Otto-von-Guericke-University Magdeburg

Faculty of Computer Science

Department of Computational Intelligence

Master's Thesis

Cost-Effective Re-Layouting of a Dynamic Manufacturing Facility with Collision-Free Material Handling

Author:

Sai Lokesh Kancharla

Advisor:

Prof. Dr.-Ing. habil. Sanaz Mostaghim

Supervisor:

M.Sc. Thomas Seidelmann

Faculty of Computer Science
Chair of Computational Intelligence

Magdeburg, 26.02.2024

Abstract

In response to the evolving landscape of manufacturing, where mass customization takes precedence over mass production, the necessity for efficient layout designs becomes prominent. This thesis aspires to build upon existing methodologies for solving the Facility Layout Problem (FLP) aiming to find better cost-effective solutions for an existing facility in a dynamic manufacturing setup, while also addressing collisions among transporters in addition. Orders are generated randomly for a variety of product variants to mimic the dynamic nature of manufacturing. The FLP is formulated as a pure integer problem, with a focus not only on optimizing workstation placements but also on workstation selection within the facility. To remove collisions among Automated Guided Vehicles (AGVs) in material handling systems, the thesis adopts unidirectional guidepaths. This guidepath layout problem is formulated as a zero-one problem. Conventionally, the facility and guidepath layout encodings are concatenated and the problems are integrated into one and solved using NSGA-II. Additionally, the thesis explores a coevolutionary approach, using two separate populations, each dedicated to solving one problem. This approach initially assumed to be a superior alternative, surprisingly yielded only marginal improvements over the conventional approach.

Acknowledgements

I extend my sincere thanks to Prof. Sanaz Mostaghim for graciously agreeing to supervise my thesis. Additionally, my deepest gratitude goes to my advisor, Thomas Seidelmann, for the invaluable suggestions, feedback, and guidance that have helped shape this thesis.

I am thankful to the "Bastelstude" and all who attended it for their advice, counselling, and active involvement, which have greatly contributed to the development of this work.

I am profoundly grateful to my parents, whose unwavering support has made all of this possible. My gratitude to them is beyond expression.

A heartfelt thank you goes out to my friends for infusing joy into my journey. Their humour, delightful meals, and endless entertainment have been a source of inspiration and respite.

This thesis is the collective effort of numerous individuals, many of whom I may never know personally. To those whose tools, books, software, papers, and resources have been invaluable during this endeavour, I express my sincere appreciation. Whether directly involved or influencing from afar, I am thankful to each person who has played a role in shaping this work.

Contents

List of Figures

1 Introduction

Over time, there has been a shift in the requirements of manufacturing systems from mass production to mass customization. The survival and growth of manufacturers depend on their ability to offer a great variety of high-quality products at an acceptable price in a minimum time. This leads to increased product diversity and greater uncertainty in product flow management within the manufacturing system [25]. Optimization will always be necessary for increased productivity and profits.

Thesis Outline

To familiarize ourselves with the work of this thesis, in this chapter, we briefly introduce some concepts like the Facility Layout Problem, Material Handling Systems, and why the integration of these two is necessary. The outline of the latter part of the thesis is as follows. In Chapter 2 we begin with an overview of the topics necessary to understand this work better, extending this chapter. Chapter 3 discusses the relevant literature. A comprehensive description of the methodology is proposed in Chapter 4. The experimental setup used in evaluating the proposed methodology is described in Chapter 5. Results are evaluated and discussed in Chapter 6. Finally, Chapter 7 will summarize the thesis and give an outlook on possible future work.

Facility Layout Problem (FLP)

In general, the Facility Layout Problem refers to the proposition of finding an optimal arrangement of machines within a manufacturing facility to yield higher profits while designing the facility or optimizing an existing one. Profits could be increased by having more sales or by reducing production costs. The focus of this research is not on sales but on reducing production costs by solving the facility layout problem with appropriate objectives to lower the transportation costs of the materials within the facility and increase overall productivity. It is also important to know if the reduced costs are worth the effort put into making the necessary changes to the layout. This could be achieved by formulating and solving the facility layout problem manufacturing system with these objectives:

- Minimize the cost of transporting raw material, parts, tools, work-in-process, and finished goods between the departments
- Maximize productivity by facilitating the flow of traffic

The rationale behind the necessity of the Dynamic Facility Layout Problem is that one layout might not be optimal at all times with the evolving diversity of products and it needs to be constantly reoptimized. This necessity stems from factors such as changing product mix, fluctuating demand, and continuous technological advancements. A dynamic approach allows seamless integration of new technologies and ensures optimal spatial utilization. A dynamic facility layout is essential in the context of modern production, where personalization predominates. The adaptability it offers not only facilitates the efficient handling of diverse product configurations but also allows facilities to respond adeptly to changes in market trends and environmental considerations, supporting sustainability measures.

It is essential to note that while our problem falls under the umbrella of a dynamic Facility Layout Problem (FLP), in the context of this work, the term dynamic signifies the ability of the layout to seamlessly handle varying orders arriving over time, scheduling adjustments, and fluctuations in material flow between machines. Instead of continuously modifying the facility layout, the focus is on developing a robust facility layout. A robust layout remains static throughout but is resilient and flexible to effectively manage unforeseen changes in operational dynamics, such as fluctuating order volumes and shifting material flows, without requiring constant adjustments.

Material Handling Systems (MHS)

Material Handling Systems encompass the infrastructure designed to handle materials within the facility. Handling could involve tasks ranging from storage to transportation of products or materials required to produce the products. The earliest form of MHS used for the movement of materials in manufacturing, prior to the Industrial Revolution, primarily was manual transportation which still exists today. Post-industrial revolution, basic equipment like forklift transportation, which involves the use of powered industrial machines equipped with forks to lift, carry, stack, and move materials, emerged to facilitate the movement of materials within larger industrial spaces. Some issues regarding these traditional MHSs include the safety of manual labour, operation costs, inconsistent performance, and information sharing [22]. Conveyor belts, with types like belts, roller, chain, and screw conveyors tailored for specific applications, became instrumental in enabling the flow of materials, reducing the reliance on manual labour for material handling tasks [6]. Following World War II, there was a growing emphasis on automation and technological integration in manufacturing. Automated Guided Vehicles (AGVs) were introduced for autonomous material transportation. Computer systems became integral for inventory management, order processing, and control of material handling equipment.

Tompkins and White estimate that between 20% and 50% of the total manufacturing, and operating expenses can be attributed to material handling. Furthermore, these authors claim that a cost reduction of at least 10% to 30% can be accomplished through effective facilities planning [43]. In material handling systems, the use of Automated Guided Vehicles (AGVs) has witnessed a steady rise for several decades [14] as they effectively reduce labour costs while enhancing the flexibility of material transfer between the workstations.

The use of AGVs is not trivial and has the complexity of routing within the facility without having conflicts like collisions among them. While collisions could be avoided by many approaches of which one is the use of unidirectional guidepaths. Although the use of real-time collision avoidance for AGVs offers more flexibility, unidirectional guidepaths offer a simpler solution with easy maintenance and implementation, eliminating the need for complex sensory AGVs. Guidepaths are the predefined routes or paths that guide the movement of AGVs within a facility. These paths ensure effective and controlled navigation of AGVs when transporting materials or products across workstations within a manufacturing system.

Integration of FLP and MHS

Facilities design is the joint selection of an integrated material handling system and plant layout. It is possible to realistically evaluate layouts using the material handling cost as an objective [42]. Integrated decision-making for facility layout and material handling equipment assignment problems has a positive influence and results in overall cost reduction when compared to the two-stage approach of solving the facility layout problem first and then solving the material handling equipment assignment problem [11], as such two-stage approaches could result in a loss of dependencies as the effectiveness of a layout depends on how well the material handling systems operate, which in turn depends on the layout of the facility.

The design of a new, or relayouting of an existing manufacturing facility could be considered a hierarchical decision-making problem, known as the Bi-level Optimization Problem (BLP) where the FLP is solved on one level and the MHS is optimized on the other. Bi-level Genetic Algorithms could be used to address such BLPs [27]. Legillon et al. [24] proposed a coevolutionary algorithm, where different populations were employed for each level, encouraging cooperation between them to generate solutions for the overall problem. Significant improvement was exhibited especially at lower levels when compared to the conventional approach.

Contribution of this Thesis

In this thesis, we design an integrated approach to find an optimal layout for a dynamic manufacturing system while designing and optimizing guidepath to prevent collisions among AGVs. Most of the literature regarding Dynamic Manufacturing considers the flow of materials within the system that changes periodically. In this thesis, extending the work done by Seidelmann and Mostaghim [36], we consider the orders generated to be completely random, to achieve the true dynamic nature of manufacturing. Within this dynamic environment, collisions among AGVs are mitigated through the implementation of guidepaths.

The facility layout problem is formulated as a pure Integer Problem and the guidepath layout problem is formulated as a zero-one integer linear programming problem [12]. Kaspi and Tanchoco [19] and Raj [13] propose that the optimal approach to solving such a problem may be by mathematical approach if only the flow intensity changes with time and not the placement of workstations. But since in our case, the guidepath problem is being integrated with the facility layout problem, finding the optimal positioning of workstations is also the problem that needs to be solved, the mathematical approach would not be the most suited and we chose to solve it using simulation.

The main goal of this thesis is to find better relayouting solutions for an existing layout of a dynamic manufacturing system while also designing and optimizing guidepath, for the complete removal of collisions among material handling systems to improve the optimization of the layout. Contrary to most of the literature, the number of transporters needed is also found as a result of this. The following questions are also answered in achieving the said goal.

- How to design guidepath layouts to remove collisions among Automated Guided Vehicles (AGV)?
- How to optimally make randomly generated guidepath layout feasible with minimal changes?
- How to seamlessly integrate the facility layout and guidepath layout problems into one?
- Will coevolution be a better approach for simultaneously optimising facility and guidepath layouts?

2 Prerequisites

A manufacturing system is a complex network of machines, workstations, and processes designed for the production of goods. The effectiveness of a manufacturing system directly influences productivity, cost efficiency, and overall operational success. Facility Layout Problem (FLP), Scheduling, Material Handling Systems (MHS), and multi-objective optimization are some of the techniques that play crucial roles in enhancing the efficiency and performance of a manufacturing system.

2.1 Facility Layout Problem (FLP)

Facility Layout Problem (FLP), the problem of finding or designing an efficient and effective layout for a facility, has been a subject of research for decades, if not centuries. The idea that the relative location of something or a group of things will influence the intended usage is not believed to be new. The problem concerns the placement of physical facilities like machines, workstations, and departments within a layout. Generally, two objective functions, qualitative and quantitative, are optimized. The quantitative objective in general could be to minimize material handling costs or to maximize productivity, and the qualitative objective could be to maximize some measure of closeness rating [34], which assesses how close a potential facility location is to other relevant infrastructure.

The inherent difficulty of FLPs arises from the multitude of possible layouts for a manufacturing facility. For a small number of workstations or departments within a manufacturing system, say N, there could exist a large number of possible combinations of location patterns or layouts, the factorial of N (N!). In the earlier stages, the manual selection of layouts was based on qualitative judgement using graphic aids. Typically six to eight were selected from all possible arrangements and these selected layouts were subject to further investigation. However, this conventional approach was tedious. The combinatorial nature of the problem made it infeasible to find a solution by exhaustive computerized enumerations because of the excessive time required. A feasible algorithm to solve such combinatorial problems was lacking. A breakthrough occurred with the proposal of heuristic algorithms exemplified by Gordon and Elwood's [1] proposed approach to guide the simulation to make it proceed in the direction of the optimum value of the objective function used, providing a feasible method for tackling the combinatorial challenges posed by FLPs.

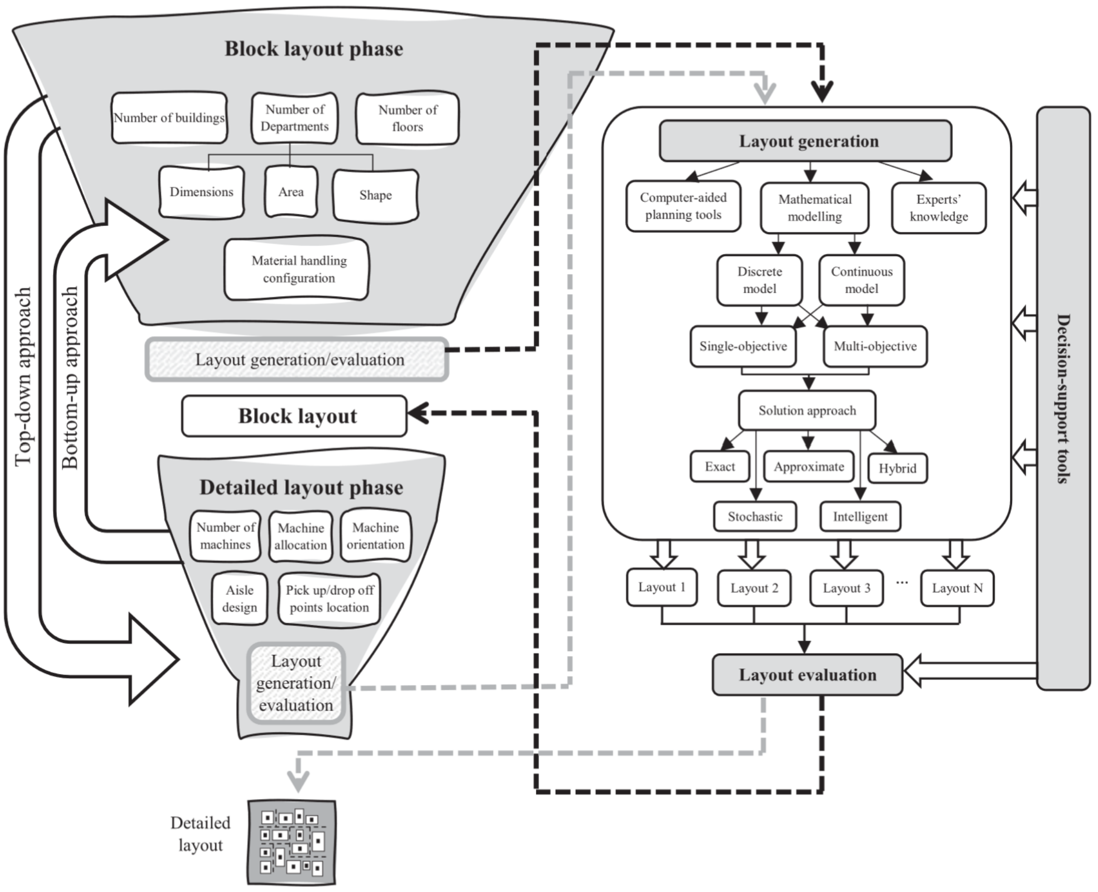


Figure 2.1: A conceptual framework aimed at guiding the formulation of a facility layout problem organizing the process into several key steps and considerations from [31]

2.1.1 Static and Dynamic nature of FLP

Depending on the approach taken in planning a facility layout, FLP can be classified as static or dynamic. In a Static Facility Layout Problem, planners assume that the flow or flow intensity of products or material across workstations within a facility remains constant throughout the time the layout is being planned for. On the other hand if the flow or the flow intensity of materials changes within the planning horizon, the problem is called Dynamic Facility Layout Problem [17].

The facility layout problem is often treated as a static one. However, the flow is unlikely to remain unchanged for a long planning horizon. Dynamic Facility Layout Problem is crucial not only for a long planning horizon and the need for dynamic treatment of the layout problem is supported by Nicloe and Hollier [26], who in their study concluded that "Radical layout changes occur frequently and that management should therefore take this into account in their forward planning". It is assumed that designers generally face the facility layout problem when they create a new manufacturing system or service, or when they expand, consolidate, or modify existing systems. But based on a survey conducted [26], nearly half of the companies surveyed had less than 2 years of average layout stability. This means, that even established manufacturing companies need to change the layout every 2 or 3 years for optimal usage.

2.2 Scheduling

Scheduling refers to the systematic assignment of tasks or jobs to the responsible facilities and transporters. In every time step, the algorithm takes in all the relevant information regarding the current state of the system. It predicts necessary decisions on which tasks should be performed and, where and when they should be performed, ie., the algorithm assigns the necessary entities for the smooth execution of these tasks and also establishes a timeline for each task. Scheduling algorithm plays a crucial role in optimal usage of resources thus enhancing the overall performance of a manufacturing facility.

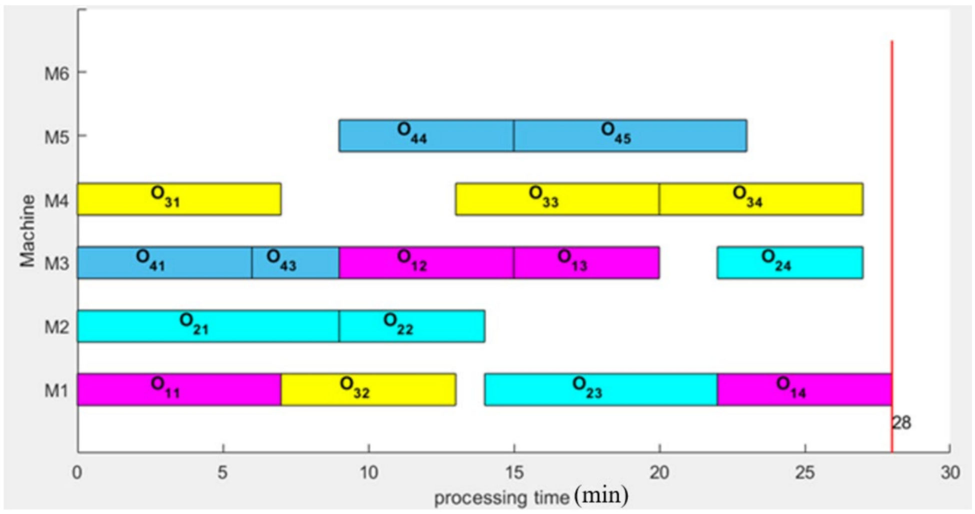


Figure 2.2: Gantt chart of a scheduling problem where O_{ij} denotes the j-th operation of the i-th product, and Mk represents the k-th machine as illustrated in [52].

To solve the scheduling problem in dynamic manufacturing systems, Seidelmann et al. [37] presented a novel simulation framework for evaluating general manufacturing system designs. It was applied to a simplified smartphone manufacturing process with minimal lot size and multiple variants of smartphones recognizing mass customization. A global planner was designed to create tasks by considering all the available workstations and transporters simultaneously. Specifically observer/controller architecture was used to reconfigure task priorities.

2.3 Material Handling Systems (MHS)

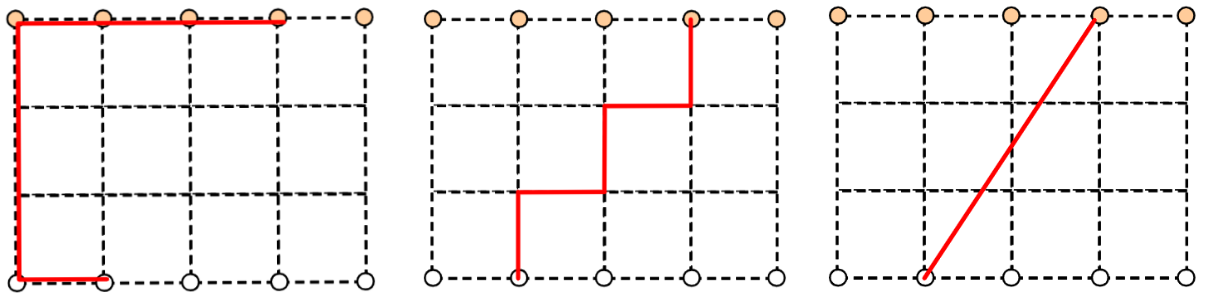
Material Handling Systems refer to the vast number of equipment and systems responsible for the storage, handling, control and movement of materials across different departments within a facility during manufacturing. MHS can account for a significant amount in the cost of manufacturing a product and thus the effective usage and avoidance of unexpected failures in such systems could lower the maintenance cost which in turn will reduce the manufacturing cost [39].

2.3.1 GuidePath: Automated Guided Vehicles (AGV)

The AGV guide path layout problem was first defined by Gaskins and Tanchoco as a zero-one integer linear programming problem. In this problem formulation, the guidepath is established as a node-arc network, where nodes denote pick-up/delivery stations and aisle intersections and arcs represent the guidepath which connected these nodes. Kaspi and Tanchoco[19] introduced a more computationally efficient approach utilizing the branch and bound technique to solve the guidepath problem effectively. However, The departmental layout was not taken into consideration, rather, the solution was solely based on a flow intensity chart.

The placement of departments is the primary concern in configuring the guide paths for an AGV. The location of pickup and delivery points are represented as nodes and the routes of AGVs can be modelled as graphs. The arcs connecting these nodes give the guide path to be followed by the AGV. Duinkerken et al. [8] discussed different routing procedures within a container terminal:

- Loop - routed in a loop
- Mesh - routed in taxicab/Manhattan way
- Cross-Over - routed along a straight line connecting origin and destination



(a) Loop Routing **(b)** Mesh Routing **(c)** Cross-over Routing

Figure 2.3: Different routing strategies of AGVs assuming that the locations of origin and destination are fixed in a container terminal explained in [8]

A study of more than 50 flexible manufacturing systems in Japan found that unidirectional loop layouts were preferred [18]. Although bidirectional systems offer more flexibility in material transfer, it is widely recognized that they require more expensive, sophisticated controls and larger investments.

2.4 Optimization

Optimization is to find the best or most effective outcome concerning some criteria or arrangement. With the criteria or arrangement being represented as a mathematical function generally referred to as objective, optimization is to find the value where this said function has the maximal or minimal value, and this value is said to be the optimal solution.

2.4.1 Multi Objective Optimization (MOO)

Multi-Objective Optimization is the process of finding optimal solutions for a problem with more than one objective, which generally are in conflict with each other and need to be considered simultaneously. The problem to be optimized is formulated as a mathematical equation or a simulation to be solved by combining all the conflicting objectives into one or, by using one of the optimization algorithms. It's crucial to note that MOO optimizes all objectives simultaneously aiming to obtain a Pareto optimal front where solutions are diverse and no solution is better than another in all objectives representing different trade-offs between objectives. In MOO, a population of solutions is typically considered, rather than focusing on a single solution. One doesn't need to specify their preferences upfront, instead, they can later select from the found Pareto front.

2.4.2 Weighted Sum

The weighted sum of objectives is a technique in which all the objectives to be optimized are assigned a weight that reflects their relative importance, such that the sum of all these weights is constrained to equal one. The weighted sum is then calculated by taking a linear combination of the individual objectives, where each objective's contribution is scaled by its assigned weight. This method allows for the aggregation of multiple objectives into a single combined objective function.

Mathematically, for a minimization problem with n objectives $(f_1, f_2, \ldots, f_n)$ and corresponding weights $(w_1, w_2, \ldots, w_n)$, where $\sum_{i=1}^{n} w_i = 1$, the weighted sum objective function (F) is calculated as follows:

$$F = w_1 \cdot f_1 + w_2 \cdot f_2 + \ldots + w_n \cdot f_n \tag{2.1}$$

Adjusting the weights can lead to different solutions allowing the decision-maker to make preferences regarding the trade-offs between different objectives. However, it's important to note that this approach typically returns only a single solution, rather than a population of solutions. Additionally, one must specify their preferences before starting the optimization process, as they need to assign weights to each objective based on their importance.

2.4.3 Evolutionary Algorithms

Evolutionary Algorithms are a class of optimization algorithms that mimic natural evolution. An evolutionary algorithm is a population-based stochastic direct search algorithm. In the optimisation context, each individual in the population represents a solution to the optimization problem, referred to as a chromosome. The algorithm collocates a population of individuals which evolve through mutation, crossover and selection while being evaluated by a suitable fitness measure to find out which of these individuals fits better into the environment being tested for. Evolutionary algorithms, with their population-based search and non-dominated sorting, are well-suited for solving MOO problems to find diverse and optimal solutions.

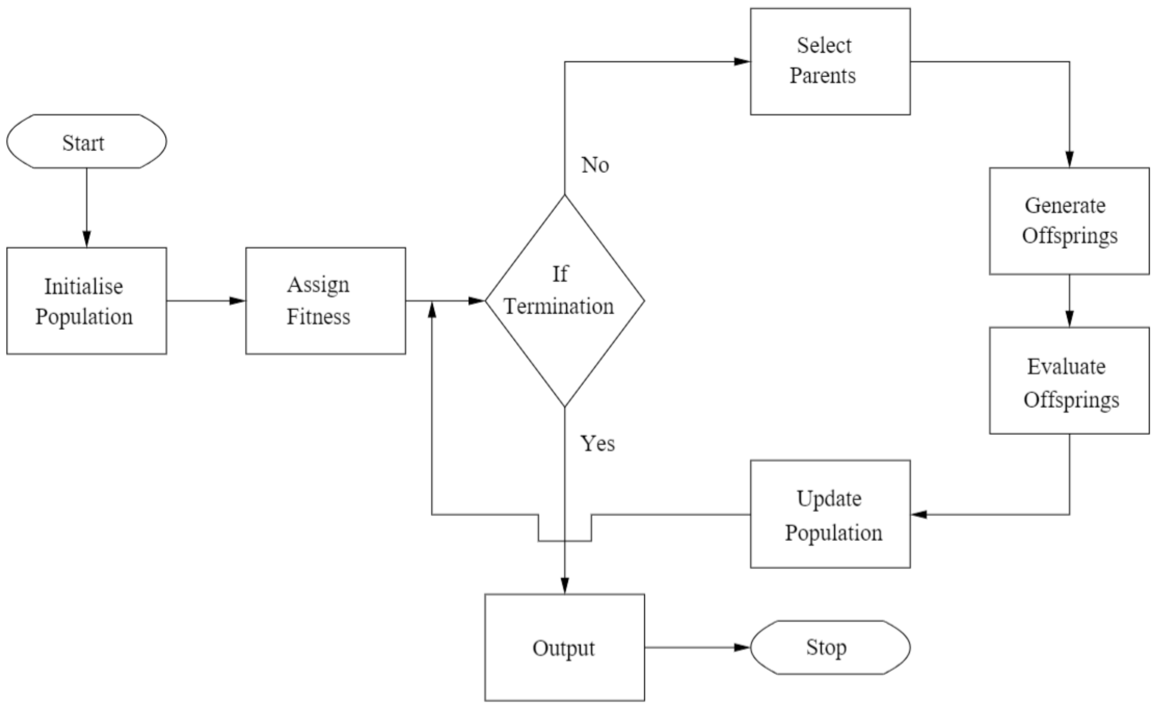


Figure 2.4: A flowchart illustrating the general structure of an Evolutionary Algorithm outlining the steps involved in the algorithm's operation [40]

2.4.4 Genetic Algorithms

Genetic Algorithms are one of the many variants of evolutionary algorithms. They are search-based heuristics inspired by natural selection and genetics. In each generation all the solutions are evaluated based on a fitness function and the better solutions from these are selected and subject to crossover and mutation to get new solutions which are again evaluated and added back to the main population if better than the parent solutions.

Selection

As the name suggests, selection is the process of selecting solutions from the population for mating. Some of the different types of selection operators are:

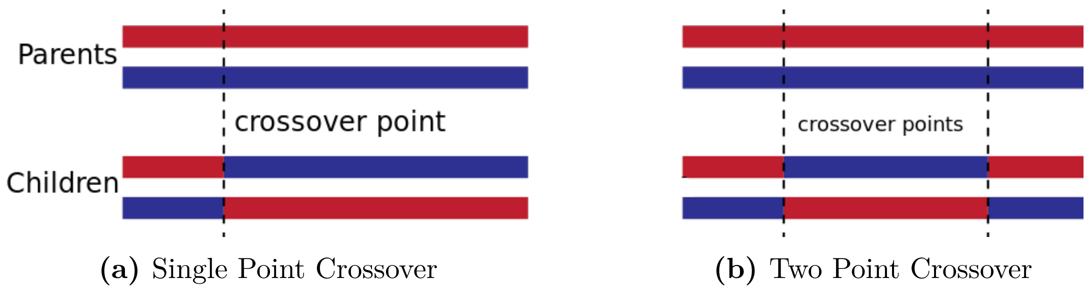
- **Tournament Selection:** Individuals are randomly chosen from the population and the one with the best fitness is selected as a parent for crossover.

- **Roulette Wheel Selection:** A probability is assigned to each individual of the population based on their fitness value and selection is performed by spinning the roulette wheel.

- **State Selection:** Used for population control, whenever a new individual is created, it competes with an existing individual and the better among them is selected to stay.

Crossover

Crossover is the process of combining two chromosomes(parents) to create two offspring (children). Parts of two-parent solutions are swapped to form two children solutions. This process also referred to as mating or recombination, exploits the knowledge gained from previous generations by combining the features of two good solutions. There are different types of crossover based on the number of points for crossover.



(a) Single Point Crossover **(b)** Two Point Crossover

Figure 2.5: Classificatios of crossover operators categorized by the number of crossover points. The first image depicts a single crossover point and the second category has two crossover points.

Mutation

Mutation introduces random changes to individuals in an optimization algorithm, enabling the exploration of new regions within the search space and preventing the algorithm from becoming trapped in local optima. Various types of mutation strategies exist, including but not limited to random, normal, and polynomial mutations. Mutation operators can be custom designed to address unique characteristics that suit the specific requirements of a problem, offering flexibility and adaptability in optimizing solutions.

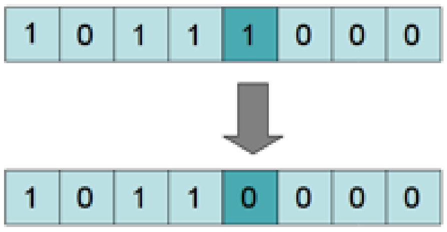


Figure 2.6: To introduce new genetic material, certain individuals undergo mutation, where a gene is randomly altered[1].

2.4.5 Coevolutionary Algorithms

Coevolutionary algorithms are based on the concept of natural complementary evolution of closely allied species [10] [54]. While traditional evolutionary algorithms consist of a single population addressing a particular optimization problem, coevolutionary algorithms are composed of different populations which evolve simultaneously by interacting with each other. Coevolutionary algorithms are commonly used in optimization problems instead of multi-level algorithms, where the performance at one level depends on the behaviour or decisions of the other levels [24]. The dynamic nature of these algorithms helps capture complex relations and interdependencies among different populations.

[1]https://www.codingame.com/playgrounds/334/genetic-algorithms/tools

3 Related Work

In this section we present an analysis of the existing work related to the Facility Layout Problem, Material Handling Systems, the integration of FLP and MHS, and Scheduling highlighting significant contributions close to the current work, to trace the development of ideas and approaches in this field, emphasizing the unique improvements we bring to the field.

3.1 Facility Layout Problem

The research on facility layout problem (FLP) started in the 1950s. In an early attempt to solve this, Koopmans and Beckmann [21] interpreted the problem as the assignment of manufacturing plants to locations with the objective of minimizing the inter-plant transportation cost. Facility Layout Problem when addressed separately as a single problem could be solved using any of the optimization problems. Bunker et al. [9] used a Coevolutionary approach to solve hierarchical problems related to facility layout optimization of a very large facility. To address the large problem sizes, the departments were divided into clusters and these clusters evolved independently in separate areas, and the position and size of these areas also evolved independently yet simultaneously using genetic algorithms.

3.1.1 Dynamic Facility Layout Problem

Multi-period facility layout problem, generally also referred to as Dynamic Facility layout problem, involves optimizing the arrangement of workstations within a facility over multiple time periods as the flow intensity across workstations within the facility changes after the fixed time period. Turanoğlu and Akkaya [44] used a new hybrid heuristic algorithm, called simulated annealing based on bacterial foraging optimization (BFO) to solve the multi-period facility layout problem in which an optimal layout is adopted for each time period.

To address the dynamic nature of the facility problem, Peng et al. [28] introduced a robust approach to create multiple layouts, each flexible across a different demand scenario. A mathematical model and an adaptive genetic algorithm were used to solve the problem while considering demand uncertainty and material flow variation between facilities. Pérez-Gosende [30] addressed the dynamic facility layout problem by presenting a multi-objective mixed-integer non-linear programming (MOMINLP) model called bottom-up mDFLP. Unlike the traditional top-down approach that initially determines the block layout (BL) and subsequently the detailed layout (DL) of each work cell, this strategy adopts a bottom-up strategy. New innovative elements such as Total Closeness

Rating (TCR) and Area Utilization Ratio (AUR) were introduced to enhance optimal space utilization.

The majority of existing literature on FLP focuses predominantly on the arrangement of machines rather than on machine selection. The integration of arrangement and selection is often overlooked. Addressing this gap, Seidelmann and Mostaghim [36] proposed a methodology for solving the FLP that encompasses not only the arrangement of machines but also the selection of machines, how many and what machines should be selected. It was found that a traditional NSGA-II [7] based on uniform random mutation has its limitations and was not suitable due to insufficient exploration of search space. A modified mutation was introduced to eliminate the inherent bias and promote a diverse set of solutions.

3.2 Scheduling

Dispatching rules are predefined heuristics used in scheduling to determine the sequence in which jobs or tasks are processed. These rules play a crucial role in job-shop scheduling, a widely studied problem in operations research and computer science. Job-shop scheduling is an optimization problem in which a set of jobs generally with varying processing times must be processed on a set of machines in a specific sequence. The objective is to efficiently allocate these jobs across the machines and find an optimal sequence adhering to specific constraints. In job-shop environments, decision-making is guided by dispatching rules, with jobs prioritized based on characteristics, machine availability, and workflow dependencies, aiming to optimize efficiency and productivity.

Rolf et al. [33] used a genetic algorithm to optimize the dispatching rules. The algorithm outperformed standard dispatching rules for a case study on printed circuit board production. Applicability in real-time decision-making for Industry 4.0 cyber-physical systems was suggested. Braune et al. [4] presented in their paper a Genetic Programming (GP) approach for generating expression trees to serve as priority dispatching rules in flexible job shop and hybrid flow shop scheduling problems. This GP approach outperformed existing literature-based priority rules. The evolved trees demonstrate extrapolation capabilities and the concept of iterative dispatching rules further enhanced performance.

3.3 Material Handling Systems

Conveyor belts are a well-established type of MHS which is widely used in various fields like manufacturing, mining, logistics and agriculture. Salawu et al. [35] used an Engineering Equation Solver (EES) to solve the mathematical equations developed to incorporate the operating parameters of the conveyor belt system. The results were simulated to select the optimal parameters for the design. Butt and Jedi [5] used Discrete Finite Element Analysis (DFEA) tools and techniques to design a mechanical conveyor system for the food industry.

Congestion in manufacturing facilities refers to the excessive amount of traffic including queues, bottlenecks, and conflicts among material handling equipment leading to delays.

Addressing this challenge of workflow congestion Zhang et al. [51] introduced a combination of probabilistic and physics-based models for workflow interruptions using object-oriented simulation models targeting rerouting strategies. The main focus was on the effectiveness of rerouting in mitigating delays due to congestion. Subsequently, Zhang et al. [50] presented the Full Assignment Problem with Congestion (FAPC) to optimize layout and flow routing simultaneously while considering congestion. Benjaafar [3] proposed a model capturing both the facility layout and congestion-related measures. Contrary to the conventional Quadratic assignment problem (QAP) a new formulation was introduced with minimizing the work-in-process (WIP) as objective. It was highlighted in the paper that the desirability of a layout is influenced not only by material handling factors but also by non-material handling factors like department utilization levels and processing times. This advocates for a more comprehensive approach to Layout design-related problems. In another contribution, Pourvaziri and Pierreval [29] introduced a novel perspective of considering WIP alongside handling and rearrangement costs for Dynamic Facility Layout Problems (DFLP). An analytical approach based on QAP formulation and open queuing network theory was used to account for the queues induced by machines and material handling systems.

3.3.1 Facility Layouts Considering Collisions

In most recent times, Xin et al. [47] suggested a new path-planning approach that makes the AGV system more robust and flexible. Most of the existing methods for path planning are static but the proposed method was dynamic. Collision paths were dynamically determined and optimized to find the best collision-free paths. An additional decomposition technique was used to make real-time decisions and reduce computational costs. While the use of real-time collision avoidance is gaining popularity, the majority of literature on collision avoidance in facility layout is on using loops or zones. A single loop covering all workstations as in Figure 3.1 is used, or, the shop floor of a facility layout is divided into a group of non-overlapping loops or zones as shown in Figure 3.2, and a single AGV is dedicated to serve in each loop to avoid collisions or conflicts. Transfer points are set up across zones to be able to transfer the products of one zone to another and vice versa.

Kumar et al. [23] discussed the design of a unidirectional loop layout problem in a Flexible Manufacturing System (FMS). A single loop was designed to cover all the workstations within the facility and the goal was to find the optimal ordering of machines in a loop to minimize the total number of loop traversals. Particle Swarm Optimization(PSO) technique was applied to solve the said loop layout problem.

Shalaby et al. [38] proposed a two-phased heuristic algorithm to divide the facility into zones. In the first phase, the best zones were searched for in the solution space based on their integrity and feasibility. In the second phase, a 0-1 integer programming model was used to select the zones that optimize the material handling costs, or, in between zone transfers, or, the average workload of zones. As a result, a bottleneck-free system with better workload balance between all AGVs was achieved. To group 'n' machines into 'N' loops for a multiple-load AGV system, AGVs that could handle more than one product, Rahimikelarijani et al. [32] illustrated a new non-linear mathematical programming model. Workload balancing among all loops and minimization

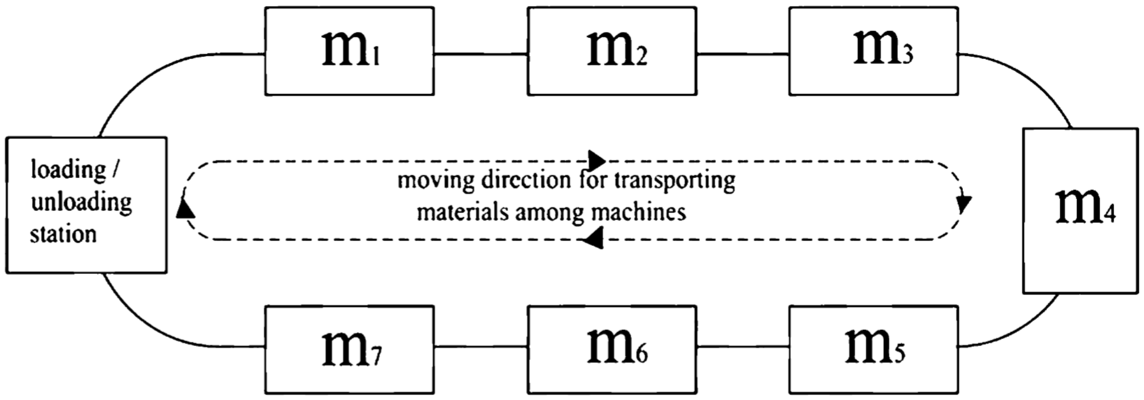

m1
m2
m3
loading / unloading station
moving direction for transporting materials among machines
m4
m7
m6
m5

Figure 3.1: Illustration of a loop layout where AGVs move in a loop to collect transport and deliver material among machines across the layout [23].

of inter-loop and intra-loop transportation were the objectives. The model found that the use of multiple-load AGV instead of single-load AGV could reduce material handling costs.

One disadvantage of the traditional tandem zone strategy is the large delay in satisfying demand when there is an imbalance of loads across zones. The imbalance could arise because, for each vehicle, its zone is fixed and can not be changed, and vehicles are also not allowed to help each other. To overcome this, Ho [16] introduced a new dynamic zone strategy which relies on two procedures:

1. **Zone Adjustment Procedure** - zones change dynamically to adjust to the production demand.
2. **Zone Assistant Procedure** - vehicles are allowed to help each other to balance workloads

A Simulated Annealing-based zone division method was used to find near-optimal zone designs. The dynamic zone strategy was found to be better than the fixed zone strategy in terms of the overall performance of systems.

For efficient management and collision-free movement of AGVs in an intelligent manufacturing workshop, Wang et al. [46] introduced a heuristic ant colony algorithm to solve scheduling and path planning problems. An approach was proposed to first allocate transporter tasks based on priority and create an initial feasible path for each AGV. Then AGV collision detection and anti-collision algorithm was designed and used to plan the global collision-free path of multiple AGVs.

3.4 Integrated FLP and MHS

An Open-Field Layout Problem (OFLP) typically refers to the challenge of determining the optimal spatial arrangement of machines within an open or unobstructed space without predefined structures or fixed constraints. Yang et al. [48] presented an optimization approach for collaborative solving of the OFLP and Automated Guided Vehicle (AGV)

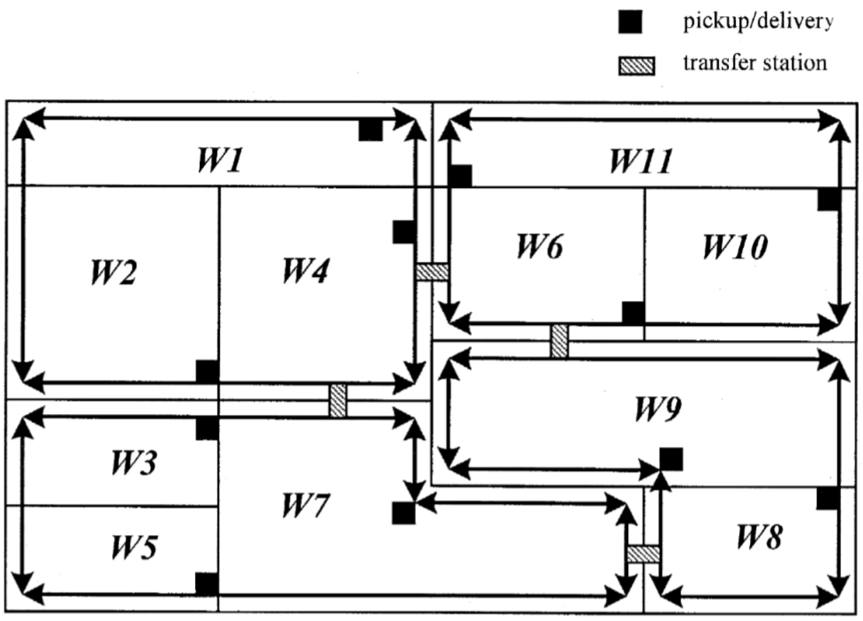


Figure 3.2: An illustration demonstrating the tandem configuration where the layout is segmented into multiple zones, each served by a dedicated AGV and designated transfer points facilitating the movement of material across zones [16].

path planning in intelligent workshops. A hybrid algorithm was proposed to find the optimal facility layout and corresponding multiple AGV paths. Discrete Simulation(DES) was utilized to simulate logistics data within the layout to capture the dynamic nature of logistics processes.

If the Dynamic Facility Layout Problem which considers changing flow between departments over multiple periods is combined with material handling system designing, the result is an even more complicated problem. To solve these synchronized problems, Kheirkhah et al. [20] proposed a Coevolutionary bilevel algorithm that tries to improve two different subpopulations related to two levels of DFLP incrementally, and periodically exchanging information with each other. The population of each level of the problem is improved by a separate Genetic Algorithm.

4 Methodology

A novel simulation framework was introduced for evaluating manufacturing system designs in [37]. Extending this foundational work, a new method was presented in [36], addressing the gap in the literature concerning the joint optimization of machine arrangement and selection. This method was built upon the simulation architecture and a planning and scheduling algorithm, called global planner, with minor adaptations tailored to specific requirements utilizing only a subset of the complete architecture and focusing on the relevant and essential components. This study is influenced by the evolving landscape of manufacturing, characterized by a shift towards mass customization and shorter product life cycles. By extending the work of the above-mentioned papers, a new methodology is presented for considering the conflicts among the material handling equipment while finding a cost-effective layout of a dynamic manufacturing facility. The same simulation architecture of [36] and a transportation-centric scheduling algorithm are used. While essential components are presented here, it is recommended to refer to the original work for a comprehensive understanding of the simulation framework and global planner architecture.

4.1 Simulation Architecture

The simulation framework is structured around entities, divided into a metamodel and a physical model. Within the metamodel, entities define the categories of machines, parts, etc., and their interactions. On the other hand, the physical model simulates real instances of these entities within the simulated manufacturing environment. The metamodel encompasses Product Kinds, Product Families, Product Interfaces, Recipes, and Skills. These entities are stateless and serve as a blueprint for the actual instances in the physical model. Meanwhile, the physical model includes entities such as the Shop Floor, Products, Workstations, Ports, Transporters, Orders, and corresponding tasks.

Figure 4.1 illustrates the two parts of the framework. The entities on top constitute a metamodel representing the manufacturing process. The lower entities construct a representation of the real world, referencing and aligning with the established metamodel. The metamodel has no knowledge of the physical model but the physical model references entities of the metamodel to define the simulated real-world entities. This configuration allows the framework to inherently model diverse scenarios, where each instance of any entity possesses a globally unique identifier. The instances of entities interact based on the specifications defined in the metamodel. Products are instances of Product Kinds and can own other Products, forming a hierarchical structure. Workstations and Transporters are key active entities that manipulate the state of the shop floor. Workstations execute Recipes based on their Skills. Recipes encapsulate manufacturing steps, detailing required

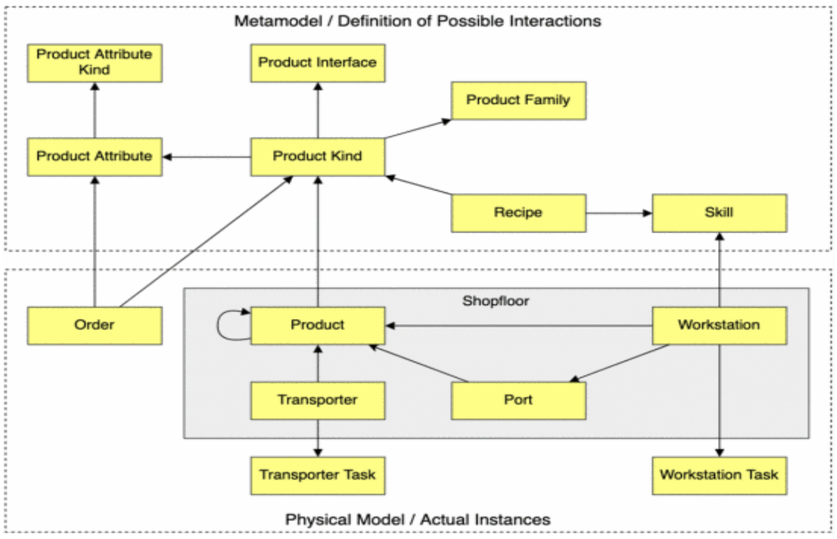


Figure 4.1: Overview of the simulation framework's entities. Those on top present a metamodel representing the manufacturing process, while those down depict a real-world representation that correlates with the established metamodel [37].

input Product Kinds, Skills, and processing times. Workstations have input and output Ports, execute Recipes, and are associated with specific Skills. Ports serve as storage connected to Workstations, and Transporters own Products and perform tasks like moving to a position and acquiring and depositing Products.

Orders represent customer demands and have priorities, deadlines, and fulfilment times. The shopfloor, represented as a two-dimensional grid, defines the spatial layout of the manufacturing area. The simulation operates iteratively until all orders are fulfilled, with processing times given in time steps, portraying the dynamic shop floor environment, focusing on workstations and transporters simultaneously, and influencing the overall system based on scheduling tasks.

4.2 Scheduling Algorithm

The scheduling algorithm, called task planner, generates tasks for workstations and transporters based on current orders and the state of the shop floor. The task planner is built upon the global planner proposed in [37] and is tailored to suit the transportation-centric approach. It effectively manages the orders collectively and orchestrates the production of products necessary for the fulfilment of the orders and their transportation. Three types of tasks are designed to enable the production workflow and product transportation.

1. **Workstation Tasks:**
    - For each order, the planner identifies the workstations and recipes required to produce the specified product.
    - The production tasks are assigned to the workstations based on their workload as a form of load-balancing, for even distribution of tasks among workstations.

- Workstation tasks are created dynamically for missing products, where the necessary inputs already exist.

2. **Port Tasks:**
    - Every workstation incorporates an input port and an output port, which in this current work are assumed to coexist within the space provided for the workstation.
    - Port tasks involve the transfer of the products between ports and transporters.
    - Port tasks are created based on product ownership and transportation needs.
3. **Transporter Tasks:**
    - Transporter tasks are generated to transport products from the output port of one workstation to the input port of another workstation ensuring the input requirements of all workstation tasks.
    - Planner evaluates the transportation needs of workstation tasks and assigns transporter tasks accordingly.
    - The planner determines the path the transporter needs to take to reach its destination based on the provided guidepath layout using a breadth-first search to find the shortest path.
    - Based on the path, necessary transport tasks are created. One transporter task corresponds to movement from one node to another node on the guidepath.
    - Transporter tasks are also created when a transporter without any tasks is occupying a workstation without any port tasks and needs to be accessed by another transporter.

The task planner, in every time step, retrieves all the unfulfilled orders and orders them based on their priority. Then Workstation tasks are scheduled based on orders and port tasks are created based on the availability of transporters. The path is determined for the transporter based on the destination of the product it has and transporter tasks are created. All the tasks are executed and the planner checks for completed workstation tasks and updates. The same is illustrated as a flow chart in figure 4.2.

The task planner also ensures that no more than one transporter occupies a position on the job floor at any time step except when the transporters are spawned, as all the transporters are spawned at the same position. If a transporter wishes to move to a position already occupied by another transporter, the former has to wait until the latter moves out. This led to an issue of the last order not being completed when one transporter was occupying the delivery station and the other transporter was waiting to move into the same delivery station to deliver the last product. The transporter occupying the delivery station does not move as it does not have any tasks, being all the orders are executed and only the delivery of the final product is remaining. To overcome this, for every transporter with no active tasks occupying a workstation with no port tasks, its surroundings are checked for new transporters. If there is any new transporter in its surroundings, a new transporter task is created for the transporter to move out of its current position into a random neighbouring position.

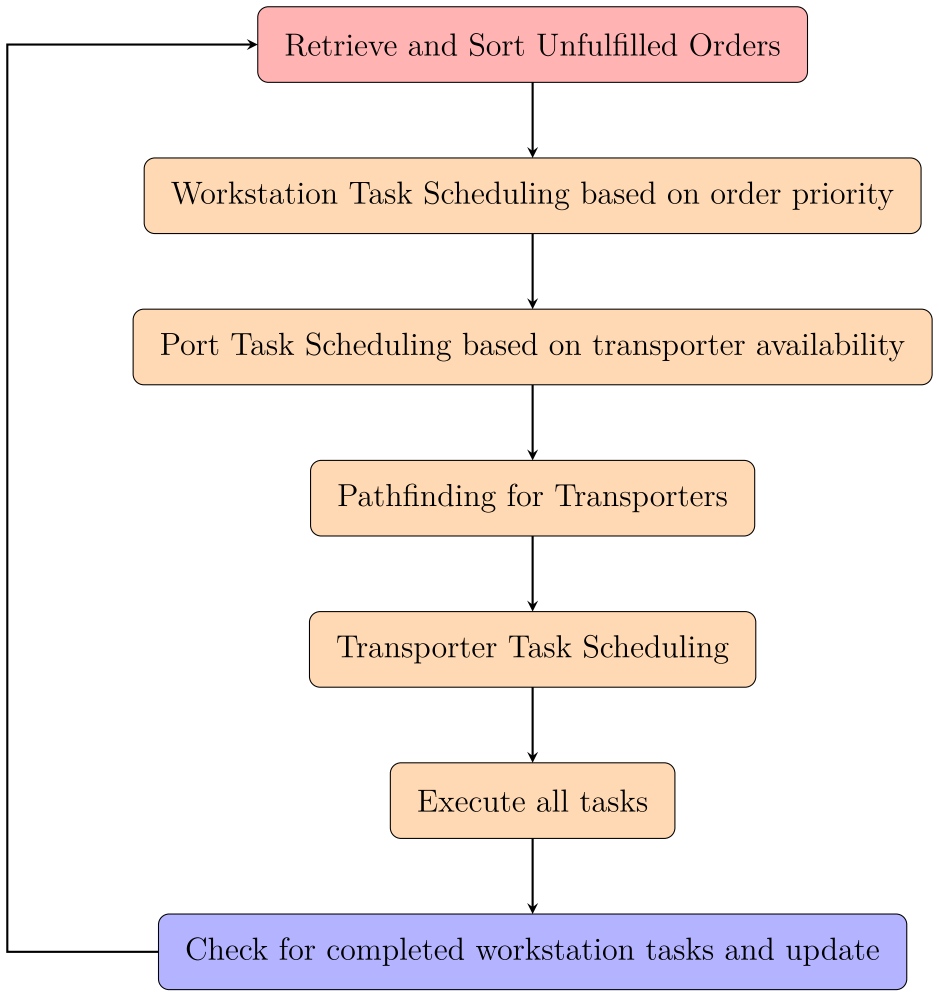

Retrieve and Sort Unfulfilled Orders
Workstation Task Scheduling based on order priority
Port Task Scheduling based on transporter availability
Pathfinding for Transporters
Transporter Task Scheduling
Execute all tasks
Check for completed workstation tasks and update

Figure 4.2: Generic flowchart depicting the iterative task planner algorithm which is responsible for unfulfilled orders retrieval, workstation and port task scheduling, transporter pathfinding, and task execution.

4.3 Optimization Approach

For optimization, suitable encodings for layouts and guidepaths are necessary, which are explained first. Some but certainly not all of the encodings generated will be feasible, so the repair function used to make them feasible, and the algorithms and operators used to optimize them are defined next. Then the objective functions and how they are calculated are described and lastly, the indicators used to evaluate and compare the performance are specified.

4.3.1 Encoding

Encoding is crucial and defines how solutions to a problem are represented. It translates the characteristics of a problem into a format suitable for genetic operations like crossover and mutation. The encoding for the joint optimization of the facility layout problem and guide path layout problem (L) is the concatenation of the two encodings, facility layout encoding (L_f) and Guidepath layout encoding (L_g).

$$L = L_f + L_g \tag{4.1}$$

Facility Layout

The facility layout encoding (L_f) is mathematically expressed as a sequence of integers, with a length of $N + 1$, where N represents the set of available positions in the facility for workstation allocation. Formally, $L_f = (f_1, f_2, ..., f_N, T)$, where $f_i \geq 0$ for $1 \leq i \leq N$ denotes the type of workstation allocated at position i. Specifically, $f_i = 0$ implies no workstation allocation, and $f_i > 1$ indicates the type of workstation according to Table 5.1. The last integer in the encoding T, signifying the number of transports to be employed within the facility, satisfies $1 \leq T < 20$.

$$L_f = (e_1, e_2, ..., e_N, T) \quad \text{where} \quad \forall i,\ e_i \geq 0,\ T \geq 1,\ T < 20,\ N = \text{number of positions} \tag{4.2}$$

GuidePath Layout

The encoding for guide path layout (L_g) is a sequence of zeroes and ones (Binary Encoding) representing the direction of arcs that constitute the guidepath. The sequence length equals the maximum number of arcs with the number of positions available in the facility being the nodes. For a modular grid, the number of arcs in a grid layout is determined by:

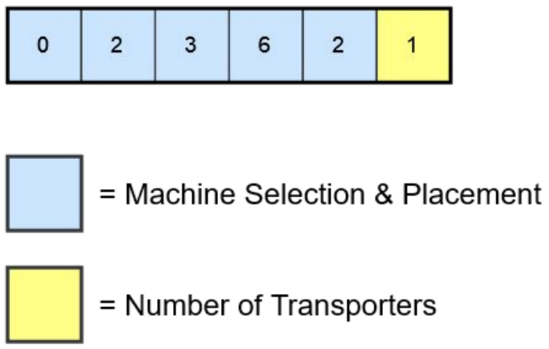
$$\begin{aligned} \text{Number of arcs} = \text{Number of Rows} \times \text{Number of Columns} - 1) \\ + \text{Number of Columns} \times (\text{Number of Rows} - 1) \end{aligned} \tag{4.3}$$

$$L_g = (g_1, g_2, ..., g_M) \quad \text{where} \quad \forall j,\ g_j \in \{0, 1\},\ M = \text{number of arcs} \tag{4.4}$$

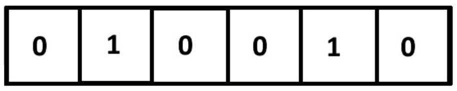
Here, 0 maps to the arcs extending from left to right or from top to bottom(lower position to higher position) while 1 maps to the arcs of opposite direction, either from top to bottom or right to left(higher position to lower position).

4.3.2 Repair Function

As one of the primary goals is to find a cost-effective layout, the initial population is created as a copy of the existing layout(source solution, L_o) and the random mutation operator, as illustrated in Algorithm 2, is applied once. The initial layout needs to be optimized and guidepaths are selected as a means to avoid collisions ie., the initial layout does not have any guidepath implemented. However, a guidepath layout is integrated with the initial layout for ease of implementation, as this guidepath layout of the initial layout (source solution) is not used in calculating the relayouting cost. Because of the randomness in initializing the population, not every population is a feasible solution in



(a) Facility Layout Encoding



(b) Guidepath Encoding

Figure 4.3: For encoding, facility layout is represented by integers, while guidepaths are denoted by zeros and ones.

terms of both facility layout and guide path layout, so we check for the feasibility of every solution and use the repair function to ensure the feasibility of every solution before evaluation.

In the simulation used, the type of workstations determines the tasks they can perform which in turn determines the products that can be produced. A layout not having the minimum required workstations will not be able to fulfil all the potential orders because of the lack of certain skills. The scheduling algorithm can not assign tasks properly on such layouts which makes them infeasible. Also, the guidepath layout is not trivial and not all layouts are feasible. It is not simple to find a feasible layout. A guide path is said to be infeasible when it is not strongly connected, ie., not every position is accessible from every other position. The guidepath layout not being strongly connected results in the planner not being able to find the paths for some tasks, thus making the layout infeasible. These problems of infeasibility were solved by using a repair function explained in Algorithm 1.

The repair function is divided into two parts. The first part modifies the facility layout as proposed in [36], where the skills present are obtained and those lacking are added accordingly. The second part repairs the guidepath layout. The feasibility of the guidepath could be ensured by the connectivity and reachability of the graph representing the layout. A graph is well connected if no node (position) is a sink or a source, i.e., every node has at least one incoming arc and one outgoing arc. For connectivity, every position is checked if well connected and for those which are not well connected, the direction of one of the arcs starting or ending at that position is reversed and checked for connectivity. A graph is said to be reachable if it is strongly connected, i.e., every node is accessible from every other node. To ensure connectivity, all the strongly connected components of the graph are calculated using Tarjan's Algorithm [41] and these components are checked if well connected as a whole, i.e., not a sink or a source.

Algorithm 1 Repair Funcion

```
1:  $L$ = Integrated encoding to be repaired $\leftarrow L_1 = L_{f1} + L_{g1}$
2:  $L_f$ = encoding of facility layout
3:  $L_g$ = encoding of guidepath layout
4:  $B_l$ = vector of lower boundaries for L
5:  $B_u$ = vector of upper boundaries for L
6:  $S$ = list containing each skill of the scenario once
7:  $S_c$ = list of all zeros, same size as $S$
8:  $A_i$ = list of arcs with direction represented by 0
9:  $A_u$ = Empty list of arcs, same size as $A_i$
10: $G_g$ = directed grid graph of arcs
11: $P_{nc}$ = list of all positions not well connected
12: $C_p$ = list of strongly connected cycles of a $G_g$
13: $q$ = queue to store nodes/positions whore arcs are to be reversed
14: $q_a$ = queue to store lists of arcs to store changes of reversing separately
15: $M_r$ = map to store positions and its arcs reversed
16:
17: for $i = 0$ to size of $L - 1$ do
18:     $L(i) = \max(L(i), Bl(i))$
19:     $L(i) = \min(L(i), Bu(i))$
20: end for
21:
22: for $i = 0$ to size of $L_f - 2$ do
23:     if $L_f(i) > 0$ then
24:         get skill $s$ of workstation $w$ corresponding to $L_f(i)$
25:         get index $j$ of $s$ in $S$
26:         $S_c(j) += 1$
27:     end if
28: end for
29:
30: for $i = 0$ to size of $S$ do
31:     while $S_c(i) == 0$ do
32:         $r$ = random integer between 0 and size of $(L_f - 2)$
33:         if $L_f(r) > 0$ then
34:             get skill $s$ of workstation corresponding to $L_f(r)$
35:             get index $j$ of $s$ in $S$
36:             if $S_c(j) \leq 1$ then
37:                 continue
38:             end if
39:         end if
40:         get random workstation $w$ with skill $S(i)$
41:         get index $e$ of the encoding that represents $w$
42:         $L_f(r) = e$
43:         $S_c(i) = 1$
44:     end while
45: end for
46:
47: $A_u \leftarrow A_i.copy()$
```

— **Continued on next page**

Algorithm 1 (Continued)

```
48: for $i = 0$ to size of $A_i$ do
49:     if $L_g(i) == 1$ then
50:         change the direction of arc, $A_u(i)$ to 1
51:     end if
52: end for
53:
54: $G_g \leftarrow$ directed grid graph of $A_u$
55: while $G_g$ is not strongly connected do  ▷ Every position reachable from every other position
56:     while All nodes of $A_u$ are not well connected do
57:         $P_{nc} \leftarrow$ not well connected nodes of $A_u$
58:                 ▷ Nodes which do not have at least one incoming and one outgoing arc
59:         for all $position$ in $P_{nc}$ do
60:             if $position$ is not well connected then
61:                 $q \leftarrow$ add($position$)
62:                 $q_a \leftarrow$ add($A_u$)
63:                 while $q$ is not empty do
64:                     $current \leftarrow$ first element of $q$
65:                     $currentarcs \leftarrow$ first element of $q_a$
66:                     $terminals \leftarrow$ arcs of $currentarcs$ starting or ending at $current$
67:                     for all $terminalArc$ in $terminals$ do
68:                         $opposite \leftarrow$ opposite end of $terminalArc$
69:                         reverse the direction of $terminalArc$
70:                         if $opposite$ is well connected then $A_u \leftarrow currentarcs$
71:                         else
72:                             if $M_r$does not contain $opposite$ then
73:                                 $M_r \leftarrow$ put($opposite$, its arcs reversed)
74:                                 $q \leftarrow$ add($opposite$)
75:                                 $q_a \leftarrow$ add($currentarcs$)
76:                             end if
77:                         end if
78:                     end for
79:                 end while
80:             end if
81:         end for
82:     end while
83:     $C_p \leftarrow$ all strongly connected cycles of $G_g$          ▷ Tarjan's Algorithm[41]
84:     while $C_p$ is not empty do
85:         for all $cycle$ in $C_p$ do
86:             if size of $cycle$ == number of nodes in $A_u$ then
87:                 break
88:             end if
89:             if $cycle$ is a sink or a source then        ▷ no incoming or outgoing arcs
90:                 $q \leftarrow$ add all positions of $cycle$     ▷ $q_a \leftarrow$ add($A_u$) for each position
91:                 same as steps 61-72
92:             end if
93:         end for
94:     end while
95: end while
```

4.3.3 Algorithm and Operators

NSGA-II [7] is well established as the most widely used Evolutionary Algorithm (EA) for Multi-Objective Optimization (MOO) problems and also in [36], better operators for the Facility Layout Problem were introduced while applying the NSGA-II with binary tournament selection, so NSGA-II with binary tournament selection was chosen to be used in every approach. The different operators used and approaches experimented with are explained below.

Random Mutation

All the values of encoding are mutated randomly with a probability, p_m, where the values are randomly set to an integer between their lower and upper bounds. The pseudocode for the same is in Algorithm 2 below. It is used in initializing the population for the algorithms, as mentioned earlier, and also to mutate the solutions formed after coevolution.

Algorithm 2 Random Mutation Operator

```
1: $L$ = Integrated encoding to be mutated $\leftarrow L_1 = L_{f1} + L_{g1}$
2: $B_l$ = vector of lower boundaries for L
3: $B_u$ = vector of upper boundaries for L
4: $p_m$ = probability to randomize a value of encoding
5:
6: for $i = 0$ to size of $(L - 1)$ do
7:     if random real number between 0 and 1 $< p_m$ then
8:         $L(i) = L_g(i - L_g)$ = random integer between $B_l(i)$ and $B_u(i)$
9:     end if
10: end for
```

Custom Mutation: GuidePath Layout

The guidepath encoding values are only mutated randomly with a probability, p_g. The values are randomly set to 0 or 1 as the guide path encoding consists of only 0s and 1s. The actual probability of mutation will be $< p_g$ as there is a chance that the value is set to the same integer as it was. It is used in the branch of coevolution where the guidepath layout is optimized and the pseudocode is in Algorithm 3.

Mutation: Facility Layout

Introduced in [36] to eliminate bias, this adapted mutation operator which mutates only the facility layout encoding has three probabilities p_w to randomize workstation allocation, p_r to swap positions of two workstations and p_t to change the number of transporters. This operator is used in the branch of coevolution where the facility layout is optimized. The pseudocode is in Algorithm 4 below.

Algorithm 3 Custom Mutation Operator for GuidePath Layout

```
1: $L$ = Integrated encoding of facility and guidepath layouts $\leftarrow L_1 = L_{f1} + L_{g1}$
2: $L_f$ = encoding of facility layout
3: $L_g$ = encoding of guide path layout to be mutated
4: $p_g$ = probability to randomize the direction of an arc of guide path
5:
6: for $i =$ Size of $(L_f)$ to size of $(L - 1)$ do
7:     if random real number between 0 and 1 $< p_g$ then
8:         $L(i) = L_g(i - L_f)$ = random integer between 0 and 1
9:     end if
10: end for
```

Algorithm 4 Custom Mutation Operator for Facility Layout

```
1: $L$ = Integrated encoding of facility and guidepath layouts $\leftarrow L_1 = L_{f1} + L_{g1}$
2: $L_f$ = encoding of facility layout to be mutated
3: $B_u$ = vector of upper boundaries for $L_f$
4: $p_w$ = probability to randomize a workstation allocation
5: $p_r$ = probability to re-position a workstation
6: $p_t$ = probability to change the number of transporters
7:
8: for $i = 0$ to size of $L_f$ - 2 do
9:     if random real number between 0 and 1 $< p_c$ then
10:         if random real number between 0 and 1 $< 0.5$ then
11:             $L_f$(i) = 0
12:         else
13:             $L_f(i)$ = random integer between 0 and $B_u$(i)
14:         end if
15:     end if
16:     if random real number between 0 and 1 $< p_r$ then
17:         $j$ = random integer between 0 and size of ($L_f$ - 2)
18:         swap values of $L_f(i)$ and $L_f(j)$
19:     end if
20: end for
21:
22: if random real number between 0 and 1 $< p_t$ then
23:     if random real number between 0 and 1 $< 0.5$ then
24:         $L_f$(size of ($L_f$ - 1)) += 1
25:     else
26:         $L_f$(size of ($L_f$ - 1)) -= 1
27:     end if
28: end if
```

Mutation: Integrated Facility and GuidePath Layouts

For mutating the values of the combined encoding of facility and guidepath layouts, a combination of Algorithm 4 and Algorithm 3 is used which is defined in Algorithm 5.

Algorithm 5 Custom Mutation Operator for Integrated Facility and GuidePath Layouts

```
1: $L_g$ = Integrated encoding of facility and guidepath layouts $\leftarrow L_1 = L_{f1} + L_{g1}$
2: $L_f$ = encoding of facility layout to be mutated
3: $L_g$ = encoding of guide path layout to be mutated
4: $B_u$ = vector of upper boundaries for $L_f$
5: $p_w$ = probability to randomize a workstation allocation
6: $p_r$ = probability to re-position a workstation
7: $p_t$ = probability to change the number of transporters
8: $p_g$ = probability to randomize the direction of an arc of guide path
9:
10: for $i = 0$ to size of $L_f$ - 2 do
11:     if random real number between 0 and 1 < $p_c$ then
12:         if random real number between 0 and 1 < 0.5 then
13:             $L_f$(i) = 0
14:         else
15:             $L_f(i)$ = random integer between 0 and $B_u$(i)
16:         end if
17:     end if
18:     if random real number between 0 and 1 < $p_r$ then
19:         $j$ = random integer between 0 and size of ($L_f$ - 2)
20:         swap values of $L_f(i)$ and $L_f(j)$
21:     end if
22: end for
23:
24: if random real number between 0 and 1 < $p_t$ then
25:     if random real number between 0 and 1 < 0.5 then
26:         $L_f$(size of ($L_f$ - 1)) += 1
27:     else
28:         $L_f$(size of ($L_f$ - 1)) -= 1
29:     end if
30: end if
31:
32: for $i = L_f$ to size of $(L - 1)$ do
33:     if random real number between 0 and 1 < $p_g$ then
34:         $L(i) = L_g(i - L_f)$ = random integer between 0 and 1
35:     end if
36: end for
```

Crossover: Facility Layout

A single-point crossover operator with crossover probability, p_c for the facility layout optimization branch of coevolution where the guidepath encoding remains unchanged and only the facility layout encoding is used for crossover. The pseudocode is in Algorithm 6.

Algorithm 6 Single Point Crossover for Facility Layout

```
1: $P_1$ = first parent solution $\leftarrow L_1 = L_{f1} + L_{g1}$
2: $p_2$ = second parent solution $\leftarrow L_2 = L_{f2} + L_{g2}$
3: $o_1$ = first offspring solution
4: $o_2$ = second offspring solution
5: $p_c$ = crossover probability
6:
7: $o_1 \leftarrow P_1.copy()$
8: $o_2 \leftarrow P_2.copy()$
9: **if** random real number between 0 and 1 $< p_c$ **then**
10:     $cutoffPoint$ = random integer between 0 and size of ($L_f$ - 1)
11:     **for** $i = 0$ to $cutoffPoint$ **do**
12:         $o_1$(i) $\leftarrow P_2$(i)
13:         $o_2$(i) $\leftarrow P_1$(i)
14:     **end for**
15: **end if**
16: return $o_1$, $o_2$
```

Crossover: GuidePath Layout

A single-point crossover operator with crossover probability, p_c for the guidepath layout optimization branch of coevolution where the facility encoding remains unchanged and only the guidepath layout encoding is used for crossover. The pseudocode is in Algorithm 7.

Algorithm 7 Single Point Crossover for GuidePath Layout

```
1: $P_1$ = first parent solution $\leftarrow L_1 = L_{f1} + L_{g1}$
2: $P_2$ = second parent solution $\leftarrow L_2 = L_{f2} + L_{g2}$
3: $o_1$ = first offspring solution
4: $o_2$ = second offspring solution
5: $p_c$ = crossover probability
6:
7: $o_1 \leftarrow P_1.copy()$
8: $o_2 \leftarrow P_2.copy()$
9: **if** random real number between 0 and 1 $< p_c$ **then**
10:     $cutoffPoint$ = random integer between size of $L_f$ and size of (L - 1)
11:     **for** $i = 0$ to $cutoffPoint$ **do**
12:         $o_1$(i) $\leftarrow P_2$(i)
13:         $o_2$(i) $\leftarrow P_1$(i)
14:     **end for**
15: **end if**
16: return $o_1$, $o_2$
```

Crossover: Coevolution

A single-point crossover operator for the coevolution of solutions where the facility layout encoding of one parent and the guidepath layout of the other parent is combined to form an offspring and vice versa for the other offspring. The pseudocode is in Algorithm 8.

Algorithm 8 Single Point Crossover for Coevolution

```
1: $P_1$ = first parent solution $\leftarrow L_1 = L_{f1} + L_{g1}$
2: $P_2$ = second parent solution $\leftarrow L_2 = L_{f2} + L_{g2}$
3: $o_1$ = first offspring solution
4: $o_2$ = second offspring solution
5:
6: $o_1 \leftarrow P_1.copy()$
7: $o_2 \leftarrow P_2.copy()$
8: $cutoffPoint$ = size of $L_f$
9: for $i = 0$ to $cutoffPoint$ do
10:     $o_1$(i) $\leftarrow P_2$(i)
11:     $o_2$(i) $\leftarrow P_1$(i)
12: end for
13: return $o_1$, $o_2$
```

Crossover: Integrated Facility and GuidePath Layouts

A single-point crossover operator for the combined encoding of facility and guidepath layouts. The pseudocode is in Algorithm 9.

Algorithm 9 Single Point Crossover for integrated Facility and GuidePath layouts

```
1: $P_1$ = first parent solution $\leftarrow L_1 = L_{f1} + L_{g1}$
2: $P_2$ = second parent solution $\leftarrow L_2 = L_{f2} + L_{g2}$
3: $o_1$ = first offspring solution
4: $o_2$ = second offspring solution
5: $p_c$ = crossover probability
6:
7: $o_1 \leftarrow P_1.copy()$
8: $o_2 \leftarrow P_2.copy()$
9: if random real number between 0 and 1 $< p_c$ then
10:     $cutoffPoint$ = random integer between 0 and size of (L - 1)
11:     for $i = 0$ to $cutoffPoint$ do
12:         $o_1$(i) $\leftarrow P_2$(i)
13:         $o_2$(i) $\leftarrow P_1$(i)
14:     end for
15: end if
16: return $o_1$, $o_2$
```

Co-evolutionary Approach

In the Co-evolutionary approach, two NSGA-II algorithms are used to optimize the facility layout and guidepath layout simultaneously yet separately and exchange information between the two algorithms periodically. Three different variants of coevolution are experimented with. They differ in which solutions are used for the exchange of information during the coevolution step. The three ways solutions are selected for co-evolving, corresponding to the "Coevolve the two populations" step outlined in the Figure 4.4, are:

1. **Random Reproduction**
   - Two solutions are randomly selected, one from each algorithm.
   - Single point crossover Operator described in algorithm 8 is applied to these two solutions and the resulting solutions are added to the offspring solution.
   - This is repeated until the offspring population reaches the required size.
   - The offspring population is joined with the existing populations of the two algorithms separately.
   - The size of this combined population is adjusted to the required population size by removing poor solutions based on the Ranking and Crowding Selection comparator.
   - As a result the two algorithms are initialized with slightly different populations after each coevolution step.
2. **Non-Dominated Reproduction**
   - Two non-dominated solutions are iteratively selected, one from each algorithm.
   - Single point crossover Operator described in algorithm 8 is applied to these two solutions and the resulting solutions are added to the offspring solution.
   - This is repeated and if all the non-dominated solutions are utilized, solutions are randomly selected until the offspring population reaches the required size.
   - The offspring population is joined with the existing populations of the two algorithms separately.
   - The size of this combined population is adjusted to the required population size by removing poor solutions based on the Ranking and Crowding Selection comparator.
   - As a result the two algorithms are initialized with slightly different populations after each coevolution step.
3. **non-Dominated Exchange**
   - There is no crossover in this method.
   - The non-dominated solutions of one algorithm are joined with the final population of the other algorithm and vice versa.

- The sizes of these two combined populations are adjusted to the required population size by removing poor solutions based on the Ranking and Crowding Selection comparator.

- As a result the two algorithms are initialized with slightly different populations after each coevolution step.

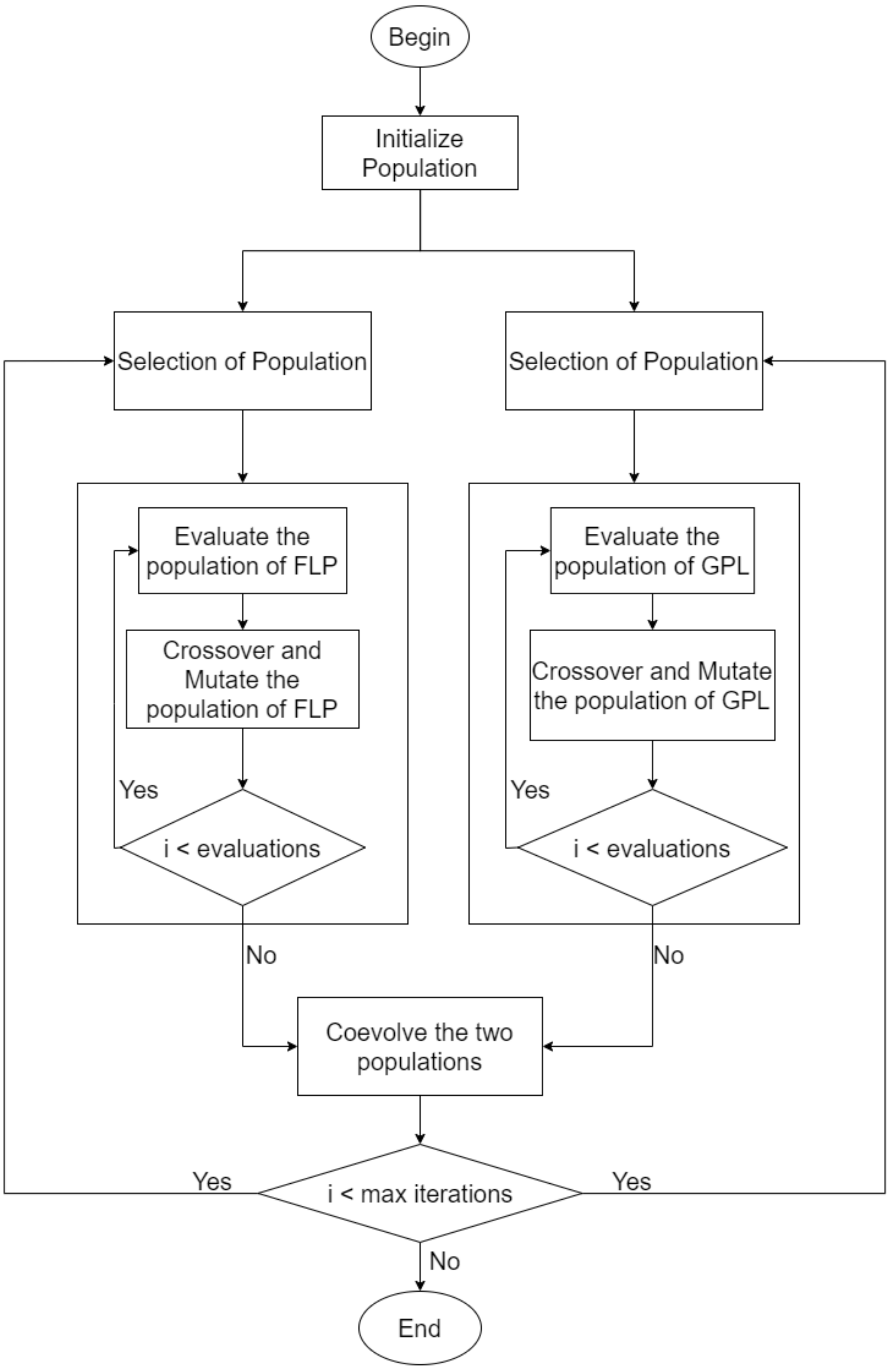

Begin
Initialize Population
Selection of Population
Selection of Population
Evaluate the population of FLP
Crossover and Mutate the population of FLP
Yes
i < evaluations
No
Evaluate the population of GPL
Crossover and Mutate the population of GPL
Yes
i < evaluations
No
Coevolve the two populations
Yes
i < max iterations
Yes
No
End

Figure 4.4: A flowchart depicting the structure of a coevolutionary algorithm used in this work outlining the steps involved in the algorithm's operation

Traditional/Conventional Approach

In the conventional approach, a single NSGA-II algorithm is used to optimize both the facility layout and guidepath layout by solving the integrated problem with combined encoding. The solutions represented by concatenated encodings are obtained by using special operators, single point crossover and custom mutation defined in algorithm 9 and algorithm 5 respectively.

4.3.4 Objective Functions

Three objective functions as in [36] are used for the evaluation of the layouts, where any initial layout already exists. The same objectives are used for both the algorithms in coevolution and for the conventional optimization. The objectives are:

1. **Re-layouting Cost:**
    - The re-layouting cost is calculated, considering equal costs for all transporters and certain costs for new workstations.
    - The total rectilinear distance determines the cost of moving a workstation travelled during the relocation process.
    - The exact calculation method is illustrated in Algorithm 10
2. **Flow Time:**
    - Flow time measures the average number of time steps required for the fulfilment of an order.
    - It provides insight into the efficiency of the proposed layouts in facilitating order processing.
3. **Idle Time:**
    - Idle time is determined as the average ratio of idle time steps without task assignments to those with task assignments.
    - Both transporters and workstations are considered.
    - Idle time reflects periods of inactivity or underutilization in the overall system.

Algorithm 10 Finding Re-Layouting Cost as defined in [36]

```
1: $L_{of}$ = Original layout of the scenario
2: $L_o$ = Layout for which the cost is to be computed
3: $X_o$ = An empty list of integers
4: $X$ = An empty list of integers
5: $c$ = Computed re-layouting cost = 0(initially)
6:
7: repeat
8:     $d = \infty$
9:     $i = -1$
10:    $i_o = -1$
11:    for $j = 0$ to size of $L - 2$ do
12:        if $j \notin X$ then
13:            for $k = 0$ to size of $L_{of} - 2$ do
14:                if $k \notin X_o$ then
15:                    $w$ = workstation represented by $(L_f(j))$
16:                    $w_o$ = workstation represented by $(L_{of}(k))$
17:                    $d_c$ = rectilinear distance from $w$ to $w_o$
18:                    if $d_c < d$ then
19:                        $d = d_c$
20:                        $i = j$
21:                        $i_o = k$
22:                    end if
23:                end if
24:            end for
25:        end if
26:    end for
27:    if $i > 0$ then
28:        $c+ = d$
29:        add $i$ to $X$
30:        add $i_o$ to $X_o$
31:    end if
32: until size of $X$ didn't increase
33:
34: for all $v \in L$ such that $v \notin X$ do
35:     $c+ =$ cost of workstation represented by $v$
36: end for
37:
38: $t_o \leftarrow L_{of}$(size of $L_{of} - 1$)
39: $t \leftarrow L_f$(size of $L_f - 1$)
40: if $t > to$ then
41:     $c \leftarrow c + (t - t_o) \times$ cost of transporters
42: end if
43: return $c$
```

4.4 Performance Indicators

The performance of the above-mentioned different optimization approaches is assessed using three performance indicators: Hypervolume, Inverted Generational Distance (IGD) and C- Metric [2]. For calculating these, all evaluated solutions are archived without interfering with the optimization process.

- **Hypervolume** is described as the size of the dominated space [53]. Higher hypervolume indicates a better distribution of solutions across the objective space and closeness of solution to optima as well.
- **IGD** indicates how closely solutions approach the Pareto front. Lower IGD indicates better convergence, ie., solutions are closer to optimal.
- **C-metric** or set coverage considers the domination of the solutions in the Pareto front obtained from one algorithm over the other and vice versa [15] [49].

5 Experimental Setup

The experimental setup explores a simplified manufacturing process within the framework of mass customization with a focus on a modular smartphone manufacturing scenario with a minimal set of modular components, the same as the case studied in [36] with the implementation of guidepath layout to avoid collisions among AGVs in addition. Positions available for the placement of workstations,$P_n = 20$, which are arranged in a grid format with dimensions of 5 units in width and 4 units in height (5×4), incorporating a spacing of 2 units between each machine in both the horizontal and vertical directions. The overall dimensions of the manufacturing facility, when considering the grid layout and spacing, $s_x = 17$ units in width (5 machines with 6 gaps of 2 units) and $s_y = 14$ units in height (4 machines with 5 gaps of 2 units). With these 20 positions as nodes, 31 arcs are needed (from equation 4.3) for designing the guide path. Therefore, the encoding consists of 52 integers, with the first 20 maps to the positions on the shop floor, the last 31 maps to the direction of the arcs, and the 1 in between maps to the number of transporters (equations 4.1, 4.2 and 4.4).

The mathematical relation between the encoding and the positions on the shop floor is defined below:

$$\begin{aligned}
\text{let,}\quad & i \text{ be the index of the integer in the encoding of the facility layout (0 to 19)} \\
& \% \text{ be the modulo operator which gives remainder} \\
& \text{x coordinate of the position } = 3 + (i\%5) \times 3 \\
& \text{y coordinate of the position } = 3 + (i/5) \times 3
\end{aligned}$$

The arcs are organized in a specific order for representation in the encoding. The arcs are encoded in two phases:16 horizontal arcs and 15 vertical arcs

1. **Horizontal Arcs:**
    - The arcs traverse rows from left to right
    - Start at the bottom left corner and move across each row to the bottom right
    - After reaching the bottom right move to the next row on the top
2. **Vertical Arcs:**
    - The arcs traverse columns from bottom to top
    - Start at the bottom left corner and move up each column to the top left
    - After reaching the top left move to the next column on the right

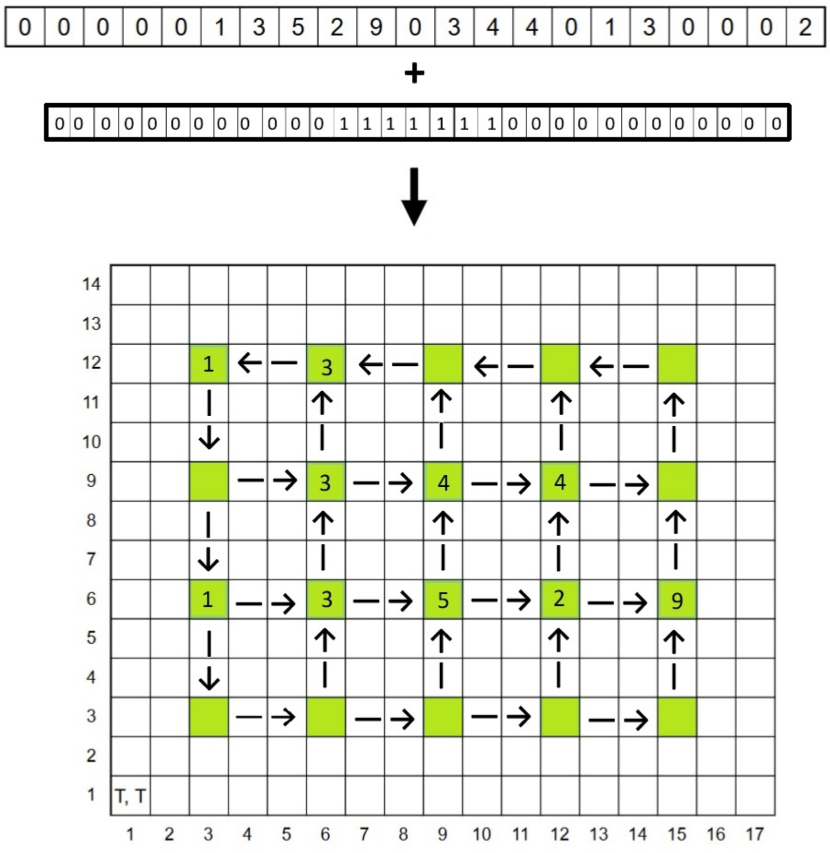


Figure 5.1: How encoding translates to facility and guidepath layouts

Figure 5.1 illustrates how the information is translated from the encoding to the facility and guidepath layouts. Positions available for the placement of workstations are highlighted in green and the directions of arcs of the guidepath are represented by arrows. The transporters always start at the first cell, (1,1). This will have a minor overall impact as a maximum of 25 steps could be wasted in the worst-case scenario which is not much as each solution generally requires at least 10,000 time steps and the impact can be ignored[36]. The layout presented in this image is the one considered as the initial layout that needs to be optimized.

The scenario used necessitates the production of 34 distinct product variants. The list of all the workstations and their corresponding value in the encoding is tabulated in the table 5.1. Six different types of skills/tasks are performed by 12 different workstations. It could be noticed that each skill/task can be performed by two workstations with varying processing time and setup costs. All the transporters are equivalent in their functionality and have a cost of 50.

Table 5.2 outlines the necessary details of the product family. All the products that could be manufactured in this smartphone manufacturing setup and the skills and inputs required for manufacturing these products are listed. It was referred to as a recipe in [36]. In this hypothetical scenario, smartphones are assembled from various parts and subsequently shipped. The initial raw products stored in "Storage Access" are "Mainboard 1", "Mainboard 2", "Mainboard 3", "2 GiB RAM", "4 GiB RAM", "Dual Core CPU", "Quad Core CPU", "Plastic Ingot" and "Metal Ingot". These products can be combined

Encoding	Workstation/Skill	Processing Time	Cost
1	Storage Access	60	40
2	Soldering	60	40
3	Injection Molding	60	40
4	Die Casting	60	40
5	Bolting	60	40
6	Shipping	60	40
7	Storage Access	30	80
8	Soldering	30	80
9	Injection Molding	30	80
10	Die Casting	30	80
11	Bolting	30	80
12	Shipping	30	80

Table 5.1: Workstations in the setup and their details

Product	Required Input	Required Skill
Mainboard 1	-	Storage Access
Mainboard 2	-	Storage Access
Mainboard 3	-	Storage Access
2 GiB RAM	-	Storage Access
4 GiB RAM	-	Storage Access
Dual Core CPU	-	Storage Access
Quad Core CPU	-	Storage Access
Plastic Ingot	-	Storage Access
Metal Ingot	-	Storage Access
Main Module	MB, RAM, CPU	Soldering
Case(Plastic)	Plastic Ingot	Injection Molding
Case(Metal)	Metal Ingot	Die Casting
Smartphone	Main Module, Case	Bolting

Table 5.2: Products and their requirements

to assemble a smartphone. The main module is created at the "Soldering" workstation by combining the necessary mainboard, RAM and CPU. Cases made of either plastic or metal, are produced respectively at "Injection Molding" and "Die Casting" workstations. One of these cases is combined with the main module at the "Bolting" workstation to make a "Smartphone". Finally, the completed smartphone is shipped from the "Shipping" Workstation to successfully fulfil an order.

At any given time simulation can process up to 10 orders simultaneously. As soon as an existing order is completed, a new order is generated randomly. For evaluation, simulation is run until 300 randomized orders for these 34 smartphone variants are fulfilled for each solution and the objective function values are archived. It is advised to refer back to the original work in [37] and [36] for more details and a better grasp of product kinds, workstations and recipes. In the coevolutionary framework, 25 iterations of coevolution are performed. Within each iteration, there are two branches each initialized with a population size of 100. The evolution within each branch proceeds for 50 generations. Consequently, the total number of evaluations performed in this process is 250,000. The conventional approach is also initialized with a population size of 100. The evolution proceeds for 2,500 generations to make the total evaluations in this process 250,000, for a fair comparison.

The probabilities for mutation and crossover operators are as follows:

- Randomization probability for workstations, $p_w = 0.1$
- Repositioning probability for workstations, $p_r = 0.2$
- Transporter changing probability, $p_t = 0.3$
- Randomization probability for guipdepath, $p_g = 0.6$
- Crossover probability, $p_c = 0.9$

In the conventional approach, the randomization probabilities for workstations and guide-path are both considered to be 0.1 to illustrate the effect of randomizing on guide-path.

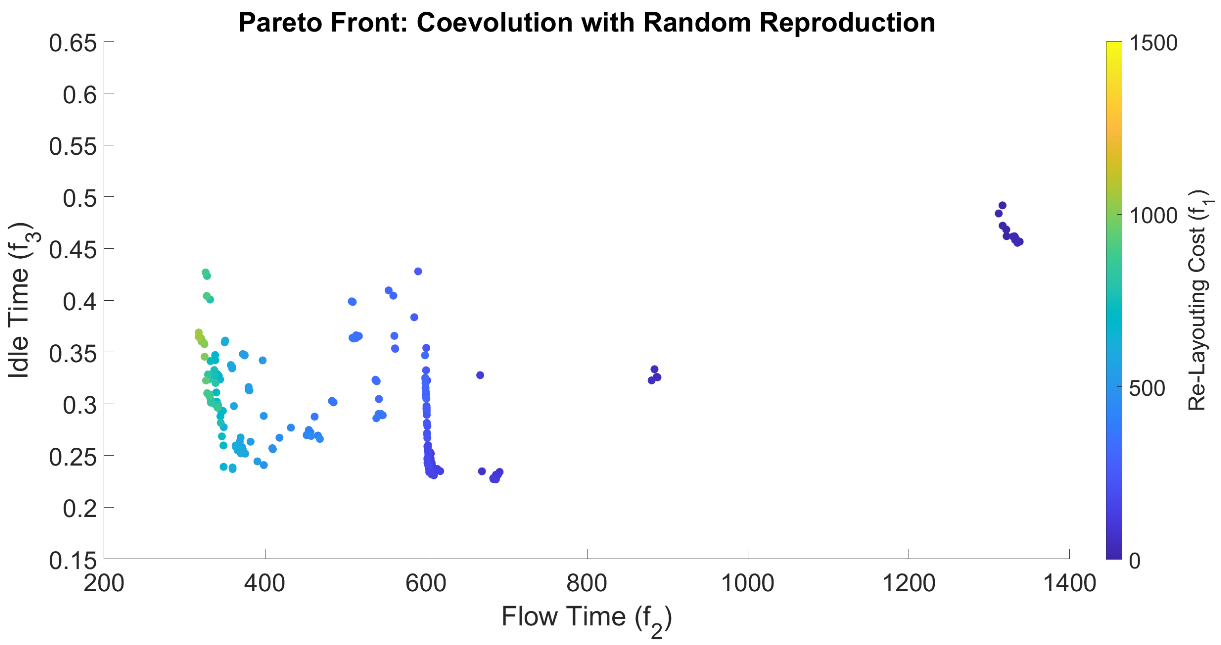
6 Results and Discussion

In this chapter, we evaluate the performance of the different scenarios outlined in the methodology, focusing on their effectiveness within the setup discussed in Chapter 5. Significantly, a comparative analysis between the conventional approach and three distinct variants of the coevolutionary approach outlined in Chapter 4 is conducted. The performance analysis of the different coevolutionary approaches is presented first and the better performance of these is compared with the conventional approach.

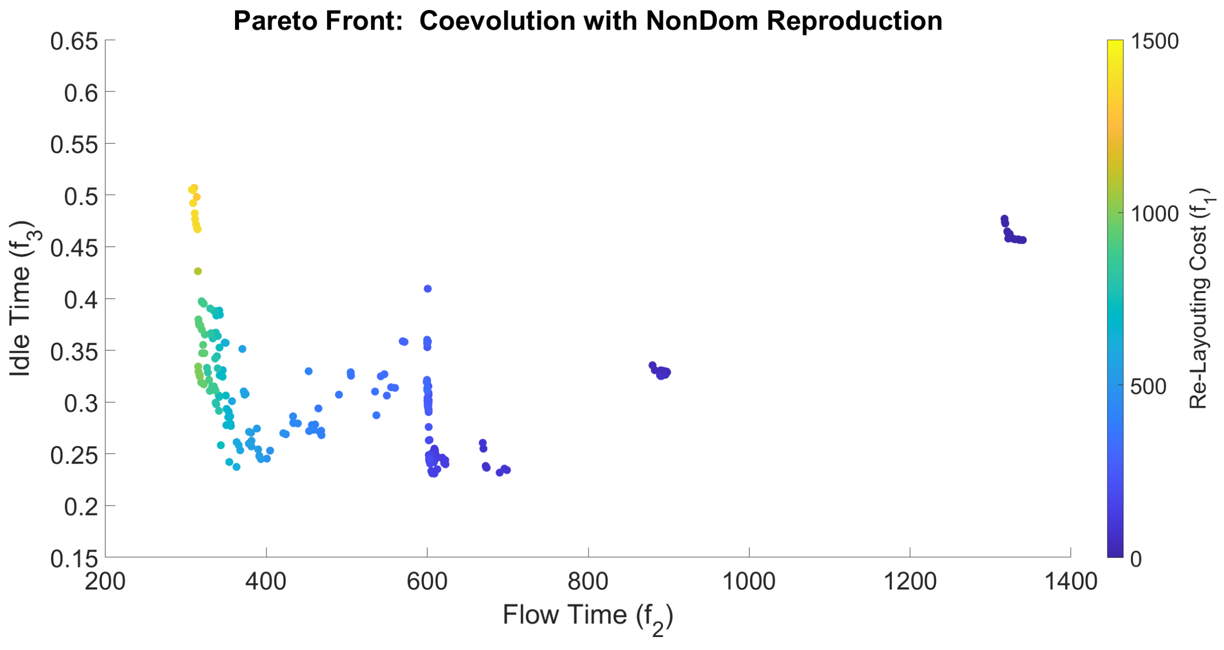
6.1 Performance Analysis: Coevolutionary Approaches

Pareto fronts: non-dominated solutions

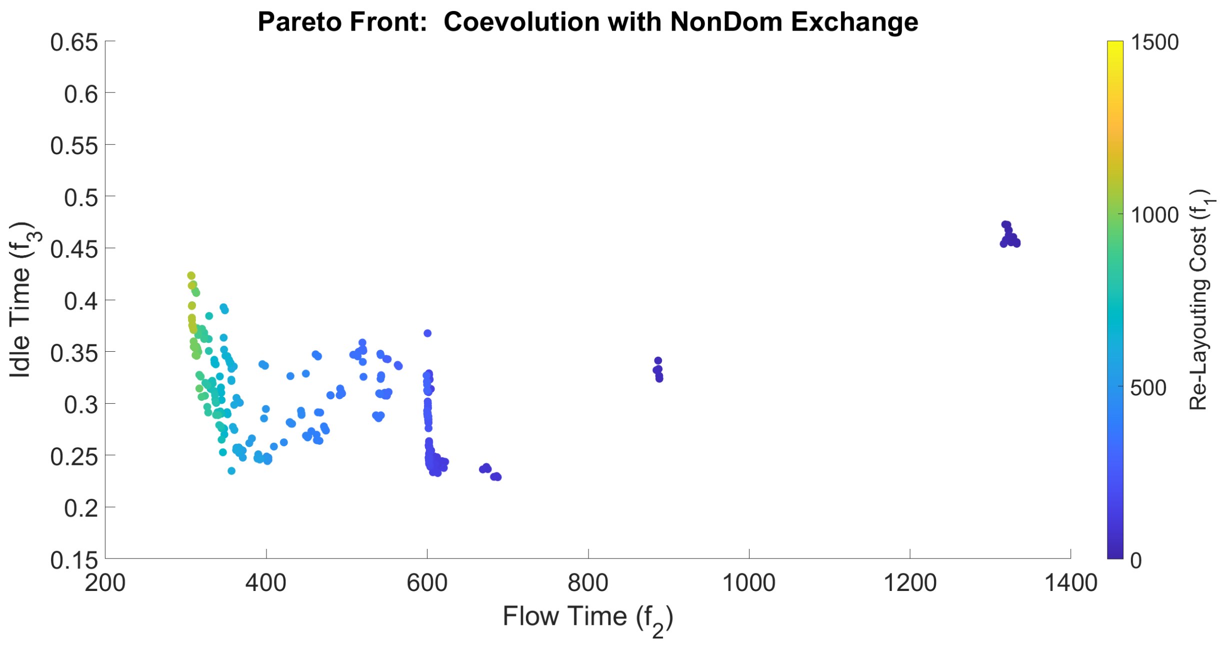
The Pareto fronts of the three coevolutionary approaches are illustrated in figures 6.1a, 6.1b, and 6.1c.



(a) Coevolution with random reproduction



(b) Coevolution with nonDom reproduction



(c) Coevolution with nonDom exchange

Figure 6.1: Pareto Fronts of different coevolutionary approaches

The diversity of solutions remains comparable across all three methods. Moreover, it's observed that across all approaches, there exist non-dominated solutions with notably high flow time and idle time, yet comparatively lower relayouting costs. This signifies an improvement in flow time and idle time at the expense of increased relayouting costs. Table 1 lists the objective values for the optimal solutions with one of the objectives being the main objective, i.e., the best value for that objective in that approach. The trade-off lists the objective values of the solution whose objective values are closest to the medians of those for that run. The solution with 0 re-layouting costs is the initial layout of the facility that is to be optimized. For these solutions, the ideal time and flow time are rather high which indicates that there is a necessity to optimize the existing layout. The optimization required is not only for the placement of the workstations but also for the addition of new workstations and transporters. The trade-off values indicate compromise solutions that balance all the objectives. Optimal solutions considering all the solutions of the three approaches are also listed.

Main Objective	Re-Layouting Cost	Flow Time	Idle Time
Coevolution: random reproduction			
Re-Layouting Cost	0	1335	0.45677
Flow Time	1050	317.48	0.36887
Idle Time	113	686.84	0.22709
Trade-Off	320	541.41	0.30463
Coevolution: nonDom reproduction			
Re-Layouting Cost	0	1325.2	0.45908
Flow Time	1374	307.48	0.50509
Idle Time	144	608.8	0.23086
Trade-Off	366	490.18	0.30712
Coevolution: nonDom exchange			
Re-Layouting Cost	0	1320.8	0.47235
Flow Time	1088	306.44	0.42352
Idle Time	89	687.55	0.22872
Trade-Off	375	479.91	0.3076
Coevolution: Overall			
Re-Layouting Cost	0	1321.4	0.4619
Flow Time	1088	306.44	0.42352
Idle Time	113	686.84	0.22709
Trade-Off	369	464.65	0.29383

Table 6.1: Optimal Solutions for Different Coevolutionary Approaches

Hypervolue and IGD

In all coevolutionary approaches, 25 iterations of coevolution are performed, with 50 generations for each branch, totalling 2500 generations. The hypervolume of each approach is calculated by considering the non-dominating solutions from all three approaches as a reference. Figure 6.2 displays the hypervolume of each approach for every generation. Specifically, within each coevolutionary iteration spanning 100 generations, the first 50 generations are dedicated to the facility layout optimization branch, followed by the subsequent 50 generations that are focused on guidepath optimization. This sequential pattern repeats throughout the iterations. Notably, during generations 50 to 100, the hypervolume is relatively low. This is attributed to the sole optimization of the guidepath before coevolution starts. The majority of the influence on objectives is from the facility layout and without coevolution, guidepath optimization performs comparatively poorly.

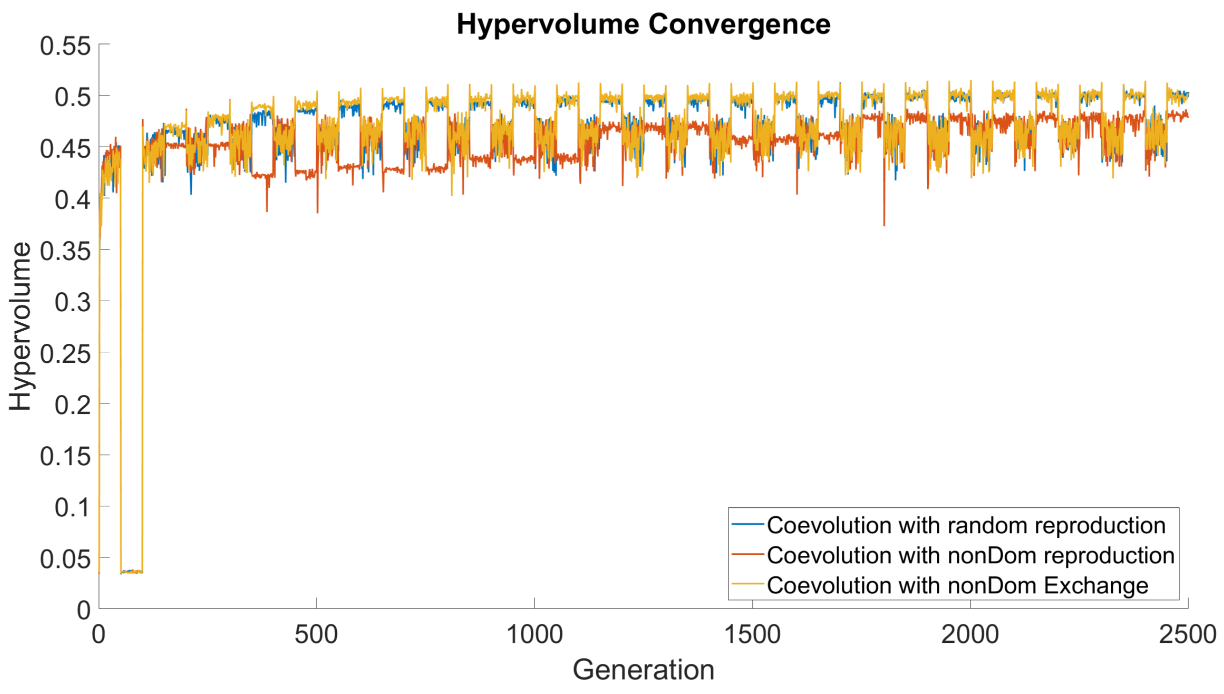


Figure 6.2: Hypervolume Convergence of different coevolutionary approaches

The averages of hypervolume and IGD are listed in the Table 2. Among the approaches, NonDom Exchange has the highest hypervolume (0.46771) and lowest IGD (65.959), suggesting that it achieves better coverage and is closer to the Pareto front compared to the other approaches. Random Reproduction has a similar hypervolume but a higher IGD. Comparatively, NonDom Reproduction performed poorly.

	Hypervolume	**IGD**
Random Reproduction	0.46617	73.294
NonDom Reproduction	0.44865	98.086
NonDom Exchange	0.46771	65.959

Table 6.2: Performance Indicators for different approaches

C-Metric

Table 6.3 presents the C-Metric values for various coevolutionary approaches. The values, expressed as percentages ($\times$ 100), indicate the proportion of non-dominated solutions from the column approach dominated by those from the row approach. Notably, NonDom Exchange demonstrates the highest C-Metric compared to the other approaches.

	Random Rep.	NonDom Rep.	NonDom Exc.
Random Rep.	-	0.4512	0.2274
NonDom Rep.	0.1875	-	0.0870
NonDom Exc.	0.4107	0.5650	-

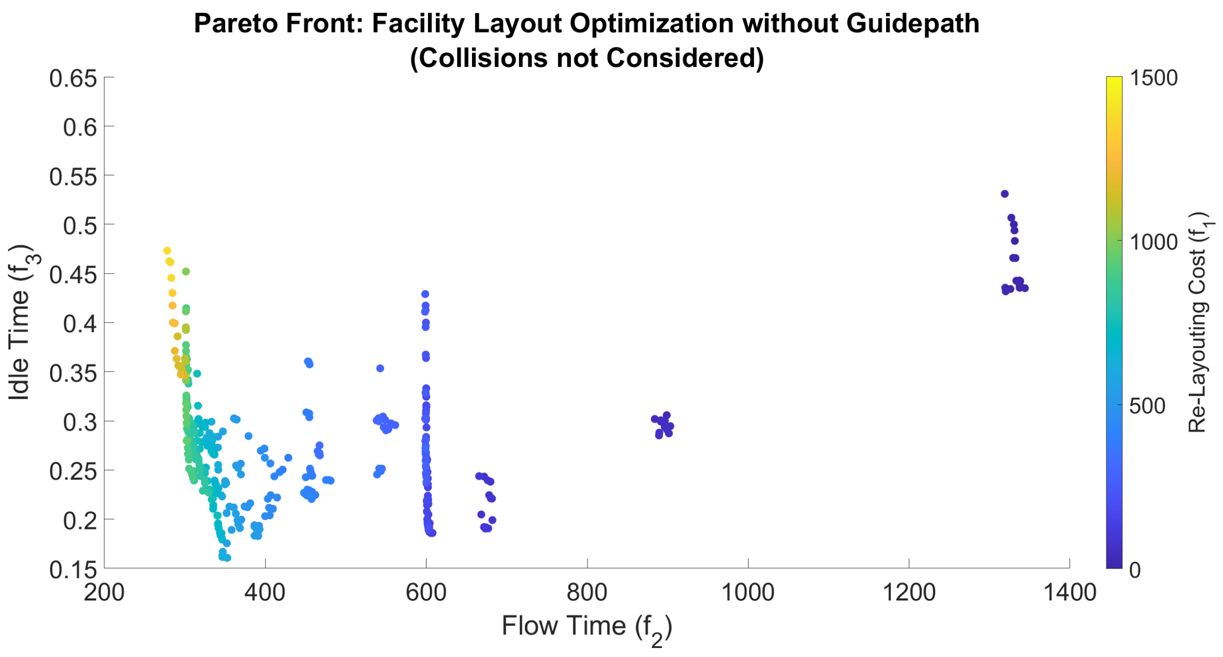
Table 6.3: Set Coverage of different coevolutionary approaches

6.2 Performance Analysis: Coevolutionary vs Conventional

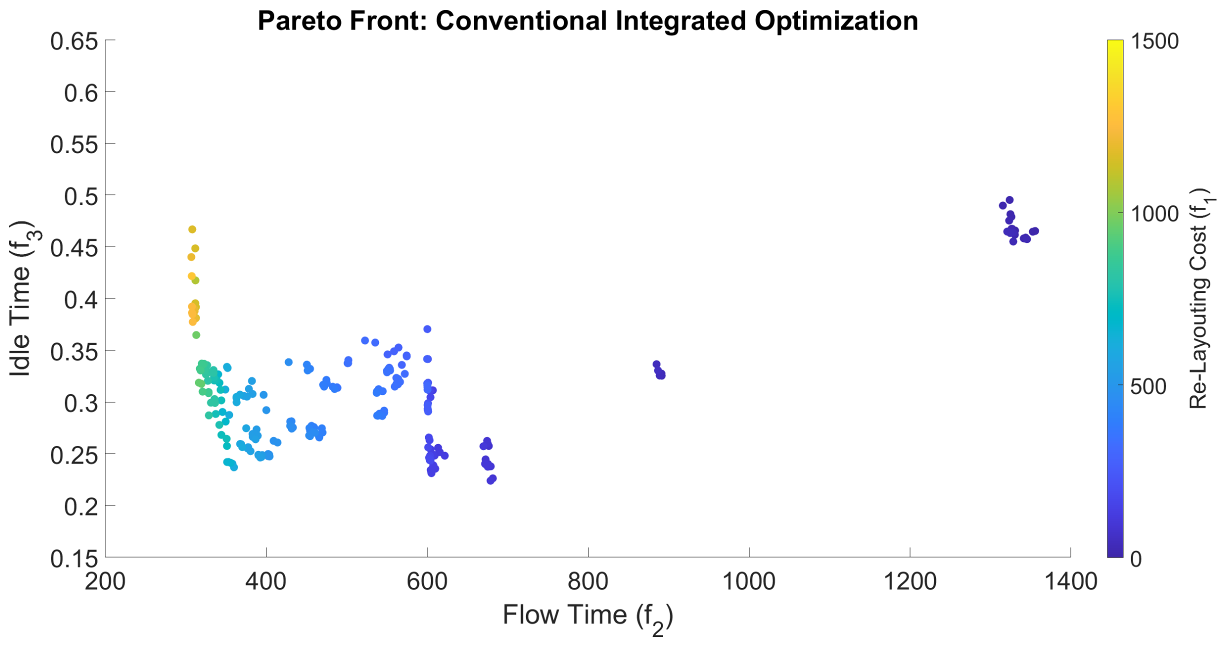
Based on the results presented earlier, the NonDom exchange variant of coevolutionary approaches is considered to be better performing and is compared with the conventional approach in this section. Additionally, the results from a straightforward facility layout optimization without guidepath implementation are included for reference. It is anticipated that this latter approach will outperform the others, as the transporters in this scenario have the freedom to move in any direction without collision considerations.

Pareto fronts: non-dominated solutions

Figures 6.3a and 6.3b show all the non-dominated solutions (Pareto fronts) for the facility layout and conventional optimization approaches respectively. While the number of solutions in the Pareto of facility layout optimization is higher, the diversity remains comparable.



(a) Facility Layout optimization without guidepath



(b) Conventional Optimization

Figure 6.3: Pareto fronts of facility layout optimization without guidepath (collisions not considered) and conventional optimization approaches

Table 6.4 presents the optimal solutions with each objective treated as the main objective and a trade-off objective, as previously described. As expected, the optimal solutions for the facility layout optimization outperform those of other approaches.

Main Objective	Re-Layouting Cost	Flow Time	Idle Time
	Facility Layout optimization without guidepath		
Re-Layouting Cost	0	1330.1	0.49991
Flow Time	1373	277.89	0.47316
Idle Time	613	352.69	0.16065
Trade-Off	458	394.24	0.26969
	Coevolutionary optimization		
Re-Layouting Cost	0	1330.4	0.53068
Flow Time	1097	305.57	0.45269
Idle Time	616	350.3	0.2225
Trade-Off	360	539.5	0.31012

Table 6.4: Optimal Solutions from different approaches

Hypervolue and IGD

Figure 6.4 shows the hypervolume of each approach for each generation. The hypervolume of each approach is calculated by considering the non-dominating solutions from all three approaches as a reference. The facility layout optimization without guidepath and conventional approaches ran for 2500 generations to match the evaluations of the coevolutionary approaches.

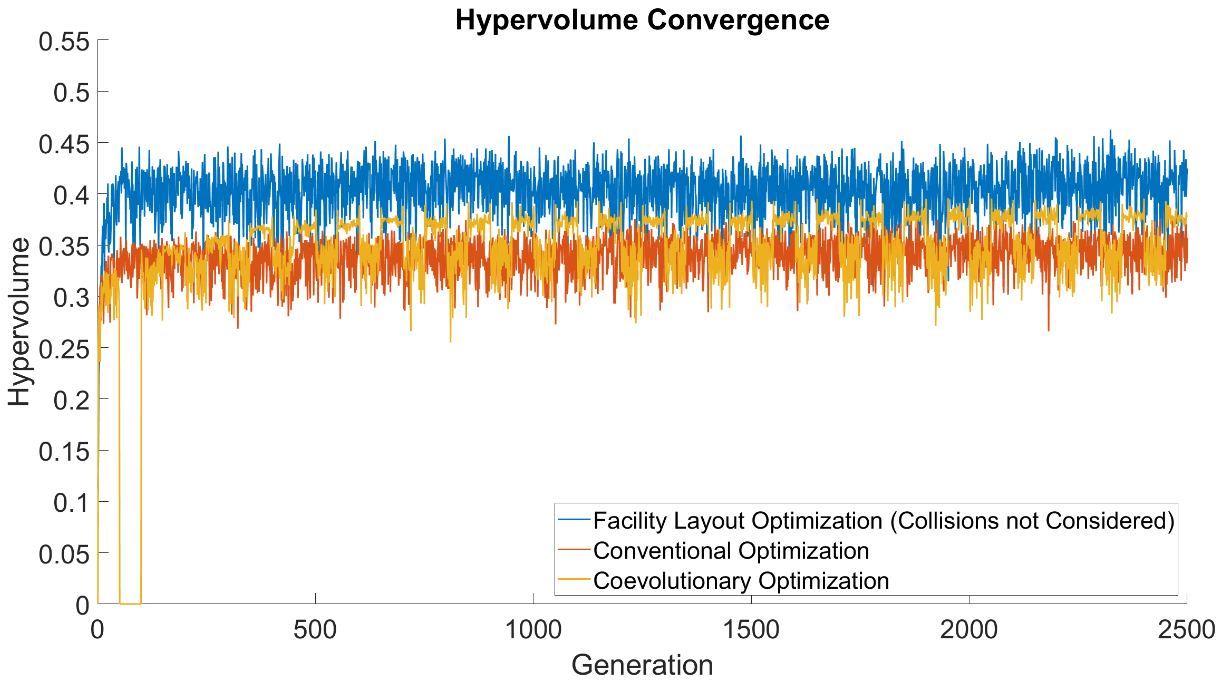


Figure 6.4: Hypervolume convergence of facility layout optimization without guidepath (collisions not considered), coevolutionary optimization with nondom exchange and conventional optimization approaches

The averages of hypervolume and IGD are listed in the Table 6.5. Facility Layout optimization performs better in terms of hypervolume and IGD compared to Coevolutionary and Conventional optimization. While the Coevolutionary approach has a better hypervolume, the conventional approach has better IGD.

	Hypervolume	**IGD**
Facility	0.40324	53.688
Coevolutionary	0.34481	86.527
Conventional	0.33913	67.273

Table 6.5: Performance Indicators of the other approaches

C-Metric

With Pareto, hypervolume and IGD being close C-metric (set coverage) was calculated for a more appropriate evaluation of the coevolutionary and conventional approaches.

$$\text{C-Metric(Coevolutionary, Conventional)} = 0.4407$$
$$\text{C-Metric(Conventional, Coevolutionary)} = 0.2375$$

About 44% of the non-dominated solutions of the conventional approach are being dominated by the non-dominated solutions of the coevolutionary approach and conversely, about 24% in the reverse scenario.

6.3 Discussion

The facility layout optimization without a guidepath approach is better for all the performance indicator values, as anticipated because there are no restrictions on the routes of the transporters. In the absence of a guidepath, transporters follow the shortest possible path. With a guidepath, the overall distance travelled by the transporter will be higher because the length of the feasible path will not always align with the shortest Manhattan path. However, it does not imply that the absence of collisions makes it inherently better. This is because, without guidepaths, collisions are not considered in the optimization process. While the performance indicators may suggest better results, the facility layout optimization without a guidepath overlooks the crucial aspect of collision avoidance. In contrast, guidepath provides a structured and collision-free routing solution, contributing to a safer and more controlled material handling system within the facility. The main goal of this thesis is to optimize the layout of a dynamic manufacturing facility considering collisions, i.e., removing collisions by implementing a guidepath.

NonDom Exchange demonstrated superior performance among the various Coevolutionary approaches, while NonDom Reproduction exhibited comparatively poorer results. The poor performance of the NonDom Reproduction approach may be attributed to several factors. One possible explanation is that the non-dominated solutions generated by each algorithm have already adapted to their respective layouts or guidepaths. As a result, performing crossover operations between these solutions may not lead to significant improvements or good solutions. Consequently, the NonDom Reproduction approach may struggle to produce high-quality offspring solutions compared to other approaches such as Random Reproduction or NonDom Exchange. Despite the similar performance between Random Reproduction and NonDom Exchange in terms of Hypervolume, NonDom Exchange outperformed in terms of IGD and C-metric suggesting that NonDom Exchange generated solutions that are more dominating and closer to the true Pareto front compared to Random Reproduction. A comparison of the hypervolume values for the various approaches outlined in Tables 2 and 6.5 reveals an inconsistency in the hypervolume value of the NonDom Exchange approach. This variation is because of the differences in the reference utilized for hypervolume calculation.

While randomization is necessary to explore different guidepath layouts, The effect of the value of randomization probability for guidepath (p_c) is relatively minimal. From the experiments conducted, it was observed that in each run, approximately 85% of the time, the generated guidepath was not feasible and required repair, while the facility layout needed repair around 33% of the time. This highlights the fragility of the guidepath and underscores the importance of using an effective repair function. Although the current repair function for guidepaths aims to make minimal changes using breadth-first search, there is room for improvement. The modifications made by the repair function to ensure the feasibility of guidepaths also make it restrictive and for the tested layout size, the number of feasible guidepaths is limited. One potential improvement could involve implementing a bidirectional guidepath, which may reduce unnecessary longer paths by allowing transporters to travel in both directions. This could optimize transporter paths, especially in cases where the destination is physically closer but requires a longer path due to the one-directional guidepath constraint.

The findings indicate that achieving comparable results requires considerably fewer evaluations. Approximately 100,000 evaluations would be adequate to yield similar outcomes. Notably, there was not much distinction observed between the two approaches in terms of their Pareto fronts, indicating that they both performed well and exhibited diversity in the solutions generated. Both the coevolutionary and conventional approaches produced scattered non-dominated solutions across the objective space, with similar convergence patterns in the hypervolume and IGD analysis. However, based on the C-metrics, the solutions obtained from the coevolutionary approach outperformed those from the conventional approach. This is evident from the higher percentage of solutions in the Pareto front of the coevolutionary approach dominating those in the Pareto front of the conventional approach, which is nearly double the percentage in the reverse scenario.

7 Conclusion and Future Work

This thesis aims to provide cost-effective and better solutions for a dynamic manufacturing system by optimizing an existing layout considering collisions among transporters. Specifically, the use of unidirectional guidepaths to prevent collisions among Automated Guided Vehicles (AGVs) in Material Handling Systems is explored. Optimal solutions are achieved by addressing the facility layout and guidepath layout problems as an integrated problem. This integration is pivotal, as the arrangement of the facility significantly influences the functionality of the guidepath. Conversely, an effective guidepath enhances the productivity of the layout, thereby improving the overall optimization of the manufacturing system.

The facility layout problem is conceptualized as a mixed integer problem, where each integer denotes a type of workstation to be placed at a specific location mapped to the position of the integer in the encoding. On the other hand, the guidepath layout is formulated as a zero-one integer problem, where zeroes and ones signify the direction of the arc in the guidepath. Similar to the facility layout encoding, the encoding maps the position of the zero or one to the respective arc direction. To ensure the feasibility of both randomly generated facility layouts and guidepath layouts, a custom repair function is employed. This function identifies any missing skills and subsequently adds them to make the layout feasible. The guidepath layout is modelled as a grid graph, where each node represents a location on the grid, and the edges represent possible movements between adjacent locations ie., arcs of the guidepath. To ensure the feasibility, it is essential to enhance the reachability and connectivity of this grid graph. To achieve this with the minimum number of changes made to the guidepath, a breadth-first approach is used.

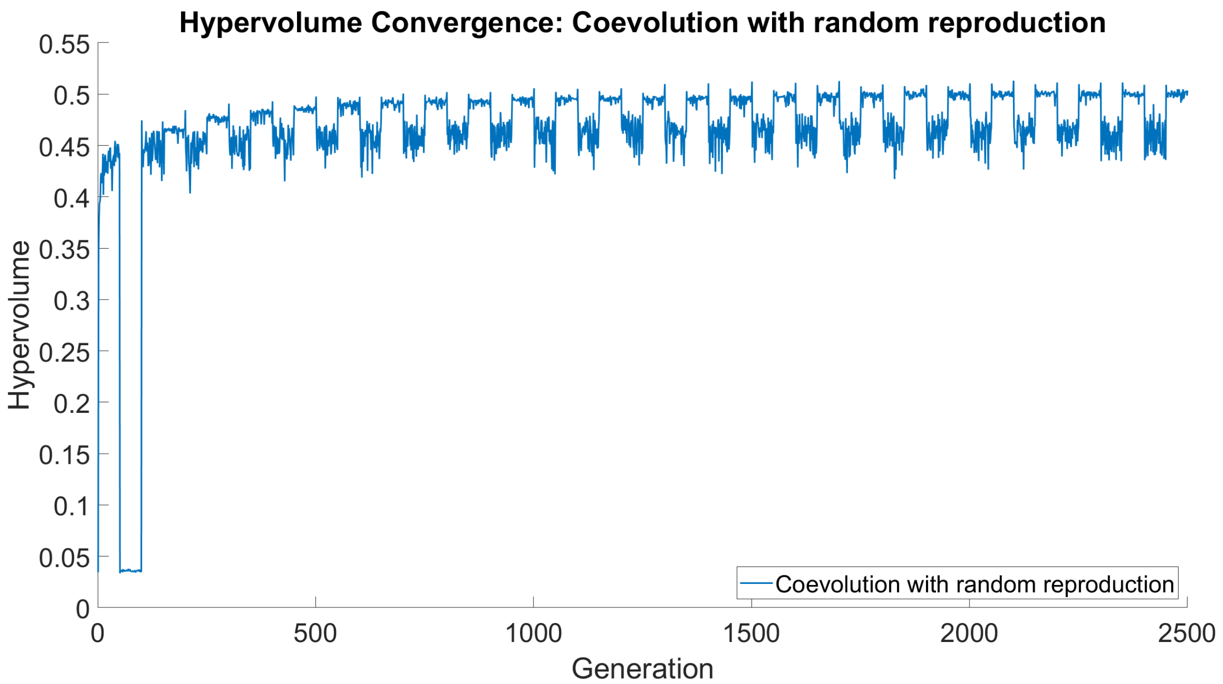
For integrating the problems, two approaches were explored. One was to concatenate the encodings for the two problems into one and optimize it as any conventional problem. The other was to use coevolution to optimize facility layout and guide paths separately with different populations while periodically exchanging information between them. Various methods of information exchange were experimented with, among which directly replacing the non-dominated solutions of one algorithm (facility layout optimization) with the underperforming solutions of the other algorithm (guidepath optimization) and vice versa proved to be effective. Both approaches exhibited similar performance, with the coevolutionary approach slightly outperforming the conventional approach. In conclusion, for setups of comparable size, the utilization of coevolution provides slight benefits. However, it is important to note that the assessment was limited by time constraints, preventing the exploration of larger-scale problems. It is anticipated that for larger setups, the coevolutionary approach could demonstrate superior performance compared to the conventional approach.

Future Work

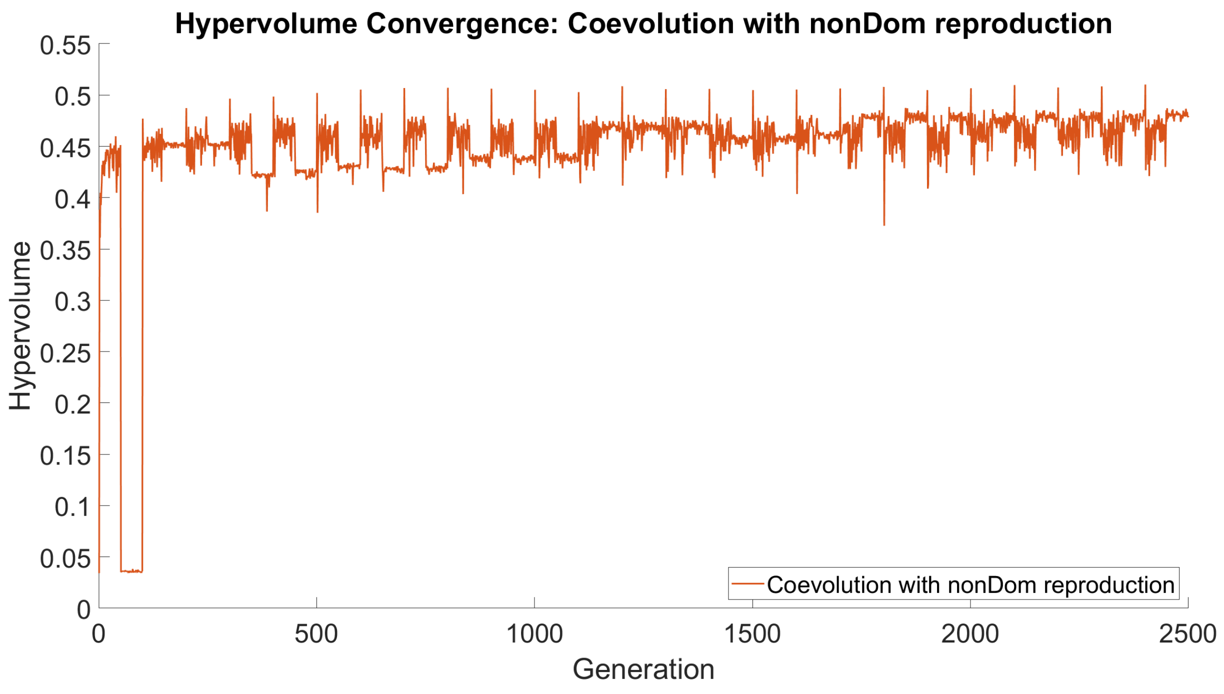
The work presented in this thesis can be extended in many ways. One such is to conduct a thorough scalability analysis by applying the coevolutionary approach to a large-scale problem. Scaling up the size of the facility, the addition of new types of workstations, and increasing the number of variants of products that can be produced can provide valuable insights into the performance of coevolution. Consideration of energy costs is effective for optimizing the number of transporters and is increasingly recognised [45]. Future work could be on the integration of the said energy costs into the optimization process. The facility layout and guidepath layout problems could also be optimized with different objectives rather than having the same as in the case of this thesis. Beyond guidepaths, alternate strategies like using dynamic tandem zones for mitigating collisions among material handling systems could be explored. Investigating the effectiveness of different collision avoidance strategies could provide more options to tailor solutions to specific environments. Scheduling, which was treated as a separate operation could by itself be made an optimization problem or be integrated into the optimization framework. This approach may lead to improved coordination among the entities within the manufacturing systems.

Appendices

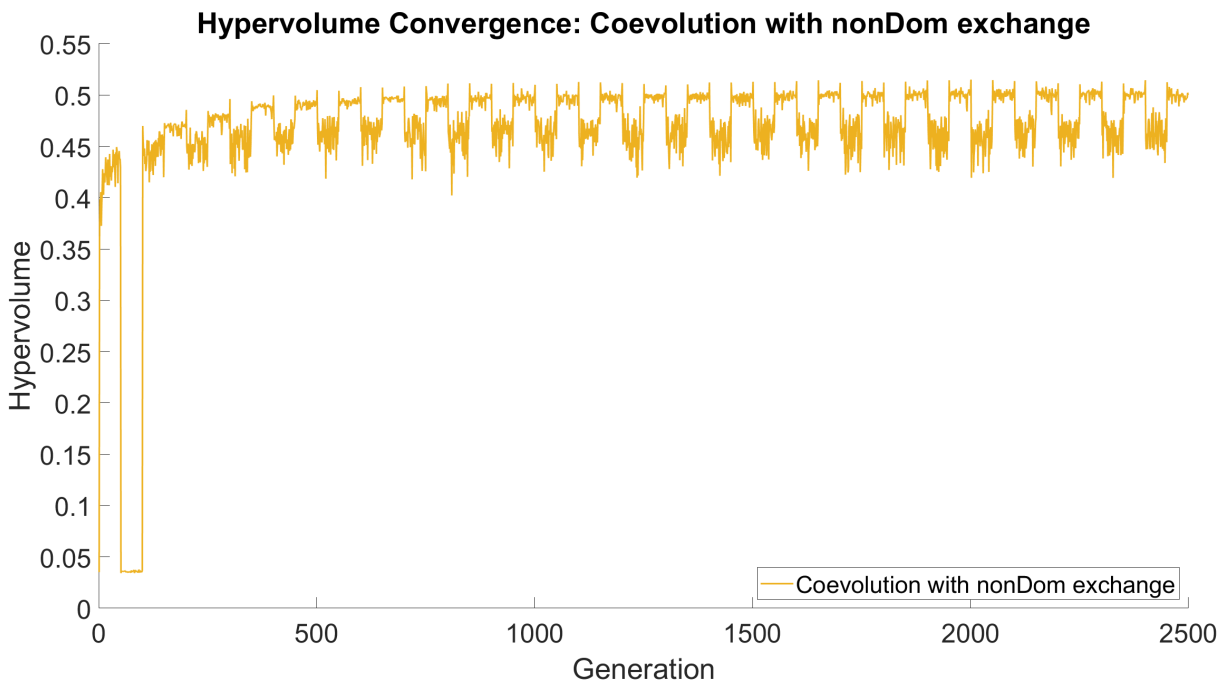
In this appendix, we present individual plots of the hypervolumes for each coevolutionary approach, building upon the combined plot presented in Chapter 6.



Hypervolume convergence of coevolution with random reproduction

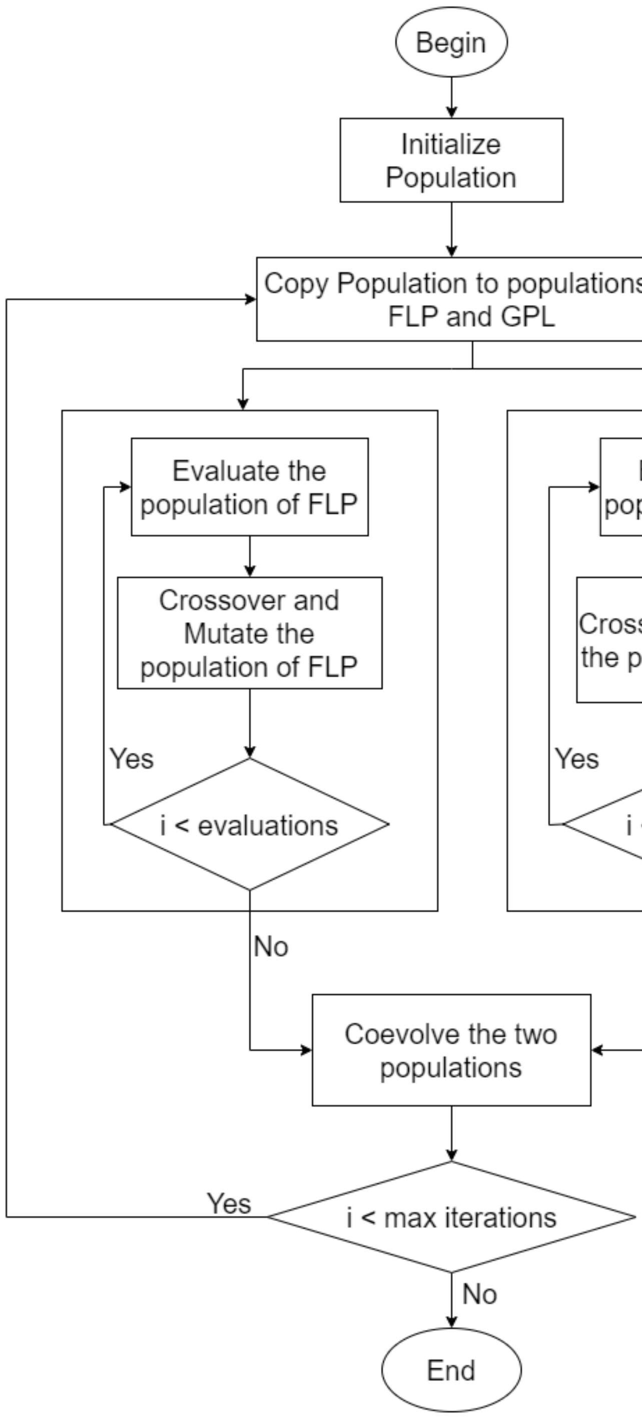


Hypervolume convergence of coevolution with nonDom reproduction

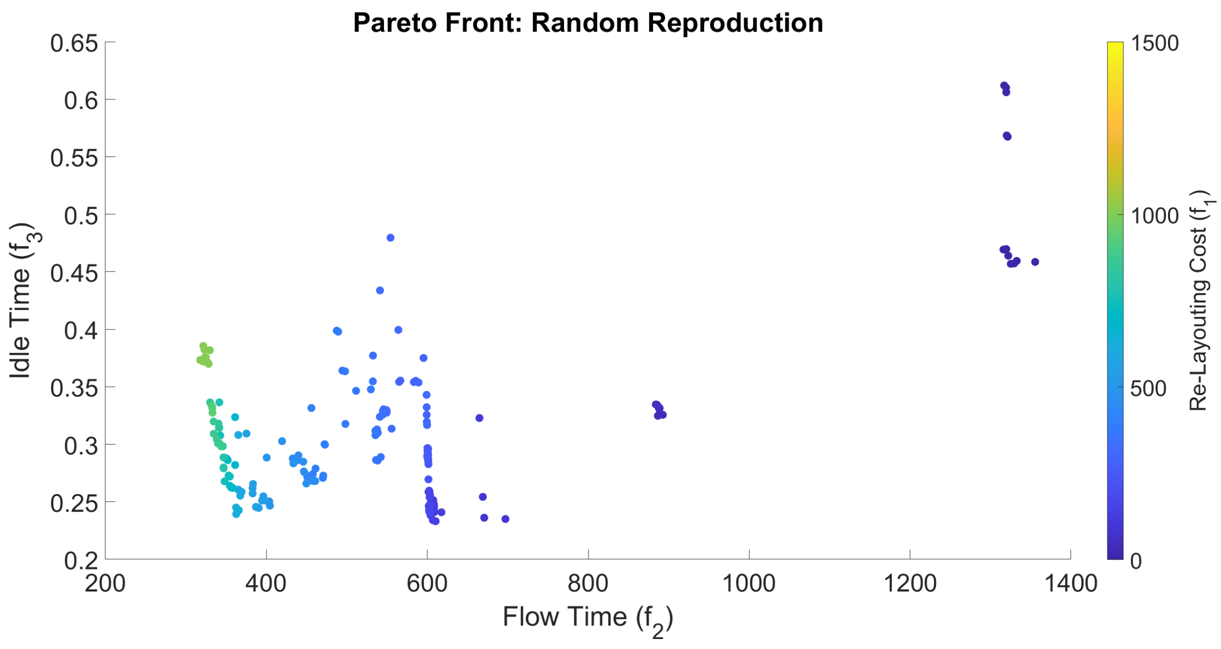


Hypervolume convergence of coevolution with nonDom exchange

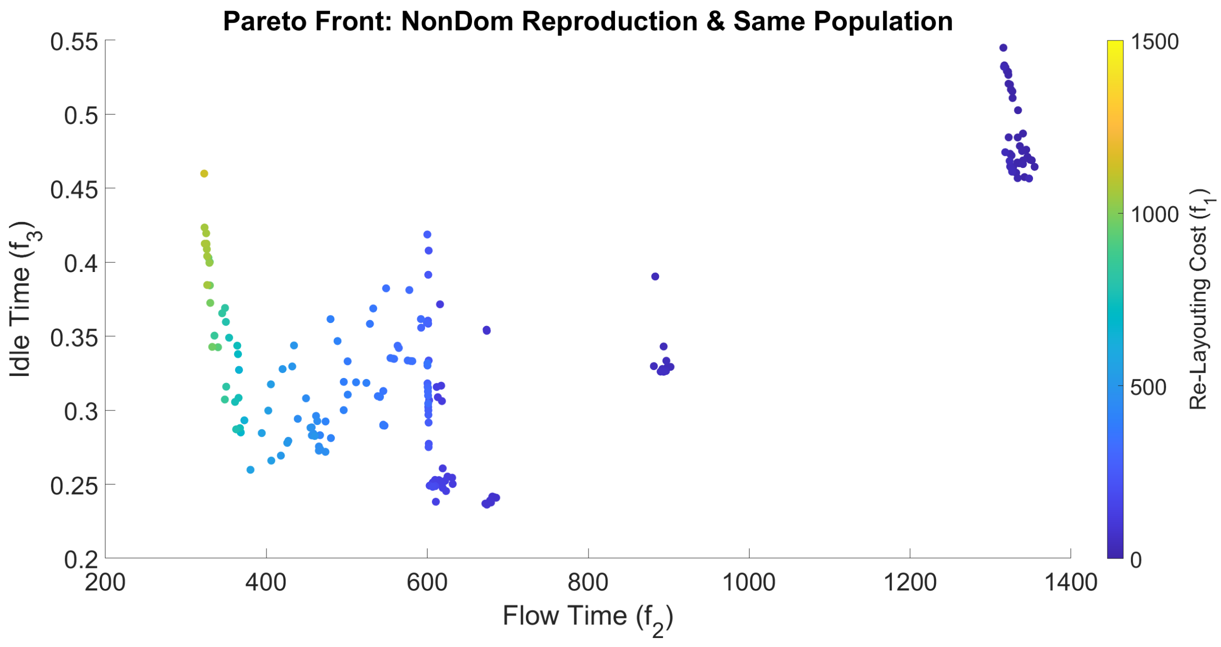
Additionally, we analyze the performance of the coevolution scenarios, random reproduction and nonDom reproduction, when both branches are initialized with the same population after every step of information exchange as depicted. Our findings suggest that coevolutionary approaches with identical population initialization exhibit inferior performance compared to those maintaining distinct populations for each branch of coevolution. So, these approaches were not discussed previously in this thesis.


Begin
Initialize Population
Copy Population to populations of FLP and GPL
Evaluate the population of FLP
Crossover and Mutate the population of FLP
Yes
i < evaluations
Evaluate the population of GPL
Crossover and Mutate the population of GPL
Yes
i < evaluations
No
No
Coevolve the two populations
Yes
i < max iterations
No
End

A flowchart depicting the structure of a coevolutionary algorithm where the branches are initialized with identical populations after coevolution



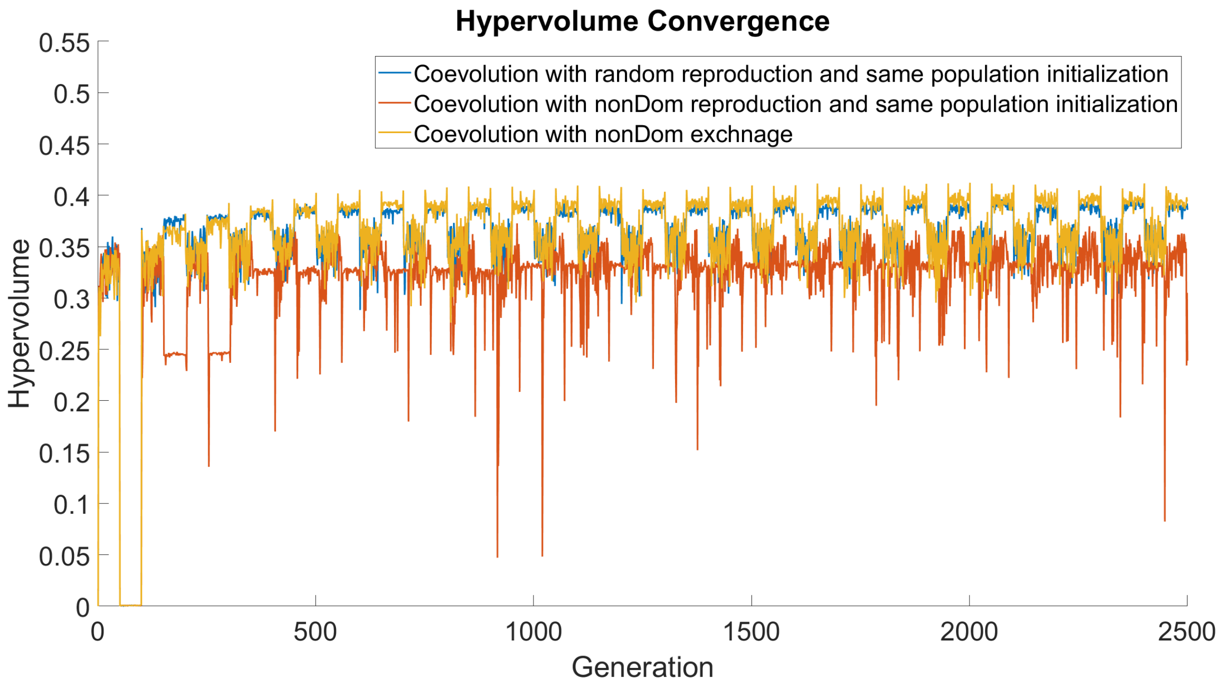
Pareto front of coevolution with random reproduction and identical population initialization



Pareto front of coevolution with nonDom reproduction and identical population initialization

Main Objective	Re-Layouting Cost	Flow Time	Idle Time
Coevolution: random reproduction & same population initialization			
Re-Layouting Cost	0	1319.2	0.6101
Flow Time	1012	317.73	0.37326
Idle Time	132	610.37	0.23315
Trade-Off	360	542.2	0.2889
Coevolution: nonDom reproduction & same population initialization			
Re-Layouting Cost	0	1317.1	0.53297
Flow Time	1136	322.76	0.4599
Idle Time	86	674.05	0.23623
Trade-Off	342	600.2	0.33022

Table 1: Optimal Solutions for Different Coevolutionary Approaches



Hypervolume convergence of coevolutionary approaches with identical population initialization compared with coevolution with nonDom exchnage

	Hypervolume	**IGD**
Random Reproduction & same pop.	0.35974	73.328
NonDom Reproduction & same pop.	0.3177	120.53
NonDom Exchange	0.36207	64.849

Table 2: Performance Indicators for different approaches

Bibliography

[1] Gordon C. Armour and Elwood S. Buffa. “A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities”. In: *Management Science* 9.2 (1963), pp. 294–309. ISSN: 00251909, 15265501. URL: http://www.jstor.org/stable/2627408 (cit. on p. 5).

[2] Charles Audet, Jean Bigeon, Dominique Cartier, Sébastien Le Digabel, and Ludovic Salomon. “Performance indicators in multiobjective optimization”. In: *European Journal of Operational Research* 292 (2020). hal-03862074, pp. 397–422. DOI: 10.1016/j.ejor.2020.11.016 (cit. on p. 36).

[3] Saifallah Benjaafar. “Modeling and Analysis of Congestion in the Design of Facility Layouts”. In: *Management Science* 48.5 (2002), 679–704. ISSN: 0025-1909. URL: https://www.jstor.org/stable/822506 (cit. on p. 15).

[4] Roland Braune, Frank Benda, Karl F. Doerner, and Richard F. Hartl. “A genetic programming learning approach to generate dispatching rules for flexible shop scheduling problems”. In: 243 (Jan. 2022), p. 108342. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2021.108342 (cit. on p. 14).

[5] Javaid Butt and Sunny Jedi. “Redesign of an In-Market Conveyor System for Manufacturing Cost Reduction and Design Efficiency Using DFMA Methodology”. In: *Designs* 4.1 (2020), p. 6. DOI: 10.3390/designs4010006 (cit. on p. 14).

[6] Simon Chinguwa, Wilson R. Nyemba, Kudzai Boora, and Charles Mbohwa. “Feasibility study of the materials handling and development of a sustainable conveying system in plastics recycling and manufacture”. In: *Procedia Manufacturing* 33 (2019), pp. 383–390. ISSN: 2351-9789. DOI: 10.1016/j.promfg.2019.04.047 (cit. on p. 2).

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017 (cit. on pp. 14, 27).

[8] Mark Duinkerken, Jaap Ottjes, and Gabriel Lodewijks. “Comparison of Routing Strategies for AGV Systems using Simulation”. en. In: *Proceedings of the 2006 Winter Simulation Conference*. Monterey, CA, USA: IEEE, Dec. 2006, 1523–1530. ISBN: 978-1-4244-0501-5. DOI: 10.1109/WSC.2006.322922 (cit. on p. 8).

[9] Thomas Dunker, Günter Radons, and E. Westkamper. “A coevolutionary algorithm for a facility layout problem”. In: *International Journal of Production Research - INT J PROD RES* 41 (Oct. 2003), 3479–3500. DOI: 10.1080/0020754031000118125 (cit. on p. 13).

[10] William H. Durham. *Coevolution: Genes, Culture, and Human Diversity*. Stanford: Stanford University Press, 1991, p. 656. URL: http://www.sup.org/books/title/?id=2588 (cit. on p. 12).

[11] Adem Erik and Yusuf Kuvvetli. “Integration of material handling devices assignment and facility layout problems”. In: *Journal of Manufacturing Systems* 58 (Jan. 2021), 59–74. ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2020.11.015 (cit. on p. 3).

[12] Robert J. Gaskins and Jose M. A. Tanchoco. “Flow path design for automated guided vehicle systems”. In: *International Journal of Production Research* 25 (1987), pp. 667–676. URL: https://api.semanticscholar.org/CorpusID:108633187 (cit. on p. 3).

[13] Suman Gothwal and Tilak Raj. “Optimisation of AGVs path layout in flexible manufacturing system using 0-1 linear integer programming”. In: *International Journal of Process Management and Benchmarking* 8 (Jan. 2018), p. 182. DOI: 10.1504/IJPMB.2018.090796 (cit. on p. 3).

[14] Gary Hammond. *Automated Guided Vehicle Systems*. English. Springer Verlag, 1986. ISBN: 0387166777 (cit. on p. 2).

[15] Michael Pilegaard Hensen and Andrzej Jaszkiewicz. “Evaluating the quality of approximation to the non-dominated set”. In: (Mar. 1998) (cit. on p. 36).

[16] Ying-Chin Ho. “A dynamic-zone strategy for vehicle-collision prevention and load balancing in an AGV system with a single-loop guide path”. en. In: *Computers in Industry* 42.2–3 (June 2000), 159–176. ISSN: 01663615. DOI: 10.1016/S0166-3615(99)00068-8 (cit. on pp. 16, 17).

[17] Hasan Hosseini-Nasab, Sepideh Fereidouni, Seyyed Mohammad Taghi Fatemi Ghomi, and Mohammad Bagher Fakhrzad. “Classification of facility layout problems: a review study”. In: *Springer-Verlag London Ltd.* (2017). Received: 1 March 2017 / Accepted: 27 July 2017. DOI: 10.1007/s00170-017-0431-5 (cit. on p. 6).

[18] Ramanujam Jaikumar. *Flexible Manufacturing Systems: A Managerial Perspective.* Division of Research, Graduate School of Business Administration, Harvard University, 1984. URL: https://books.google.de/books?id=fehSuAAACAAJ (cit. on p. 8).

[19] Moshe Kaspi and J. M. A. Tanchoco. “Optimal flow path design of unidirectional AGV systems”. en. In: *International Journal of Production Research* 28.6 (June 1990), 1023–1030. ISSN: 0020-7543, 1366-588X. DOI: 10.1080/00207549008942772 (cit. on pp. 3, 8).

[20] Amirsaman Kheirkhah, Hamidreza Navidi, and Masume messi bidgoli. “Dynamic Facility Layout Problem: A New Bilevel Formulation and Some Metaheuristic Solution Methods”. In: *Engineering Management, IEEE Transactions on* 62 (Aug. 2015), 396–410. DOI: 10.1109/TEM.2015.2437195 (cit. on p. 17).

[21] Tjalling C. Koopmans and Martin Beckmann. “Assignment Problems and the Location of Economic Activities”. In: *Econometrica* 25.1 (1957), pp. 53–76. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1907742 (visited on 11/21/2023) (cit. on p. 13).

[22] Prabhanjan P Kulkarni, Saurabh S Gudhate, and Praveen K Mali. “Design and Analysis of Material Handling System”. In: *IJIRST – International Journal for Innovative Research in Science & Technology* 2.12 (2016), p. 43. URL: http://www.ijirst.org (cit. on p. 2).

[23] R.M Kumar, P. Asokan, and Somasundaram Kumanan. "Design of loop layout in flexible manufacturing system using non-traditional optimization technique". In: *The International Journal of Advanced Manufacturing Technology* 38 (Aug. 2008), 594–599. DOI: 10.1007/s00170-007-1032-x (cit. on pp. 15, 16).

[24] Francois Legillon, Arnaud Liefooghe, and El-Ghazali Talbi. "CoBRA: A cooperative coevolutionary algorithm for bi-level optimization". In: June 2012, p. 8. ISBN: 978-1-4673-1510-4. DOI: 10.1109/CEC.2012.6256620 (cit. on pp. 3, 12).

[25] P. Leitão, J. Barbosa, and D. Trentesaux. "Bio-inspired multi-agent systems for reconfigurable manufacturing systems". English. In: *Engineering Applications of Artificial Intelligence* 25.5 (2012), pp. 934–944. ISSN: 09521976 (cit. on p. 1).

[26] L. M. Nicol and R. H. Hollier. "Plant Layout in Practice". In: *Material Flow* 1.3 (1983), pp. 177–188 (cit. on p. 6).

[27] V. Oduguwa and R. Roy. "Bi-level optimisation using genetic algorithm". In: *Proceedings 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS 2002).* Sept. 2002, 322–327. DOI: 10.1109/ICAIS.2002.1048121 (cit. on p. 3).

[28] Yunfang Peng, Tian Zeng, Lingzhi Fan, Yajuan Han, and Beixin Xia. "An Improved Genetic Algorithm Based Robust Approach for Stochastic Dynamic Facility Layout Problem". In: 2018 (Dec. 2018), 1–8. DOI: 10.1155/2018/1529058 (cit. on p. 13).

[29] Hani Pourvaziri and Henri Pierreval. "Dynamic facility layout problem based on open queuing network theory". en. In: *European Journal of Operational Research* 259 (June 2017), 538–553. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2016.11.011 (cit. on p. 15).

[30] Pablo Pérez-Gosende, Josefa Mula, and Manuel Díaz-Madroñero. "A bottom-up multi-objective optimisation approach to dynamic facility layout planning". In: 0 (Jan. 2023), 1–18. ISSN: 0020-7543. DOI: 10.1080/00207543.2023.2168308 (cit. on p. 13).

[31] Pablo Pérez-Gosende, Josefa Mula, and Manuel Díaz-Madroñero. "Facility layout planning. An extended literature review". In: *International Journal of Production Research* 59.12 (2021), pp. 3777–3816. DOI: 10.1080/00207543.2021.1897176 (cit. on p. 6).

[32] Behnam Rahimikelarijani, Mohammad Saidi-Mehrabad, and Farnaz Barzinpour. "A Mathematical Model for Multiple-Load AGVs in Tandem Layout". en. In: *Journal of Optimization in Industrial Engineering* 13.1 (June 2020). DOI: 10.22094/joie.2018.537.37 (cit. on p. 15).

[33] Benjamin Rolf, Tobias Reggelin, Abdulrahman Nahhas, Sebastian Lang, and Marcel Müller. "Assigning Dispatching Rules Using a Genetic Algorithm to Solve a Hybrid Flow Shop Scheduling Problem". In: *Procedia Manufacturing* 51 (2020), pp. 76–81. DOI: 10.1016/j.promfg.2020.02.051 (cit. on p. 14).

[34] Meir J. Rosenblatt. "The Dynamics of Plant Layout". In: *Management Science* 32.1 (1986), pp. 76–86. ISSN: 00251909, 15265501 (cit. on p. 5).

[35] Ganiyat Salawu, Glen Bright, and Chiemela Onunka. “MODELLING AND SIMULATION OF A CONVEYOR BELT SYSTEM FOR OPTIMAL PRODUCTIVITY”. English. In: *International Journal of Mechanical Engineering and Technology (IJMET)* 1.11 (2020), 115–121. URL: https://www.infona.pl//resource/bwmeta1.element.ID-f923ee25-94d5-445d-a399-dae5699effb1 (cit. on p. 14).

[36] Thomas Seidelmann and Sanaz Mostaghim. “Finding Cost-Effective Re-Layouting Solutions in Modern Brownfield Facility Layout Planning”. en. In: *2022 IEEE Congress on Evolutionary Computation (CEC)*. Padua, Italy: IEEE, July 2022, 1–8. ISBN: 978-1-66546-708-7. DOI: 10.1109/CEC55065.2022.9870345 (cit. on pp. 3, 14, 19, 24, 27, 34, 35, 37–39).

[37] Thomas Seidelmann, Jens Weise, and Sanaz Mostaghim. “Meeting Demands for Mass Customization: A Hybrid Organic Computing Approach”. en. In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. Orlando, FL, USA: IEEE, Dec. 2021, 1–8. ISBN: 978-1-72819-048-8. DOI: 10.1109/SSCI50451.2021.9659946 (cit. on pp. 7, 19, 20, 39).

[38] Mohamed A. Shalaby, Tamer Y. El Mekkawy, and Sherif A. Fahmy. “Zones Formation Algorithm in Tandem AGV Systems: A Comparative Study”. In: *International Journal of Production Research* 44.3 (2006), pp. 505–521. DOI: 10.1080/00207540500268566 (cit. on p. 15).

[39] Daniel Siegel, Hamed D. Ardakani, Jangmyung Lee, Young Su Chang, and Junghoon Lee. “A review of predictive monitoring approaches and algorithms for material handling systems”. In: *International Journal of Advanced Logistics* 3.3 (2014), pp. 87–99. DOI: 10.1080/2287108X.2014.985775 (cit. on p. 7).

[40] Ankur Sinha, Kalyanmoy Deb, Pekka Korhonen, and Jyrki Wallenius. “Progressively interactive evolutionary multi-objective optimization method using generalized polynomial value functions”. In: *IEEE Congress on Evolutionary Computation*. 2010, pp. 1–8. DOI: 10.1109/CEC.2010.5586278 (cit. on p. 10).

[41] Robert Tarjan. “Depth-First Search and Linear Graph Algorithms”. In: *SIAM Journal on Computing* 1.2 (1972), pp. 146–160. DOI: 10.1137/0201010 (cit. on pp. 24, 26).

[42] James A. Tompkins and Ruddell Reed. “An applied model for the facilities design problem”. In: *International Journal of Production Research* 14 (1976), pp. 583–595 (cit. on p. 3).

[43] James A. Tompkins and John A. White. *Facilities Planning*. John Wiley & Sons Inc, 1984. ISBN: 0471032999 (cit. on p. 2).

[44] Betül Turanoğlu and Gökay Akkaya. “A new hybrid heuristic algorithm based on bacterial foraging optimization for the dynamic facility layout problem”. In: *Expert Systems with Applications* 98 (May 2018), 93–104. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2018.01.011 (cit. on p. 13).

[45] Hsiao Fan Wang and Ching Hsiang Chan. “Multi-objective optimisation of automated guided dispatching and vehicle routing system”. en. In: *International Journal of Modelling in Operations Management* 4.1/2 (2014), p. 35. ISSN: 2042-4094, 2042-4108. DOI: 10.1504/IJMOM.2014.063585 (cit. on p. 52).

[46] Y.J. Wang, X.Q. Liu, J.Y. Leng, J.J. Wang, Q.N. Meng, and M.J. Zhou. “Study on scheduling and path planning problems of multi-AGVs based on a heuristic algorithm in intelligent manufacturing workshop”. en. In: *Advances in Production Engineering & Management* 17.4 (Dec. 2022), 505–513. ISSN: 18546250, 18556531. DOI: 10.14743/apem2022.4.452 (cit. on p. 16).

[47] Jianbin Xin, Xuwen Wu, Andrea D’Ariano, Rudy Negenborn, and Fangfang Zhang. “Model Predictive Path Planning of AGVs: Mixed Logical Dynamical Formulation and Distributed Coordination”. en. In: *IEEE Transactions on Intelligent Transportation Systems* (2023), 1–12. ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2023.3254147 (cit. on p. 15).

[48] Cheng Yang, Songkai Liu, and Zhigang Xu. “A simulation-based optimization method for facility layout considering the AGV path”. In: *Journal of Physics: Conference Series* 2430 (Feb. 2023), p. 012019. DOI: 10.1088/1742-6596/2430/1/012019 (cit. on p. 16).

[49] Gary G. Yen and Zhenan He. “Performance Metric Ensemble for Multiobjective Evolutionary Algorithms”. In: *IEEE Transactions on Evolutionary Computation* 18.1 (2014), pp. 131–144. DOI: 10.1109/TEVC.2013.2240687 (cit. on p. 36).

[50] Min Zhang, Rajan Batta, and Rakesh Nagi. “Designing manufacturing facility layouts to mitigate congestion”. In: 43 (Oct. 2011), 689–702. ISSN: 0740-817X. DOI: 10.1080/0740817X.2010.546386 (cit. on p. 15).

[51] Min Zhang, Rajan Batta, and Rakesh Nagi. “Modeling of Workflow Congestion and Optimization of Flow Routing in a Manufacturing/Warehouse Facility”. In: *Management Science* 55.2 (2009), 267–280. ISSN: 0025-1909. URL: https://www.jstor.org/stable/40539144 (cit. on p. 15).

[52] Xu Zhang, Hua Zhang, and Jin Yao. “Multi-Objective Optimization of Integrated Process Planning and Scheduling Considering Energy Savings”. In: *Energies* 13.23 (2020), p. 6181. DOI: 10.3390/en13236181 (cit. on p. 7).

[53] Eckart Zitzler and Simon Künzli. “Indicator-Based Selection in Multiobjective Search”. In: *Parallel Problem Solving from Nature*. 2004. URL: https://api.semanticscholar.org/CorpusID:14163788 (cit. on p. 36).

[54] Petr Škoda and Fathalrahman Adam, eds. *Knowledge Discovery in Big Data from Astronomy and Earth Observation*. Elsevier, 2020, pp. 445–453. ISBN: 978-0-12-819154-5. DOI: 10.1016/B978-0-12-819154-5.00037-0 (cit. on p. 12).

Statement of Authorship / Selbstständigkeitserklärung

I hereby declare that I have prepared this Master's thesis independently and exclusively using the literature and aids provided. The passages taken directly or indirectly from external sources are identified as such.

The thesis has not been submitted to another examination authority or published elsewhere in the same or a similar form.

/

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbstständig und außschließlich unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Stellen sind als solche kenn-tlich gemacht.

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

Place / Ort Date / Datum	Signature / Unterschrift