing times which are currently produced in the line. We consider different numbers of stations ($S = \{4, 7, 10\}$). We use two sets of instances from Otto et al. (2013), with 20, 50, and 100 tasks. Note that, these are the number of tasks for the current product family, and these numbers will increase for future product evolutions in future generations. We assume $I = 2$ product models in generation 0, and we assign random values to δ and σ_1, but $\sigma_2 = 1 - \sigma_1$.

The instances' sizes are determined by the 3-tuple (I, S, O), where I, S, and O represent the number of product models, stations, and tasks, respectively. Therefore, for each size of instances $((2, 4, 20 - 24 - 29)$, $(2, 7, 50 - 60 - 72)$, and $(2, 10, 100 - 120 - 144))$, 10 instances are generated and solved. Precisely, $20 - 24 - 29$ in $(2, 4, 20 - 24 - 29) - size$ instances means the upper bound of the number of generated tasks in the current, first, and second generations are 20, 24, and 29, respectively, because $\delta = 20\%$. Moreover, additional tests are performed for sensitivity analysis and managerial insights by changing some parameters of the model.

The compatibility matrices CE_o and CR_e are randomly generated. However, the equipment with higher ability is more expensive compared to the ones that can perform only a small set of tasks. Moreover, automated equipment is more expensive than manual ones. Note that, the passage of time and amortization are taken into account to generate the cost values. As there are no-cost values in the data set (Otto et al., 2013), costs are uniformly

distributed. We tried to logically generate these cost values respecting the reasonable ratio of different types of costs (buying, installing, etc.). The ratio has been asked and approved by our industrial partners. The main reason for selecting the uniform distribution was to have more simplicity and not get biased results. Using a uniform distribution, all cost values within a given range are selected with equal chance which led us to avoid any corresponding biases of costs. Moreover, it is not easy to find proper cost data sets as needed in our studied problem in the literature. Therefore, the uniform distribution can be suitable in our case when dealing with limited information about the cost distribution from the literature. The cost values are generated as follows:

- α_{eg} and α'_{rg} are randomly selected from the interval $[1000, 2500]$ and $[20000, 40000]$, respectively.
- β_{eg} and β'_{rg} are randomly selected from the interval $[600, 2200]$ and $[28000, 32000]$, respectively.
- γ_{eg} and γ'_{rg} are randomly selected from the interval $[200, 500]$ and $[7000, 8000]$, respectively.
- λ_{eg} and λ'_{rg} are randomly selected from the interval $[100, 250]$ and $[3500, 4000]$, respectively.

5.2. Performance of approaches

Herein, we evaluate the performance of the proposed $MILP$ and AA in terms of the solution quality and execution time. Table 1 includes two parts for different sizes of instances and indicates the number of solved instances within the time limit, the average integrality gap (in %) for non-solved instances, the average CPU time (in s) for the $MILP$, the average CPU times spent in the master and sub-problems, and the average number of iterations that the master problem is resolved with an updated set of scenarios (N_{master}) for the AA. Table 1

shows that the MILP cannot solve some large-size instances to optimality, and the integrality gap of CPLEX is given. Note that for this size the time limit for the CPLEX is set to 60000s. Assembly line design and balancing decisions are made every several years, and the user does not require a fast response time. Therefore, such a time limit is reasonable for the user in such decision-making processes. Moreover, it helps us to benchmark the AA against good quality solutions to avoid bias in our analysis. Also, the last column shows that the number of iterations of AA is smaller for some large-size instances than small-size cases. The reason is that we set less number of N iterations with no solution improvement to stop the algorithm. However, AA still finds a better solution much quicker than $MILP$ (the solution quality is given and discussed in Table 2). Table 1 shows that AA can solve the instances much faster than $MILP$. Note that the CPU time for the sub-problem given in this Table is the sum of all iterations of doing the sub-problem (not for a single run of the sub-problem).

Table 1: Average computational time (s) of the approaches.

Size	$MILP$			AA		
(P, S, O)	N° solved	Integrality	CPU	CPU time (s)		N_{master}
	instances	opt. gap (%)	time (s)	Master problem	Sub-problem	
(2, 4, 20-24-29)	10/10	0.0	747.1	136.4	0.2	7.7
(2, 7, 50-60-72)	10/10	0.0	8356.3	1054.8	2.8	8.6
(2, 10, 100-120-144)	8/10	4.8	41065.6	4959.7	19.1	5.1

Table 2 shows the final solution (cost objective function value) provided by $MILP$ and AA. The objective function value (OFV) of the $MILP$ is the average cost for the instances of different sizes. Two values in two different columns (*OFV (master problem)* and *OFV (sub-problem)*) are given for the solution of AA. The values in column *OFV (master problem)* are the average final objective function values obtained by AA where the worst scenario found by the sub-problem is integrated with other existing scenarios. The values in column *OFV (sub-problem)* are the average best values obtained by the sub-problem for only the worst

scenario found by the sub-problem. Since the developed AA performs as a heuristic to solve the problem, it may not be able to find the same value as in $MILP$ but it finds a very close objective value to the value obtained by $MILP$. The first Gap (%) column shows the average difference between the final solution of AA and the best solution of $MILP$. The gap is calculated using Formula (37). Moreover, the last column $N_{huristic}$ shows the number of instances (out of 10 for each size) where the OFV of the sub-problem is not equal to the one for the master problem. In these instances, the sub-problem is not able to find the optimal worst scenario in the last iteration, and the suboptimal worst scenario found by the sub-problem was already in the set of scenarios in the master problem. The last Gap (%) column also uses the same Formula (37) to calculate the difference between the OFVs of the sub-problem and the master problem for each instance, but the cost values of the master and sub-problems are replaced to the cost values of $MILP$ and $AA-master$, respectively.

$$Gap = \frac{Cost(MALRP^{MILP}) - Cost(MALRP^{AA-master})}{Cost(MALRP - W^{MILP})} 100\% \tag{37}$$

Table 2: Solution quality of the approaches.

Size	$MILP$	AA		Gap (%)	$N_{huristic}$	Gap (%)
(P, S, O)	OFV	OFV (master problem)	OFV (sub-problem)	$MILP/AA$		AA (master/sub)
(2, 4, 20-24-29)	80455.2	80285.0	79895.1	0.2	2/10	0.5
(2, 7, 50-60-72)	84323.5	81657.2	80892.0	2.9	4/10	1.0
(2, 10, 100-120-144)	86883.2	89831.6	89831.6	- 11.0	0/10	0.0

The performance of the proposed AA shows that the approach is working very well for small and medium-size instances where it finds the solution with less cost. In the same context, the $MILP$ gives the smaller cost for large-size instances and it performs better for the large size. However, the comparison of approaches that solve robust optimization by approximating the worst case with heuristics is not straightforward. For instance, the larger cost observed for AA may mean that AA performs quite well to generate the worst-case

solution for the large-size instances (worst with higher cost value) solution compared to the $MILP$ (with 11% cost difference) much faster in a reasonable time. That is, the scenarios that correspond to the worst-case cost may not be part of the sampled scenario tree. Note that the AA was not able to find the higher cost value compared to the solution of $MILP$ for only 1 instances among 10 in large-size instances. Moreover, the sub-problem provides the same result as the master problem which means that the algorithm converges to the same value.

5.3. Managerial insights

This section provides several insights stemming from the approaches' performance evaluation. First, it compares the performance of the proposed robust model and the performance of the classical model for the problem at hand. Second, a sensitivity analysis is given for the different values of some parameters of the model. The values reported in this section concern only a set of small size instances $((2, 4, 20 - 24 - 29) - size)$.

5.3.1. Comparison of the robust model and a classical model

This section describes a classical model (named $MALRP_{Classic}$) to compare our robust model $MALRP$ with the classical approach $MALRP_{Classic}$.

The $MALRP_{Classic}$ refers to the case where the design and reconfiguration planning decisions are made at each period for each production generation. In this case, the user does not foresee future generations, and decisions are not made at the design stage considering all future production generations regarding the possible scenarios. Here, the model optimizes the design/redesign of the line between each consecutive generation for each product family with the same set of scenarios as in the $MILP$. We fix the design of equipment and the resources for each corresponding period, then we optimize the reconfiguration planning for the following periods. Finally, we sum up all the assembly costs of the worst scenario of each generation over the life cycle of the line (all generations). We also evaluate the performance of

the classical approach via a simulation model. The simulation does the same as the classical model considering a given scenario at each period.

To compare our robust model with the classical one, we resolve the example introduced in 3.4 for the classical model. The robust model outperforms the classical model with a cost of 73247 versus 74014. The reason is because of the period-by-period decision-making in the classical model. The cost parameters related to equipment and robots that have been used in the solution of the classical model are given as; $\alpha_{20} = 1027$, $\alpha_{60} = 2096$, $\alpha_{01} = 1349$, $\alpha_{21} = 1263$; $\beta_{20} = 614$, $\beta_{60} = 1667$, $\beta_{01} = 933$, $\beta_{21} = 882$; $\gamma_{20} = 207$, $\gamma_{60} = 438$, $\gamma_{01} = 257$, $\gamma_{21} = 276$; $\lambda_{20} = 66$, $\lambda_{60} = 82$, $\lambda_{01} = 152$, $\lambda_{21} = 142$; $\alpha'_{20} = 32077$, $\alpha'_{21} = 33924$; $\beta'_{20} = 21902$, $\beta'_{21} = 23602$; $\gamma'_{20} = 7000$, $\gamma'_{21} = 7701$, $\lambda'_{20} = 3260$, and $\lambda'_{21} = 3697$. The detailed calculation of cost functions are given in Appendix B.

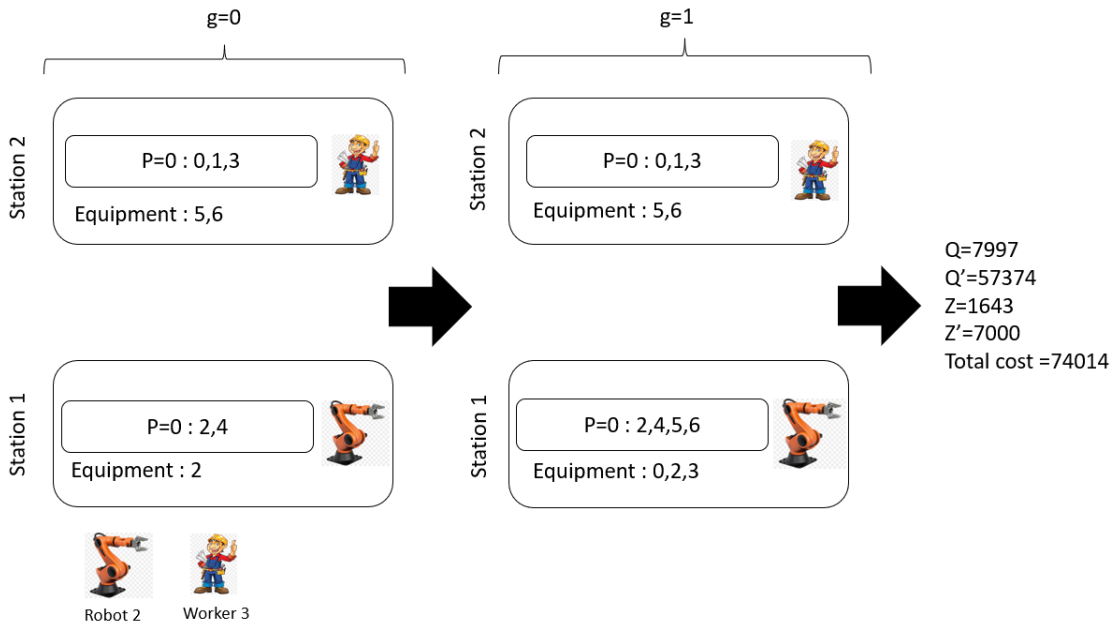

Figure 7: The solution of the illustrative example solved by the classical model.

[Alt-Text:]The solution of the illustrative example with the classical model (including the required resources, equipment, task assignments, and cost values) for two possible scenarios of the current design and one future production generation.

Table 3 shows that $MALRP$ results in significantly lower costs than $MALRP_{Classic}$ (16.7%). This table also shows that $MALRP_{Classic}$ is well simulated using the evaluation model and the evaluated model $MALRP^{Ev}_{Classic}$ provides a very close cost value to the $MALRP_{Classic}$ (5.6%). These gap values are calculated using the adaption of Formula (37). Note that the costs in the simulation are lower because the expected cost values are reported.

Table 4 shows that the studied robust model $MALRP$ results, in significantly lower costs than $MALRP_{Classic}$, but it consumes more computational time. However, we show in the previous section that AA significantly reduced the time to solve the mode. For example, for the corresponding small size of instances, AA lasts in average 136.6s (in Table 1) which is

Table 3: Comparison of our model $MALRP$ and the classical model $MALRP_{Classic}$.

Gap (%)	
$MALRP_{Classic}/MALRP$	$MALRP_{Classic}\ /MALRP^{Ev}_{Classic}$
16.7	5.6

even faster than the $MALRP_{Classic}$ which lasts in average 180.9s. Table 3 shows that our robust model can save 16.7% of the total costs compared to the classical model. However, concerning the detailed cost values in Table 4, it can be seen around 18% of such cost-saving belongs to the purchasing/selling costs of robots and workers, and installation/un-installation cost of equipment, robots, and workers. On the contrary, the classical model $MALRP_{Classic}$ saves around 4.8% of the purchasing/selling costs of the equipment (Q) over all instances, whereas the total average value of such a cost in the $MALRP_{Classic}$ is still slightly higher than our morel $MALRP$. The robust model purchase flexible equipment that will ease future reconfigurations. As a result, the $MALRP_{Classic}$ invests more (around 20% than the $MALRP$) on the design cost of the line by purchasing costs of equipment and resources ($Q + Q'$) and then for the reconfiguration of the line by (un-)installation costs of equipment and resources ($Z + Z'$), especially for the equipment.

Table 4: Comparison of the detailed cost values and CPU time between our model $MALRP$ and the classical model $MALRP_{Classic}$.

Problem	Q	Q'	Z	Z'	CPU time (s)
$MALRP$	9816.0	61022.6	1866.0	8377.2	747.1
$MALRP_{Classic}$	9972.2	78803.6	2730.6	12237.6	180.9
$MALRP^{Ev}_{Classic}$	9123.0	72675.4	1882.8	14517.0	0.7

5.3.2. Sensitivity analysis

This section analyzes the sensitivity of the solution of the $MALRP$ and the classical model $MALRP_{Classic}$ to the value of δ that controls the number of tasks in the new product variants. We consider a large value of δ ($\delta = 50\%$) and compare the results with the results of instances with $\delta = 20\%$. Note that a large value of δ corresponds to the situation where the product family changes a lot from one generation to the next. The graphs in Figures 8 show how increasing the number of tasks in the new product variants impacts the costs of the studied model $MALRP$ and also the $MALRP_{Classic}$.

The left part a of the figure shows the purchasing costs of the equipment and resources increase when the number of required tasks increases for the product models. Moreover, in part b, our proposed model does not change by increasing the number of required tasks and it keeps almost the same (un-)installation (reconfiguration) cost of the equipment and resources, whereas this cost value increases in the classical model, significantly.

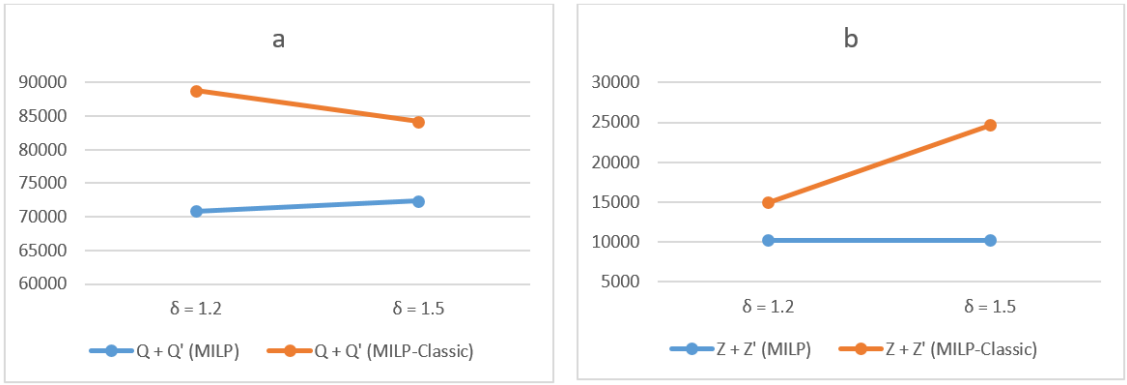

Figure 8: The impact of changing δ on the average cost values in both models $MALRP$ and $MALRP_{Classic}$: a) values of $Q + Q'$, b) values of $Z + Z'$.

[Alt-Text:]Two sides: the left side shows how changing the value of δ from 1.2 to 1.5 influences the average cost values $Q + Q'$ in both models $MALRP$ and $MALRP_{Classic}$ and the right side shows the same analysis for the cost values $Z + Z'$.

The performance of the MILP based on scenario sampling depends on the number of scenarios. Thus, we investigate the impact of omission/inclusion of scenarios on the cost value of the worst case. We eliminate a portion of scenarios ($a\%$ from all number of scenarios) in the scenario tree, and we see how the results change. Table 5 shows how the total and detailed cost values change from the main problem ($MILP - MALRP$) to other variants of the problem with $a\%$ less number of scenarios ($MILP - MALRP^{a\%}$), where $a = 10, 20, 50$. The total cost value (OFV column) and all the detailed costs decrease by removing a set of scenarios, overall. However, in some cases, the (un-)installation cost of equipment (Z) and resources (Z') at stations increase which is because of the sample of scenarios and the reconfiguration costs may vary.

Table 5: Checking the impact of omission/inclusion of scenarios on the solution.

Elimination portion	**OFV**	**Q**	**Q'**	**Z**	**Z'**
$MILP - MALRP$	80455.2	9816.0	61022.6	1866.0	8377.2
$MILP - MALRP^{5\%}$	80120.0	9016.0	60722.2	1772.8	8609.0
$MILP - MALRP^{10\%}$	79675.1	8934.3	60561.8	1671.8	8507.2
$MILP - MALRP^{20\%}$	79012.5	8350.0	60554.9	1711.8	8395.8
$MILP - MALRP^{50\%}$	78611.2	8199.2	60513.3	1738.6	8160.1

This analysis shows the conservatism of the obtained solution. The results in Table 5 are logical because when the decision maker removes some scenarios the solution does not become worse and the total cost decreases because the worse scenarios which rarely happen in the assembly line may be ignored. We can conclude that our robust model is able to account for the risk-averse profile of the decision-maker by sampling the correct number of scenarios. When the less likely scenarios which are costly are omitted the total design and reconfiguration cost decrease.

6. Conclusion

This study deals with a mixed-model assembly line reconfiguration planning problem ($MALRP$). The problem occurs in a practical case study of the automotive industry. The line produces a set of product models from a product family, and it must be reconfigured when any changes in demand happen, or when the product family changes. The design and reconfiguration of the line are costly. Therefore, we studied a robust optimization problem where the evolution of product families in future periods is uncertain. To the best of our knowledge, we are the first to study the robust assembly line design under uncertainty on the product family. We provide a novel mixed-integer linear programming ($MILP$), and it minimizes the total cost of the initial design and future reconfigurations. The $MILP$ is based on a scenario tree that provides realizations of the product family in the future. To generate the tree, we provide a tool to sample random scenarios for future product models. An adversarial approach (AA) is also developed to approximately solve the larger instances.

Several computational results and analyses are provided over benchmark instances generated from the literature. The results show that AA performs much faster and more efficiently than the proposed $MILP$. The AA performs well for small-size instances where it can find better solutions with lower cost value. For the large size instances, the $MILP$ cannot find the optimal solution within a certain time limit, while it provides the solution with less cost compared to AA. Also, the AA is better at finding the worst-case scenario for large-size instances (worst with higher cost value) solution compared to the $MILP$, with significantly less computational time. Moreover, we show that the proposed robust model leads to significantly a lower design and reconfiguration cost for the worst-case scenario compared to the classical model where the design and reconfiguration of the line are optimized at each period without foreseeing the next generations. We analyze the impact of the variability of the product family on the cost of the line over its life cycle. The results show that the reconfiguration

costs of a flexible line (designed with the robust approach) do not vary significantly when product families change a lot in different generations. However, the robust solution requires a larger initial investment. Also, we tried to consider the impact of omission or inclusion of scenarios on the final solution of the proposed robust optimization model, where the user may find a less costly solution by omission of some less likely scenarios.

We identify several promising avenues for future research. An interesting future research effort on this study is to develop an efficient exact algorithm to solve larger instances, optimally. Moreover, the proposed generic approach to generate the scenario tree can be enhanced by characterizing the set of product families for a given joint precedence graph. Finally, learning techniques could be developed to design and evaluate the quality of the proposed uncertainty set.

Acknowledgement

The authors would like to thank the project ASSISTANT (https://assistant-project.eu/) that is funded by the European Commission, under grant agreement number 101000165, H2020–ICT-38-2020, artificial intelligence for manufacturing. The authors also would like to thank the CCIPL computing center for the generous computing resource allocation.

Disclosure statement

No potential conflict of interest is reported by the authors.

Data Availability Statement

Data supporting the findings of this study are available at a reasonable request from the authors.

References

Abbas, M., ElMaraghy, H., 2018. Co-platforming of products and assembly systems. Omega 78, 5–20.

AlGeddawy, T., ElMaraghy, H., 2012. A co-evolution model for prediction and synthesis of new products and manufacturing systems. Journal of Mechanical Design 134.

Alptekinoğlu, A., Corbett, C.J., 2008. Mass customization vs. mass production: Variety and price competition. Manufacturing & Service Operations Management 10, 204–217.

Altemeier, S., Helmdach, M., Koberstein, A., Dangelmaier, W., 2010. Reconfiguration of assembly lines under the influence of high product variety in the automotive industry–a decision support system. International Journal of Production Research 48, 6235–6256.

Becker, C., Scholl, A., 2006. A survey on problems and methods in generalized assembly line balancing. European Journal of Operational Research 168, 694–715.

Benkamoun, N., Huyet, A.L., Kouiss, K., 2013. Reconfigurable assembly system configuration design approaches for product change, in: Proceedings of 2013 international conference on industrial engineering and systems management (IESM), IEEE. pp. 1–8.

Bienstock, D., Özbay, N., 2008. Computing robust basestock levels. Discrete Optimization 5, 389–414.

Biswas, S., Chakrabortty, R.K., Turan, H.H., Elsawah, S., 2023. Consideration of uncertainties in a dynamic modeling system integrated with a deep learning based forecasting approach. CIRP Journal of Manufacturing Science and Technology 44, 27–44.

Boysen, N., Fliedner, M., Scholl, A., 2009. Production planning of mixed-model assembly lines: overview and extensions. Production Planning and Control 20, 455–471.

Breckle, T., Manns, M., Kiefer, J., 2021. Assembly system design using interval-based customer demand. Journal of Manufacturing Systems 60, 239–251.

Bryan, A., Hu, S.J., Koren, Y., 2013. Assembly system reconfiguration planning. Journal of Manufacturing Science and Engineering 135.

Castañé, G., Dolgui, A., Kousi, N., Meyers, B., Thevenin, S., Vyhmeister, E., Östberg, P.O., 2022. The assistant project: Ai for high level decisions in manufacturing. International Journal of Production Research , 1–19.

Gembarski, P.C., Plappert, S., Lachmayer, R., 2021. Making design decisions under uncertainties: probabilistic reasoning and robust product design. Journal of Intelligent Information Systems 57, 563–581.

Hashemi-Petroodi, S.E., Thevenin, S., Dolgui, A., 2022. Mixed-model assembly line design with new product variants in production generations. IFAC-PapersOnLine 55, 25–30.

Hu, S.J., Zhu, X., Wang, H., Koren, Y., 2008. Product variety and manufacturing complexity in assembly systems and supply chains. CIRP annals 57, 45–48.

Johansen, J., 1990. Production management systems—a cim perspective. by jimmie browne, john harhen and james shivnan (addison-wesley publishing company,. 1988)[pp. 284]£ 20.00. Production Planning & Control 1, 261–261.

Khettabi, I., Benyoucef, L., Boutiche, M.A., 2021. Sustainable reconfigurable manufacturing system design using adapted multi-objective evolutionary-based approaches. The International Journal of Advanced Manufacturing Technology 115, 3741–3759.

Koren, Y., Gu, X., Guo, W., 2018. Reconfigurable manufacturing systems: Principles, design, and future trends. Frontiers of Mechanical Engineering 13, 121–136.

Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., Van Brussel, H., 1999. Reconfigurable manufacturing systems. Annals of the CIRP 48, 2.

Kucukkoc, I., Zhang, D.Z., 2014. Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. International Journal of Production Research 52, 3665–3687.

Liu, M., Liu, X., Chu, F., Zheng, F., Chu, C., 2020. Robust disassembly line balancing with ambiguous task processing times. International Journal of Production Research 58, 5806–5835.

Liu, X., Yang, X., Lei, M., 2021. Optimisation of mixed-model assembly line balancing problem under uncertain demand. Journal of Manufacturing Systems 59, 214–227.

Manzini, M., Unglert, J., Gyulai, D., Colledani, M., Jauregui-Becker, J.M., Monostori, L., Urgo, M., 2018. An integrated framework for design, management and operation of reconfigurable assembly systems. Omega 78, 69–84.

Mehrabi, M.G., Ulsoy, A.G., Koren, Y., 2000. Reconfigurable manufacturing systems: Key to future manufacturing. Journal of Intelligent Manufacturing 11, 403–419.

Molina, A., Rodriguez, C.A., Ahuett, H., Cortés, J., Ramírez, M., Jiménez, G., Martinez, S., 2005. Next-generation manufacturing systems: key research issues in developing and integrating reconfigurable and intelligent machines. International Journal of Computer Integrated Manufacturing 18, 525–536.

Otto, A., Otto, C., Scholl, A., 2013. Systematic data generation and test design for solution algorithms on the example of salbpgen for assembly line balancing. European Journal of Operational Research 228, 33–45.

Pereira, J., 2018. The robust (minmax regret) assembly line worker assignment and balancing problem. Computers & Operations Research 93, 27–40.

Pil, F.K., Holweg, M., 2004. Linking product variety to order-fulfillment strategies. Interfaces 34, 394–403.

Pirmoradi, Z., Wang, G.G., Simpson, T.W., 2014. A review of recent literature in product family design and platform-based product development. Advances in product family and product platform design: methods & applications , 1–46.

Rahman, A.A.A., 2020. Revolution of production system for the industry 4.0, in: Mass Production Processes. IntechOpen.

Schuh, G., Salmen, M., Kuhlmann, T., Wiese, J., 2017. Highly iterative product development within the tool and die making industry. Procedia CIRP 61, 576–581.

Şeker, Ş., Özgürler, M., Tanyaş, M., 2013. A weighted multiobjective optimization method for mixed-model assembly line problem. Journal of Applied Mathematics 2013.

Singh, A., Gupta, S., Asjad, M., Gupta, T., 2017. Reconfigurable manufacturing systems: journey and the road ahead. International Journal of System Assurance Engineering and Management 8, 1849–1857.

Sivasankaran, P., Shahabudeen, P., 2017. Comparison of single model and multi-model assembly line balancing solutions. International Journal of Computational Intelligence Research 13, 1829–1850.

Stief, P., Etienne, A., Dantan, J.Y., Siadat, A., 2023. A methodology for production system design driven by product modelling and analysis–application in the automotive industry. International Journal of Production Research 61, 1341–1357.

Thomopoulos, N.T., 1967. Line balancing-sequencing for mixed-model assembly. Management Science 14, B–59.

Wang, W., Koren, Y., 2012. Scalability planning for reconfigurable manufacturing systems. Journal of Manufacturing Systems 31, 83–91.

Wei, W., Ji, J., Wuest, T., Tao, F., 2017. Product family flexible design method based on dynamic requirements uncertainty analysis. Procedia CIRP 60, 332–337.

Wu, W., Huang, Z., Zeng, J., Fan, K., 2022. A decision-making method for assembly sequence planning with dynamic resources. International Journal of Production Research 60, 4797–4816.

Yanıkoğlu, İ., Gorissen, B.L., den Hertog, D., 2019. A survey of adjustable robust optimization. European Journal of Operational Research 277, 799–813.

Zhang, H., Qin, S., Li, R., Zou, Y., Ding, G., 2020. Progressive modelling of feature-centred product family development. International Journal of Production Research 58, 3701–3723.

Zhang, Z., Tang, Q., Chica, M., Li, Z., 2023. Reinforcement learning-based multiobjective evolutionary algorithm for mixed-model multimanned assembly line balancing under uncertain demand. IEEE Transactions on Cybernetics .

Appendix A. Illustrative example - input parameters

The input parameters used in the illustrative example are as follows:

$\delta = 0.2$ and $\sigma = 0.4$.

$\alpha_{30} = 1001$;$\alpha_{40} = 1140$; $\alpha_{50} = 2068$; $\alpha_{31} = 1457$; $\alpha_{41} = 1330$; $\alpha_{51} = 2293$ and $\alpha_{61} = 2299$.

$\beta_{30} = 771$; $\beta_{40} = 700$; $\beta_{50} = 1641$; $\beta_{31} = 882$; $\beta_{41} = 807$; $\beta_{51} = 1866$ and $\beta_{61} = 1952$.

$\gamma_{30} = 217$; $\gamma_{40} = 220$; $\gamma_{50} = 446$; $\gamma_{31} = 295$; $\gamma_{41} = 285$ $\gamma_{51} = 453$ and $\gamma_{61} = 470$.

$\lambda_{30} = 87$;$\lambda_{40} = 84$; $\lambda_{50} = 87$; $\lambda_{31} = 177$; $\lambda_{41} = 119$; $\lambda_{51} = 138$ and $\lambda_{61} = 112$.

$\alpha'_{30} = 25297$;$\alpha'_{10} = 32803$; $\alpha'_{31} = 27236$ and $\alpha'_{11} = 34467$.

$\beta'_{30} = 0$;$\beta'10 = 21811$; $\beta'_{31} = 0$ and $\beta'_{11} = 24680$.

$\gamma'_{30} = 0$;$\gamma'_{10} = 7286$; $\gamma'_{31} = 0$ and $\gamma'_{11} = 7565$.

$\lambda'_{30} = 0$;$\lambda'_{10} = 3353$; $\lambda'_{31} = 0$ and $\lambda'_{11} = 3666$.

Appendix B. Illustrative example - calculation of cost functions

The calculations for two scenarios (Equations (B.1) - (B.4) for scenario 1 and (B.6) - (B.9) for scenario 2) considered in the illustartive example provided in section 3.4 are detailed below. Therefore, the final total cost for two scenarios are calculated in Equations (B.5) and (B.10), respectively.

$$Q_1 = 1001 \left[\sum_{s\in\mathcal{S}} b^0_{s30} = 1\right] + 1140 \left[\sum_{s\in\mathcal{S}} b^0_{s40} = 1\right] + 2068 \left[\sum_{s\in\mathcal{S}} b^0_{s50} = 1\right] = 4209 \tag{B.1}$$

$$Q'_1 = 32803 \left[\sum_{s\in\mathcal{S}} w^0_{s10} = 1\right] + 25297 \left[\sum_{s\in\mathcal{S}} w^0_{s30} = 1\right] = 58100 \tag{B.2}$$

$$Z_1 = 217 \left[\sum_{s\in\mathcal{S}} b^0_{s30} = 1\right] + 220 \left[\sum_{s\in\mathcal{S}} b^0_{s40} = 1\right] + 446 \left[\sum_{s\in\mathcal{S}} b^0_{s50} = 1\right] = 883 \tag{B.3}$$

$$Z_1' = 7286 \left[\sum_{s\in\mathcal{S}} \; w_{s10}^0 = 1\right] = 7286 \tag{B.4}$$

$$Q_1 + Q_1' + Z_1 + Z_1' = 4209 + 58100 + 883 + 7286 = 70478 \tag{B.5}$$

$$Q_2 = 1001 \left[\sum_{s\in\mathcal{S}} \; b_{s30}^0 = 1\right] + 1140 \left[\sum_{s\in\mathcal{S}} \; b_{s40}^0 = 1\right] + 2068 \left[\sum_{s\in\mathcal{S}} \; b_{s50}^0 = 1\right] + 2299 \left[b_{610}^{+1} = 1\right] = 6508 \tag{B.6}$$

$$Q_2' = 32803 \left[\sum_{s\in\mathcal{S}} \; w_{s10}^0 = 1\right] + 25297 \left[\sum_{s\in\mathcal{S}} \; w_{s30}^0 = 1\right] = 58100 \tag{B.7}$$

$$Z_2 = 217 \left[\sum_{s\in\mathcal{S}} \; b_{s30}^0 = 1\right] + 220 \left[\sum_{s\in\mathcal{S}} \; b_{s40}^0 = 1\right] + 446 \left[\sum_{s\in\mathcal{S}} \; b_{s50}^0 = 1\right] + 470 \left[b_{s610}'^{-1} = 1\right] = 1353 \tag{B.8}$$

$$Z_2' = 7286 \left[\sum_{s\in\mathcal{S}} \; w_{s10}^0 = 1\right] = 7286 \tag{B.9}$$

$$Q_2 + Q_2' + Z_2 + Z_2' = 6508 + 58100 + 1353 + 7286 = 73247 \tag{B.10}$$

Moreover, the calculations corresponding to the illustartive example provided in section 5.3.1 are detailed below (Equations (B.11) - (B.14)). The final total cost is computed in

Equation (B.15).

$$Q_1 = 1027 \left[\sum_{s\in\mathcal{S}} b^0_{s20} = 1\right] + 2068 \left[\sum_{s\in\mathcal{S}} b^0_{s50} = 1\right] + 2096 \left[\sum_{s\in\mathcal{S}} b^0_{s60} = 1\right] + 1349 \left[b^{+1}_{010} = 1\right] + 1457 \left[b^{+1}_{310} = 1\right] = 7997 \tag{B.11}$$

$$Q'_1 = 32077 \left[\sum_{s\in\mathcal{S}} w^0_{s20} = 1\right] + 25297 \left[\sum_{s\in\mathcal{S}} w^0_{s30} = 1\right] = 57374 \tag{B.12}$$

$$Z_1 = 207 \left[\sum_{s\in\mathcal{S}} b^0_{s20} = 1\right] + 446 \left[\sum_{s\in\mathcal{S}} b^0_{s50} = 1\right] + 438 \left[\sum_{s\in\mathcal{S}} b^0_{s60} = 1\right] + 257 \left[{b'^{-}_{s010}}^{1} = 1\right] + 295 \left[b'^{-1}_{s310} = 1\right] = 1643 \tag{B.13}$$

$$Z'_1 = 7000 \left[\sum_{s\in\mathcal{S}} w^0_{s20} = 1\right] = 7000 \tag{B.14}$$

$$Q_1 + Q'_1 + Z_1 + Z'_1 = 7997 + 57374 + 1643 + 7000 = 74014 \tag{B.15}$$

Appendix C. The nomenclature and definitions of sets, parameters, and variables

All the sets, indices, parameters, and decision variables are defined in detail as follows: [htbp]

Table C.6: The list of sets and indices proposed in $MILP$.

Sets and indices	
$\mathcal{SC}$	Set of scenario n $\in \{1, ..., N\}$.
$\mathcal{G}$	Set of production generation g $\in \{0, ..., G\}$.
$\mathcal{PS}_g$	Set of product families p,p' in generation g (p,p' $\in \{1, ...,\mathrm{PS}_g\}$).
$\mathcal{I}$	Set of product models i $\in \{1, ..., I\}$.
$\mathcal{S}$	Set of stations s $\in \{1, ..., S\}$.
$\mathcal{E}$	Set of equipment e $\in \{1, ..., E\}$.
$\mathcal{R}$	Set of resources r $\in \{1, ..., R\}$.
$\mathcal{O}$	Set of assembly tasks o $\in \{1, ..., O\}$.
O_i	Set of assembly tasks required for product model i.
O'_p	Set of assembly tasks required for product family p.
CR_e	Set of resources certified to use the equipment e.
CE_o	Set of equipment capable to perform task o.

Table C.7: The list of parameters used in $MILP$.

Sets and indices	
C	The takt time.
A_i	The set of precedence ralations between pairs of tasks (pairs of tasks o and o' where task o' precedes task o) required for product model i.
A'_p	The weighted integrated precedence graph of all product models in family p.
α_{eg}, α'_{rg}	The purchasing cost of equipment e and robot r (or hiring cost of worker r) in production generation g, respectively.
β_{eg}, β'_{rg}	The selling price of equipment e and robot r in production generation g, respectively.
λ_{eg}, λ'_{rg}	The installation cost of equipment e and robot r in production generation g, respectively.
γ_{eg}, γ'_{rg}	The un-installation cost of equipment e and robot r in production generation g, respectively.
m^g_{ip}	The volume of the market demand (percentage) for product model i in product family p in generation g.
pt^g_{oep}	The average time of each task o executing by equipment e in the joint graph of the product family p at generation g.
pt^g_{oeip}	The processing time of each task o of the certain product model i when it is executed by equipment e in product family p at generation g.

Table C.8: The list of decision variables used in $MILP$.

Decision variables	
Y	Continuous variable of the worst (maximum) total design and re-configuration cost value.
Q_n	Continuous variable of the purchase/selling cost of the equipment.
Q'_n	Continuous variable of the purchase/selling cost of the robots and hiring/firing the workers.
Z_n	Continuous variable of the installing/uninstalling cost of the equipment.
Z'_n	Continuous variable of the installing/uninstalling cost of the robots and workers.
w^g_{srp}	binary variable equal to 1 if resource r is assigned to station s for producing product family p in generation g, and 0 otherwise.
x^g_{sop}	binary variable equal to 1 if task o performed on product family p is performed at station s during production generation g, and 0 otherwise.
b^g_{sep}	Binary variable equal to 1 if equipment e is installed at station s for producing product family p in generation g, and 0 otherwise.
$b^{+\,g}_{epp'}$	Binary variable equal to 1 if equipment e must be provided at the line in generation g switching from product family p' (in previous generation g-1) to p (in generation g), and 0 otherwise.
$b^{-\,g}_{epp'}$	Binary variable equal to 1 if equipment e must be eliminated from the line in generation g switching from product family p' (in previous generation g-1) to p (in generation g), and 0 otherwise.
$b'^{+\,g}_{sepp'}$	Binary variable equal to 1 if equipment e must be provided at station s in generation g switching from product family p' (in previous generation g-1) to p (in generation g), and 0 otherwise.
$b'^{-\,g}_{sepp'}$	Binary variable equal to 1 if equipment e must be eliminated from station s in generation g switching from product family p' (in previous generation g-1) to p (in generation g), and 0 otherwise.
$w^{+\,g}_{rpp'}$	Binary variable equal to 1 if resource r must be provided at the line in generation g switching from product family p' (in previous generation g-1) to p (in generation g), and 0 otherwise.
$w^{-\,g}_{rpp'}$	Binary variable equal to 1 if resource r must be eliminated from the line in generation g switching from product family p' (in previous generation g-1) to p (in generation g), and 0 otherwise.
$w'^{+\,g}_{srpp'}$	Binary variable equal to 1 if resource r must be provided at station s in generation g switching from product family p' (in previous generation g-1) to p (in generation g), and 0 otherwise.
$w'^{-\,g}_{srpp'}$	Binary variable equal to 1 if resource r must be eliminated from the station s in generation g switching from product family p' (in previous generation g-1) to p (in generation g), and 0 otherwise.

Appendix D. Mathematical model for the sub-problem ($MILP_{sub}$)

The mathematical formulation of the $MILP_{sub}$ is given as follows:

$$\max\ Q + Q' + Z + Z' \tag{D.1}$$

s.t.

$$Q = \sum_{e\in\mathcal{E}} \left[\alpha_{eg-1}\ \left[\sum_{s\in\mathcal{S}}\ b_{se}^{*g-1}\right] + \alpha_{eg}\ b_e^{+g} + \beta_{eg}\ b_e^{-g}\right] \qquad g\in\mathcal{G}, |G|=1 \tag{D.2}$$

$$Q' = \sum_{r\in\mathcal{R}} \left[\alpha'_{rg-1}\ \left[\sum_{s\in\mathcal{S}}\ w_{sr}^{*g-1}\right] + \alpha'_{rg}\ w_r^{+g} + \beta'_{rg}\ w_r^{-g}\right] \qquad g\in\mathcal{G}, |G|=1 \tag{D.3}$$

$$Z = \sum_{e\in\mathcal{E}} \lambda_{eg-1} \sum_{s\in\mathcal{S}}\ b_{se}^{*g-1} + \sum_{s\in\mathcal{S}} \sum_{e\in\mathcal{E}} \left[\lambda_{eg}\ b_{se}^{\prime+g} - \gamma_{eg}\ b_{se}^{\prime-g}\right] \qquad g\in\mathcal{G}, |G|=1 \tag{D.4}$$

$$Z' = \sum_{r\in\mathcal{R}} \lambda'_{rg-1} \sum_{s\in\mathcal{S}}\ w_{sr}^{*g-1} + \sum_{s\in\mathcal{S}} \sum_{r\in\mathcal{R}} \left[\lambda'_{rg}\ w_{sr}^{\prime+g} - \gamma'_{rg}\ w_{sr}^{\prime-g}\right] \qquad g\in\mathcal{G}, |G|=1 \tag{D.5}$$

$$\sum_{s\in\mathcal{S}}\ b_{se}^{g} - \sum_{s\in\mathcal{S}}\ b_{se}^{*g-1}\ \leq\ b_e^{+g} \qquad e\in\mathcal{E}, g\in\mathcal{G}, |G|=1 \tag{D.6}$$

$$\sum_{s\in\mathcal{S}}\ b_{se}^{*g-1} - \sum_{s\in\mathcal{S}}\ b_{se}^{g}\ \leq\ b_e^{-g} \qquad e\in\mathcal{E}, g\in\mathcal{G}, |G|=1 \tag{D.7}$$

$$b_{se}^{g} - b_{se}^{*g-1}\ \leq\ b_{se}^{\prime+g} \qquad s\in\mathcal{S},\ e\in\mathcal{E}, g\in\mathcal{G}, |G|=1 \tag{D.8}$$

$$b_{se}^{*g-1} - b_{se}^{g}\ \leq\ b_{se}^{\prime-g} \qquad s\in\mathcal{S},\ e\in\mathcal{E}, g\in\mathcal{G}, |G|=1 \tag{D.9}$$

$$\sum_{s\in\mathcal{S}}\ w_{sr}^{g} - \sum_{s\in\mathcal{S}}\ w_{sr}^{*g-1}\ \leq\ w_r^{+g} \qquad r\in\mathcal{R}, g\in\mathcal{G}, |G|=1 \tag{D.10}$$

$$\sum_{s\in\mathcal{S}}\ w_{sr}^{*g-1} - \sum_{s\in\mathcal{S}}\ w_{sr}^{g}\ \leq\ w_r^{-g} \qquad r\in\mathcal{R}, g\in\mathcal{G}, |G|=1 \tag{D.11}$$

$$w_{sr}^{g} - w_{sr}^{*g-1}\ \leq\ w_{sr}^{\prime+g} \qquad s\in\mathcal{S}, r\in\mathcal{R}, g\in\mathcal{G}, |G|=1 \tag{D.12}$$

$$w_{sr}^{*g-1} - w_{sr}^{g}\ \leq\ w_{sr}^{\prime-g} \qquad s\in\mathcal{S}, r\in\mathcal{R}, g\in\mathcal{G}, |G|=1 \tag{D.13}$$

$$\sum_{s\in\mathcal{S}}\ x_{so}^{g}\ =\ 1 \qquad o\in\mathcal{O}'\cup\mathcal{O}'', g\in\mathcal{G}, |G|=1 \tag{D.14}$$

$$\sum_{s\in\mathcal{S}} b^g_{se} \;\leq\; 1 \qquad e\in\mathcal{E}, g\in\mathcal{G}, |G|=1 \tag{D.15}$$

$$\sum_{s\in\mathcal{S}} w^g_{sr} \;\leq\; 1 \qquad r\in\mathcal{R}, g\in\mathcal{G}, |G|=1 \tag{D.16}$$

$$\sum_{r\in\mathcal{R}} w^g_{sr} \;\leq\; 1 \qquad s\in\mathcal{S}, g\in\mathcal{G}, |G|=1 \tag{D.17}$$

$$x^g_{so} \;\leq \sum_{e\in CE_o} b^g_{soe} \qquad s\in\mathcal{S}, o\in\mathcal{O}'\cup\mathcal{O}'' g\in\mathcal{G}, |G|=1 \tag{D.18}$$

$$b^g_{soe} \;\leq \sum_{r\in CR_e} w^g_{sr} \qquad s\in\mathcal{S}, o\in\mathcal{O}'\cup\mathcal{O}'', e\in CE_o, g\in\mathcal{G}, |G|=1 \tag{D.19}$$

$$b^g_{soe} \;\leq\; b^g_{se} \qquad s\in\mathcal{S}, o\in\mathcal{O}'\cup\mathcal{O}'', e\in CE_o, g\in\mathcal{G}, |G|=1 \tag{D.20}$$

$$\sum_{o\in\mathcal{O}'\cup\mathcal{O}''}\sum_{e\in CE_o} {pt'_{oe}}^g b^g_{soe} \;\leq\; C \qquad s\in\mathcal{S}, g\in\mathcal{G}, |G|=1 \tag{D.21}$$

$$\sum_{s\in\mathcal{S}} s\; x^g_{so} - \sum_{s'\in\mathcal{S}} s'\; x^g_{s'o'} \;\leq M(1-M_{oo'}) \qquad o\in\mathcal{O}'\cup\mathcal{O}'', o'\in O'', g\in\mathcal{G}, |G|=1 \tag{D.22}$$

$$b^{+g}_e, b^{-g}_e \in [0,1] \qquad e\in\mathcal{E}, g\in\mathcal{G}, |G|=1 \tag{D.23}$$

$$b'^{+g}_{se}, b'^{-g}_{se} \in [0,1] \qquad s\in\mathcal{S}, e\in\mathcal{E}, g\in\mathcal{G}, |G|=1 \tag{D.24}$$

$$w^{+g}_r, w^{-g}_r \in [0,1] \qquad r\in\mathcal{R}, g\in\mathcal{G}, |G|=1 \tag{D.25}$$

$$\pi^g_{soe} \in [0,1] \qquad s\in\mathcal{S}, o\in\mathcal{O}', e\in CE, g\in\mathcal{G}, |G|=1 \tag{D.26}$$

$$x^g_{so} \in \{0,1\} \qquad s\in\mathcal{S}, o\in\mathcal{O}'\cup\mathcal{O}'', g\in\mathcal{G}, |G|=1 \tag{D.27}$$

$$b^g_{se} \in \{0,1\} \qquad s\in\mathcal{S}, e\in\mathcal{E}, g\in\mathcal{G}, |G|=1 \tag{D.28}$$

$$w^g_{sr} \in \{0,1\} \qquad s\in\mathcal{S}, r\in\mathcal{R}, g\in\mathcal{G}, |G|=1 \tag{D.29}$$

$$b^g_{soe} \in \{0,1\} \qquad s\in\mathcal{S}, o\in\mathcal{O}'\cup\mathcal{O}'', e\in\mathcal{E}, g\in\mathcal{G}, |G|=1 \tag{D.30}$$

$$M_{oo'} \in \{0,1\} \qquad o\in\mathcal{O}'\cup\mathcal{O}'', o'\in\mathcal{O}'' \tag{D.31}$$

$$Q, Q', Z, Z' \geq 0 \tag{D.32}$$

Appendix E. The framework and details of the adversarial approach

Precisely, the master problem solves the $MILP$ for a smaller number of scenarios and provides the current design ($G = \{0\}$), then the first sub-problem optimizes the first reconfiguration of the line as well as searching for the worst product family in the first generation ($G = \{1\}$) and it keeps fixed, the design of the line, as given by the master problem. A single worst product family for this generation is found. The second sub-problem does the same for the next generation ($G = \{2\}$) by keeping fixed the design and first reconfiguration plan of the line. It finds the worst product family for the new generation. Therefore, the whole scenario for both generations which has been found by sub-problems is added to the set of scenarios existing in the master problem and we re-solve the master problem. Finally, comparing the new solution and the previous solution of the master problem, the algorithm decides whether to stop or continue. The developed approach is detailed in Algorithm 1 and drawn in Figure E.9.

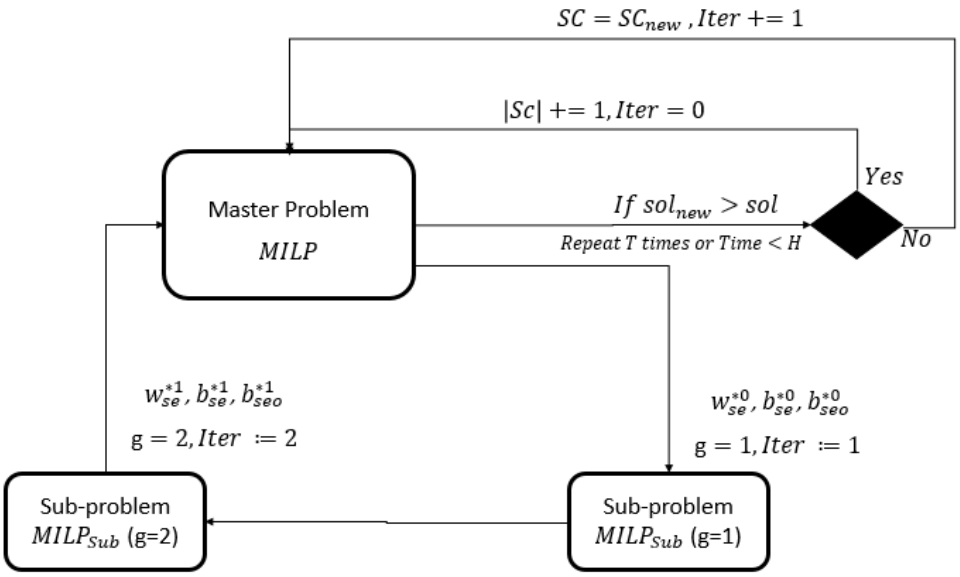

Figure E.9: The framework of the adversarial approach.

[Alt-Text:]The Figure shows the data flow between the master problem and sub-problems and also more details of the Adversarial Approach (AA) developed in this study.