



UiT The Arctic University of Norway

Department of Industrial Engineering

Develop an autonomous product-based reconfigurable manufacturing system

Morten Monland Olavsbråten

Master's thesis in Industrial Engineering, INE-3900, May 2022

Abstract

With the ever-emerging market including mass customization and product variety, reconfigurable manufacturing systems (RMS) have been presented as the solution. A manufacturing system that combines the benefits of the two classic manufacturing systems to increase responsiveness and reduce production time and costs. To cope with the lack of physical systems, an RMS system have been built at UiT Narvik. Today, both reconfiguration and deciding layout must be executed manually by a human. A task that is both incredibly time consuming and far from optimal. A method of automating the layout generation and thus the manufacturing system is presented in this thesis. To the author's knowledge such experiment has not been performed previously. Layouts is generated with a NSGA-II algorithm in Python by minimizing objectives from a developed mathematical model. The results have been tested with a MiR-100 mobile robot placing five modules in two different layouts. The results have been compared with a digital visualization for validation. In addition to the visualization, videos of the physical system's automated layout generation are presented. The results concludes that the method both generates feasible layouts as well as enhancing the automation of the system.

Keywords: RMS, MiR-100, layout problem

Acknowledgement

I would like to thank my supervisor Hao Yu and my co-supervisor Halldor Arnarson for good discussions and guidance throughout my thesis work, and Halldor for assistance when testing.

I would like to thank research fellows Shayan Dadman and Hussein Al-Sallami for good discussions. Marius and Syed for countless hours spent together at school, always in a good mood. And the rest of the class for a good study environment. In addition, I would like to thank the employees in the department and at the laboratory for facilitating for the completion of my thesis.

Table of Contents

Abstract	i
Acknowledgement.....	ii
1 Introduction	1
1.1 Background.....	2
1.2 Problem statement	2
1.3 Research objective	3
2 Literature review	4
2.1 Designing RMS	4
2.1.1 Layout design	4
2.1.2 RMT and machine selection.....	5
2.2 General optimization in RMS.....	6
2.2.1 Process planning.....	6
2.2.2 Initial configuration.....	7
2.2.3 Reconfiguration.....	8
2.2.4 Reconfiguration point.....	10
2.3 Presented RMS systems.....	10
2.3.1 Automation.....	10
2.3.2 Virtualization of RMS	11
2.4 Discussion and conclusion.....	12
3 Modelling the problem	13
3.1 Assumptions	13
3.2 Constraints	13
3.3 Dependent and independent variables	14
3.4 Mathematical modelling	14

3.4.1	Notations	14
3.4.2	Version 1	15
3.4.3	Version 2	16
4	Method	17
4.1	Software.....	17
4.1.1	Pymoo.....	17
4.2	Algorithm.....	17
4.3	Visualization.....	18
4.3.1	Python visualization	19
4.3.2	Visual components visualization.....	19
4.4	Physical system.....	19
4.5	Creation of operation scripts.....	22
4.5.1	Validating the mathematical model.....	22
4.5.2	Upscaled	26
4.5.3	RMS system placed with a fixed equipment.....	28
4.5.4	Placement in the facility	31
5	Results	32
5.1	Validating the mathematical model.....	32
5.2	Upscaled	36
5.2.1	5 Modules	36
5.2.2	Changing the location of reference points.....	38
5.2.3	11 modules	41
5.3	RMS system connected with fixed machinery	42
5.4	Finding placement in the facility	44
5.5	Physical system.....	46
6	Discussion and Conclusion	52

6.1	Discussion.....	52
6.2	Conclusion.....	53
6.2.1	Future works.....	53
	References.....	54

List of Tables

Table 1:	Dependent and independent variables that the mathematical model has to support..	14
Table 2:	Type of equipment and sizes of each module.....	19
Table 3:	Different ways of constraining to create different layouts.....	40
Table 4:	YouTube links for the physical system placing layout 1 & 2.....	51

List of Figures

Figure 1:	The five modules and the MiR-100 mobile robot that collectively represent the hardware of the RMS system.	2
Figure 2:	Figure on the left shows the marker on the module in an upright position. The figure on the right shows the marker down.	20
Figure 3:	Figure on the left shows the mobile robot in an un-dock position and figure on the right shows the dock-position.	20
Figure 4:	Script describing the connection with the OPC UA server.	21
Figure 5:	Script that utilizes the two layouts based on the coordinates found by the developed model and method.	21
Figure 6:	Describes how the resultant-vectors will search when running the script.....	22
Figure 7:	Flowchart describing how the script containing the algorithm will operate.	23
Figure 8:	Show the objectives from the mathematical model with three modules.	24
Figure 9:	Script with the implementation of the mathematical model as a class.	25
Figure 10:	The objectives shown with two groups.	27
Figure 11:	Group 2 as implemented in the script.	28
Figure 12:	Objectives shown from the reference being a fixed equipment to modules within the RMS system.	29
Figure 13:	Model for RMS system and fixed equipment implemented in Python script.....	30

Figure 14: Flowchart describing the process of finding near-optimal placement in the facility.	31
Figure 15: The Pareto-front.....	32
Figure 16: The Pareto-front with the near-optimal solution showed as a red dot.....	33
Figure 17: Best point in the search with the sizes of the objectives.....	33
Figure 18: Angle for v_1 and v_2	33
Figure 19: Length of the decomposed v_1 and v_2	34
Figure 20: V_1 and v_2 presented on the finished layout with sizes from p_2 and p_3 to centre of modules.	34
Figure 21: Shows the decomposition and adding the distance from the corner (P2 and P3) to the centre of the module in Python.....	34
Figure 22: Distances from the reference point P1 to the centre of module 2 (3D-printer) and 3 (Conveyor).	35
Figure 23: Figure on the left shows the plotted layout in Python with a caption of the equipment. Figure on the right shows the layout in Visual Components.	35
Figure 24: Another angle of the visualization in Visual Components.	35
Figure 25: The Pareto-front for group 2.....	36
Figure 26: Sizes of the objectives for group 2 from the preferred solution based on compromise programming.	36
Figure 27: Sizes from the zero module in the system (reference module group 1) to the two modules placed when solving for group 2.	36
Figure 28: Figure on top shows the plot of the 5 modules in Python. The bottom figure shows the visualization in Visual Components.....	37
Figure 29: Viewing the visualization from the end.....	38
Figure 30: A different view of the Visual Components visualization.....	38
Figure 31: Show the updated reference point for the objectives in group 2.	39
Figure 32: The unit circle with the four sectors showing possible directions for modules to be placed in.	39
Figure 33: Plotted following the constraint described in version 1 in table 3.....	40
Figure 34: Figure on the left shows the plot from version 2 in table 4. Figure on the right shows' version 3 in table 4.....	40

Figure 35: Python plot of the results with 11 modules. Red module represents 3D-printer. Yellow modules represent robot arms. Purple module is the lift-conveyor. Blue are conveyors and green represents a table.	41
Figure 36: Visual Components visualization of the results with 11 modules.....	41
Figure 37: Another angle from the visualization of the 11 modules results.	42
Figure 38: Pareto-front from the script results for the model that collaborates between the RMS system and fixed equipment.	42
Figure 39: Decomposed the resultant-vectors forming the objectives. Shows the distance from the reference point on the CNC machine to the modules.....	43
Figure 40: Comparison between the Python plot (Left) and the Visual Components visualization (Right).....	43
Figure 41: Another angle of the visualization of the RMS system together with the CNC machine.	43
Figure 42: Values for the surface covering the layout that is the basis for finding the placement in the facility.	44
Figure 43: Show the same layout as in the upscaled 5 modules results. But the coordinates on the x-axis and y-axis are showing the feasible range in the laboratory.....	44
Figure 44: The space in the figure indicates the feasible placement in the laboratory. The figure shows the layout being placed in the bottom-left corner.....	45
Figure 45: Show the same approach have been successfully implemented on the 11 modules results.	45
Figure 46: Present the coordinates that was sent to the execution script for the two layouts..	46
Figure 47: Plot (Left) and visualization (Right) for layout 1 how it should be placed in the physical system.	46
Figure 48: Figure shows layout 1 by the physical system in the laboratory.....	47
Figure 49: A different angle of Layout 1 in the laboratory.....	47
Figure 50: Comparison between the physical system (Left) and the visualization (Right) from almost the same angle.	48
Figure 51: Plot (Left) and visualization (Right) for layout 2 how it should be placed in the physical system.	48
Figure 52: Layout 2, physical system.....	49
Figure 53: Another angle of layout 2 performed by the physical system.	49

Figure 54: Visualization of layout 2 that is compared with the physical system showed in figure 51. 50

Figure 55: The physical system from the same angle as the visualization in figure 50..... 50

1 Introduction

Availability and customizability for products are higher than ever. Mass customization, with product variety and process variety, has become one of the leading production strategies [1]. Reconfigurable manufacturing system (RMS) was introduced in 1995 by Dr. Koren [2, 3] as the new production paradigm that can obtain mass customization [4, 5].

RMS combines the two traditional manufacturing systems: Dedicated manufacturing line (DML) and flexible manufacturing system (FMS) to create a rapid manufacturing system that is flexible [2]. RMS as a system also includes reconfigurable manufacturing tools (RMT), where tools can be reconfigured, and reconfigurable inspection machine (RIM). There has been formulated six core characteristics that have to be fulfilled when designing an RMS system [6, 7]. RMS use the part-family as the system focus and decisionmaker when producing, and both hardware and software must be reconfigurable. The RMS system can be understood differently, and there is not one solution for how it shall look like or be operated.

Since the term's inception, numerous studies on RMS have been published. The lack of framework for interpreting, operating, and automating RMS creates a gap between research and commercialization. The previous studies have been too focused on optimization, rather than the physical tasks of RMS. To cope with the lack of physical RMS systems, research fellow Halldor Arnarson started creating an RMS system at UiT Narvik. The system consists of both hardware and software. The hardware are five rectangular modules with different equipment attached, and a MiR-100 mobile robot. The hardware can be seen in figure 1. The developed software is made up of a structure that connects the mobile robot with the modules.

When the system has been tested, the layout for the modules has been decided and executed manually. To bring the system further, the method presented will be towards solving the layout problem in a physical RMS system. The thesis is structured with first presenting the current state of the literature. Then, the layout problem is modelled, and the method for utilizing the model. From the method, the results are shown. Lastly, it concludes with a conclusion and recommendations for further investigations and research.

1.1 Background

There have been performed tests with the mobile robot moving the modules, but the movements of the mobile robot were decided manually. Layout generation and automated initializing of layouts has not yet been investigated. A connection between mobile robot and modules have been established, but the layout and reconfiguration were configured manually and randomly. A procedure that is both time-consuming and inefficient. This experiment can be seen in the video:

https://www.youtube.com/watch?v=UXUlaawd8Ps&t=112s&ab_channel=HalldorArnarson.

By improving the automation of the layout generation and thus the physical system, the study wants to contribute to the commercialization of the concept RMS and further develop the physical system at UiT. The systems hardware is shown below in figure 1.



Figure 1: The five modules and the MiR-100 mobile robot that collectively represent the hardware of the RMS system.

1.2 Problem statement

The system at UiT has been tested with the MiR-100 placing modules in a layout, but all the inputs were decided and inserted manually. A task that is both time-consuming and far from optimal. There exists no recipe for layout generation and how they should be formed together during production. Neither does any automated reconfiguration when switching part families.

1.3 Research objective

The thesis is divided into two parts. The first part is to conduct a literature review. The literature review was both to get acquainted to the system, as well as verifying that the following research in part 2 had yet to be performed. Part I also required the development of a Python script. The script was to display the modules as a representation. This script was the foundation for the Python plot that have been used in the results. Part I milestones:

- Literature review covering previous research performed for topics covering layout design, operation, and automation in RMS.
- Python script that plots five squares which represents the modules that exists in the laboratory.

The second part is the part where the problem will be studied. With the use of an algorithm, a Python script will generate coordinates for the modules. Then, the new generated coordinates will be visualized and tested in the physical system. The milestones for part II:

- Generate a mathematical model that describes the problem.
- Create a Python script with the implementation of the algorithm and generate coordinates for the five modules.
- Visualize the script generated layout.
- Transfer the results to the physical system in the lab and compare results with the visualization.

The scope was set in the pre-study report to face the layout problem for the RMS system, by developing a mathematical model that could be solved with an optimization algorithm.

2 Literature review

The literature review highlights previous studies performed in the field of design, optimization, and automation of RMS. The review gains knowledge revolving what has been done and strengthen the purity of the research. The review is revolved around optimization of different parts in RMS and automation of the system. Web of Science is used to find scientific papers and keywords are generated with topics + “reconfigurable manufacturing system”. Some papers were discovered in the reference list from papers read.

2.1 Designing RMS

Design is critical for RMS to be considered as a cost-effective system for an enterprise [6]. Study shows that 20-50% of total cost is material handling cost, and with the proper layout, handling cost can be reduced with 10-30% [8]. The design process can be broken into three parts: Layout design, material handling system design/selection, and control system specification[9, 10].

2.1.1 Layout design

The layout design problem in RMS will only consider the current and upcoming planned production [11]. When production surpasses the initial plan, the layout problem for RMS will be solved based on current production. Traditionally, layout problems have been solved with a single-objective approach. However, RMS includes new variables such as the material handling and relocation cost, as well as the traditional constraints including work in progress (WIP), cost, and product lead-time [11]. The layout problem in RMS must consider three steps: Initial layout that is reconfigurable, indicators that can tell when the system needs reconfiguration, and the transition from the layout being used to a new layout [9]. According to Goyal and Jain [12], the most opted layout design for RMS is a flow line approach.

The layout design problem is often modelled as Quadratic Assignment Problem (QAP), which is classified as a NP-complete problem. It is described as one of the hardest optimization problems to solve [13]. Therefore, most of the studies presume it is an optimization problem with single or multiple objectives. These problems are often solved with metaheuristic techniques such as variations to genetic algorithm (GA), simulated annealing (SA), or bio-inspired multi-agent systems like particle swarm optimization (PSO) and ant colony optimization (ACO) [9, 14].

Yamada et al. [15] used PSO with a penalty system to reduce the manufacturing time. It was one of the first layout optimization studies in the field and the layout consisted of manufacturing cells instead of modules. Yamada [16] later used the same PSO approach with a penalty system to optimize the reconfiguration time of the RMS system when one manufacturing process is finalized and the system is resetting to a new manufacturing process. Guan et al. [8] used a revised version of the electromagnetism-like mechanism with encoded particles that moved discretely, instead of a continuous system. They described the layout design of RMS as a QAP problem and optimized the design layout based on material handling cost. Workstations were described as moved with an AGV robot, but no physical tests or simulations were performed.

Benderbal and Benyoucef [17] optimized machine layout and machine selection simultaneously using archived multi-objective simulated annealing (AMOS). They divided the layout problem into two sub-problems: First concerning the evolution of the product. The second the machine layout's evolution according to the first sub-problem. The optimal machine layout was determined based on the characteristics and needs for the product, the process plan, and the chosen RMT's. They found the optimal RMT layout with a search involving a penalty system. Prior to this, Benderbal et al. [18] used almost the same approach for layout design with an importance index number given to the different machines, and a penalty system. They emphasized the gap in the literature on layout design for RMS.

2.1.2 RMT and machine selection

RMT give the opportunity to use multiple machine-configurations in the same production. RMT has been described as highly important for RMS' reconfigurability and can be grouped into basic modules and auxiliary modules; Auxiliary modules can be changed, and a machine with this technology is said to have multiple configurations [19]. Different approaches have been proposed for optimal machine selection.

Goyal et al. [19] used a hybrid approach with non-dominated sorting genetic algorithm-2 (NSGA-II) algorithm to determine optimal machine selection. They used technique for order of preference by similarity to ideal solution (TOPSIS) to sort results. New performance measures were developed to ensure near as possible optimal selection. Benderbal et al. [20] used a similar approach with NSGA-II algorithm and TOPSIS for machine selection to minimize time and maximize flexibility. Bensmaine et al. [21] used the NSGA-II algorithm to

reduce total time and total cost. It was based on the process plan. Goyal et al. [22] highlighted that focusing too heavily on cost for RMT may result in a less responsive system, and that performance criteria for RMT and RMT convertibility should be oriented more toward responsiveness. Benderbal et al. [23] introduced the modularity index, and used an AMOSA approach for minimizing completion time, minimizing cost, and maximizing the modularity of the system when machine selection. They used 4 basic modules and 6 auxiliary modules and stated the increased responsiveness of the system. Xie et al. [24] applied a novel GA to find optimal RMT configuration and process plan concurrently. The importance of producing RMT configuration and process plans simultaneously was highlighted for increased productivity. Production cost was minimized, and they improved the nonmonetary performance.

2.2 General optimization in RMS

In 2020, Sabioni et al. [25] published a state-of-the-art literature review, listing all previous works performed on optimization in RMS. This chapter has been separated into four sub-chapters based on their literature review and analysis of the literature: Process planning, configuration, reconfiguration and reconfiguration point.

2.2.1 Process planning

The process plan contains the sequence the product should be produced in, and how the operation should be organized [20]. Often in the literature the configuration and machine selection are found based on the process planning. Bensmaine et al. [26] used a simulation based NSGA-II for process planning in RMS to minimize total cost and time when producing a certain amount of one part type. Dahane et al. [27] used the same technique but with multiple part types, and machine reliability as a constraint. They combined process planning with machine selection. Chaube et al. [28] modified the NSGA-II algorithm and analysed the requirements of a new product. By comparing requirements with the machine-functionality they could determine production feasibility.

Touzout et al. [29] solved the multi-objective process plan generation problem by minimizing cost and time and including greenhouse gases as a sustainability criterion. They used the multi-objective integer linear programming (MOILP) method and compared the results to those produced by AMOSA and NSGA-II. Bensmaine et al. [30] proposed a new heuristic that examines every available machine for new production, directed towards process planning and

scheduling. Benderbal et al. [20] introduced the flexibility index that would help avoid any disruption caused by machine unavailability. Benderbal et al. [18] later improved generating process plans with implementation of unavailability constraints for machines when solving the machine layout problem. Benderbal et al. [31] also introduced a new system index: the robustness index. By using the index, machine selection was based on availability. Benderbal et al. [32] later used NSGA-II to solve a multi-objective including said robustness index and minimizing time. Choi et al. [33] made a production planning method with multi-objective: minimizing energy consumption and maximizing throughput. They suggested a solution with both automated guided vehicles (AGV) and conveyors.

2.2.2 Initial configuration

Configuration of RMS is the co-operation between machine layout, machine selection and process planning. The configuration is the procedure in which the system is assembled. Configuration of the RMS system can be grouped into three: Machine-level, system-level, and both combined [4]. Machine-level is a single machine configuration, while on the system-level configuration can both include configuration of layout and without. Huang et al. [34] describes the configuration design of RMS as having three main blocks. System configuration design, cell configuration design, and RMT configuration design. They used living system theory (LST) to optimize the overall configuration design of RMT based on information flow, material flow, and energy flow.

Lv et al. [35] studied series and parallel configurations, and stated that those in solitude is not sufficient. Therefore, the hybrid parallel-series configuration should be applied for RMS. Reliability was used as a factor for optimizing system configuration and they applied their theories successfully in an enterprise in Shanghai. Maniraj et al. [36] found optimal configuration based on a specific part applying ACO. The objective function was solved by minimizing capital cost. Dou et al. [37] used GA with graph theory to optimize configuration for single-product flow-line configurations by minimizing capital cost. The genetic algorithm proved to be effective for middle-to large scale problems and can also be efficient for reconfigurations. Dou et al. [38] applied a two-stage optimization approach with full topological sorting algorithm and machine graph augmentation algorithm to generate optimal configuration with minimizing capital cost. It was proven to only be effective for small to medium sized problems, single objective and only for one part for one demand period. Goyal

et al. [39] optimized the configuration for a single-flow RMS line using a multi-objective PSO. They tweaked the PSO algorithm with NSGA-II. Their optimization had fitness values for cost and utilization. Ashraf et al. [40] created a framework for finding the optimal configuration in RMS when there are constraints. They stated configuration as one of the key issues for RMS that needed solving to make RMS a mature manufacturing system. With use of NSGA-II algorithm they found optimal configuration. The solution could also be used to help develop process planning.

2.2.3 Reconfiguration

Reconfiguration describes the rerouting of the manufacturing system when changing from one part family to another. The manufacturing systems reconfiguration activities can be grouped into hard and soft. Hard reconfiguration is physical, adding/removing machines and machine layout, while soft reconfiguration is logical, such as reprogramming machines, changing software etc. In the literature there have been a higher focus on the hard reconfiguration [9].

Objectives to minimize/maximize, also called performance measures, are important to ensure maximum output when reconfiguring. Mittal et al. [41] explained how literature often lacked reconfiguration cost as one of the performance measures. If optimization was based on reconfiguration cost, the company could save cost when reconfiguring between different part families. Xiaowen et al. [42] applied GA to optimize reconfiguration when the system moves from producing one part to another. In the case study five machines was used on both parts, and reconfiguration of the system was changing RMT between the operations. Saxena and Jain [43] applied a three-phase approach to determine the near-optimal RMS configuration. The first phase was planning and modelling. Phase 2 was finding possible configurations. In phase 3 they applied the AIS algorithm to find near optimal solution by minimizing the multi-objectives: Minimizing capital cost, operating cost and maintenance cost, and reconfiguration cost. Youssef and ElMaraghy [44] combined GA with reactive tabu search (RTS) to optimize selection of configuration. They highlighted the importance of selecting the optimal configuration when reconfiguration. Later, integrating machine availability for finding optimal configuration was researched by the same authors [45]. It was shown that availability affects the optimal configuration and equipment needed, and with availability considered the cost between different configurations fluctuated further. Zhao et al. [46] developed a stochastic framework where they stated three problems for optimal configuration/reconfiguration:

Optimal configurations problem, optimal selection policy problem, and performance measures. Zhao et al. [47] later used the stochastic framework to find optimal configuration from multiple feasible configurations, using both a IPEC and a IPSA algorithm. Moghaddam et al. [48] used a two-phase approach to develop a configuration design based on scalability by utilizing modular RMT's. They used integer linear programming (ILP) formulation to find the best RMT configurations, and a mixed integer linear programming (MILP) formulation to find the greatest transformations for the RMT configurations. Objectives were to minimize reconfiguration cost. Configuration design was found based on single product and it was preferable to find a solution for new products within the part family. Moghaddam et al. [49] later developed a configuration design based on the same modular RMT's, that could produce different parts in a part family. The configuration design was found with MILP and ILP, where the objectives were to minimize reconfiguration cost and overall cost. Sabioni et al. [4] used a hybrid approach with modified Brute-force algorithm (MBFA) and GA to optimize configuration on both machine and system-level simultaneously. According to the authors, this had not been done previously in the literature, finding optimal machine layout and machine selection at the same time. Optimal configuration was based on customer requirements while minimizing cost.

Sabioni et al. [5] optimized the configuration for mass-customized modular products and RMS concurrently, by combining nonlinear integer programming with GA based on customer requirements. The GA decided the following: Selection of module instance and required operations, process planning and machine selection, and layout configuration. Dou et al. [50] found a bi-objective optimization of configuration and scheduling when producing multiple parts in a part family simultaneously. The objectives were to minimize cost and tardiness. They found in a case study that both configuration generation and scheduling had to be performed concurrently for the solution to be as optimal as possible for the reconfigurable flow line producing multiple parts. Hasan et al. [51] developed a methodology for objectives to be minimized when finding configurations for multiple part families. They also presented a suggestion of the sequence of part families focusing on maximum benefit in the system. Asghar et al. [52] created a framework using a multi-objective GA to generate process plans for different parts in a product family and finding the optimal configurations based on the process plans. The framework included process plans, kinematics for machine configuration, and reconfiguration changeability. Abbasi and Houshmand [53] and Bryan et al. [54] used pre-fixed configurations for the system when reconfigured to make new part families. This approach is

easier and lowered the total cost, but the system loses its fully potential from the core characteristics point of view.

2.2.4 Reconfiguration point

Intuitively the reconfiguration point is when the production changes from one part family to another. However, some studies have been proposed to describe the process and timing of the reconfiguration. Using discrete event systems combining with a supervisor, Schmidt [55] found an optimal solution for the fastest way a new configuration could start. The new configuration started while the old configuration was finishing the current process. Based on dynamic complexity, Huang et al. [56] found RMS' state catastrophe using Cusp catastrophe's state condition to give the decision maker information about when the system should reconfigure. Hence, gain responsiveness.

2.3 Presented RMS systems

The optimization problems presented in the studies demand a system to be applied on. Even though there are few industrial commercialized cases of RMS systems, some interpretations of RMS systems have been introduced. Sanderson et al. [57] presented a system close to an RMS system, called SMART (Smart Manufacturing and Reconfigurable Technologies). The system allows different configurations and is based on the SMC Training HAS-200 system. Such systems are often based on FESTO's CP-factory, which is marketed as an Industry 4.0 technology. They have a modular structure and can be upscaled or downscaled. They have been presented by multiple authors [58, 59]. Adamietz et al. [60] created a prototype for a container-integrated RMS system that were reconfigurable and transportable. However, the reconfiguration part was not automated. These smart factories are close to RMS systems, but they are not developed on RMS' characteristics premises.

2.3.1 Automation

The idea behind automatic layout generation is that the manufacturing system can automatically be reconfigured based on the layout that is generated. Kim et al. [61] created a modular factory testbed and found reconfiguration as being the bottleneck of the system. Automation is key to fulfil RMS potential being a smart manufacturing system. Autonomous systems is able to perform operations with less detailed programming and without human control [62].

A way to automate the RMS system is by adopting an Autonomous Industrial Mobile Manipulator (AIMM) into the system. AIMM is an industrial manipulator that is connected to an autonomous mobile robot. This creates a setup that can move freely as well as performing tasks [63]. The definition of AIMM fits great for being a part of RMS' structure [64]. Andersen et al. used a LH6 manipulator mounted to a MiR100 mobile platform to create a mobile manipulator [65]. The MiR100 mobile platform was able to transport the LH6 manipulator to the location it was going to perform tasks. They showed how an AGV can be used in an industrial automated task. Arnarson and Solvang [66] published an article where they proposed splitting the traditional AIMM into two parts. AIMM was intended as a manipulator fixed to a mobile robot. By fixing the manipulator to a movable module, the mobile robot can relocate the module, and thereby be able to move multiple different modules. Chen et al. [67] proposed a new method for reconfigurable material handling system with the use of multiple mobile robots. It was solved with graph theory and first-order logic calculus and the mobile robots created the configuration modelling. Inoue et al. [68] used a mobile robot manipulator as a key component of the RMS system that could reconfigure the layout. When including the manipulator, they managed to achieve results without human powered reconfiguration of the manufacturing system. Based on their approach, a mobile robot manipulator using an AGV robot is feasible, both for high speed and high accuracy.

2.3.2 Virtualization of RMS

To secure safety and feasibility when working with automation, virtualization is an important tool. With a digital twin of the RMS system, layout design, configuration, machine selection, and reconfiguration could easily be done through a simulation in a “plug & play manner” [69]. In the simulation the designers can choose what objectives the solution should be solved on, constraints, and requirements. Zhang et al. [70] proposed a five-dimensional model-driven reconfigurable digital-twin (RDT) framework and built a digital twin for a manufacturing system solved on reconfigurability. They proved that the RDT could be used for a manufacturing system and reconfigurable tasks with the use of expandable model structure and an optimization approach. Leng et al. [71] created a prototype system where the manufacturing system was reconfigurable, and the reconfiguration was driven by the digital twin. This was done in two parts; a semi-physical simulation gathered data that was sent for optimization, and the semi-physical system received the results for verification and implementation. However, the system was not reconfigurable on its own and it is highly complex. In the state-of-the-art

paper covering DT frameworks for RMS made by Hajjem et al. [72], they concluded that the reconfiguration of the software part is not thoroughly looked into, and the relationship between flexibility with scalability, and modularity have to be considered for future DT work in RMS.

Zheng et al. [73] implemented a virtual visualization for the RMS facility layout problem using plant software. They built different modules including process module, control module, layout module, storage module, optimization module, and monitoring module. And Petroodi et al. [74] showed how the simulation approach can be applied with the collaboration of discrete event simulator and optimization of process planning and layout design. The optimization was done using a simulated annealing process through python, and the reconfigurability was gained by including mobile robots.

2.4 Discussion and conclusion

After reviewing papers within RMS there are roadblocks towards generating an RMS system. The development part is clearly lacking attention. With almost 30 years of research in the field, only a handful of the reviewed papers discuss how the modular design should look like or present any physical system. The presented systems are SMART factories and can barely be identified as an RMS system. Maganha et al. [9] investigated and showed in their state-of-the-art paper, the lack of research in RMS layout design. And literature based on layout optimization when reconfiguring has not described how it should be implemented in a physical system.

Automation should be implemented in the system for it to be regarded reconfigurable and responsive. The majority of the research reviewed has been revolved around optimization, where the solution is presented numerically, and the operational part has been neglected. Guan et al. [8] optimized layout with an AGV robot in mind, but they did not cover the operational part of the AGV robot. Yamada et al. [15] compared infrared and supersonic sensor on the robot, but it was only a simulation. To make the RMS system more valuable and tempting for the industry to implement, the development tasks and automation needs a higher degree of investigation. As stated in the literature, there have been a higher focus on the hardware/software part with optimizing machine selection, machine layout, and algorithms for configuration, rather than the implementation and application. The literature has not presented any method nor solutions to the physical layout problem.

3 Modelling the problem

Chapter 3 will describe the modelling of the problem. The problem is described as n-number of modules that needs to be placed in a feasible layout for production with the aid of an MiR-100.

3.1 Assumptions

Developing a product-based RMS system is a broad term. Especially when there is no foundation for how it should look like or be operated. Neither in the software nor the hardware. RMS systems today is developed based on guidelines found in the literature and subjective opinions. Assumptions are to be made to simplify the problem.

The modelling is proposed with the physical system in mind. Coordinates for the generated layout is the output of the mathematical model. The objective function is minimizing the distance and area between modules. Formulation of the model should not be specific to the quantity of modules in the lab but be formulated so it can be extended to include more than the five modules that exists today. To avoid having an exponential growing model and thus having the need of high computer power, modules are grouped in two or three. Meaning, if the problem is upscaled, the model will stay the same, only added more groups.

The global working order for modules is chosen by the product. In addition to the working order of the global system, each equipment have one optimal working direction, locally. For this problem, the working direction for each equipment is fixed. The facility is described as the feasible space where the RMS system can be placed physically in the lab. One module is chosen as the reference module and one point on that module will be the zero point for that group. If more groups are added they will use one of the already placed modules as the reference module.

3.2 Constraints

Constraints in the modelling ensures the validation and verification of the model when running the physical system.

1. No overlap between modules.
2. Working order of modules are decided by product family.
3. Working direction for each module is decided by working order and reach of equipment placed on each module.

4. Working direction is fixed.
5. Facility is 10x5 meters.
6. Three objectives are maximum quantity per group.

3.3 Dependent and independent variables

The modelling is proposed with regards to the variables displayed in table 1.

Independent variables	Dependent variables
Change of part family.	Modules must be added, removed, and/or moved to a new location.
Fixed equipment introduced.	Objectives must be changed slightly.
New modules introduced.	Constraints must be defined to match the sizes of new modules.
Other equipment is present in the facility.	The layout should be relocated to a feasible position.

Table 1: Dependent and independent variables that the mathematical model has to support.

3.4 Mathematical modelling

A mathematical model provides a method for solving physical problem with mathematical tools [75]. The problem is a physical problem, but for the algorithm to create new coordinates for the layout, the problem must be defined mathematically. The mathematical model should make the algorithm search for feasible coordinates in x and y direction. There are numerous approaches for developing a model that will do so, however for this problem, a vector-based approach has been chosen. The mathematical model is proposed based on the problem, the assumptions, the variables, and the constraints.

3.4.1 Notations

Notations that are being used in the mathematical model:

$$n = \text{series of integers} \quad (3.1)$$

$$x_n = \text{variable in } x - \text{direction}$$

$$y_n = \text{variable in } y - \text{direction}$$

$$xl = \text{lower limit for variables}$$

$$xu = \text{upper limit for variables}$$

$$\theta_n = \text{angle}$$

$$v_{xn} = x_n \times \cos\theta_n$$

$$v_{yn} = y_n \times \sin\theta_n$$

$$v_n = v_{xn} + v_{yn}$$

y_{dis} = shortest distance between modules in y – direction

x_{dis} = shortest distance between modules in x – direction

3.4.2 Version 1

Three resultant-vectors placed from one module to another (v_1 , v_2 , and v_3) describes the connection and distance between modules. Thus, they are the objectives to be minimized. Thereby, minimizing the distance between the modules. By constraining the problem to maximum three functions per group, the three resultant-vectors will be connected as a triangle. Three points p_1 , p_2 , and p_3 describes start/end for v_1 , v_2 , and v_3 . The minimization problem:

$$\text{Minimize } \sum \text{Distance between modules} \quad (3.2)$$

$$\text{Minimize } \sum_{n=3} f_1, f_2, \dots, f_n$$

$$f_1 = v_1$$

$$f_2 = v_2$$

$$f_3 = v_2 - v_1$$

$$\text{Minimize } \sum_{n=3} \langle v_{x1} + v_{y1} \rangle, \langle v_{x2} + v_{y2} \rangle, (v_2 - v_1)$$

Since this problem have three functions that is to be minimized simultaneously it is a multi-objective problem. Search heuristics are often used for multi-objective problems. For this problem there are no known values in the functions so the search heuristics will have to test multiple solutions to find near optimal. Variables in the search is implemented as shown in equation (3.3).

$$xl \leq x_1, x_2, y_1, x_2 \leq xu \quad (3.3)$$

$$xl \leq \theta_1, \theta_2 \leq xu$$

No overlapping, constraint nr. 1, must be constrained in the model. Thus, the optimization algorithm searching for near optimal solution cannot be allowed to use values that leads to modules crashing. It can be constrained in x-direction or y-direction, or both. Dependent on what direction, the size of either x-vector or y-vector between modules must be of a size that leaves the shortest gap between two modules larger than 0.

$$0 \leq Y_{dis} \quad (3.4)$$

$$0 \leq X_{dis}$$

3.4.3 Version 2

The first version of the mathematical model gave positive output, but for further enhancing the model, a revision was made. Third objective was unnecessary and by constraining differently, the resultant-vectors does not have to be connected. Objective 3 is thus to reduce the area produced by the two modules. Version 2 of the mathematical model is shown in equation (3.5):

Minimize: \sum Distance between modules, \sum Area taken by the placed modules (3.5)

$$\text{Minimize } \sum_{n=3} f_1, f_2, \dots, f_n$$

$$f_1 = v_1$$

$$f_2 = v_2$$

$$f_3 = (\text{abs}(v_{x1}-v_{x2})) \times (\text{abs}(v_{y1}-v_{y2}))$$

$$\text{Minimize } \sum_{n=3} \langle v_{x1} + v_{y1} \rangle, \langle v_{x2} + v_{y2} \rangle, (\text{abs}(v_{x1}-v_{x2})) \times (\text{abs}(v_{y1}-v_{y2}))$$

Lower/upper limit and constraints are executed as in version 1.

4 Method

The method will cover the work towards validation and verification (V & V) of the mathematical model and use the model to generate layouts for the physical system. The V & V will be performed in two stages. First, the method will be proposed for visualizing the layouts digitally. After verifying the solution is feasible, part two will include the testing of the solution in the physical system. The layout of the physical system should be near-identical to the visualization. The mathematical model is proposed as a solution that can be extended within the system, by adding groups. To validate this proposition, the first solution will contain three modules. Then, another group will be added to include the five modules that exists in the physical system. This chapter will showcase the methods that have been used for obtaining the results.

4.1 Software

There are an enormous number of software possibilities for finding a solution to this problem. Because this is a mathematical problem, the program must be capable of solving mathematical problems as well as being able to include search heuristics. All scripts for operating and running the physical system is built in Python. Python is chosen based on the already established connection between mobile robot and modules through Python. Because of its popularity, there have been developed multiple frameworks/libraries with optimization algorithms coded and ready to use that are available from GitHub [76-78].

4.1.1 Pymoo

Pymoo is a framework for optimization problems in Python [78]. It has 17 different algorithms that covers single, multi and many objective-optimization. Pymoo have the opportunity for both benchmarking and creating/solving own problems. The framework is updated regularly and giving the user the opportunity to choose algorithm without having to implement a new library.

4.2 Algorithm

An algorithm is going to be the decisionmaker for the new layout. Previous studies that have opted an optimization approach have used different types of search heuristics. While studying differences, it was found that Deb et al. [79] compare NSGA-II with Pareto-archived evolution strategy (PAES) and strength-Pareto evolutionary algorithm (SPEA), and concluded that the NSGA-II was dominant compared to the other two. Two different studies performed a

comparison test between NSGA-II and multi-objective PSO (MOPSO). Hojjati et al. [80] concluded that the NSGA-II gave better results while Nourbakhsh et al. [81] stated the opposite. The difference however was minor. When comparing NSGA-II with NSGA-III, Ciro et al. [82] stated the latter is better constructed to solve problems with more than two objectives. Pham et al. [83] showed a slightly higher performance as well with the NSGA-III. Ishibuchi et al. [84] compared NSGA-II and NSGA-III on multiple test problems and concluded that both were superior on different type of problems. They also experienced different results when implementing NSGA-III with different libraries. There are limited information on implementation of NSGA-III on own problems, but Blank et al. [78] created a framework for using NSGA-II in Pymoo. Therefore, NSGA-II will be used based on the framework, and former successful implementations.

NSGA-II is a non-dominated sorting genetic algorithm and described as a multi-objective optimization algorithm[79]. Meaning it can minimize/maximize two or three objectives. Compared to a genetic algorithm, the mating and survival selection are modified. The crowding distance is used to determine solutions. Furthermore, the NSGA-II uses tournament mating selection who compares the rank first, followed by the crowding distance. This is done to increase pressure on the selection process. The algorithm has a population size that determines the number of solutions generated. For each iteration, the algorithm provides an output of a dominant solution. The dominant solution is called Pareto-optimal. By its definition, a Pareto-optimal solution with multi-objectives is: “A Pareto optimum is a state where no consumer can be made better off without making another consumer worse off” [85]. The Pareto optimum solutions form a Pareto-front and one solution in the pareto-front is the closest to optimal. To find this one solution, a multi-criteria decision maker (MCDM) is applied. Here, in the form of compromise programming. To perform the compromise programming, weights are applied in the decomposition, to seek the compromise between the objectives. The chosen solution is therefore found as a minimization between the ideal and the desired solution [86].

4.3 Visualization

To swiftly validate the model and solution before applying it in the real system, two digital tools are used. A Python script generates modules represented as colored rectangles and a representation has been developed in Visual components [87].

4.3.1 Python visualization

The visualization in Python is strictly for showing the layout. Each module with the size of its physical twin is created as a colored rectangle. When the script produces new coordinates for the modules, these coordinates are used for visualizing the results. Visualizing the new layout gives the opportunity to see that no modules are overlapping, and that the layout is feasible.

4.3.2 Visual components visualization

While the visualization in Python is strictly for the purpose of visualizing the proposed layout, it has limited connection with the physical system. In Visual Components the facility has the same coordinates as the mobile robot map in the physical system. The modules are constructed with the same size as its physical twins and their appearance is intended to be similar. It will be used to visualize the layout and help with planning how the physical system should operate.

4.4 Physical system

The physical systems hardware contains of five modules and a MiR-100 mobile robot. The modules and sizes of the modules are presented in table 2.






	Equipment	Size in x-direction	Size in y-direction	
	Nachi robot	0.75	0.5	m
	Scara robot	0.88	0.6	m
	Creality CR-30	0.61	0.9	m
	Conveyor	1.5	0.52	m
	Lift conveyor	1.08	0.8	m

Table 2: Type of equipment and sizes of each module.

To allow the system the opportunity to work autonomously, the system is connected to an Open Platform Communications Unified Architecture (OPC UA). All the modules are equipped with a Raspberry Pi or small computer that can receive information from the server. In the script, a command for docking makes the mobile robot drives towards the module. When the mobile robot has stopped in front of the module, a marker attached to the module will be put in position 1. The mobile robot drives underneath the module and stops when it hits the marker. The marker in position 0 (down) and 1 (up) is shown in figure 2.



Figure 2: Figure on the left shows the marker on the module in an upright position. The figure on the right shows the marker down.

The mobile robot is equipped with two beams that with the help of a motor can be extended outwards. When extended the bars are in dock-position, shown in the right on figure 3. The modules have a construction that the beam fits into, which gives the mobile robot the ability to drag modules to a new location. The mobile robot drags the module to the position found in the script. When the mobile robot has successfully placed the module in correct position, the bars go back to zero-position (undock-position). Left of figure 3 shows the undock-position.

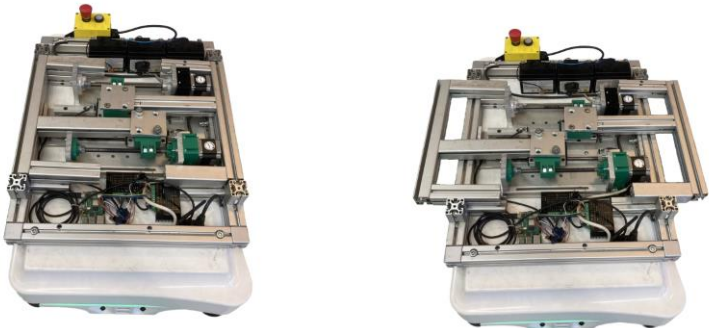


Figure 3: Figure on the left shows the mobile robot in an un-dock position and figure on the right shows the dock-position.

For the execution of the system, a script is generated where Python connects to the OPC UA server. See figure 4. The new layout that is generated is being sent to a script that executes the mission. See figure 5. The system's operation script instructs the mobile robot to dock, move to the new set of coordinates, then undock.

```

1 from opcua import Client
2
3 def connect_to OPCUA():
4     ip_address = "10.0.0.251"
5     port_number = "4820"
6     full_path = "opc.tcp://{0:s}:{1:s}".format(ip_address, port_number)
7
8     client = Client(full_path)
9     client.connect()
10    print("connect to opcua")
11
12    objects = client.get_objects_node()
13
14    for i in objects.get_children():
15        # MIR controller in OPCUA server -----
16        if (i.get_browse_name().to_string() == "2:mir228_controller"):
17            mir228_controller = i
18            # Finding all the main folders -----
19            for i in mir228_controller.get_children():
20                if (i.get_browse_name().to_string() == "2:mir228_mission"):
21                    mir228_mission = i
22                    # Going in each folder to find the objects -----
23                    for i in mir228_mission.get_children():
24                        if (i.get_browse_name().to_string() == "2:mir228_next_mission"):
25                            mir228_next_mission = i
26                        if (i.get_browse_name().to_string() == "2:mir228_busy"):
27                            mir228_busy = i
28
29    return mir228_next_mission, mir228_busy

```

Figure 4: Script describing the connection with the OPC UA server.

```

1 import RobotRunning2
2 import time
3
4 Layout_1 = [
5     ["nachi_dock"],
6     ["pos", "x, y, 0"],
7     ["nachi_un_dock"],
8
9     ["print3d_dock"],
10    ["pos", "x, y, 0"],
11    ["print3d_un_dock"],
12
13    ["lcon_dock"],
14    ["pos", "x, y, 0"],
15    ["lcon_un_dock"],
16
17    ["scara_dock"],
18    ["pos", "x, y, 0"],
19    ["scara_un_dock"],
20
21    ["conlift_dock"],
22    ["pos", "x, y, 0"],
23    ["conlift_un_dock"],
24 ]
25
26 Layout_2 = [
27    ["conlift_dock"],
28    ["pos", "x, y, 0"],
29    ["conlift_un_dock"],
30
31    ["scara_dock"],
32    ["pos", "x, y, 0"],
33    ["scara_un_dock"],
34
35    ["lcon_dock"],
36    ["pos", "x, y, 0"],
37    ["lcon_un_dock"],
38
39    ["print3d_dock"],
40    ["pos", "x, y, 0"],
41    ["print3d_un_dock"],
42
43    ["nachi_dock"],
44    ["pos", "x, y, 0"],
45    ["nachi_un_dock"],
46 ]
47
48 def execute_mir_mission(mir228_next_mission, mir228_busy, mission):
49
50    for k in range(len(mission)):
51        mir228_next_mission.set_value(mission[k])
52        #wait until mission is done
53        run = 1
54        while(run == 1):
55            time.sleep(1)
56            if(mir228_busy.get_value() == 0):
57                print("done with ",mission[k])
58                run = 0
59
60    mir228_next_mission, mir228_busy = RobotRunning2.connect_to OPCUA()
61
62    execute_mir_mission(mir228_next_mission, mir228_busy, Layout_1)
63
64    time.sleep("The time between layout 1 and layout 2")

```

Figure 5: Script that utilizes the two layouts based on the coordinates found by the developed model and method.

4.5 Creation of operation scripts

Scripts for generating layout for the physical system is divided into multiple parts. This chapter will describe the method behind the scripts, and how the results will be obtained. The first part is how the mathematical model will be validated. The second part describes upscaling the mathematical model and using the results for the physical system. Since it is a product-based RMS system, the module-order is decided by the product. The product that the system is based on is a shaker. The shaker will be produced in a 3D-printer, and then moved to an assembly step by combining pick and place robot-arms and conveyors. A method for placing the RMS system with a fixed machine is proposed as well. The last part contains a method for finding near-optimal placement in the facility for the system.

4.5.1 Validating the mathematical model

Based on the mathematical model, two resultant-vectors will emerge from a module that is the reference module. When finished in the 3D-printer, the shaker will be moved by the Nachi robot with a limited reach. Therefore, the Nachi robot is the decision-maker.

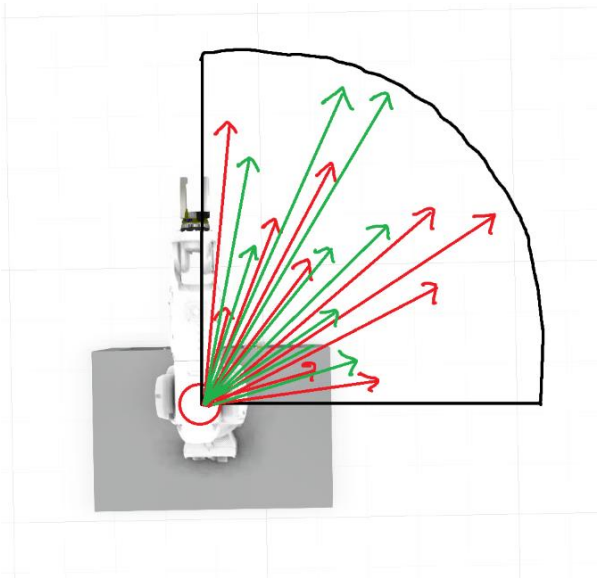


Figure 6: Describes how the resultant-vectors will search when running the script.

Figure 6 illustrates how the resultant-vectors emerges from the reference-module. Red arrow illustrates the size of v_1 , and green arrow illustrates v_2 . The end point of the arrows indicates the position of the reference point on the modules that needs positioning. Objective 3, the area, is the area between the dominant green and red arrow. A flowchart describing the script operation can be seen in figure 7.

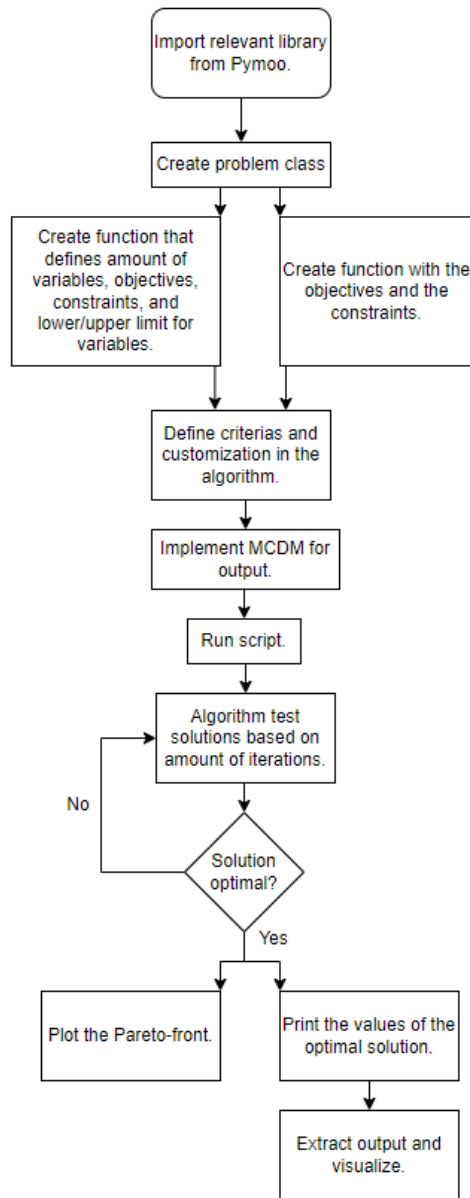


Figure 7: Flowchart describing how the script containing the algorithm will operate.

The objectives are to minimize the resultant-vector v_1 , v_2 , and minimize the area as shown in figure 8. The Nachi robot is the reference module, and the reference point can be seen as p_1 . Hence, the 3D-printer and conveyor are placed dependent on the Nachi robot. Because the Nachi robot has limited reach and its intention is to pick pieces from the printer and place them on the conveyor, that is the optimal equipment for reference. Corners for resultant-vector endpoints were chosen based on the shortest distance from how the author intended the layout to look like. However, any placement on the modules can be used for reference points.

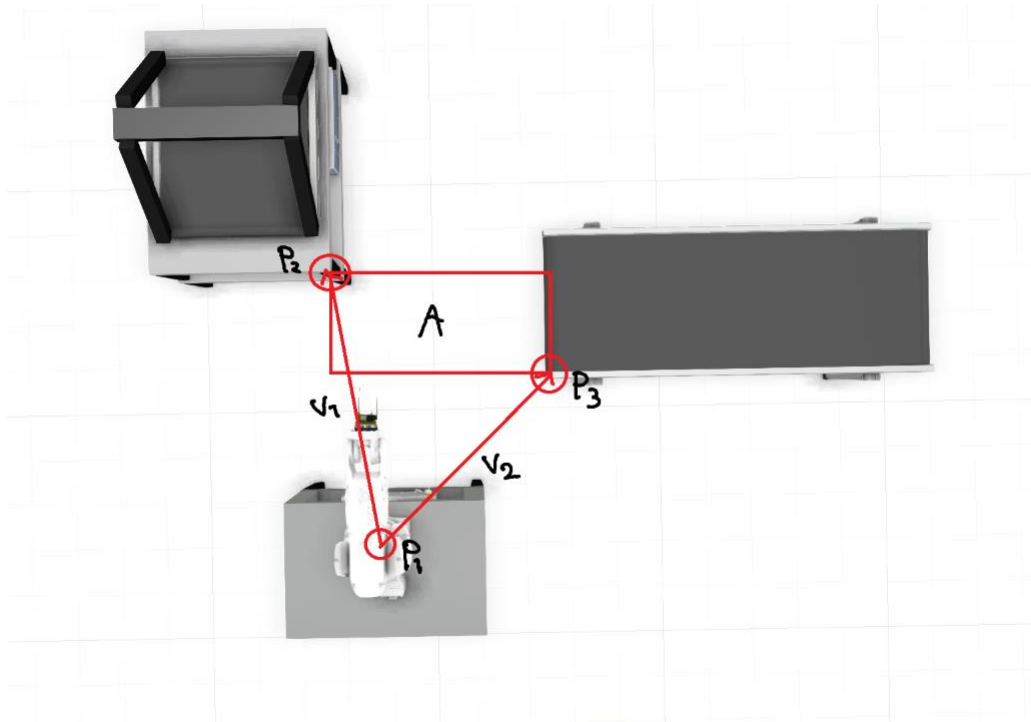


Figure 8: Show the objectives from the mathematical model with three modules.

The Nachi robot has a reach of 0.912m, which is the maximum value for the variables. Despite the Nachi robot's possibility to operate within 270°, the variable's viable input is limited for simplicity.

$$0 \leq x_1, x_2, y_1, y_2 \leq 0.912 \quad (4.6)$$

$$0 \leq \theta_1, \theta_2 \leq 90^\circ$$

Two constraints are implemented in the code to avoid any overlap between modules. 3D-printer is constrained in y-direction. The distance between center of Nachi-robot, to the end of the module it is placed on in y-direction is 0.15m. The constraint is set to 0.2m as a safety factor. Constraint 2 is constraining the conveyor in x-direction. The Nachi robot is placed in center on the module from x-direction perspective, being 0.3m from the edge of the module. Also for the x-constraint, a safety factor has been added.

$$0.2 \leq \langle v_{y1} \rangle \quad (4.7)$$

$$0.35 \leq \langle v_{x2} \rangle$$

The constraints are following this principle in future methods. The intention is allowing the algorithm to only search where there are no other modules, but empty floorspace. Worth noting that the constraints in the script are identical to the constraints in equation (4.7), but the class demands the greater-than-sign to be opposite way to generate correct results.

```

8 class Group1(Problem):
9
10     def __init__(self):
11         super().__init__(n_var=6,
12                          n_obj=3,
13                          n_constr=2,
14                          xl=np.array([0, 0, 0., 0., 0., 0.]),
15                          xu=np.array([90, 90, 0.912, 0.912, 0.912, 0.912]))
16
17         self.f1_cos = None
18         self.f1_sin = None
19         self.f2_cos = None
20         self.f2_sin = None
21
22     def _evaluate(self, x, out, *args, **kwargs):
23         self.f1_cos = [np.cos(np.radians(b)) for b in x[:,0]]
24         self.f1_sin = [np.sin(np.radians(v)) for v in x[:,0]]
25
26         self.f2_cos = [np.cos(np.radians(m)) for m in x[:,1]]
27         self.f2_sin = [np.sin(np.radians(n)) for n in x[:,1]]
28
29         modul2_x = (x[:,2] * self.f1_cos)
30         modul2_y = (x[:,3] * self.f1_sin)
31
32         modul3_x = (x[:,4] * self.f2_cos)
33         modul3_y = (x[:,5] * self.f2_sin)
34
35         area_x = abs(modul2_x-modul3_x)
36         area_y = abs(modul2_y-modul3_y)
37
38         f1 = modul2_x + modul2_y
39         f2 = modul3_x + modul3_y
40         f3 = (area_x)*(area_y)
41
42         g1 = modul2_y<0.2
43         g2 = modul3_x<0.4
44
45
46         out["F"] = anp.column_stack([f1, f2, f3])
47         out["G"] = anp.column_stack([g1, g2])
48
49     def pareto_front(self, *args, **kwargs):
50         return self._pareto_front.exec(self, *args, **kwargs)
51
52 algorithm = NSGA2(
53     pop_size=200,
54     n_offsprings=100,

```

Figure 9: Script with the implementation of the mathematical model as a class.

As seen in figure 9, the objectives, constraints, and variables are created in a class, named “Group1”. The Pymoo library have implemented a results command that deploys the algorithm to produce output from the class. There are some customizations implemented in the algorithm that can alter the results.

Kalyanmoy Deb [88] showed that NSGA-II have moved very close to the true Pareto-optimal front after 100 generations. Based on this, the algorithm was programmed to terminate the search after generation 100. The population size is set to 200 and offspring to 100, which results in 20100 function evaluations ($200*100+100$). The sampling has been set to “real random” for the initial population. This is because the vectors and area represent the foundation of a future layout, yet there are no known variables prior to the search. A simulated binary crossover has been included. This will create offspring during the evolution and ultimately enhance the results of the search[89]. To further secure realistic results, and increase the diversity in the population, a mutation operator is implemented in addition. The mutation works together with the binary crossover and create a few mutated results that will be evaluated in the evolution of the results[89].

The operators have stayed consistent for all searches and results.

4.5.2 Upscaled

While the method for generating a layout for three modules are for validating the mathematical model and potential revisions of the mathematical model, the upscaled approach are the basis for the physical system. The former validation is used as guiding principles in this approach. By expanding to five modules, the mathematical model can be verified as having the ability to be upscaled/downscaled. Here, layout generation is divided into two groups. For the first group, the algorithm finds the near-optimal coordinates for the 3D-printer and conveyor. For the second group, the same algorithm finds the near-optimal coordinates for the Scara-robot and the lift-conveyor. The objectives are formulated similarly, but the constraints are alike. The two groups are solved in the same script.

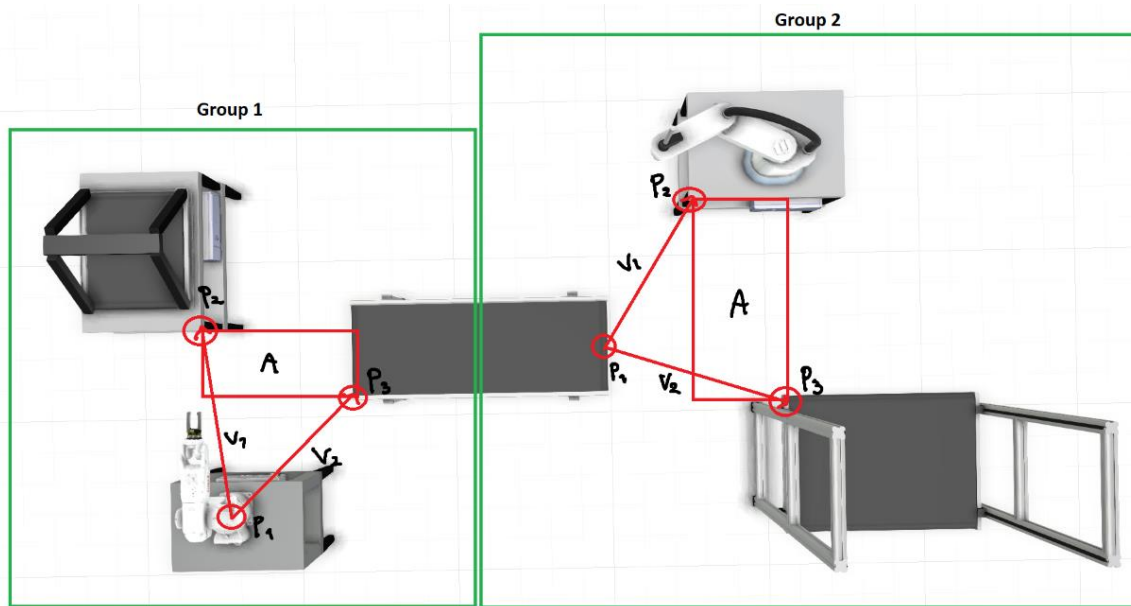


Figure 10: The objectives shown with two groups.

The constraints for group 2 are shown in equation (4.8):

$$0.3 \leq v_{y1} \tag{4.8}$$

$$0.1 \leq v_{x2}$$

$$0.2 \geq v_{y2}$$

These constraints prevent the modules from overlap. The constraints are formulated based on the validation, with one module in x-direction, and the other in y-direction. However, module 2 in group 2 (lift-conveyor) is constrained in both x and y-direction to avoid overlapping. Group 1 is implemented as in figure 9, and figure 11 show group 2 implemented similarly but with different constraints.

```

93 class Group2(Problem):
94
95     def __init__(self):
96         super().__init__(n_var=6,
97                          n_obj=3,
98                          n_constr=3,
99                          xl=anp.array([0, 0, 0., 0., 0., 0.]),
100                          xu=anp.array([90, 90, 1, 1, 1, 1]))
101
102         self.f3_cos = None
103         self.f3_sin = None
104         self.f4_cos = None
105         self.f4_sin = None
106
107     def _evaluate(self, x, out, *args, **kwargs):
108         self.f3_cos = [np.cos(np.radians(b)) for b in x[:,0]]
109         self.f3_sin = [np.sin(np.radians(v)) for v in x[:,0]]
110
111         self.f4_cos = [np.cos(np.radians(m)) for m in x[:,1]]
112         self.f4_sin = [np.sin(np.radians(n)) for n in x[:,1]]
113
114         modul5_x = (x[:,2] * self.f3_cos)
115         modul5_y = (x[:,3] * self.f3_sin)
116
117         modul6_x = (x[:,4] * self.f4_cos)
118         modul6_y = (x[:,5] * self.f4_sin)
119
120         area_x_1 = abs(modul5_x-modul6_x)
121         area_y_1 = abs(modul5_y-modul6_y)
122
123         f4 = modul5_x + modul5_y
124         f5 = modul6_x + modul6_y
125         f6 = (area_x_1)*(area_y_1)
126
127         g6 = modul5_y<0.3
128         g7 = modul6_x<0.1
129         g8 = modul6_y>0.2
130
131         out["F"] = anp.column_stack([f4, f5, f6])
132         out["G"] = anp.column_stack([g6, g7, g8])
133
134
135     def pareto_front(self, *args, **kwargs):
136         return self._pareto_front.exec(self, *args, **kwargs)

```

Figure 11: Group 2 as implemented in the script.

4.5.3 RMS system placed with a fixed equipment

Methods for generating layout in 4.5.1 and 4.5.2 have been restricted to only include the modules that forms the RMS system. In addition, it is preferably that the system can collaborate with fixed equipment. There will be situations where it is wanted to use heavy equipment that is fixed in the facility and not feasible nor possible to be moved with a mobile robot. Therefore, it is wanted to establish a scenario that describes this issue. Figure 12 illustrates a scenario where a CNC machine (5x2 meter) is fixed in the workspace, and 3 modules are to be placed accordingly to the reference point, p₁.

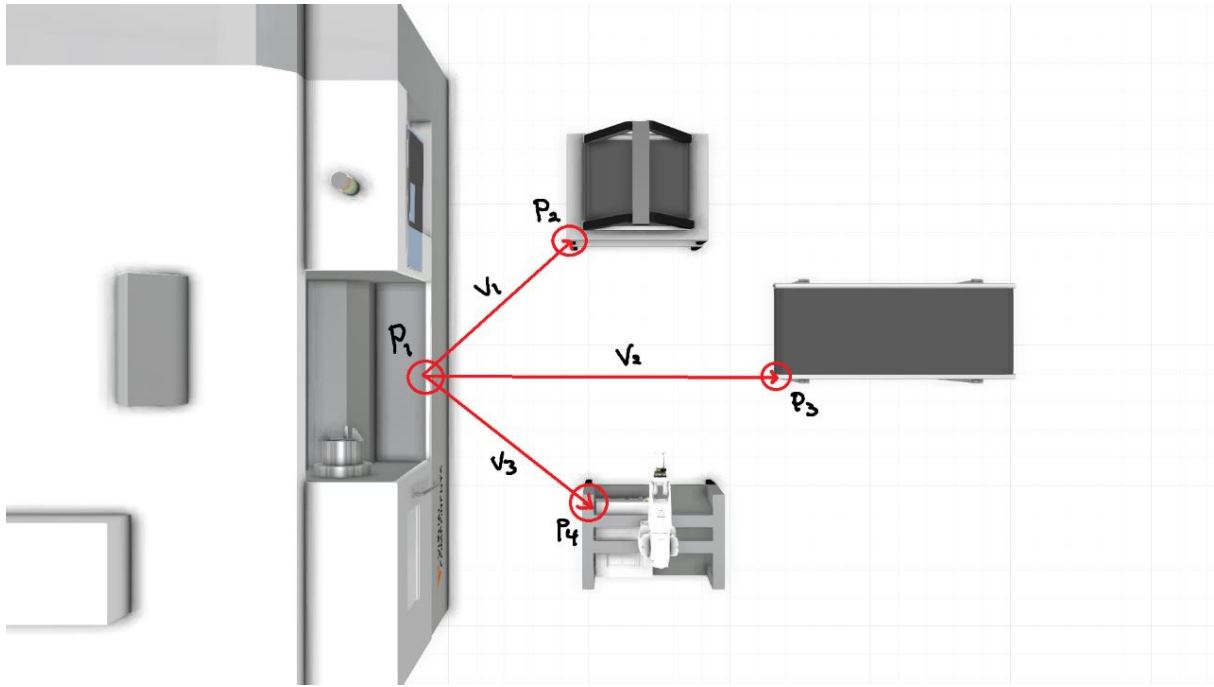


Figure 12: Objectives shown from the reference being a fixed equipment to modules within the RMS system.

Since there are three modules to be placed, the mathematical model is tweaked slightly. This also strengthen the versatility of the model.

$$\text{Minimize } \sum_{n=3} v_1, v_2, \dots, v_n \quad (4.9)$$

$$f_1 = v_1$$

$$f_2 = v_2$$

$$f_3 = v_3$$

$$\text{Minimize } \sum_{n=3} \langle v_{x1} + v_{y1} \rangle, \langle v_{x2} + v_{y2} \rangle, \langle v_{x3} + v_{y3} \rangle$$

The constraints are keeping the modules from overlapping.

$$0.5 \leq \langle v_{y1} \rangle \quad (4.10)$$

$$0.5 \geq \langle v_{y2} \rangle$$

$$1.1 \leq \langle v_{x3} \rangle$$

Constraint 4.10.1 and 4.10.2 secures that the 3D-printer and Nachi robot does not overlap in a y-direction. Constraint 4.10.3 prevents the conveyor from overlapping with the other modules. The modified mathematical model implemented in the script is showed in figure 13.

```

8   class CNC(Problem):
9
10  def __init__(self):
11      super().__init__(n_var=9,
12                      n_obj=3,
13                      n_constr=3,
14                      xl=anp.array([0, 0, 0, 0., 0., 0., 0., 0., 0.]),
15                      xu=anp.array([90, 90, 90, 2., 2., 2., 2., 2., 2.]))
16
17      self.f1_cos = None
18      self.f1_sin = None
19      self.f2_cos = None
20      self.f2_sin = None
21      self.f3_cos = None
22      self.f3_sin = None
23
24  def _evaluate(self, x, out, *args, **kwargs):
25      self.f1_cos = [np.cos(np.radians(b)) for b in x[:,0]]
26      self.f1_sin = [np.sin(np.radians(v)) for v in x[:,0]]
27
28      self.f2_cos = [np.cos(np.radians(m)) for m in x[:,1]]
29      self.f2_sin = [np.sin(np.radians(n)) for n in x[:,1]]
30
31      self.f3_cos = [np.cos(np.radians(k)) for k in x[:,2]]
32      self.f3_sin = [np.sin(np.radians(l)) for l in x[:,2]]
33
34      modul1_x = (x[:,3] * self.f1_cos)
35      modul1_y = (x[:,4] * self.f1_sin)
36
37      modul2_x = (x[:,5] * self.f2_cos)
38      modul2_y = (x[:,6] * self.f2_sin)
39
40      modul3_x = (x[:,7] * self.f3_cos)
41      modul3_y = (x[:,8] * self.f3_sin)
42
43      f1 = modul1_x + modul1_y
44      f2 = modul2_x + modul2_y
45      f3 = modul3_x + modul3_y
46
47      g1 = modul1_y<0.5
48      g2 = modul2_y>0.5
49      g3 = modul3_x<1.2
50
51      out["F"] = anp.column_stack([f1, f2, f3])
52      out["G"] = anp.column_stack([g1, g2, g3])
53
54  def pareto_front(self, *args, **kwargs):
55      return self._pareto_front.exec(self, *args, **kwargs)

```

Figure 13: Model for RMS system and fixed equipment implemented in Python script.

4.5.4 Placement in the facility

The layout is generated locally within the RMS system. When generating the layout in the facility, the facility has a feasible region where the layout can be placed. A method for finding the optimal placement in the facility is proposed, following the flowchart presented in figure 14.

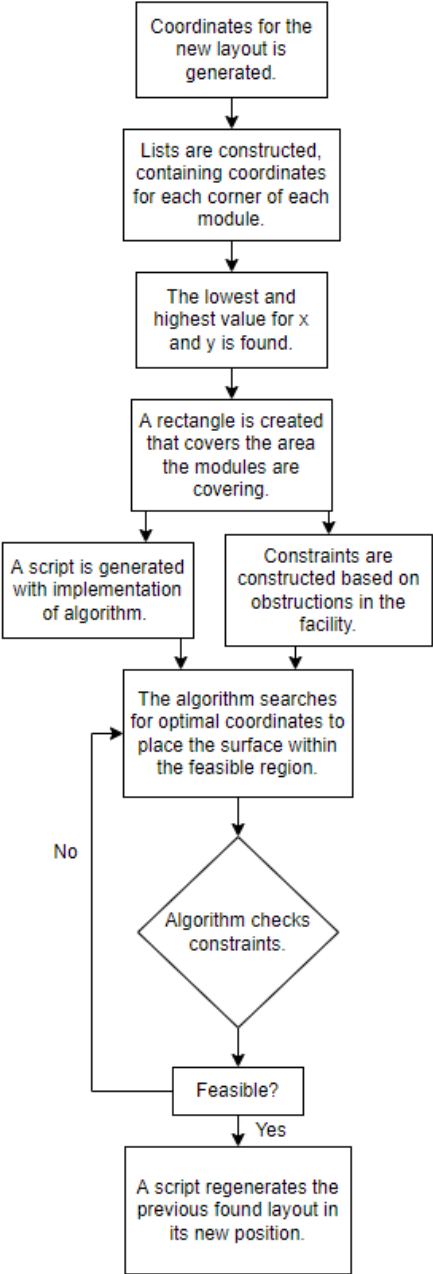


Figure 14: Flowchart describing the process of finding near-optimal placement in the facility.

5 Results

The results will display the results from the methods proposed. The results are divided into multiple parts and the last part show the results from the physical test with the link to two YouTube videos of the layout generation in the physical system.

5.1 Validating the mathematical model

When running the script presented in the method section, a Pareto front is plotted as a part of the output. The Pareto front shows the solutions the population have converged towards.

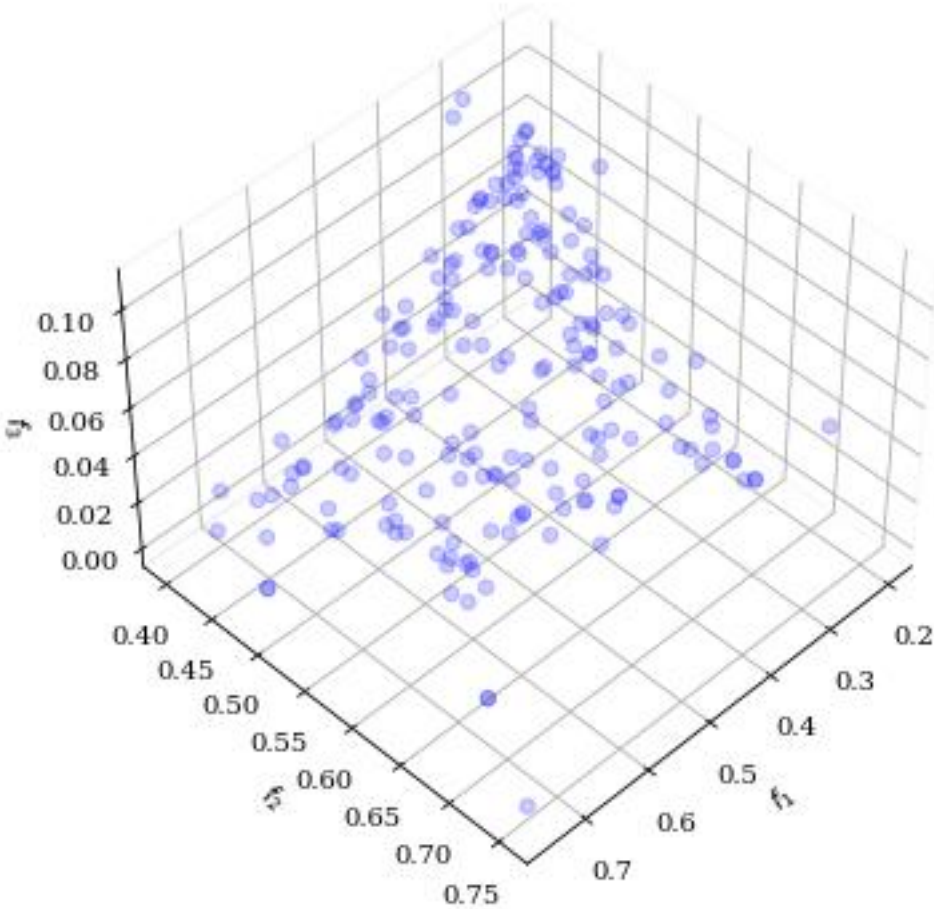


Figure 15: The Pareto-front.

Each blue dot represents one population. Even though all are dominant and feasible, there are only one solution that is the closest to optimal. By utilizing the compromise programming the closest to optimal solution is found. This solution is displayed as the red dot in figure 16.

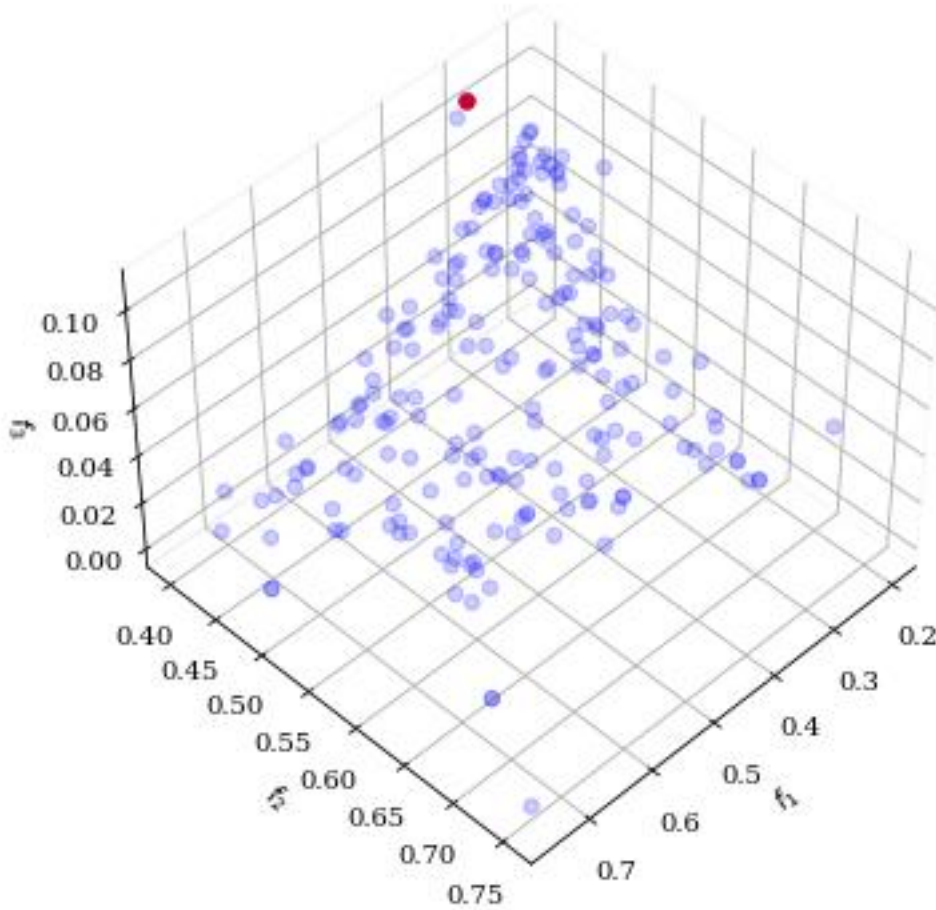


Figure 16: The Pareto-front with the near-optimal solution showed as a red dot.

This solution is stored and printed. The three values represent f_1 , f_2 , and f_3 .

```
Best point is iteration nr.: 0 - [0.30422604 0.40010819 0.10818217]
```

Figure 17: Best point in the search with the sizes of the objectives.

F_1 and f_2 are the resultant-vectors. They must be decomposed to find the distance between the reference points, in x and y-direction. The angle of the resultant-vectors related to the x-axis is stored and presented in Figure 18.

```
Angle for resultant-vector v1: 60.89786978041004
Angle for resultant-vector v2: 31.909655300293586
```

Figure 18: Angle for v_1 and v_2 .

```

Best point is iteration nr.: 0 - [0.30422604 0.40010819 0.10818217]
Distance from P1 to P2 in x direction is: 0.14796577106575035 meter
Distance from P1 to P2 in y direction is: 0.2658187641613324 meter
Distance from P1 to P3 in x direction is: 0.3396448937438059 meter
Distance from P1 to P3 in y direction is: 0.21148974648688185 meter

```

Figure 19: Length of the decomposed v_1 and v_2 .

Figure 20 illustrates the resultant-vectors in red, and the distance from p_2 to center of module 2, and p_3 to center of module 3, in yellow.

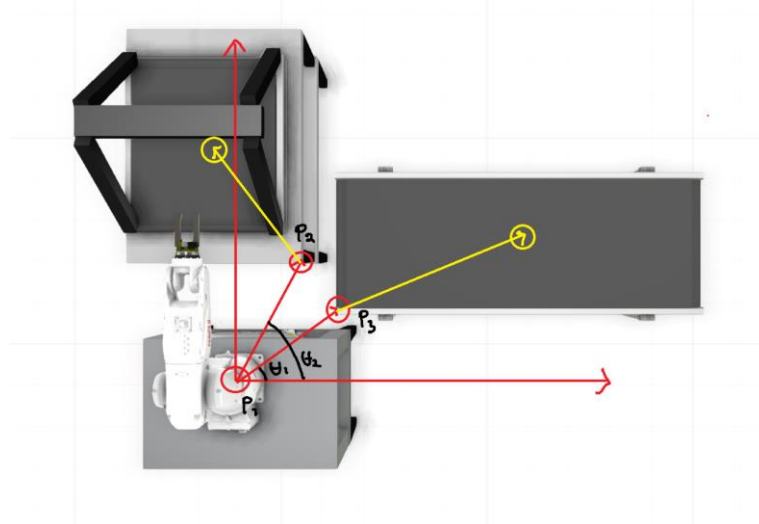


Figure 20: V_1 and v_2 presented on the finished layout with sizes from p_2 and p_3 to centre of modules.

As explained in the method section, p_2 and p_3 are positioned in the modules' corners. Both the visualization in Python and Visual Components, and the physical system uses center coordinates as input. Therefore, the decomposed vectors must be added or subtracted the distance from p_2 and p_3 to the module center, represented as yellow vectors in figure 20. Adding or subtracting depends on where the center is located in reference to what corner is used for reference.

```

85 size_3DPrinter_x = 0.61
86 size_3DPrinter_y = 0.9
87 size_Conveyor_x = 1.5
88 size_Conveyor_y = 0.52
89
90 Module2_distance_x = ((results.F[I][0]) * (my_problem.f1_cos[I])) - (size_3DPrinter_x/2)
91 Module2_distance_y = ((results.F[I][0]) * (my_problem.f1_sin[I])) + (size_3DPrinter_y/2)
92 Module3_distance_x = ((results.F[I][1]) * (my_problem.f2_cos[I])) + (size_Conveyor_x/2)
93 Module3_distance_y = ((results.F[I][1]) * (my_problem.f2_sin[I])) + (size_Conveyor_y/2)

```

Figure 21: Shows the decomposition and adding the distance from the corner (P_2 and P_3) to the centre of the module in Python.

Values printed in figure 22 are used for plotting the layout in Python and visualizing the layout in Visual components. The layout and comparison can be seen in figure 23.

```
Distance from Nachi robot to 3D-printer in x direction is: -0.15703422893424965 meter
Distance from Nachi robot to 3D-printer in y direction is: 0.7158187641613324 meter
Distance from Nachi robot to conveyor in x direction is: 1.089644893743806 meter
Distance from Nachi robot to conveyor in y direction is: 0.47148974648688186 meter
```

Figure 22: Distances from the reference point P1 to the centre of module 2 (3D-printer) and 3 (Conveyor).

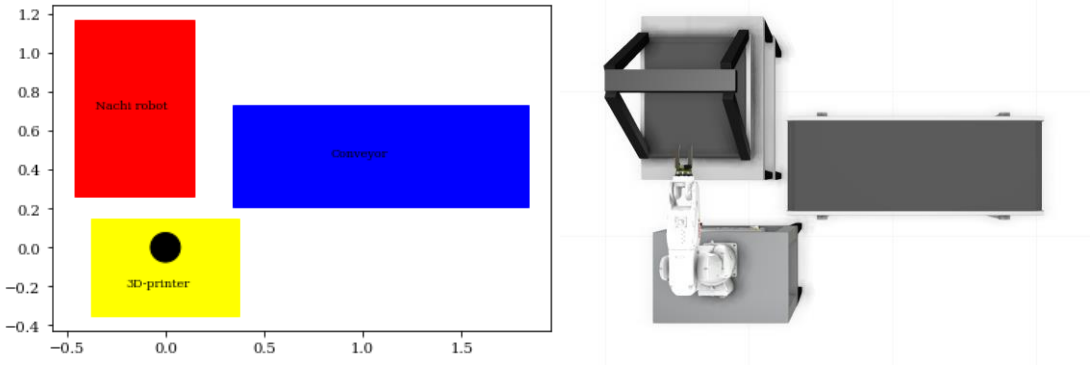


Figure 23: Figure on the left shows the plotted layout in Python with a caption of the equipment. Figure on the right shows the layout in Visual Components.

The black circle represents the position of the Nachi robot on the module. The coordinate system simply reflects the entire size of the layout and does not determine where the layout is located inside the facility.

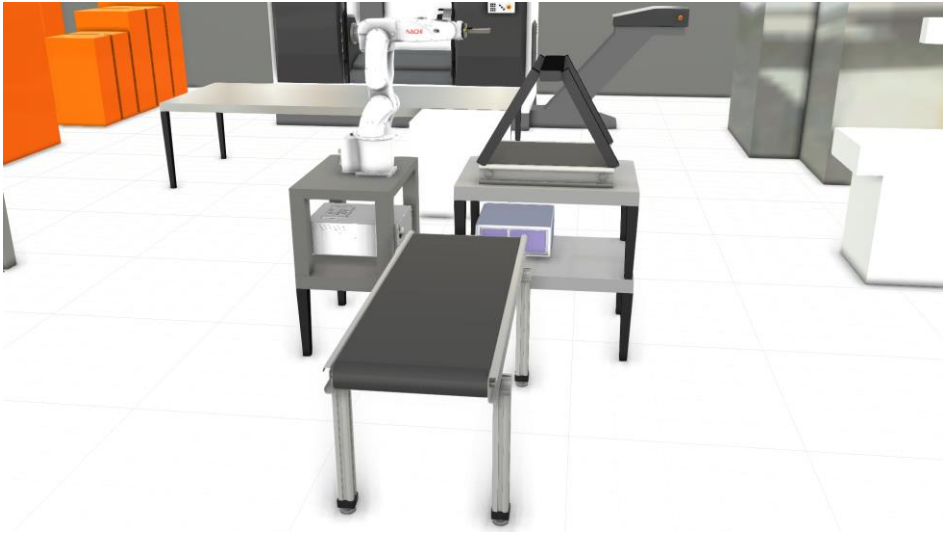


Figure 24: Another angle of the visualization in Visual Components.

The visualizations are good verification tools for the results. No modules are overlapping, and the Nachi robot can reach both modules.

5.2 Upscaled

5.2.1 5 Modules

With the upscaled version, obtaining results are divided in two. Group 1 is implemented similarly as previous, and the output is visualized in 5.1. The algorithm finds the results for group 1 first, then for group 2. Both results are printed simultaneously. Figure 25 shows the Pareto-front for group 2.

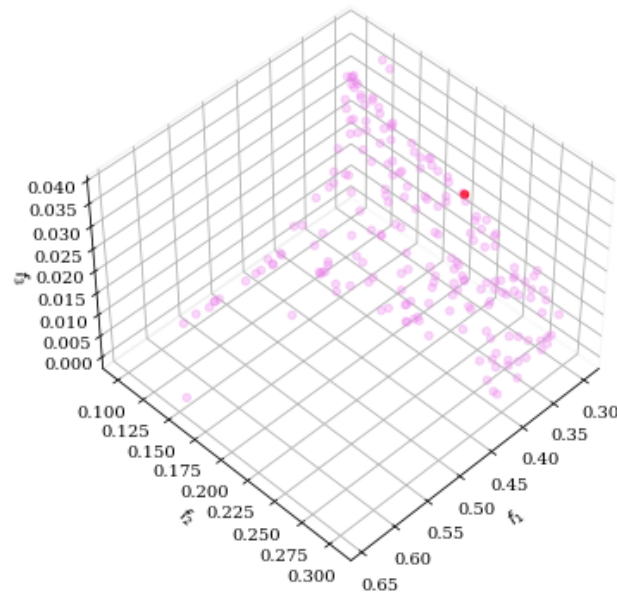


Figure 25: The Pareto-front for group 2.

Best point for group 2 is iteration nr.: Point 4 - [0.30000552 0.20480553 0.02076594]

Figure 26: Sizes of the objectives for group 2 from the preferred solution based on compromise programming.

The system is global, and group 2's layout is generated based on the layout for group 1. The position for the conveyor is stored from group 1's results and this size is added to the length of the decomposed vectors. The Scara robot and lift conveyor is placed dependent on the Nachi robots position (zero-position for the system).

```
Distance from Nachi robot to Scara robot in x direction is: 2.2827092113016456 meter
Distance from Nachi robot to Scara robot in y direction is: 1.071479611365838 meter
Distance from Nachi robot to lift-conveyor in x direction is: 2.455069529568567 meter
Distance from Nachi robot to lift-conveyor in y direction is: 0.26190095489396364 meter
```

Figure 27: Sizes from the zero module in the system (reference module group 1) to the two modules placed when solving for group 2.

The results are visualized both in the Python model as well as in Visual components. Second yellow module represents the Scara robot, and the purple module represents the lift conveyor. The visualization shows the results is feasible for the physical system.

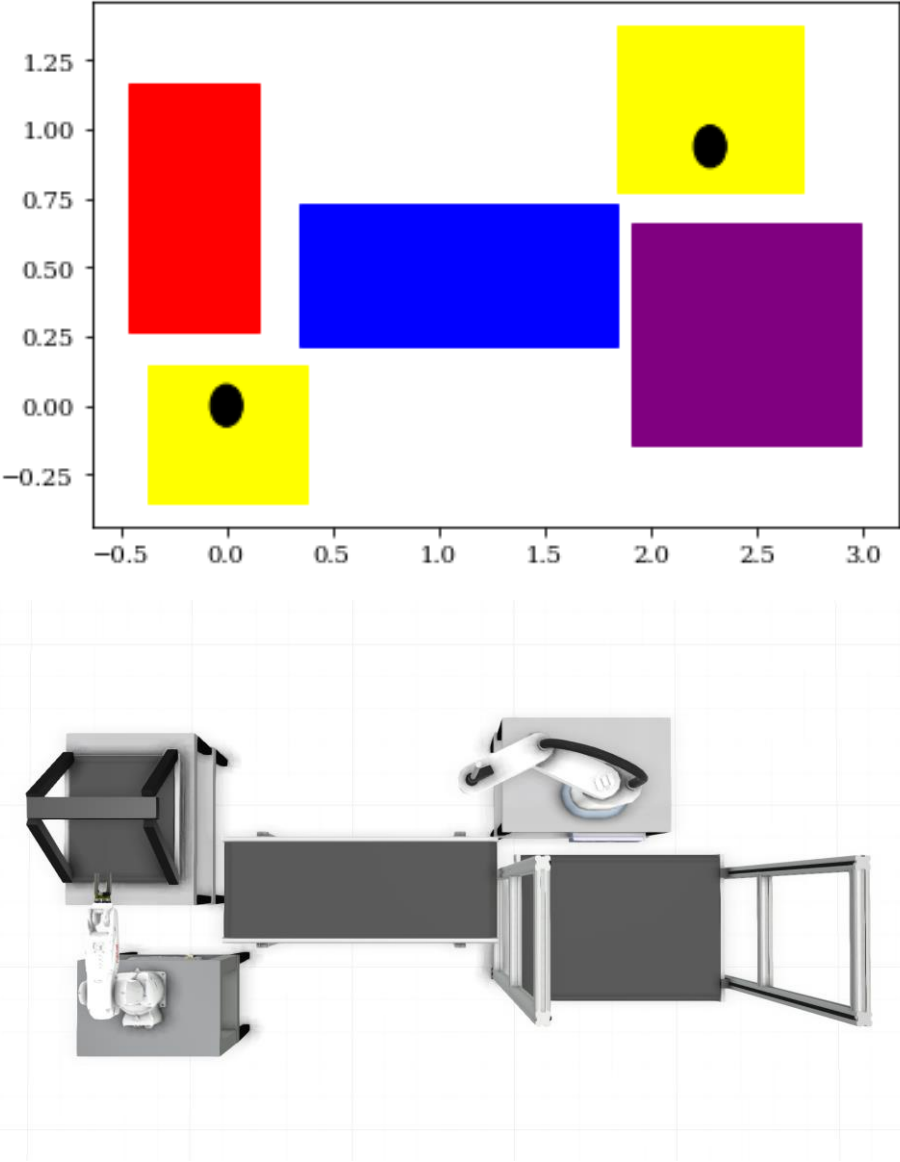


Figure 28: Figure on top shows the plot of the 5 modules in Python. The bottom figure shows the visualization in Visual Components.



Figure 29: Viewing the visualization from the end.

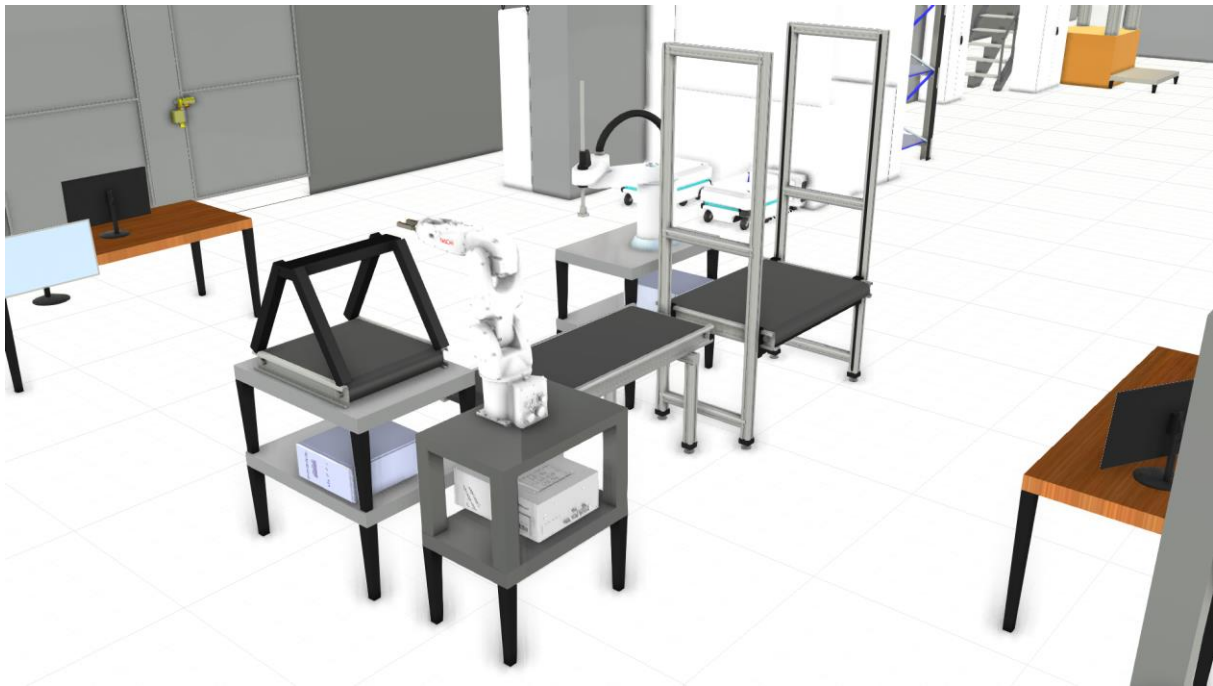


Figure 30: A different view of the Visual Components visualization.

5.2.2 Changing the location of reference points

Even though the mathematical model creates feasible results for an upscaled layout, the reference point p_1 is changed to develop a more standardized approach. Start and end points for the search vectors shown in figure 10 is placed in the end of the conveyor. While the placement

for p_1 in group 2 produces feasible output, it is not an optimal approach. Furthermore, to have the ability to upscale and downscale, while emphasizing automation, placing the reference point in the middle standardizes the approach for adding/removing modules.

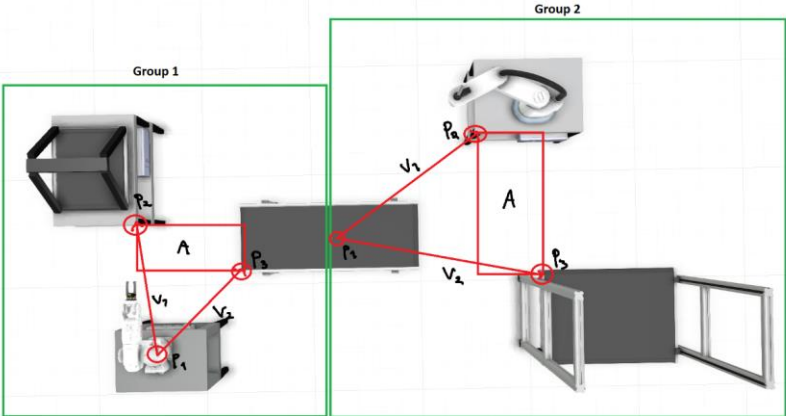


Figure 31: Show the updated reference point for the objectives in group 2.

When placing p_1 of group 2 in the center of the reference module, it gives the layout multiple ways of being executed. With the new reference point, some different layouts are tested. For the presented layouts, the algorithm has only searched for angle-values between 0° and 90° (Sector 1 in the unit circle.). When presenting different layouts, some tests have been performed with the algorithm searching within other sectors in the unit circle. However, the algorithm gives an error when increasing the angle variable above 180 degrees. This problem is solved by simulating the algorithm is searching in another sector by multiplying the decomposed x or y vector (Dependent on which sector the module is placed.). The coordinate system and the sectors are placed on the conveyor in figure 32. The conveyor is positioned as shown in figure 31.

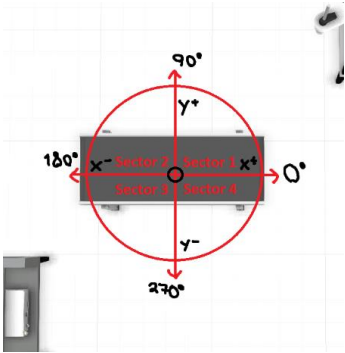


Figure 32: The unit circle with the four sectors showing possible directions for modules to be placed in.

Based on the new reference point, three different options for a layout for five modules is generated by changing the way the modules are constrained:

Version	Scara robot (yellow)	Lift conveyor (purple)
1	Positive y-direction.	Negative y-direction.
2	Positive y-direction.	Positive x-direction.
3	Positive x-direction.	Negative y-direction.

Table 3: Different ways of constraining to create different layouts.

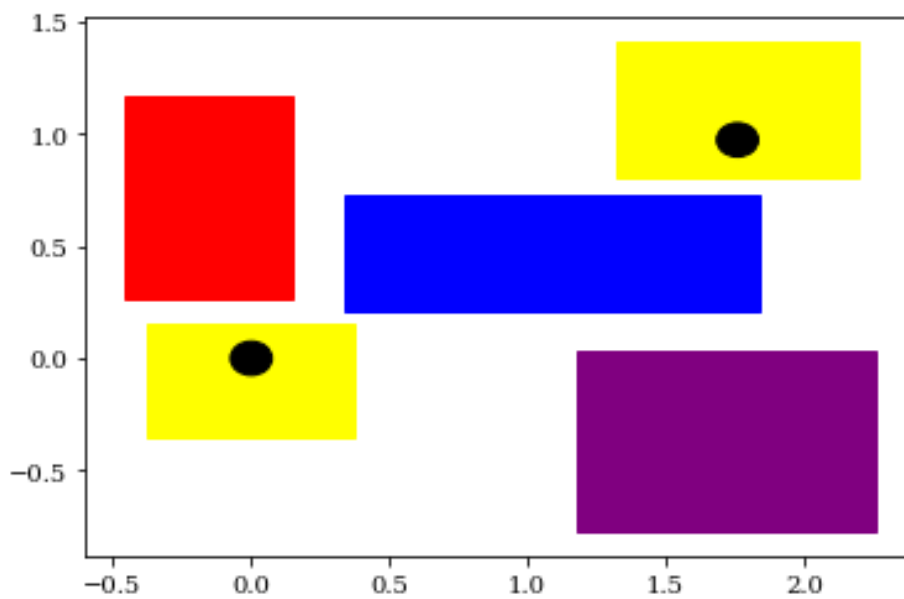


Figure 33: Plotted following the constraint described in version 1 in table 3.

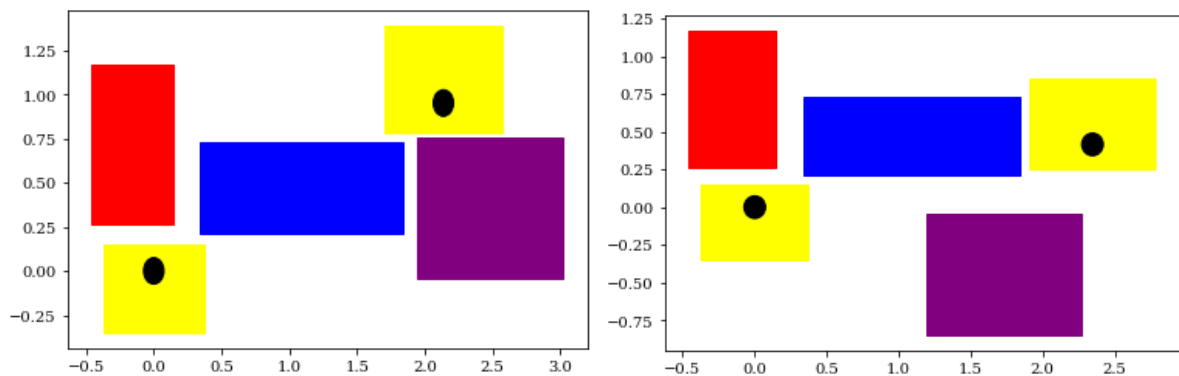


Figure 34: Figure on the left shows the plot from version 2 in table 4. Figure on the right shows' version 3 in table 4.

5.2.3 11 modules

Many manufacturing systems include more equipment and workstations than what is presented in this system. By upscaling to 11 modules there is simulated a manufacturing system with more assembly steps for the shaker. To obtain the results, it will be grouped into five groups. After the script have generated output for one group, a reference point will be placed in the center of the optimal module. Here, the optimal module is the one having a limited reach. Thus, the robots.

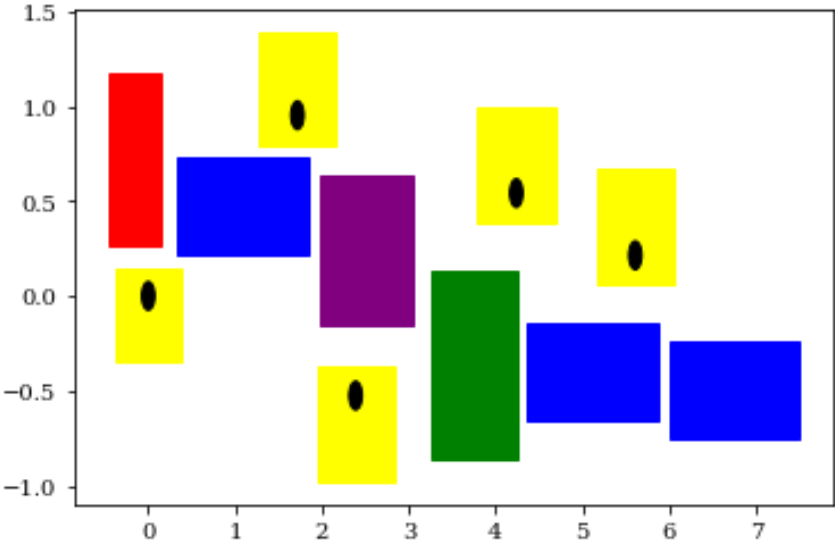


Figure 35: Python plot of the results with 11 modules. Red module represents 3D-printer. Yellow modules represent robot arms. Purple module is the lift-conveyor. Blue are conveyors and green represents a table.

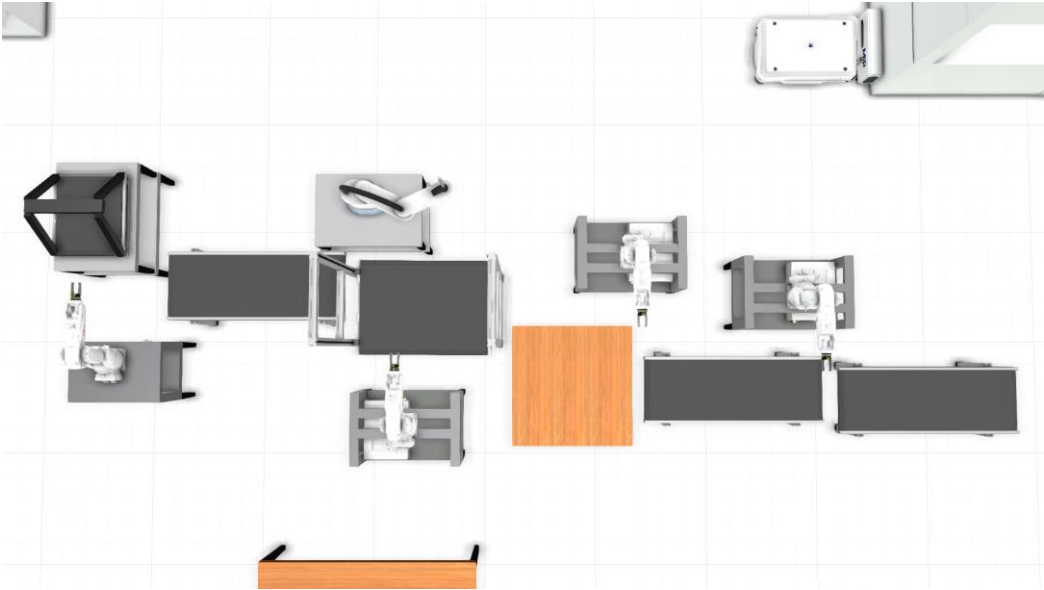


Figure 36: Visual Components visualization of the results with 11 modules.

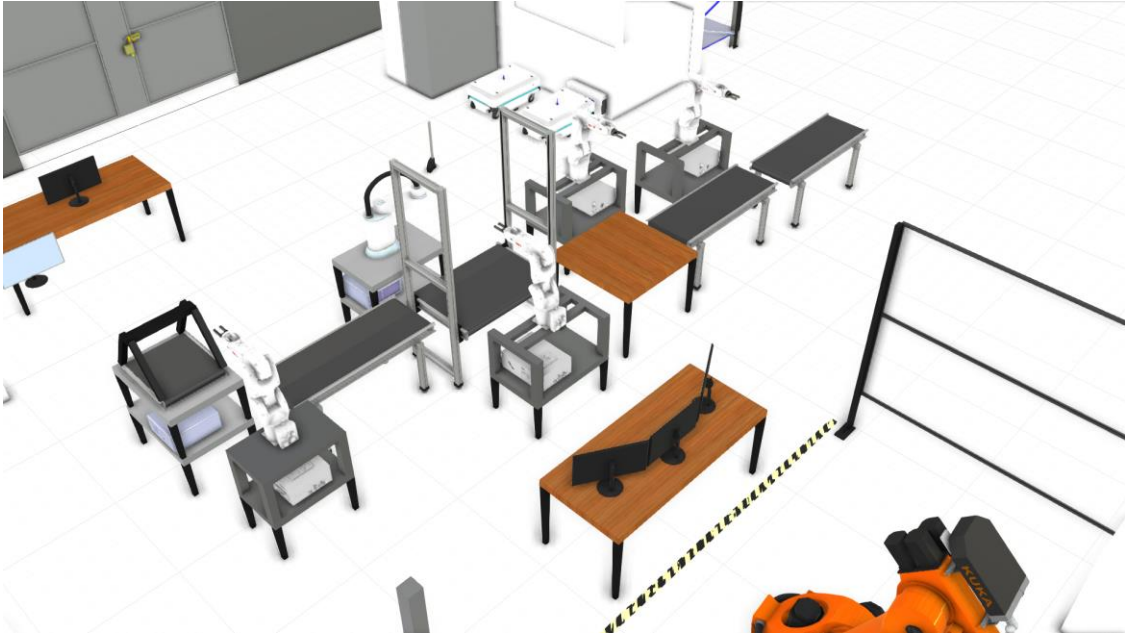


Figure 37: Another angle from the visualization of the 11 modules results.

5.3 RMS system connected with fixed machinery

When combining the RMS system with the fixed CNC machine, the script is formulated as for the validation of the mathematical model, but with the objectives that is presented in the method section. The formula for obtaining results is the same as for prior gathered results.

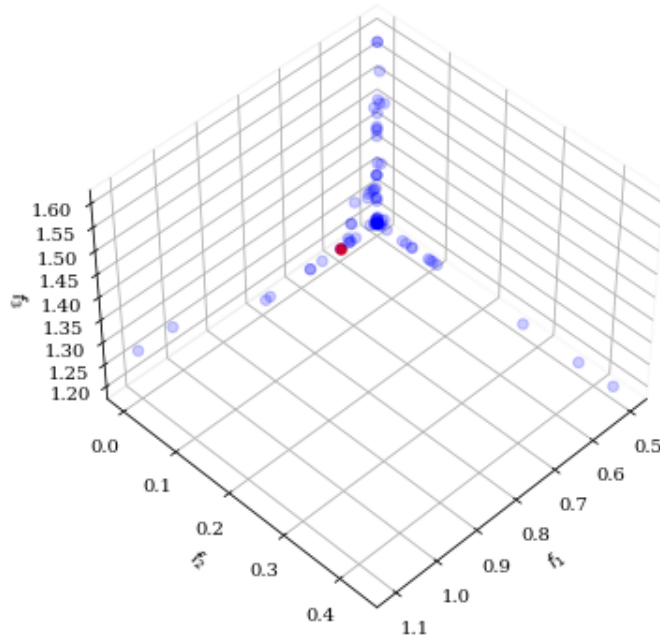


Figure 38: Pareto-front from the script results for the model that collaborates between the RMS system and fixed equipment.

Best point is iteration nr.: Point 22 - [5.97735122e-01 1.72286417e-05 1.20000392e+00]
 Distance from CNC to 3D-printer in x direction is: 0.6919759551880305 meter
 Distance from CNC to 3D-printer in y direction is: 0.8515664765622237 meter
 Distance from CNC to Nachi in x direction is: 0.40000000272779823 meter
 Distance from CNC to Nachi in y direction is: -0.29998277135848367 meter
 Distance from CNC to Conveyor in x direction is: 1.94998116244045 meter
 Distance from CNC to Conveyor in y direction is: 0.26739047891809314 meter

Figure 39: Decomposed the resultant-vectors forming the objectives. Shows the distance from the reference point on the CNC machine to the modules.

The distances presented in figure 39 are visualized both in Python and in Visual components.

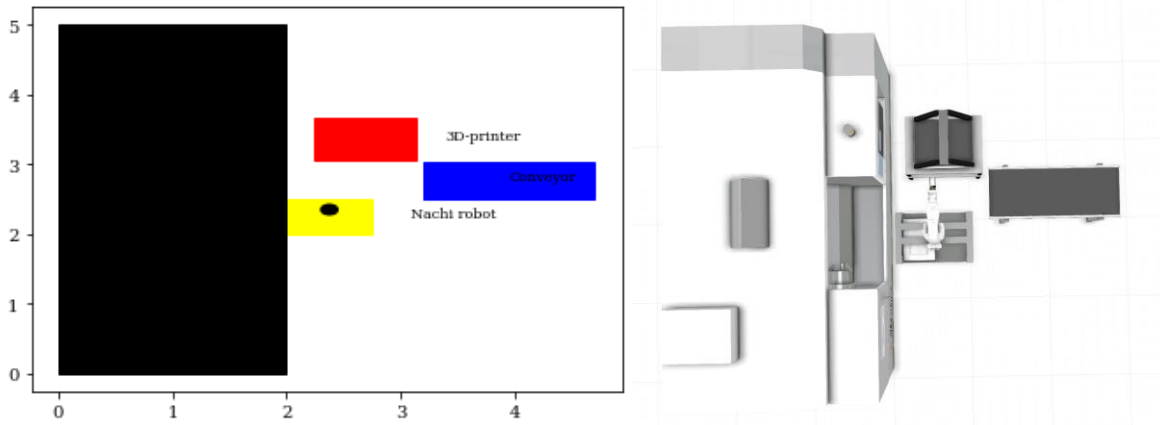


Figure 40: Comparison between the Python plot (Left) and the Visual Components visualization (Right).

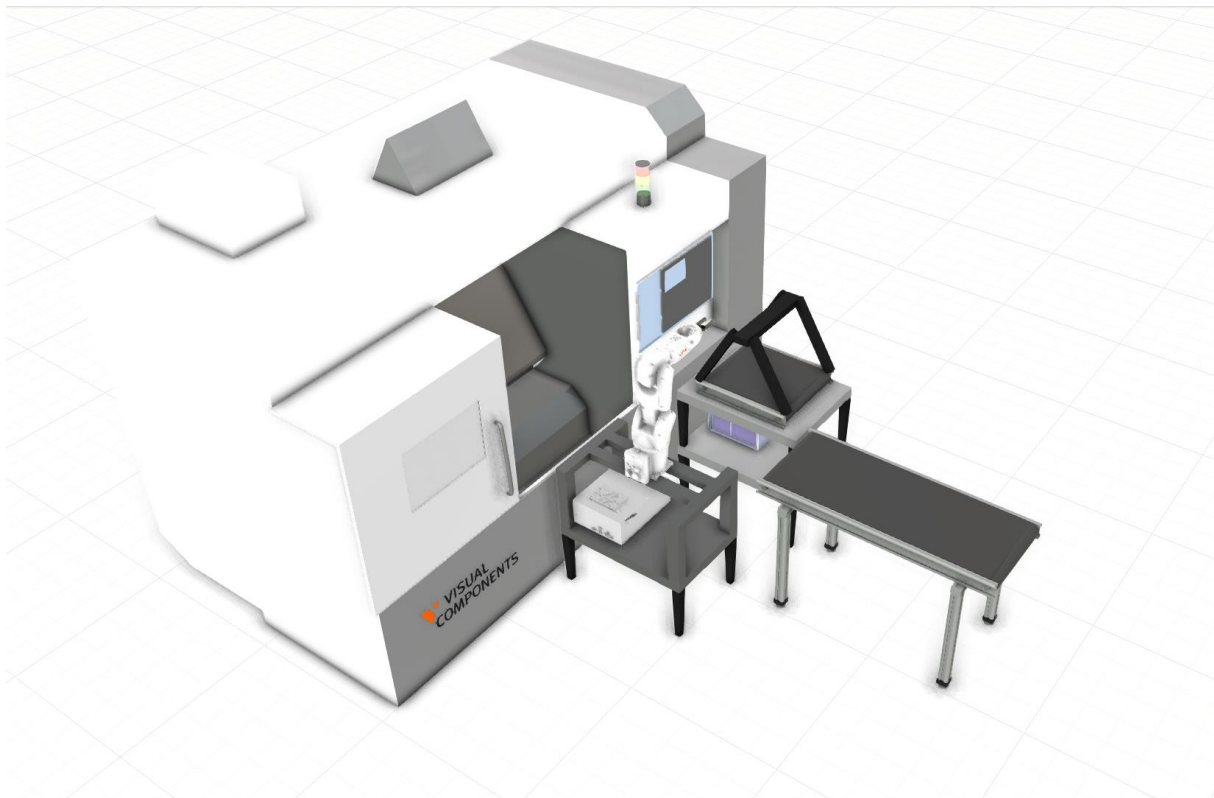


Figure 41: Another angle of the visualization of the RMS system together with the CNC machine.

5.4 Finding placement in the facility

As stated in constraint nr. 5, the facility is 10x5 meter. The physical system is the reference for finding the optimal placement. Since [0,0] is located in the corner of the lab, $x = [20,30]$ and $y = [20,30]$ has been used for feasible placement in the facility. For now, the feasible range is an open space, and the most optimal position when minimizing is therefore [20,20]. The coordinates and center for this surface can be seen in figure 42.

Values for surface containing layout: Rectangle(xy=(20, 20), width=3.4521, height=2.07148, angle=0)
 Center for surface containing layout: (21.726051879251408, 21.03573980568292)

Figure 42: Values for the surface covering the layout that is the basis for finding the placement in the facility.

The center for each module is stored from the layout generation. By adding this value with the center for the surface, the new position is given. The modules are placed similar to each other as previously, but in an optimal position in the facility.

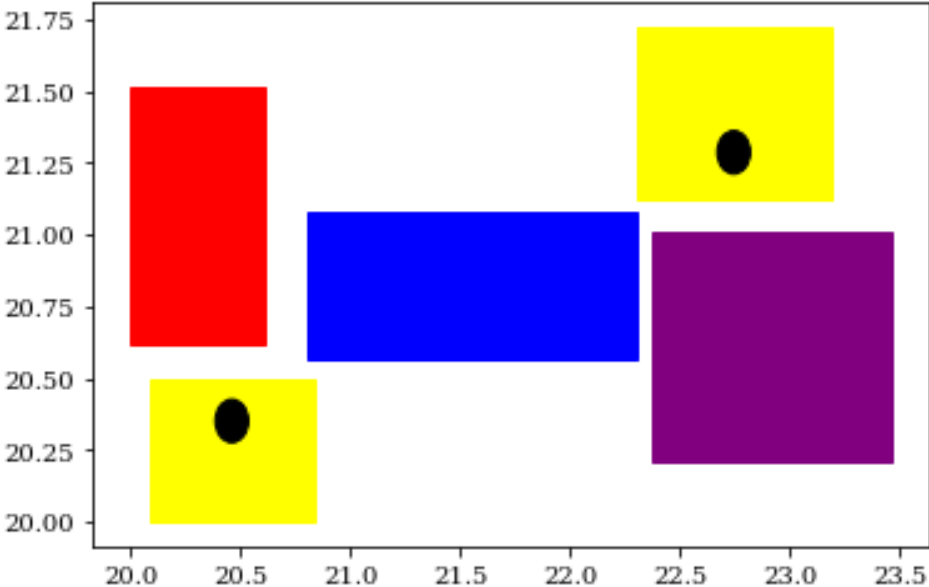


Figure 43: Show the same layout as in the upscaled 5 modules results. But the coordinates on the x-axis and y-axis are showing the feasible range in the laboratory.

The position in the feasible range in the facility it can be seen it is placed as close to the bottom left corner as possible.

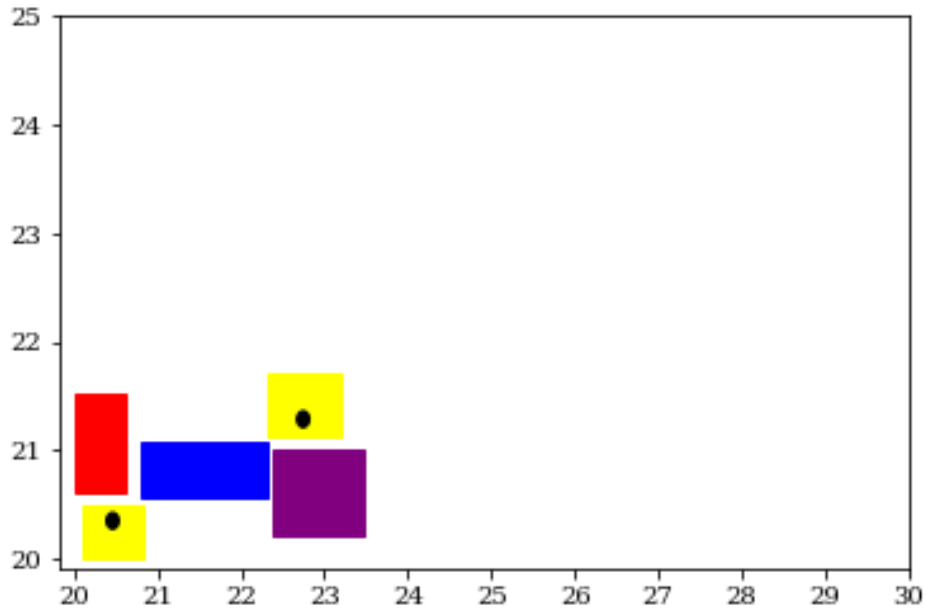


Figure 44: The space in the figure indicates the feasible placement in the laboratory. The figure shows the layout being placed in the bottom-left corner.

This approach can be extended. The same technique was used on the upscaled version including 11 modules. The same range for feasible region have been used.

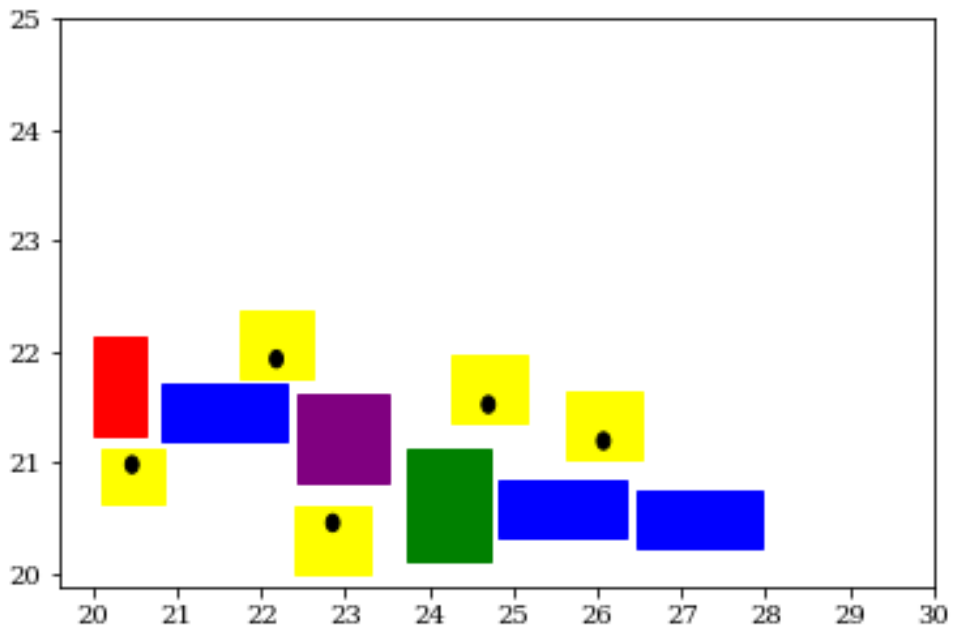


Figure 45: Show the same approach have been successfully implemented on the 11 modules results.

For these results it is assumed that the facility is empty. However, if there are any fixed equipment in conflict with the layout, it must be constrained.

5.5 Physical system

Two layouts will be presented with the physical system. Layout 1 is the same layout which was found in 5.2.1. When performing the physical test, it was discovered that the mobile robot could not place the modules as close as shown in the digital visualization. The sizes of the modules were added 0.3m in x and y to avoid this problem. Layout 2 is generated to record the reconfiguration of the physical system. Layout 1 have been placed in an optimal placement in the laboratory. Because of some other equipment existing in the laboratory when the experiment was conducted, the coordinates alter from the results in 5.4. Figure 46 illustrates the coordinates that was extracted from the script and figure 47 displays the visualization of layout 1.

```
Configuration 1:
[['Nachi x:           ' '22.5']
 ['3D-printer x:      ' '22.248']
 ['Conveyor x:        ' '23.69']
 ['Scara x:           ' '25.093']
 ['Lift conveyor x:   ' '25.265']]
[['Nachi y:           ' '22.25']
 ['3D-printer y:      ' '23.316']
 ['Conveyor y:        ' '23.061']
 ['Scara y:           ' '23.761']
 ['Lift conveyor y:   ' '22.752']]
Configuration 2:
[['Nachi x:           ' '28.0']
 ['3D-printer x:      ' '28.747']
 ['Conveyor x:        ' '29.066']
 ['Scara x:           ' '30.516']
 ['Lift conveyor x:   ' '30.085']]
[['Nachi y:           ' '23.75']
 ['3D-printer y:      ' '23.65']
 ['Conveyor y:        ' '24.606']
 ['Scara y:           ' '25.005']
 ['Lift conveyor y:   ' '23.703']]
```

Figure 46: Present the coordinates that was sent to the execution script for the two layouts.

These values are then converted to a string so they can be executed in the physical system.

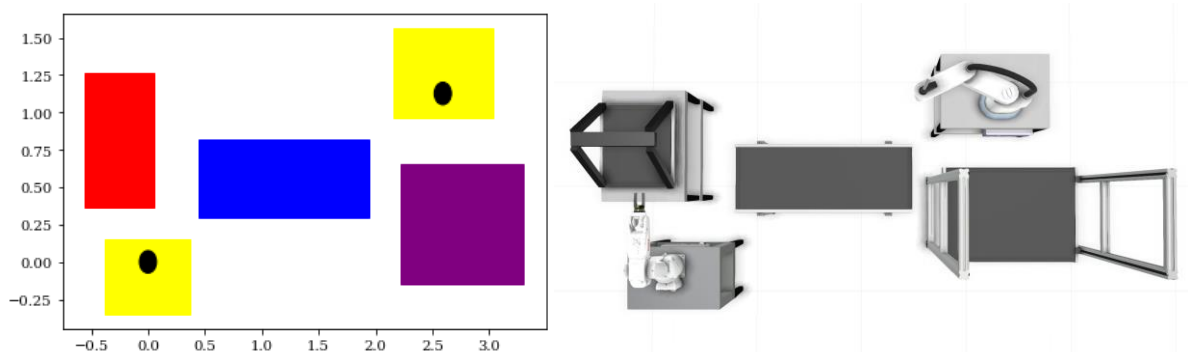


Figure 47: Plot (Left) and visualization (Right) for layout 1 how it should be placed in the physical system.

Layout 1 physical system:



Figure 48: Figure shows layout 1 by the physical system in the laboratory.



Figure 49: A different angle of Layout 1 in the laboratory.

A comparison between the physical system and the visualization in Visual Components can be seen in figure 50.



Figure 50: Comparison between the physical system (Left) and the visualization (Right).

After layout 1 was completed, the system reconfigured into layout 2. However, layout 2 is not placed in the optimal placement in the facility. To demonstrate the reconfiguration, layout 2 was simply positioned at a secure distance from layout 1. Visualization for layout 2 can be seen in figure 51 and the physical system in figure 52 and 53.

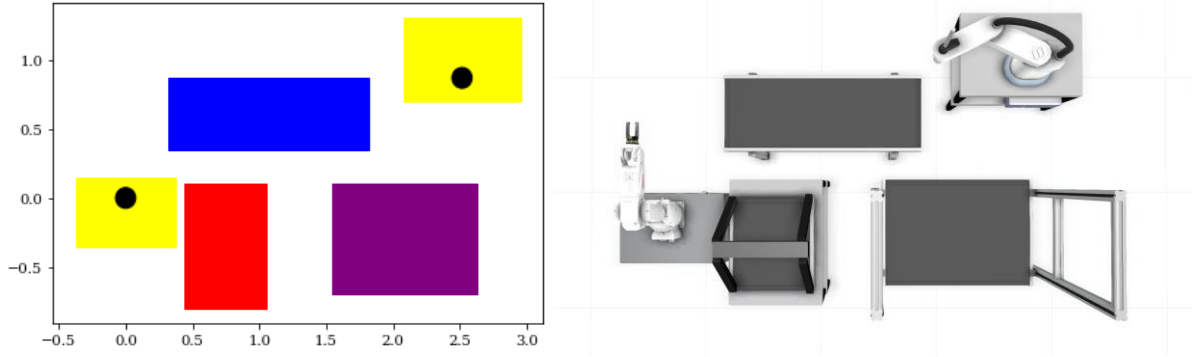


Figure 51: Plot (Left) and visualization (Right) for layout 2 how it should be placed in the physical system.



Figure 52: Layout 2, physical system.



Figure 53: Another angle of layout 2 performed by the physical system.

A comparison between the visualization in Visual components (figure 54), and the physical system (figure 55) is presented for layout 2.



Figure 54: Visualization of layout 2 that is compared with the physical system showed in figure 55.



Figure 55: The physical system from the same angle as the visualization in figure 54.

In table 4, link for the videos showing the physical system creating layout 1 & 2 can be gathered.

Layout 1	https://www.youtube.com/watch?v=voFilQ-QfH8&t=1s&ab_channel=UiTNarvik-IndustrialEngineering
Layout 2	https://www.youtube.com/watch?v=3OfvNkiVtr4&t=31s&ab_channel=UiTNarvik-IndustrialEngineering

Table 4: YouTube links for the physical system placing layout 1 & 2.

6 Discussion and Conclusion

6.1 Discussion

A general model has been proposed to generate layouts for a physical RMS system as a response to the layout problem in RMS. An algorithm generates the layouts, and a method for determining the near-optimal location to place the layout has been introduced. The model proposed requires limited amount of computer-power and generates a new layout within seconds. To validate and verify the methods used and the results acquired, tests on a physical system have been undergone and the results have been compared between the physical system and digital visualizations. To the author's knowledge, no experiment has been done previously with automating layout generation as a response to the layout problem in RMS.

A connection between the program creating layouts and the physical system have been used, emphasizing the automation of the system. The model containing the scripts have been proved to be general. The model can easily be upscaled by adding groups in the script. Even though placing five modules are not a complex problem, the method was proposed as an answer to create a foundation for the automation problem within RMS that clearly lacks in the literature. One of the major reasons for applying the NSGA-II as the problem solver, is that with an upscaled problem that is more complex, the same method can be applied. For showing the method's strength and versatility, different scenarios have been tested and visualized. The RMS system placed in a layout that collaborates with a CNC machine is shown, as well as multiple layouts using the same modules.

Though the results are verified comparing the physical system with the visualization, some parts need more investigation. For maximizing the method proposed, a benchmark solution should be investigated. Then, the effectiveness of the method can be extended further. To further generalize the method, the fixed angle of each module should be included in the mathematical model, and then be found in the search.

The direction of the constraint has been chosen manually for each module. When creating a layout that includes a higher number of equipment and thus more groups, the direction each new group is placed towards have an impact on the total area the system occupy. Also, the equipment order becomes more complex. Therefore, the method should be extended further to include direction selection for new groups to be placed.

6.2 Conclusion

The results concludes that the proposed method can be used for tackling the layout problem in the RMS system. Two different layouts were generated based on manufacturing a drinking shaker, using a developed mathematical model in a Python script. The results were found utilizing the NSGA-II algorithm. The mathematical model was validated, and different tests were completed. The output from the Python script were coordinates for the new layout, placed in a near-optimal feasible region in the facility. In addition to the two layouts for the physical system, an upscaled visualization of 11 modules is shown.

A connection in Python was used to send the coordinates to the physical system. In the physical system, a MIR-100 mobile robot autonomously configured layout 1, and then reconfigured the system into layout 2. The results from the physical test were compared with the visualization to validate and verify. Both tests have been filmed and the YouTube link can be found in the results section. The results enhance both the automation and the effectiveness of the system.

6.2.1 Future works

The method and mathematical model should be strengthened and developed further. The model should be benchmarked to find where the method can be augmented. The angle of placement for each module should also be generated automatically by the script.

In the literature there have been multiple studies on optimal equipment order based on products. To generalize the method further for tackling more complex problems, a connection should be made with such methods and the method presented in this thesis.

References

- [1] J. Daaboul, C. Da Cunha, A. Bernard, and F. Laroche, "Design for mass customization: Product variety vs. process variety," (in English), *CIRP Ann-Manuf. Technol.*, Article vol. 60, no. 1, pp. 169-174, 2011, doi: 10.1016/j.cirp.2011.03.093.
- [2] Y. Koren, X. Gu, and W. Guo, "Reconfigurable manufacturing systems: Principles, design, and future trends," *Frontiers of Mechanical Engineering*, vol. 13, no. 2, pp. 121-136, 2018.
- [3] Y. Koren *et al.*, "Reconfigurable manufacturing systems," *CIRP annals*, vol. 48, no. 2, pp. 527-540, 1999.
- [4] R. C. Sabioni, J. Daaboul, and J. Le Duigou, "Concurrent optimisation of modular product and Reconfigurable Manufacturing System configuration: a customer-oriented offer for mass customisation," (in English), *Int. J. Prod. Res.*, Article; Early Access p. 17, 2020, doi: 10.1080/00207543.2021.1886369.
- [5] R. C. Sabioni, J. Daaboul, and J. Le Duigou, "An integrated approach to optimize the configuration of mass-customized products and reconfigurable manufacturing systems," (in English), *Int. J. Adv. Manuf. Technol.*, Article vol. 115, no. 1-2, pp. 141-163, Jul 2021, doi: 10.1007/s00170-021-06984-w.
- [6] Y. Koren and M. Shpitalni, "Design of reconfigurable manufacturing systems," *Journal of manufacturing systems*, vol. 29, no. 4, pp. 130-141, 2010.
- [7] M. Bortolini, F. G. Galizia, and C. Mora, "Reconfigurable manufacturing systems: Literature review and research trend," *Journal of manufacturing systems*, vol. 49, pp. 93-106, 2018.
- [8] X. P. Guan, X. Z. Dai, B. J. Qiu, and J. Li, "A revised electromagnetism-like mechanism for layout design of reconfigurable manufacturing system," (in English), *Comput. Ind. Eng.*, Article vol. 63, no. 1, pp. 98-108, Aug 2012, doi: 10.1016/j.cie.2012.01.016.
- [9] I. Maganha, C. Silva, and L. Ferreira, "The layout design in reconfigurable manufacturing systems: a literature review," (in English), *Int. J. Adv. Manuf. Technol.*, Review vol. 105, no. 1-4, pp. 683-700, Nov 2019, doi: 10.1007/s00170-019-04190-3.
- [10] A. Oke, K. Abou-El-Hossein, and N. J. Theron, "The design and development of a reconfigurable manufacturing system," *South African Journal of Industrial Engineering*, vol. 22, no. 2, pp. 121-132, 2011.
- [11] G. Meng, S. S. Heragu, and H. Zijm, "Reconfigurable layout problem," (in English), *Int. J. Prod. Res.*, Article vol. 42, no. 22, pp. 4709-4729, Nov 2004, doi: 10.1080/0020754042000264590.
- [12] K. K. Goyal and P. K. Jain, "Design of reconfigurable flow lines using MOPSO and maximum deviation theory," (in English), *Int. J. Adv. Manuf. Technol.*, Article vol. 84, no. 5-8, pp. 1587-1600, May 2016, doi: 10.1007/s00170-015-7760-4.
- [13] R. E. Burkard, S. E. Karisch, and F. Rendl, "QAPLIB - A quadratic assignment problem library," (in English), *J. Glob. Optim.*, Article vol. 10, no. 4, pp. 391-403, Jun 1997, doi: 10.1023/a:1008293323270.
- [14] P. Leitao, J. Barbosa, and D. Trentesaux, "Bio-inspired multi-agent systems for reconfigurable manufacturing systems," (in English), *Eng. Appl. Artif. Intell.*, Article vol. 25, no. 5, pp. 934-944, Aug 2012, doi: 10.1016/j.engappai.2011.09.025.

- [15] Y. Yamada, K. Ookoudo, Y. Komura, Ieee, and Ieee, "Layout optimization of manufacturing cells and allocation optimization of transport robots in reconfigurable manufacturing systems using particle swarm optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nv, Oct 27-31 2003, NEW YORK: Ieee, 2003, pp. 2049-2054. [Online]. Available: <Go to ISI>://WOS:000187883300331. [Online]. Available: <Go to ISI>://WOS:000187883300331
- [16] Y. Yamada and Ieee, "Dynamic reconfiguration of reconfigurable manufacturing systems using Particle Swarm Optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, Orlando, FL, May 15-19 2006, NEW YORK: Ieee, in IEEE International Conference on Robotics and Automation ICRA, 2006, pp. 1444-1449, doi: 10.1109/robot.2006.1641912. [Online]. Available: <Go to ISI>://WOS:000240886903017
- [17] H. H. Benderbal and L. Benyoucef, "A New Hybrid Approach for Machine Layout Design Under Family Product Evolution for Reconfigurable Manufacturing Systems," in *9th IFAC/IFIP/IFORS/IIE/INFORMS Conference on Manufacturing Modelling, Management and Control (IFAC MIM)*, Berlin, GERMANY, Aug 28-30 2019, vol. 52, AMSTERDAM: Elsevier, 2019, pp. 1379-1384, doi: 10.1016/j.ifacol.2019.11.391. [Online]. Available: <Go to ISI>://WOS:000504282400234
- [18] H. H. Benderbal, M. Dahane, and L. Benyoucef, "Exhaustive Search Based Heuristic for Solving Machine Layout Problem in Reconfigurable Manufacturing System Design," in *16th IFAC Symposium on Information Control Problems in Manufacturing (INCOM)*, Bergamo, ITALY, Jun 11-13 2018, vol. 51, AMSTERDAM: Elsevier Science Bv, 2018, pp. 78-83, doi: 10.1016/j.ifacol.2018.08.238. [Online]. Available: <Go to ISI>://WOS:000445651000015
- [19] K. K. Goyal, P. K. Jain, and M. Jain, "Optimal configuration selection for reconfigurable manufacturing system using NSGA II and TOPSIS," (in English), *Int. J. Prod. Res.*, Article vol. 50, no. 15, pp. 4175-4191, 2012, doi: 10.1080/00207543.2011.599345.
- [20] H. H. Benderbal, M. Dahane, and L. Benyoucef, "Flexibility-based multi-objective approach for machines selection in reconfigurable manufacturing system (RMS) design under unavailability constraints," (in English), *Int. J. Prod. Res.*, Article vol. 55, no. 20, pp. 6033-6051, 2017, doi: 10.1080/00207543.2017.1321802.
- [21] A. Bensmaine, M. Dahane, and L. Benyoucef, "A non-dominated sorting genetic algorithm based approach for optimal machines selection in reconfigurable manufacturing environment," (in English), *Comput. Ind. Eng.*, Article vol. 66, no. 3, pp. 519-524, Nov 2013, doi: 10.1016/j.cie.2012.09.008.
- [22] K. K. Goyal, P. K. Jain, and M. Jain, "A novel methodology to measure the responsiveness of RMTs in reconfigurable manufacturing system," *Journal of Manufacturing Systems*, vol. 32, no. 4, pp. 724-730, 2013.
- [23] H. H. Benderbal, M. Dahane, and L. Benyoucef, "Modularity assessment in reconfigurable manufacturing system (RMS) design: an Archived Multi-Objective Simulated Annealing-based approach," (in English), *Int. J. Adv. Manuf. Technol.*, Article vol. 94, no. 1-4, pp. 729-749, Jan 2018, doi: 10.1007/s00170-017-0803-2.
- [24] N. Xie, A. P. Li, and W. Xue, "Cooperative Optimization of Reconfigurable Machine Tool Configurations and Production Process Plan," (in English), *Chin. J. Mech. Eng.*, Article vol. 25, no. 5, pp. 982-989, Sep 2012, doi: 10.3901/cjme.2012.05.982.

- [25] R. C. Sabioni, J. Daaboul, and J. L. Duigou, "Optimization of Reconfigurable Manufacturing Systems Configuration: A Literature Review," in *International Joint Conference on Mechanics, Design Engineering & Advanced Manufacturing*, 2020: Springer, pp. 426-435.
- [26] A. Bensmaine, M. Dahane, and L. Benyoucef, "Simulation-Based NSGA-II Approach for Multi-Unit Process Plans Generation in Reconfigurable Manufacturing System," in *16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Toulouse, FRANCE, Sep 05-09 2011, NEW YORK: Ieee, in IEEE International Conference on Emerging Technologies and Factory Automation-ETFA, 2011. [Online]. Available: <Go to ISI>://WOS:000297542900071. [Online]. Available: <Go to ISI>://WOS:000297542900071
- [27] M. Dahane and L. Benyoucef, "An Adapted NSGA-II Algorithm for a Reconfigurable Manufacturing System (RMS) Design Under Machines Reliability Constraints," in *Metaheuristics for Production Systems*, E.-G. Talbi, F. Yalaoui, and L. Amodeo Eds. Cham: Springer International Publishing, 2016, pp. 109-130.
- [28] A. Chaube, L. Benyoucef, and M. K. Tiwari, "An adapted NSGA-2 algorithm based dynamic process plan generation for a reconfigurable manufacturing system," (in English), *J. Intell. Manuf.*, Article vol. 23, no. 4, pp. 1141-1155, Aug 2012, doi: 10.1007/s10845-010-0453-9.
- [29] F. A. Touzout and L. Benyoucef, "Multi-objective sustainable process plan generation in a reconfigurable manufacturing environment: exact and adapted evolutionary approaches," (in English), *Int. J. Prod. Res.*, Article vol. 57, no. 8, pp. 2531-2547, Apr 2019, doi: 10.1080/00207543.2018.1522006.
- [30] A. Bensmaine, M. Dahane, and L. Benyoucef, "A new heuristic for integrated process planning and scheduling in reconfigurable manufacturing systems," (in English), *Int. J. Prod. Res.*, Article vol. 52, no. 12, pp. 3583-3594, Jun 2014, doi: 10.1080/00207543.2013.878056.
- [31] H. H. Benderbal, M. Dahane, and L. Benyoucef, "A new robustness index formachines selection in reconfigurable manufacturing system," in *2015 International Conference on Industrial Engineering and Systems Management (IESM)*, 2015: IEEE, pp. 1019-1026.
- [32] H. Haddou-Benderbal, M. Dahane, and L. Benyoucef, "Hybrid heuristic to minimize machine's unavailability impact on reconfigurable manufacturing system using reconfigurable process plan," *IFAC-PapersOnLine*, vol. 49, no. 12, pp. 1626-1631, 2016.
- [33] Y. C. Choi and P. Xirouchakis, "A holistic production planning approach in a reconfigurable manufacturing system with energy consumption and environmental effects," (in English), *Int. J. Comput. Integr. Manuf.*, Article vol. 28, no. 4, pp. 379-394, Apr 2015, doi: 10.1080/0951192x.2014.902106.
- [34] S. H. Huang *et al.*, "Multi-scale Configuration Design Method of Reconfigurable Manufacturing System Based on Living System Theory," in *IEEE International Conference on Industrial Engineering and Engineering Management (IEEE IEEM)*, Bangkok, THAILAND, Dec 16-19 2018, NEW YORK: Ieee, in International Conference on Industrial Engineering and Engineering Management IEEM, 2018, pp. 1719-1724. [Online]. Available: <Go to ISI>://WOS:000458674600343. [Online]. Available: <Go to ISI>://WOS:000458674600343
- [35] C. Lv, A. P. Li, and L. Y. Xu, "Research and optimization of reconfigurable manufacturing system configuration based on system reliability," (in English),

- Kybernetes*, Article; Proceedings Paper vol. 39, no. 6, pp. 1058-1065, 2010, doi: 10.1108/03684921011046834.
- [36] M. Maniraj, V. Pakkirisamy, and R. Jeyapaul, "An ant colony optimization-based approach for a single-product flow-line reconfigurable manufacturing systems," (in English), *Proc. Inst. Mech. Eng. Part B-J. Eng. Manuf.*, Article vol. 231, no. 7, pp. 1229-1236, May 2017, doi: 10.1177/0954405415585260.
- [37] J. P. Dou, X. Z. Dai, X. D. Ma, Z. D. Meng, and Ieee, "A GA-based Approach to Optimize Single-Product Flow-Line Configurations of RMS," in *IEEE International Conference on Mechatronics and Automation*, Takamatsu, JAPAN, Aug 05-08 2008, NEW YORK: Ieee, 2008, pp. 653-658. [Online]. Available: <Go to ISI>://WOS:000267671500119. [Online]. Available: <Go to ISI>://WOS:000267671500119
- [38] J. P. Dou, X. Z. Dai, and Z. D. Meng, "Graph theory-based approach to optimize single-product flow-line configurations of RMS," (in English), *Int. J. Adv. Manuf. Technol.*, Article vol. 41, no. 9-10, pp. 916-931, Apr 2009, doi: 10.1007/s00170-008-1541-2.
- [39] K. K. Goyal, P. K. Jain, and M. Jain, "APPLYING SWARM INTELLIGENCE TO DESIGN THE RECONFIGURABLE FLOW LINES," (in English), *Int. J. Simul. Model.*, Article vol. 12, no. 1, pp. 17-26, Mar 2013, doi: 10.2507/ijstimm12(1)2.220.
- [40] M. Ashraf and F. Hasan, "Configuration selection for a reconfigurable manufacturing flow line involving part production with operation constraints," (in English), *Int. J. Adv. Manuf. Technol.*, Article vol. 98, no. 5-8, pp. 2137-2156, Sep 2018, doi: 10.1007/s00170-018-2361-7.
- [41] K. K. Mittal, D. Kumar, and P. K. Jain, "A systematic approach for optimum configuration selection in reconfigurable manufacturing system," *Journal of The Institution of Engineers (India): Series C*, vol. 99, no. 6, pp. 629-635, 2018.
- [42] X. Xiaowen, Z. Beirong, and X. Wei, "Configuration Optimization Method of Reconfigurable Manufacturing Systems," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 6, no. 8, pp. 1389-1393, 2013.
- [43] L. K. Saxena and P. K. Jain, "A model and optimisation approach for reconfigurable manufacturing system configuration design," (in English), *Int. J. Prod. Res.*, Article vol. 50, no. 12, pp. 3359-3381, 2012, doi: 10.1080/00207543.2011.578161.
- [44] A. M. A. Youssef and H. A. ElMaraghy, "Optimal configuration selection for reconfigurable manufacturing systems," (in English), *Int. J. Flexible Manuf. Syst.*, Article vol. 19, no. 2, pp. 67-106, Jun 2007, doi: 10.1007/s10696-007-9020-x.
- [45] A. M. A. Youssef and H. A. Elmaraghy, "Availability consideration in the optimal selection of multiple-aspect RMS configurations," (in English), *Int. J. Prod. Res.*, Article vol. 46, no. 21, pp. 5849-5882, 2008, doi: 10.1080/00207540701261626.
- [46] X. B. Zhao, J. C. Wang, and Z. B. Luo, "A stochastic model of a reconfigurable manufacturing system Part 1: A framework," (in English), *Int. J. Prod. Res.*, Article vol. 38, no. 10, pp. 2273-2285, Jul 2000, doi: 10.1080/00207540050028098.
- [47] X. B. Zhao, J. C. Wang, and Z. B. Luo, "A stochastic model of a reconfigurable manufacturing system - Part 2: Optimal configurations," (in English), *Int. J. Prod. Res.*, Article vol. 38, no. 12, pp. 2829-2842, Aug 2000. [Online]. Available: <Go to ISI>://WOS:000088569200015.
- [48] S. K. Moghaddam, M. Houshmand, and O. F. Valilai, "Configuration design in scalable reconfigurable manufacturing systems (RMS); a case of single-product flow

- line (SPFL)," (in English), *Int. J. Prod. Res.*, Article vol. 56, no. 11, pp. 3932-3954, 2018, doi: 10.1080/00207543.2017.1412531.
- [49] S. K. Moghaddam, M. Houshmand, K. Saitou, and O. F. Valilai, "Configuration design of scalable reconfigurable manufacturing systems for part family," (in English), *Int. J. Prod. Res.*, Article vol. 58, no. 10, pp. 2974-2996, May 2020, doi: 10.1080/00207543.2019.1620365.
- [50] J. P. Dou, J. Li, and C. Su, "Bi-objective optimization of integrating configuration generation and scheduling for reconfigurable flow lines using NSGA-II," (in English), *Int. J. Adv. Manuf. Technol.*, Article vol. 86, no. 5-8, pp. 1945-1962, Sep 2016, doi: 10.1007/s00170-015-8291-8.
- [51] F. Hasan, P. Jain, and D. Kumar, "Optimum configuration selection in Reconfigurable Manufacturing System involving multiple part families," *Opsearch*, vol. 51, no. 2, pp. 297-311, 2014.
- [52] E. Asghar, U. K. U. Zaman, A. A. Baqai, and L. Homri, "Optimum machine capabilities for reconfigurable manufacturing systems," (in English), *Int. J. Adv. Manuf. Technol.*, Article vol. 95, no. 9-12, pp. 4397-4417, Apr 2018, doi: 10.1007/s00170-017-1560-y.
- [53] M. Abbasi and M. Houshmand, "Production planning and performance optimization of reconfigurable manufacturing systems using genetic algorithm," (in English), *Int. J. Adv. Manuf. Technol.*, Article vol. 54, no. 1-4, pp. 373-392, Apr 2011, doi: 10.1007/s00170-010-2914-x.
- [54] A. Bryan, S. J. Hu, and Y. Koren, "Assembly System Reconfiguration Planning," (in English), *J. Manuf. Sci. Eng.-Trans. ASME*, Article vol. 135, no. 4, p. 13, Aug 2013, Art no. 041005, doi: 10.1115/1.4024288.
- [55] K. W. Schmidt and Ieee, "Optimal Configuration Changes for Reconfigurable Manufacturing Systems," in *52nd IEEE Annual Conference on Decision and Control (CDC)*, Florence, ITALY, Dec 10-13 2013, NEW YORK: Ieee, in IEEE Conference on Decision and Control, 2013, pp. 7621-7626. [Online]. Available: <Go to ISI>://WOS:000352223508091. [Online]. Available: <Go to ISI>://WOS:000352223508091
- [56] S. H. Huang, G. X. Wang, X. W. Shang, and Y. Yan, "Reconfiguration point decision method based on dynamic complexity for reconfigurable manufacturing system (RMS)," (in English), *J. Intell. Manuf.*, Article vol. 29, no. 5, pp. 1031-1043, Jun 2018, doi: 10.1007/s10845-017-1318-2.
- [57] D. Sanderson, J. C. Chaplin, L. de Silva, P. Holmes, and S. Ratchev, "Smart Manufacturing And Reconfigurable Technologies: Towards an Integrated Environment for Evolvable Assembly Systems," in *1st IEEE International Workshop on Foundations and Applications of Self-* Systems (FAS-W)*, Augsburg, GERMANY, Sep 12-16 2016, NEW YORK: Ieee, 2016, pp. 263-264, doi: 10.1109/fas-w.2016.61. [Online]. Available: <Go to ISI>://WOS:000391523100048
- [58] O. Madsen and C. Moller, "The AAU Smart Production Laboratory for teaching and research in emerging digital manufacturing technologies," in *7th Conference on Learning Factories (CLF)*, Darmstadt, GERMANY, Apr 04-05 2017, vol. 9, AMSTERDAM: Elsevier Science Bv, in Procedia Manufacturing, 2017, pp. 106-112, doi: 10.1016/j.promfg.2017.04.036. [Online]. Available: <Go to ISI>://WOS:000416991700014
- [59] C. da Cunha, O. Cardin, G. Gallot, and J. Viaud, "Designing the Digital Twins of Reconfigurable Manufacturing Systems: application on a smart factory," in *17th IFAC*

- Symposium on Information Control Problems in Manufacturing (INCOM)*, Budapest, HUNGARY, Jun 07-09 2021, vol. 54, AMSTERDAM: Elsevier, 2021, pp. 874-879, doi: 10.1016/j.ifacol.2021.08.103. [Online]. Available: <Go to ISI>://WOS:000716937600144
- [60] R. Adamietz, T. Giesen, P. Mayer, A. Johnson, R. Bibb, and C. Seifarth, "Reconfigurable and transportable container-integrated production system," (in English), *Robot. Comput.-Integr. Manuf.*, Article vol. 53, pp. 1-20, Oct 2018, doi: 10.1016/j.rcim.2018.02.008.
- [61] D. Y. Kim *et al.*, "A modular factory testbed for the rapid reconfiguration of manufacturing systems," (in English), *J. Intell. Manuf.*, Article vol. 31, no. 3, pp. 661-680, Mar 2020, doi: 10.1007/s10845-019-01471-2.
- [62] R. Rosen, G. von Wichert, G. Lo, and K. D. Bettenhausen, "About The Importance of Autonomy and Digital Twins for the Future of Manufacturing," in *15th IFAC Symposium on Information Control Problems in Manufacturing*, Ottawa, CANADA, May 11-13 2015, vol. 48, AMSTERDAM: Elsevier Science Bv, 2015, pp. 567-572, doi: 10.1016/j.ifacol.2015.06.141. [Online]. Available: <Go to ISI>://WOS:000375804100094
- [63] H. Engemann, S. Z. Du, S. Kallweit, P. Conen, and H. Dawar, "OMNIVIL-An Autonomous Mobile Manipulator for Flexible Production," (in English), *Sensors*, Article vol. 20, no. 24, p. 30, Dec 2020, Art no. 7249, doi: 10.3390/s20247249.
- [64] M. Hvilshoj, S. Bogh, O. S. Nielsen, and O. Madsen, "Autonomous industrial mobile manipulation (AIMM): past, present and future," (in English), *Ind. Robot.*, Article vol. 39, no. 2, pp. 120-135, 2012, doi: 10.1108/01439911211201582.
- [65] R. E. Andersen *et al.*, "Integration of a skill-based collaborative mobile robot in a smart cyber-physical environment," in *27th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM)*, Modena, ITALY, Jun 27-30 2017, vol. 11, AMSTERDAM: Elsevier Science Bv, in *Procedia Manufacturing*, 2017, pp. 114-123, doi: 10.1016/j.promfg.2017.07.209. [Online]. Available: <Go to ISI>://WOS:000419072100014
- [66] H. Arnarson and B. Solvang, "Reconfigurable autonomous industrial mobile manipulator system," in *2022 IEEE/SICE International Symposium on System Integration (SII)*, 2022: IEEE, pp. 772-777.
- [67] Y. Chen, X. Dai, and Z. Meng, "Modeling of reconfigurable material handling system consisting of multiple mobile robots," in *IEEE International Conference Mechatronics and Automation, 2005*, 2005, vol. 4: IEEE, pp. 1731-1735.
- [68] S. Inoue *et al.*, "High-Precision Mobile Robotic Manipulator for Reconfigurable Manufacturing Systems," *International Journal of Automation Technology*, vol. 15, no. 5, pp. 651-660, 2021.
- [69] H. H. Benderbal, A. R. Yelles-Chaouche, and A. Dolgui, "A Digital Twin Modular Framework for Reconfigurable Manufacturing Systems," in *IFIP WG 5.7 International Conference on Advances in Production Management Systems (APMS)*, Novi Sad, SERBIA, Aug 30-Sep 03 2020, vol. 592, CHAM: Springer International Publishing Ag, in *IFIP Advances in Information and Communication Technology*, 2020, pp. 493-500, doi: 10.1007/978-3-030-57997-5_57. [Online]. Available: <Go to ISI>://WOS:000681218300057
- [70] C. Y. Zhang, W. J. Xu, J. Y. Liu, Z. H. Liu, Z. D. Zhou, and D. T. Pham, "A Reconfigurable Modeling Approach for Digital Twin-based Manufacturing System," in *11th CIRP Conference on Industrial Product-Service Systems*, Peoples R China,

- May 29-31 2019, vol. 83, AMSTERDAM: Elsevier, in *Procedia CIRP*, 2019, pp. 118-125, doi: 10.1016/j.procir.2019.03.141. [Online]. Available: <Go to ISI>://WOS:000568146700019
- [71] J. W. Leng *et al.*, "Digital twin-driven rapid reconfiguration of the automated manufacturing system via an open architecture model," (in English), *Robot. Comput.-Integr. Manuf.*, Article vol. 63, p. 12, Jun 2020, Art no. 101895, doi: 10.1016/j.rcim.2019.101895.
- [72] E. Hajjem, H. H. Benderbal, N. Hamani, and A. Dolgui, "Digital Twin Framework for Reconfigurable Manufacturing Systems: Challenges and Requirements," in *IFIP WG 5.7 International Conference on Advances in Production Management Systems (APMS)*, Nantes, FRANCE, Sep 05-09 2021, vol. 631, CHAM: Springer International Publishing Ag, in *IFIP Advances in Information and Communication Technology*, 2021, pp. 553-562, doi: 10.1007/978-3-030-85902-2_59. [Online]. Available: <Go to ISI>://WOS:000719354900059
- [73] L. Zheng, L. Y. Zhu, B. Wang, L. H. Bai, and Ieee, "A Simulation Analysis of Facility Layout Problems in Reconfigurable Manufacturing Systems," in *International Conference on Computer Sciences and Applications (CSA)*, Hubei Univ Technol, Wuhan, PEOPLES R CHINA, Dec 14-15 2013, NEW YORK: Ieee, 2013, pp. 423-427, doi: 10.1109/csa.2013.106. [Online]. Available: <Go to ISI>://WOS:000357013800100
- [74] S. E. H. Petroodi, A. B. D. Eynaud, N. Klement, and R. Tavakkoli-Moghaddam, "Simulation-based optimization approach with scenario-based product sequence in a reconfigurable manufacturing system (RMS): A case study," in *9th IFAC/IFIP/IFORS/IISE/INFORMS Conference on Manufacturing Modelling, Management and Control (IFAC MIM)*, Berlin, GERMANY, Aug 28-30 2019, vol. 52, AMSTERDAM: Elsevier, 2019, pp. 2638-2643, doi: 10.1016/j.ifacol.2019.11.605. [Online]. Available: <Go to ISI>://WOS:000504282400446
- [75] D. A. Towers, D. Edwards, and M. Hamson, *Guide to mathematical modelling*. Bloomsbury Publishing, 2020.
- [76] A. Benitez-Hidalgo, A. J. Nebro, J. Garcia-Nieto, I. Oregi, and J. Del Ser, "jMetalPy: A Python framework for multi-objective optimization with metaheuristics," (in English), *Swarm Evol. Comput.*, Article vol. 51, p. 12, Dec 2019, Art no. 100598, doi: 10.1016/j.swevo.2019.100598.
- [77] R. E. Perez, P. W. Jansen, and J. Martins, "pyOpt: a Python-based object-oriented framework for nonlinear constrained optimization," (in English), *Struct. Multidiscip. Optim.*, Article vol. 45, no. 1, pp. 101-118, Jan 2012, doi: 10.1007/s00158-011-0666-3.
- [78] J. Blank and K. Deb, "Pymoo: Multi-Objective Optimization in Python," (in English), *IEEE Access*, Article vol. 8, pp. 89497-89509, 2020, doi: 10.1109/access.2020.2990567.
- [79] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182-197, 2002.
- [80] A. Hojjati, M. Monadi, A. Faridhosseini, and M. Mohammadi, "Application and comparison of NSGA-II and MOPSO in multi-objective optimization of water resources systems," *Journal of Hydrology and Hydromechanics*, vol. 66, no. 3, pp. 323-329, 2018.

- [81] A. Nourbakhsh, H. Safikhani, and S. Derakhshan, "The comparison of multi-objective particle swarm optimization and NSGA II algorithm: applications in centrifugal pumps," *Engineering Optimization*, vol. 43, no. 10, pp. 1095-1113, 2011.
- [82] G. C. Ciro, F. Dugardin, F. Yalaoui, and R. Kelly, "A NSGA-II and NSGA-III comparison for solving an open shop scheduling problem with resource constraints," *IFAC-PapersOnLine*, vol. 49, no. 12, pp. 1272-1277, 2016.
- [83] A. Štefek, V. Křivánek, and K. L. Pham, "A Selection and Comparison of Several Algorithms Implemented in the Pymoo Library," in *2021 International Conference on Military Technologies (ICMT)*, 2021: IEEE, pp. 1-7.
- [84] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima, "Performance comparison of NSGA-II and NSGA-III on various many-objective test problems," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016: IEEE, pp. 3045-3052.
- [85] G. Debreu, "Valuation equilibrium and Pareto optimum," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 40, no. 7, p. 588, 1954.
- [86] E. S. Lee and R. J. Li, "Fuzzy multiple objective programming and compromise programming with Pareto optimum," *Fuzzy sets and systems*, vol. 53, no. 3, pp. 275-288, 1993.
- [87] H. Arnarson, B. Solvang, and B. Shu, "The application of open access middleware for cooperation among heterogeneous manufacturing systems," in *2020 3rd International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS)*, 2020: IEEE, pp. 1-6.
- [88] K. Deb, "Multi-objective optimisation using evolutionary algorithms: an introduction," in *Multi-objective evolutionary optimisation for product design and manufacturing*: Springer, 2011, pp. 3-34.
- [89] K. Deb, S. Karthik, T. Okabe, and Acm, "Self-Adaptive Simulated Binary Crossover for Real-Parameter Optimization," in *Annual Conference of Genetic and Evolutionary Computation Conference*, London, ENGLAND, Jul 07-11 2007, NEW YORK: Assoc Computing Machinery, 2007, pp. 1187-+. [Online]. Available: <Go to ISI>://WOS:000268226900211. [Online]. Available: <Go to ISI>://WOS:000268226900211

