Mälardalen University
School of Innovation Design and Engineering
Eskilstuna, Sweden

Thesis for the Degree of Bachelor in Computer Science with Specialization in Intelligent Systems 15.0 credits

OPTIMIZING PRODUCTION SCHEDULING IN MANUFACTURING ENVIRONMENTS

Aron Alander
alander.aron@gmail.com

Jonathan Hjalmarsson
jonishjalmarsson@gmail.com

Examiner: Ivan Tomasic
Mälardalen University, Västerås, Sweden

Supervisor(s): Ning Xiong
Mälardalen University, Västerås, Sweden

Company Supervisor(s): John Moberg
MITC, Eskilstuna

12/06/2024

Acknowledgments

We would like to thank our supervisors John Moberg and Ning Xiong for the assistance, guidance and encouragement throughout this study. Their expertise and support have been instrumental in the completion and success of this thesis. We would also like to thank Vincent Adoue at *Mälardalen Industrial Technology Center* (MITC) for providing us with this exciting and innovative thesis work.

Abstract

Efficient production scheduling is important for both maximizing productivity and minimizing costs in manufacturing environments. This thesis presents an approach to optimizing production scheduling using Artificial Intelligence (AI) and Genetic Algorithms (GAs). The primary goal is to develop a generalized solution that can be modified and adapted to the varying needs that different production- or manufacturing lines may have. This research has two main research questions that address the problem at hand. (1) Can Genetic Algorithms be used to optimize a sequence of products in a production line? and (2) How effectively can Genetic Algorithms optimize the sequencing of production tasks in diverse production lines to minimize total order processing times?

Through experimentation with various GA configurations the results achieved suggested that GAs were appropriate for sequence optimization. The study demonstrates that GAs can optimize a production line up to almost 42% , which significantly reduced the total processing time. The thesis also highlights the importance of the representation of data, which is essential for the optimization of the sequence.

Contents

List of Figures

List of Tables

List of Acronyms

ACO Ant Colony Optimization

AI Artificial Intelligence

GA Genetic Algorithm

GPU Graphical Processing Unit

OEE Overall Equipment Effectiveness

PSO Particle Swarm Optimization

RL Reinforcement Learning

STS Single Tournament Selection

UI User Interface

1. Introduction

Artificial Intelligence (AI) has become increasingly popular in recent years and has revolutionized industries and markets worldwide [1][2]. For example in finance, AI is used for fraud detection and in healthcare it is used for patient monitoring and diagnosis [3][4][5]. The potential that AI brings has offered many opportunities for both innovation and optimizations in many fields [6][7]. In the field of manufacturing where customer expectations and the global competition escalates, AI can be of great assistance in transforming traditional practices with new solutions for both efficiency and growth [8]. AI applications in manufacturing include predictive maintenance, quality control and supply chain management. This thesis will explore how AI can be used to address the specific challenge of production scheduling tasks within manufacturing- and production lines, while providing solutions to enhance the overall efficiency of the manufacturing process [9].

1.1 Problem Formulation

Manufacturing environments have many interconnected variables and constraints that influence the scheduling process [10]. These factors include machine availability, capacity limitations and resource allocation constraints. A review of previous research and industry practices shows both the progress made but also the challenges that are faced in production scheduling. Although modern industries have tools to simulate and build production lines [11], there is a notable lack of widespread tools to optimize production [12]. The Mälardalen Industrial Technology Center (MITC) has an interest in a tool capable of effectively optimizing the sequence for a production line. Consequently, MITC has partnered with us to conduct this study. The goal for this study is to develop a modular optimization algorithm specifically designed to minimize the total processing time for orders in diversified production tasks such as production lines or assembly lines. These production lines are characterized by variations in product processing requirements and paths making it necessary for a flexible and adaptable optimization approach.

The lack of optimization frameworks for production scheduling tasks provides the basis for the first research question of the thesis. (1) *Can Genetic Algorithms be used to optimize a sequence of products in a production line?* If it is possible to use Genetic algorithms (GA) for this type of sequence optimization, it is also important to note the effectiveness of these results in comparison to the initial sequence, which leads into the second research question. (2) *How effectively can Genetic Algorithms optimize the sequencing of production tasks in diverse production lines to minimize total order processing times?* As there isn't a calculated optimal solution for direct comparison, this thesis will compare the initial solution with the optimized output to validate the overall improvement in efficiency.

1.2 Assumptions and Limitations

One of the most impactful assumptions in this study is that the travel time between nodes in the production line is set to zero. Travel time in real production lines can vary significantly [13][14]. This assumption affects the results between the generated optimal solution and if tested on a real production line since the real production line will have the travel time. One of the reasons for discarding travel time is that comparatively, the travel time is a lot less than time spent either waiting or processing in the given test cases. This is particularly true in production lines and manufacturing environments where processing times are longer. In scenarios where the processing time is shorter, the travel time would have a bigger impact on the total time. This results in a misleading output for the algorithm. The way the optimization program is made, it is possible to implement travel time by defining them in the input spreadsheet. But during this study this was not done as the production line the study is based on, does not have any provided travel time.

Another limitation is the assumption that none of the machine and resources operate without any unexpected downtime or failures. This idealized scenario does not account for real-world disruptions or failures that can impact production efficiency and scheduling [15][16].

Although a limitation is that the solution is in an idealized scenario, planning production schedules and actual production times often do not align, which is a common issue in production

planning [17]. This discrepancy happens due to the inherent complexity of production lines and the difficulty in accurately capturing all variables.

1.3 Objectives

The objectives of this study are prioritized by their importance, the primary objective being the most important, followed by the second and third objectives. The fourth objective of implementing a User Interface (UI) is considered a bonus objective that can be pursued if time allows for it. Development of a UI will not impact the research and experimentation but it will enhance the accessibility of the solution [18]. The objectives are as follows:

1. The primary objective is to develop a solution that optimizes product sequencing in a production line.
2. Secondary objective is to adapt the solution to different production line configurations and sizes.
3. The third objective is to minimize order processing time across different production lines by evaluating the algorithm's performance.
4. Lastly, a bonus objective is to make the implementation more accessible with a UI.

2. Background

This section will provide explanations for the terms that are important for this thesis and to further understand the concept of production scheduling and why it is important. The terms that will be explained are genetic algorithms, fitness, tournament selection, single point crossover, adaptive mutation, elitism, fixtures and hyperparameters.

2.1 Genetic Algorithms

Genetic Algorithms (GA) are a type of evolutionary algorithm that mimic the natural selection process observed in biological evolution [19]. These types of algorithms are used in optimization problems due to their nature of finding solutions. The algorithm works by initially creating a population with individuals represented by chromosomes that contain different possible solutions for a problem. GA's are composed of three major parts: selection, crossover and mutation. The basics of the selection process is to implement a way of selecting individuals from the overall population that will then be used as parents in the next step. The selected individuals are often selected based on their fitness, which is an indication of the overall quality of the individual. The selected parents will then move on to crossover, the crossover works by swapping genes between the different parents to create offspring that will create a new population. The offspring then go through mutation where parts of their chromosomes will be changed to ensure uniqueness and prevent stagnation. A mutation rate is applied to adjust the probability for the individual to get mutated.

2.1.1 Fitness

In GAs each individual is evaluated by calculating a fitness score which indicates the quality of the specified individual in the population [20]. Depending on the problem the best fitness score could be the highest or the lowest in the population. In a minimization problem, a lower fitness value indicates a better solution. For a maximization problem a higher fitness score is better. To calculate the fitness value a fitness function is implemented. This function may vary depending on the problem, but should reflect the objective, for example:

- **In scheduling:** The fitness function might measure the total processing time.
- **In route optimization:** The fitness function could evaluate the total distance traveled.
- **Resource allocation:** The fitness function in resource allocation problems may measure the efficiency or the effectiveness of resource distribution.

2.1.2 Tournament Selection

Tournament selection is a popular selection method in GAs that works by randomly selecting a selected amount of individuals from the population [21]. The best individual from the selected individuals will become a parent when generating a subsequent generation. By allowing weaker individuals a chance to contribute to the next generation it will broaden the search space for potential solutions and avoid premature convergence on suboptimal solutions. This approach helps the algorithm find a global optimum rather than getting stuck in a local optima.

2.1.3 Single Point Crossover

Single point crossover is a crossover method that randomly chooses a point in the parent chromosome. It then combines the genes before this point from the first parent with the genes after the point from the second parent, producing an offspring [22]. This process iterates with new crossover points for each pair until the new population is fully populated.

2.1.4 Adaptive Mutation

Adaptive mutation is a modified mutation algorithm where the mutation rate dynamically adjusts based on the number of generations elapsed [23]. As generations progress, the mutation rate increases, leading to genetic diversity and preventing stagnation within the population. This ensures a broader exploration of the solution space and enhances the chances of finding a global optimum and improves the overall effectiveness of the algorithm.

2.1.5 Elitism

To prevent the loss of the top performing individuals across generations, elitism is introduced [24]. Elitism ensures that top performing individuals from each generation are carried over to the next generation without any modifications or crossover operations. This approach helps with preventing the loss of potentially optimal solutions.

2.2 Fixtures

In the context of production line optimization, *space constraints* refer to the limits imposed by the layout and capacity of the production environment. These constraints are the maximum number of products that can be processed at the same time in a given production line. The limit is imposed by the physical space available for machinery and product movement or restrictions for the sequences due to logistics. In manufacturing settings, space constraints are a factor influencing the design and efficiency of production processes [25]. Production lines can face limitations based on the available space for equipment and storage, as well as constraints on how many products that can be simultaneously processed because of a limited number of fixtures [26]. The cost of the fixture can be a lot greater than that of the actual product that it holds [27].

Production line fixtures are physical devices or equipment used to hold and position products or components during manufacturing processes [28]. They provide a stable platform for assembling, machining or inspecting parts and ensure consistency, accuracy and efficiency in production operations. The complexity of a fixture can range from being a simple clamp to highly sophisticated automated systems. This diversity means that the cost for these fixtures can vary significantly. Simple fixtures might be relatively inexpensive, while more complex or customized fixtures such as automated robotic welding fixtures used in the automotive industry can be very costly due to their specialized design or engineering required [29]. In some cases the investment in high quality fixtures can be justified or sometimes necessary by the benefits that they provide, such as improved efficiency in production, reduced waste and higher product quality. However it is important to consider the upfront investment with the long term benefits.

As production lines often are complex and some products may have different paths than others [30][31] which also can be seen in Figure 1. Combining this with space constraints, i.e. imposing a limit for the amount of products that can be processed at the same time, for different paths may result in a more realistic representation of production lines. Implementation of path specific space constraint for optimization is important for scenarios where different paths have varying capacities. The various paths and products may have a specific fixture that it needs for processing that the other products may not need. As these fixtures can be expensive [29] it is an important part when planning and optimizing a production line.

2.3 Hyperparameters

Hyperparameters refer to preset configuration settings that dictate the behavior and performance of a machine learning or optimization algorithm [32]. These parameters can be a wide range of settings, for example:

- **Learning Rate:** Determines the step size at each iteration. Too high of a learning rate can cause the algorithm to converge too quickly leading to a suboptimal solution
- **Batch Size:** The number of training examples utilized in one iteration.

- **Population Size:** Refers to the number of individuals in a population. A larger population size allows the algorithm to explore a broader search space.
- **Mutation Rate:** Is the probability that an individual solution will go through a mutation during the optimization process.

Optimizing the hyperparameters often require experimentation and tuning as different values can lead to better or worse outcomes.

2.4 Production Scheduling

Production scheduling is an important part of manufacturing and operations management which involves planning and controlling the production process to ensure efficient use of resources [33]. It is designed to maximize productivity, minimize cost and ensure that products are delivered on time, which is why it is important to improve the field of production scheduling. By optimizing the production scheduling, production cost can be further reduced and efficiency can be improved. Due to sustainability becoming an increasingly important consideration, the optimization of the production scheduling also aims to reduce the environmental impact of the manufacturing processes, while still maintaining efficiency and productivity [34].

3. Related Works

The literature study for this project was conducted to search for work that showed different algorithms to approach this type of optimization problem. The study showed promising results by optimizing sequences in different applications compared to a random sequence. Although some approaches didn't use AI specifically, it showed that this was possible. There is also previous work done in a similar field where AI was used to optimize an assembly line using GAs which showed possibilities for this thesis. There were many different papers solving optimization problems and two of these papers were chosen.

3.1 Optimizing Product Orders Using Graph Algorithms

In the paper *Optimizing product orders using graph algorithms for improving incremental product-line analysis*, Lity et al. evaluated if an optimized sequence of product orders would reduce the total time taken compared to a random sequence of orders [35]. The paper proposed a graph based algorithm to solve the problem. By representing the products as nodes like in the traveling salesman problem (TSP) [36] it was possible to create sample graphs to find an optimal path (the shortest path) to visit all nodes. The authors proposed a greedy algorithm with two heuristic methods, nearest insertion (NEARIN) and farthest insertion (FARIN). For NEARIN the authors always chose the minimum distance to the next node, whilst FARIN took the maximum distance to the next node. Afterward the best fitting position was determined.

The result that the authors presented was that the optimal sequence was 40.1% better than the random order of products in a sequence [35].

3.2 Solving an Assembly Sequence Optimization Problem

In the paper *Optimizing product orders using graph algorithms for improving incremental product-line analysis* Fawas Alharbi and Qian Wang presented a solution to optimize assembly sequences, specifically focusing on the assembly of a car engine pump valve [37]. The paper highlights the importance of optimizing assembly line sequences for reducing assembly times and production costs. This is especially important for small and medium-sized enterprises (SME) that are trying to compete in global markets. They also acknowledge the complexity that is introduced by increasing the number of components which lead to more constraints and challenges in the assembly process.

The study discusses the effectiveness and versatility of GAs in these types of optimization problems and provides promising results by reducing the assembly time for the car engine pump valve [37]. Through five generations the total time was decreased from 570 seconds to 500 seconds. The improvement in time showcases the importance of an optimized sequence and that optimization could be done by doing small changes in the sequence.

3.3 Extension of the Previous Work

The previous work [35][37] has provided promising results by optimizing the sequence of products. This led to the choice of using a GA to find a general solution to the sequence optimization problem. In this study a random amount of specific orders that should follow a specific path of processing steps are provided, which is similar to the related work. The difference is that the solution needs to be modular so that products and processes can easily be added or removed. This will give a generalized optimization approach to address different challenges in different production environments.

4. Method

This study includes a combination of algorithmic design, simulation and testing to develop and evaluate the algorithms and their practicality in use for production line sequence optimization. The choice of methods is based on the need for a solution that is theoretically reliable and also practically applicable in real-world industrial settings. The literature study, research, experiments and implementations have been done iteratively to ensure suitability during changes throughout the study.

4.1 Discarded Methods

Throughout this study multiple methods were experimented and considered, but discarded in favor of the chosen approach. One initial candidate was Tabu Search, which initially showed great promise with efficiently searching a solution space and finding well structured solutions to the sequence optimization problem [38]. The structure resembles a more traditional sequence than the later, more randomized appearance of the sequence. Tabu Search is useful for its ability to navigate complex solution spaces and avoiding local optima by maintaining a list of tabu moves that should not be revisited. However, when the experimental phase progressed and the amount of products increased it became evident that Tabu Search's time complexity became a problem. The worst-case time complexity is $O(2^n)$, where n is the number of products [39]. Although in practice the time complexity is often lower than the worst-case scenario. The exponential growth in computational requirements became restrictive as the scenarios with large numbers of products took too much time for it to be practical. Due to this limitation, Tabu Search was ultimately discarded as a viable method for this study.

The second variation of a GA that was experimented with was Simulated Annealing (SA). SA is a probabilistic optimization algorithm inspired by the process of annealing in metallurgy [40]. SA explores the solution space by iteratively accepting potentially worse solutions to avoid getting trapped in local optima. However, despite its potential the method was discarded because the results that were achieved were not satisfactory. This is because SA is sensitive to parameter settings, particularly the cooling schedule. The cooling schedule dictates how the algorithm transitions from a wide range of solutions to a more focused and refined search towards the optimal solution.

By experimenting with different sorts of GA structures and methods it was decided that going forward a standard GA framework was going to be the backbone of the optimization algorithm.

4.2 Simulation

To understand how the production line was set up and how different products interacted with the different processes a simulation was made. This was to simulate an actual production line to help better understand the concept. Besides from the simulation a simple schematic was also made to provide a visual representation of the layout of the provided production line as seen in Figure 1.

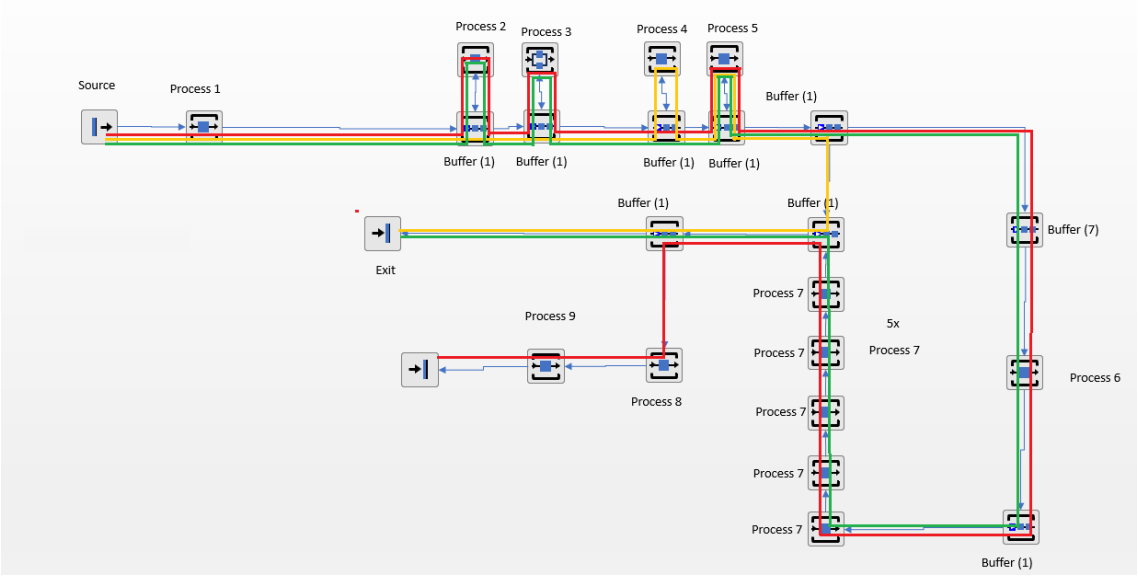

Figure 1: Presents a schematic of the production line, illustrating all the processing steps and the different paths for each colored group of products. Different processes have varying capacities, for example buffer zones are designated areas where products can temporarily wait until the processing step it needs to take is available and can be seen as the conveyor belts in the real production line.

To get a better understanding of how the products were going to be transported in the production line, a simulation of a production line was created, as seen in Figure 2. This production line formed the basis for the thesis work and assisted in examining the different product sequences and their impact on the overall efficiency.

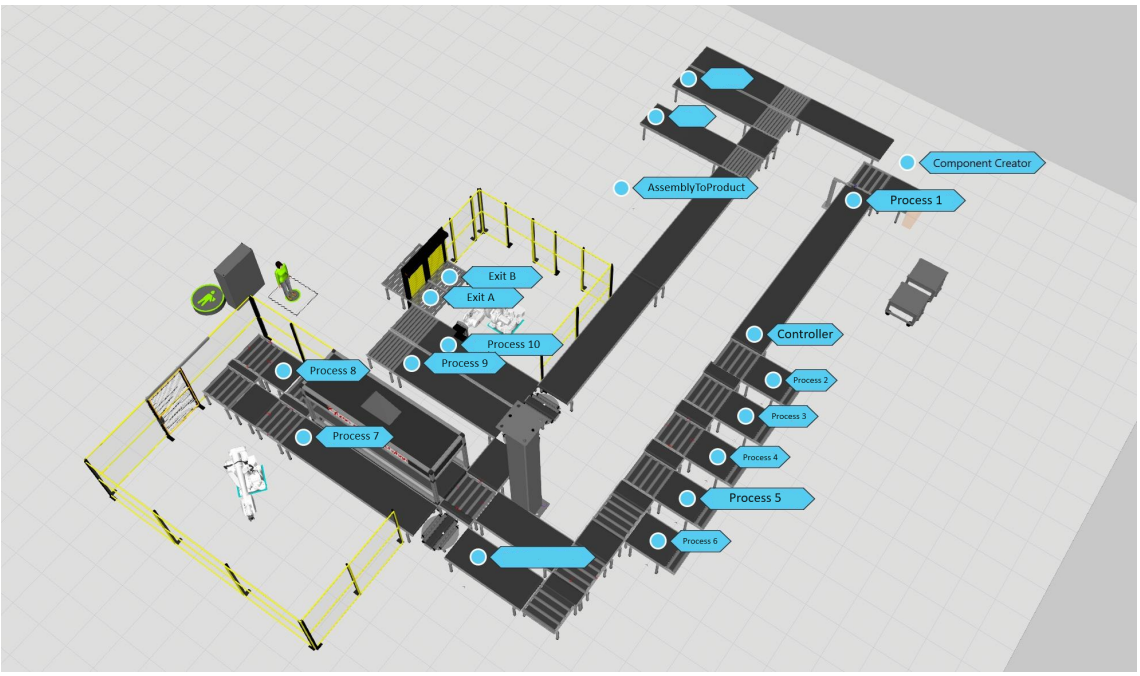

Figure 2: Displays the simulation of the production line seen in Figure 1 constructed in Visual Components [11] where the efficiency of a sequence of products could be tested. This study is based on this theoretical production line.

4.3 Data

The data that was utilized for this study was obtained from MITC in the form of an Excel spreadsheet [41] and a schematic of the production line used during this study.

The Excel sheet contains information about each product and its associated processing steps. The processing times for each step are represented numerically (in minutes) and are listed in the spreadsheet. Cells that are left blank indicate that the step does not contain any processing. The data for developing the simulation model during this study can be seen in Table 1.

Along the spreadsheet, a schematic of the production line layout was provided, as seen in Figure 1. This schematic is important for understanding the physical layout and workflow of the production line. It contains the different product paths for this case, helping to visualize the sequential and spatial relationships between the various processing steps.

To clarify the flow of the different products throughout the production line, the products are categorized into groups based on their processing paths, which are color-coded for easy distinction between them as seen in Table 1 and 3.

- Red group: Products A, B and C
- Yellow group: Products D, E and F
- Green group: Products G, H and I

Each group follows a different path in the production line, but product processing times may vary for products following the same path.

Step Name	A	B	C	D	E	F	G	H	I
Process 1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Process 2	6	6	6				6	6	6
Process 3	6	6	6				3	3	3
Process 4				6	6	6			
Process 5	6	6	6	6	6	6	6		
Process 6	8	8	8				8	8	8
Process 7	15	15	15				15	15	15
Process 8	5	5	3						
Process 9	15	15	15						

Table 1: The table displays the representation of the data in the spreadsheet *Processing Times*. Each column represents a product (labeled A-I), the rows show the various processing steps for each product. The numerical values within each cell serve as the duration for each process, measured in minutes. An empty cell means that the corresponding product should bypass that particular processing step.

The spreadsheet was then divided into multiple different sheets to further assist in loading the data into the GA. It was split into five sheets, *Process Steps*, *Product Paths*, *Processing Times*, *Product Amount*, and *Fixture Amount*. Table 1 showcases the sheet containing processing times for each product.

Step name	Capacity
Process 1	1
Buffer A (in)	1
Process 2	1
Buffer A (out)	1
Buffer B (in)	1
Process 3	2
Buffer B (out)	1
Buffer C (in)	1
Process 4	1
Buffer C (out)	1
Buffer D (in)	1
Process 5	1
Buffer D (out)	1
Buffer E	1
Buffer F	7
Process 6	1
Buffer G	1
Process 7	5
Buffer H	1
Process I	1
Process 8	1
Process 9	1

Table 2: The table displays the contents of the *Processing Steps* sheet, presenting each processing step and buffer zone's capacity. A capacity of one indicates that only one product can occupy that step at any given time.

Buffer zones are a space where products can wait for the next process to become available, the zones were incorporated to simulate the path between each process. Each buffer zone and processing step can vary in capacity. For example if a buffer or process has capacity two it means that two products can wait for their turn to enter a process or get processed at the same time. Table 2 displays the various step names and their respective capacity.

Product	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
A	Process 1	Buffer A (in)	Process 2	Buffer A (out)	Buffer B (in)	Process 3
B	Process 1	Buffer A (in)	Process 2	Buffer A (out)	Buffer B (in)	Process 3
C	Process 1	Buffer A (in)	Process 2	Buffer A (out)	Buffer B (in)	Process 3
D	Process 1	Buffer A (in)	Buffer A (out)	Buffer B (in)	Buffer B (out)	Buffer C (in)
E	Process 1	Buffer A (in)	Buffer A (out)	Buffer B (in)	Buffer B (out)	Buffer C (in)
F	Process 1	Buffer A (in)	Buffer A (out)	Buffer B (in)	Buffer B (out)	Buffer C (in)
G	Process 1	Buffer A (in)	Process 2	Buffer A (out)	Buffer B (in)	Process 3
H	Process 1	Buffer A (in)	Process 2	Buffer A (out)	Buffer B (in)	Process 3
I	Process 1	Buffer A (in)	Process 2	Buffer A (out)	Buffer B (in)	Process 3

Table 3: The table shows the *Product Paths* sheet where each product's paths are defined as steps. For example, each product starts with *Process 1* and then *Buffer A (in)*. Each step for every product is defined.

Each product had a defined path including their specified processing steps as shown in Table 3. This table shows an overview of the sequential steps that each product undergoes, providing a clear representation of the unique paths followed by each item in the production process.

Product	Amount
A	40
B	20
C	30
D	15
E	35
F	10
G	18
H	9
I	3

Table 4: *Product Amount* sheet where the amounts of each product that needs to be produced are defined. The values can be changed depending on the current demand.

The data that is provided in the figures are real values that are manipulated to anonymize the data. This is because the actual time taken, specific amounts of products, and the names of the processing steps are not relevant for the study. The spreadsheets were created to easily make the solution modular, should there be an increased demand in the amount of a specific product or if new products needed to be added or removed. As seen in Table 4 the quantity of each product can be modified as needed, to meet new demands.

Group Path	Fixture Amount
A,B,C	3
D,E,F	3
G,H,I	3

Table 5: The table shows the sheet *Fixture Amount* which was used to set up constraints for each group of products. This meant that parameters for how many products from each group that could be in the production line at a time could be monitored and set to the amount of fixtures that the production line has for each product category.

The *Fixture Amount* sheet was implemented as seen in Table 5 to establish constraints on the number of fixtures a company can utilize. For example if a company has four fixtures that are product specific for the products *A, B, C* and only two fixtures for the remaining product groups, this parameter could be changed in this spreadsheet. This ensures that the quantity of products in the production line can be adjusted.

4.4 Genetic Algorithm Development

When optimizing the workflow for the production line, there is only one thing that actually can be changed and that is the order in which the products enter production. The known information is the different amounts of products that should be produced. This can be represented as a single list containing what products that shall be produced and in the order they will be produced.

The selection of a GA for this study was made with regard to the challenges presented by production line optimization and the representation of the data in the code. GAs are appropriate for solving these types of problems due to production lines often being characterized by discrete, nonlinear progress with constraints such as varying processing times and multiple product paths [42][43].

GAs perform well in scenarios where the solution space is discrete and solutions are represented as combinations of distinct elements, for example, the sequencing of products [44]. Unlike other optimization methods that may require the problem to be linear. GAs do not need any assumptions about the problem linearity. Production lines are often nonlinear due to the complex relationships between the sequence order and production time, which come from varying process durations and dependencies between tasks. It is also possible to manage multiple constraints by incorporating them into the fitness function. For this production line optimization problem, the

fitness function handles the total processing time while also accounting for the operational constraints such as capacity and different requirements for each product type. The use of GAs allows for easy adaptations in the production environment. But also changes to the genetic parameters, such as the crossover, mutation and the selection can be made to adjust for different industrial and production requirements. The used GA was developed with a gene representation, where each gene portrays a product in a given sequence. The first gene in the list of chromosomes will be the first to be processed and the next will follow. Meaning that the structure of the list is the order that the products will be processed. The genes combined serve as the population of the GA. The products and the amount of each that shall be processed gets loaded from an excel sheet containing this information and then turned into a dictionary, for example a dictionary structured as:

$$\{\mathrm{A}:4,\mathrm{B}:2,\mathrm{C}:3,\mathrm{D}:1,\mathrm{E}:1,\mathrm{F}:1,\mathrm{G}:1,\mathrm{H}:2,\mathrm{I}:3\}$$

Where each letter represents a product, and its corresponding number the amount that will be processed. This dictionary is then turned into a list where each product is present as many times as the amount of the specified product that is supposed to be produced. So the previously described dictionary turns into a list that looks like:

$$[A, A, A, A, B, B, C, C, C, D, E, F, G, H, H, I, I, I]$$

The list is used with the fitness function and returns the calculated time it will take to process the product sequence. This list then gets genetic operations applied on it which will change the sequencing of the list and result in a changed production time.

4.5 Evaluation

One of the challenges with this optimization task is that there does not exist a current optimal path or a feasible way of calculating one. The way that evaluation was done for this study, was to introduce an alphabetically structured product sequence and a randomly shuffled one, where the alphabetical is the simple input list and the shuffled is the same list but shuffled randomly. Then using both of the sequences in the algorithm and comparing the initial predicted time taken with the optimized time prediction.

To verify the accuracy of the GAs production line calculations and constraints, manual test cases were conducted. These tests involved using the first four processes with a small batch of products allowing the final time to be manually calculated and verified as correct when compared to the output value provided by the GA.

4.6 Fitness Function

An important component of a GA is the fitness function [44][20], designed to evaluate the quality of a given solution. In the case of production line optimization, the fitness function evaluates the efficiency of various sequences of products through a production line. The function checks each sequence by simulating the production process and calculating the total time taken processing each product. The goal is to minimize the processing time, which is aligned with the objective of optimizing a production line. The fitness function returns the evaluation in form of a number, for this algorithm the time will be used as fitness because the lesser the time, the better the solution is.

4.6.1 Function Overview

The function accepts a list of sequences, with each sequence representing a potential solution. Each sequence consists of an ordered list of products that needs to be processed by the production line. All sequences are based on the original input. The function operates as follows:

- **Initialize Simulation:**
  - Load the initial sequence of products from the input.

  - Create a dictionary to track the availability of each processing step in the production line.
  - Set all processing steps as available at time zero.

- **Process Each Sequence:**
  - Iterate through each product in the sequence.
  - For each product, determine the required processing steps and their respective times.
  - Update the dictionary to reflect the availability of processing steps after each product is processed.

The time calculation works in two stages, first synchronizing steps, then summing the processing time. For each of a product's required steps the function first ensures the steps availability. If the steps are not available (indicating that it is occupied by another product), the start time for the current product is delayed to match the steps next availability. The function then adds products specific processing time at the given step to the total time taken. This process accounts for the time that every product spends at each step, this includes the time spent waiting due to step unavailability. The total processing time in a sequence is measured by the maximum time taken among all products resulting in the completion time for the entire sequence. The function returns a list of tuples with each tuple containing a sequence and its corresponding fitness score (the total processing time). The scoring mechanism allows for comparing the efficiency of different sequences directly. So that the GA can evolve more efficient sequences over subsequent generations.

5. Ethical and Societal Considerations

In addition to technical aspects of the development and implementation of the optimization algorithm it is important to address the ethical and societal implications that this algorithm might have. Although the development of this algorithm didn't reveal any obvious ethical or societal violations, it is still important to reflect on these considerations. This ensures that the solution not only performs efficiently but also aligns with social and ethical values and responsibilities.

5.1 Ethical Considerations

The optimization algorithm utilizes production data that can be anonymized and free from any personal or corporate symbols. Since the solution does not require the purpose of the data to be defined, the data used during development of the algorithm is thus anonymized. The data also does not need to be published in any way because of the nature of the process only returning a sequence and the predicted time for said sequence. The data only has to be stored in the computer that runs the algorithm. That being said, it would be possible to use this optimization algorithm with non anonymized data since the program itself does not anonymize it. It is only the data itself that is fed to the program via an Excel spreadsheet that has been anonymized. Unlike many modern AI-driven solutions that operate as "black boxes" [45], the genetic algorithm used is inherently transparent. Each step of the algorithm's genetic process, for example selection, crossover and mutation are all documented to provide clear insight in how the solutions are made. The transparency also helps with troubleshooting and when making adjustments.

5.2 Societal Considerations

While the societal considerations might not be apparent and may seem less direct than the ethical considerations, there are significant ways in which the optimization of production lines can impact society, particularly in terms of sustainability [46][47]. By increasing overall efficiency the runtime of production can be significantly reduced [48]. This reduction could translate to lower energy consumption and waste, which would directly contribute to a decrease in the environmental footprint of manufacturing processes. Besides energy consumption there are also cost savings achieved through optimized production. These savings can be reinvested in new areas such as research or development to continue becoming more eco-friendly and promoting more sustainable industry practices. Another societal consideration is the potential for improved working conditions. Optimization of a production line can lead to reduced physical strain on workers but also a more safe and efficient work environment [49]. Companies that offer a safe, efficient and comfortable work environment can find it easier recruiting new talent and retaining current employees [50]. The work environment enhances the company's reputation, making it more appealing to recruits. Existing employees are more likely to stay if they feel that their health and safety are prioritized, helping to reduce turnover rates and cost associated with recruiting new staff.

6. Finding the Optimal Product Path

This section will describe the experimental setup including the environment setup, data preparation, and the implementation of the GA along with its hyperparameters and components.

6.1 Environment Setup

The GA was implemented in Visual Studio Code [51] running Python 3.10 [52] and *pandas* version 2.2.1 [53]. The computer that ran the algorithm had a NVIDIA GTX1660 Ti 8GB graphics card (GPU).

6.2 Data Preparation

The data that was presented in the thesis was in an Excel sheet format [41] and needed to be processed and sorted into dictionaries so that the information was easily accessible and usable. This was done using the *pandas* library in Python [53]. Initially the Excel data was read into *pandas* DataFrame objects. These DataFrames were then converted into dictionaries for various purposes:

- Process steps were organized into a dictionary, mapping each step to its capacity.
- Processing times were organized into a dictionary with each step name as the key and the stored values being the time.
- Product amounts transformed into a dictionary linking each product to its quantity.
- Product paths were converted into a dictionary mapping each product to its respective path.
- Constraints were set by assigning the products and the amount for the different paths.

The preprocessing steps were necessary to be able to use the data and to ensure the solution was modular. Converting the data into dictionaries was necessary for efficient access of the optimization process.

6.3 Genetic Algorithm

The GA was initialized with a population size of 20. The individuals in this population were created by a function that made a list containing the amount of each product that was specified in the Excel sheet. The individuals were structured in an alphabetical order.

6.3.1 Genetic Algorithm Hyperparameters and Setup

The hyperparameters for the GA went through a process of experimentation to determine effective values for this study's specific problem. To keep the experiment consistent, the hyperparameters were without changes throughout the experiments and tests. The configuration of the hyperparameters were structured as follows:

- **Population size:** The population size was set to 20 to maintain computational efficiency but still having enough solution diversity.
- **Number of generations:** The number of generations was set to 500 iterations to allow the algorithm to find an optimal solution.
- **Number of parents:** 2
- **Mutation rate:** A mutation rate of 5% was chosen as the starting point for the adaptive mutation, introducing a moderate level of mutation while still maintaining genetic diversity within the population. The mutation rate increased for each generation passed.
- **Elitism ratio:** An elitism ratio of 10% was chosen which preserved the top performing individuals to promote steady improvement.

The datasets used with all the differing test cases consisted of different sequences to find differences and scenarios where the optimization has the most impact. Each dataset was designed to simulate different challenges in a production line.

- Fixed quantity: Three uniform datasets were used, where each product has the same production amount. The quantities set at 10, 20 and 50. The setup helped with evaluation of the scalability and efficiency as the volume of production increased.
- Path variability: Three datasets focused on different path majorities, labeled as *red*, *yellow* and *green*. A path majority means that there will be proportionally more of the products with that specific color label. The configuration of these paths are illustrated in Table 3. Comparing the different paths is necessary for examining how path complexity affects the optimization process.
- Variable quantity: To simulate the variability encountered in production settings, one dataset used the *RANDBETWEEN()* function in excel [54], assigning each product a random number of products between 1 and 50.
- Fixture variability: For the group specific space constraints, a modified dataset was used where different fixture amounts were specified. The purpose of this is to find the highest optimization potential and bottlenecks in the production line. The configuration of the dataset can be seen in table 5.

6.3.2 Fitness Function

To calculate the fitness of each individual in the population a fitness function was defined [55]. The algorithm iterates through each product in every individual to simulate a real production line, where rules are defined to ensure that products that need a specific process have to wait for its completion. Products that don't need a specific process can surpass the products that are in an ongoing process. A time is measured for each product and is the total time taken for a product to reach its end goal. The final fitness value output is determined by the product in the current sequence with the longest processing time, meaning it's the last product to leave the production line. This represents the total time taken for the entire sequence to complete and does not necessarily mean that the final product in a sequence is the last product in the input sequence, since products can pass each other in the production line. The value that indicates the total completion time is then saved in a list coupled with the individual. For example a sequence containing one product per product type would look like this after using the fitness function:

$$[[A, B, C, D, E, F, G, H, I], 91.5]$$

This is how the output looks like, where the sequence list is the first element in the total list and the fitness value associated with the sequence is the second element. In this case the total time is 91.5 minutes.

6.3.3 Selection

The selection method used was Tournament Selection, where three individuals from the population were selected. The best individual from these three individuals would become a parent for the next generation. This selection was done twice to receive both parents. Tournament selection was chosen because it efficiently balances selection pressure and genetic diversity.

6.3.4 Crossover

For this thesis a simple single point crossover was implemented where a splitting point was randomized between zero and the length of the individual. The splitting point was then used to combine the first part from the first parent and the second part after the splitting point from the second parent which created an offspring. The crossover was repeated until a new population was reached. Single point crossover was used for its simplicity and efficiency. Having a more advanced crossover function was tested but did not yield any improved results.

6.3.5 Mutation

An adaptive segment mutation was implemented to introduce diversity within the population and to prevent premature convergence [23]. The mutation rate was adapted based on the formula:

$$1 - \frac{\text{current generation}}{\text{max generation}}$$

This ensured that the mutation rate would be dynamically adjusted based on the generations elapsed, promoting exploration of the solution space. If a mutation was triggered, a segment of a varying length spanning from one to the size of the individual was selected. The products in the segment were then shuffled to mutate the individual. Using adaptive segment mutation allowed for adjustment of the mutation rate when the algorithm was at risk of stagnation. This has a big impact at avoiding local optima.

6.3.6 Enforcing Constraints

To maintain adherence to constraints, such as ensuring the product quantities align with the products specified in the Excel sheet *Product Amount* 4 after crossover and mutation, a function to enforce quantity constraints was implemented. The function examined each offspring and if an offspring violated any constraint, indicating an illegal sequence of products, the function returned False. Meaning that the offspring needs to be regenerated. The function returned True if the offspring complied with all the constraints, allowing it to proceed to the next stage. This enforcement mechanism guaranteed that only solutions that had legal sequences could pass.

6.3.7 Elitism

Elitism was implemented to improve the GA by saving a set amount of elites. Elites were defined by being the best individuals in a population, meaning they had the lowest fitness value. The amount of elites that were transferred to the next generation was calculated by:

$$\text{Elitism Ratio} \times \text{Population Size} \tag{1}$$

The elitism ratio was set to 0.1 meaning that two individuals from the population became elites. This ensured that the best genes were passed on to the next generation making the algorithm more efficient [56].

6.3.8 Setup for Implementing Space Constraints

To account for space constraints, the GA was adapted to include new parameters that simulate the limitations of the production line. This adaptation involves integration of a constraint that checks if there is space available for a product and waits if there is none available.

The experiment was conducted with checking the different constraint amounts, starting at one as the maximum amount of concurrent products at the same time, up to 20. So that the optimal amount of spaces could be found. Since the testing without any constraints proved the difference between the amount of products used, this experiment did not focus on that part. But to assure the effect of the amount of products used, this experiment was done by testing the constraints on 10, 20, 50 and randomized amounts per product.

6.3.9 Group Specific Space Constraint Setup

The experiments were conducted by using the same datasets as before but with an added sheet in Excel that defined the amount of fixtures that were used for each path. The product fixtures were based on the group colors as mentioned in section 4.3. So “Red”, “Yellow” and “Green” product groups with differing amounts of concurrent products were used. As experimenting with the amounts of products has already been done in previous experiments, this experiment was conducted using 50 products per product type. Finding differences and bottlenecks in the production line was done by changing the number of concurrent products per specific group. The additional Excel sheet for the dataset contained:

- Red fixture majority: 5 red fixtures, 2 yellow fixtures and 2 green fixtures
- Yellow fixture majority: 2 red fixtures, 5 yellow fixtures and 2 green fixtures
- Green fixture majority: 2 red fixtures, 2 yellow fixtures and 5 green fixtures
- Even distribution of fixtures: 3 red fixtures, 3 yellow fixtures and 3 green fixtures

7. Results

This section presents the outcomes of the conducted experiments using the GA with various constraints. The aim is to present the effectiveness of the GA in optimizing production line sequences with different constraints. The total runtime differed between the datasets, with the largest dataset (containing 50 of each product type) taking up to 60 seconds to complete. With a smaller dataset (containing 10 of each product) the time to complete was about 10 seconds.

7.1 Randomized Sequences

To compare the effectiveness of the optimization and the results, it is useful to compare the optimized sequences against a baseline. During this study, the baseline was represented by a sequence where the products are alphabetically arranged:

$$[A, A, B, B, C, C, D, D, E, E, F, F, G, G, H, H, I, I]$$

This structure inherently limits the sequence since it is structured in a way that limits different products in using their set path, as all the previous products are before in the queue. To explore the optimizations made by the GA, the initial experiment was done by randomizing the sequence of products in the datasets consisting of 10, 20 and 50 units per product type. The purpose of randomization was to create a more neutral baseline, allowing for a fairer comparison with the optimized sequence. For each randomized dataset, the sequence was shuffled 100 times to mitigate any statistical outliers that may occur. The average processing time from these 100 runs was then calculated by using the arithmetic mean formula [57].

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

This value was then used as a comparative measure against the original alphabetically ordered sequence.

Different datasets	**Initial time (minutes)**	**Time after shuffle (minutes)**	**Difference (minutes)**	**Difference (%)**
10 per product type	633.5	557	76.5	12.08%
20 per product type	1233.5	1044	189.5	15.36%
50 per product type	3033.5	2505	528.5	17.42%

Table 6: This table shows the initial unoptimized time for the different sequences and the randomly shuffled sequences total processing time. Also the differences between them in minutes and the increased efficiency by doing the shuffle.

The random shuffle resulted in an efficiency increase ranging from 12.08% to 17.42% as seen in Table 6. This displays the inefficiency of simply using the alphabetically ordered sequence. However, it is important to note that the alphabetical sequence is not necessarily reflective of real-world applications, it is merely used as a structured baseline for this experiment.

7.2 Results with no Space Constraints

Using no space constraints for the tests means that the amount of products that can be processed at the same time has no set limit. Except for the total amount of products that can be present at the production line at the same time, which is the sum of the capacity in the dataset, the capacity can be seen in Table 2.

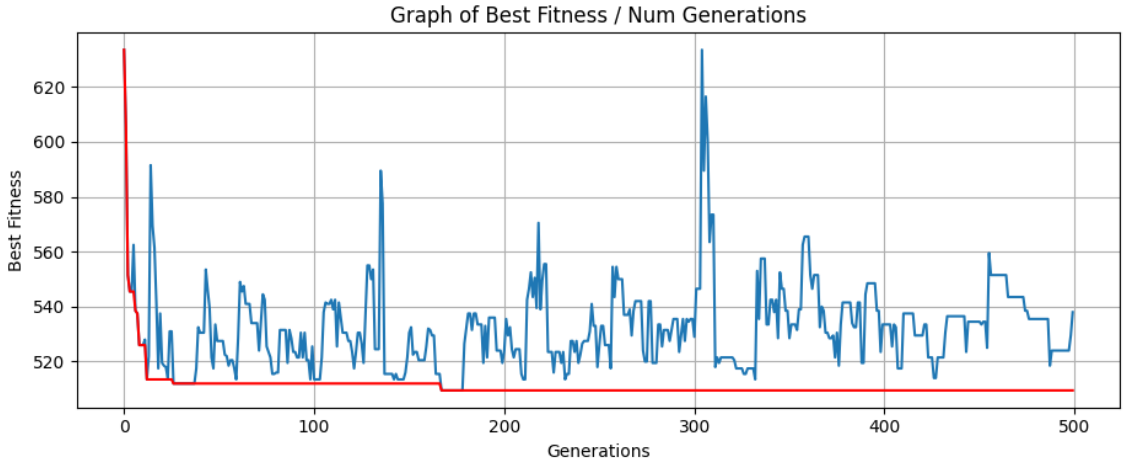

Figure 3: The graph illustrates the GAs evolution process for the dataset containing 10 products for each product type without any space constraints in the production line. The blue line indicates the lowest fitness value found in the population in each generation as the GA progresses. The red line presents the overall best processing time achieved during the optimization process.

The graph in Figure 3 displays the nature of the GA, as it converges fast and finds a good result which is represented by the red line, but takes some time to find the best. While the blue line, which represents the current best solution, searches the solution space to find a more optimal sequence. If the GA is not able to find a better sequence, i.e. it gets stuck in a local optima, it will shuffle the sequence. Which can be seen clearly around 300 generations. The optimized sequence for the 10 per product type dataset resulted in a total processing time of 509.5 minutes. Below is the resulting optimized sequence:

$$
\begin{array}{[cccccccccccccccc}
I, & H, & A, & B, & A, & C, & H, & F, & I, & C, & B, & H, & B, & A, & D, & A, \\
I, & E, & D, & G, & H, & B, & A, & G, & G, & E, & E, & C, & D, & I, & C, & E, \\
C, & F, & D, & B, & E, & B, & I, & A, & F, & C, & A, & B, & C, & D, & B, & G, \\
G, & G, & H, & G, & E, & E, & G, & F, & C, & D, & F, & I, & E, & B, & A, & C, \\
A, & I, & D, & F, & C, & B, & F, & D, & I, & G, & G, & A, & H, & H, & F, & E, \\
H, & H, & E, & F, & D, & H, & I, & I, & F, & D] & & & & & &
\end{array}
$$

For comparison, the original with a processing time of 633.5 minutes sequence was as follows:

$$
\begin{array}{[cccccccccccccccc}
A, & A, & A, & A, & A, & A, & A, & A, & A, & A, & B, & B, & B, & B, & B, & B, \\
B, & B, & B, & B, & C, & C, & C, & C, & C, & C, & C, & C, & C, & C, & D, & D, \\
D, & D, & D, & D, & D, & D, & D, & D, & E, & E, & E, & E, & E, & E, & E, & E, \\
E, & E, & F, & F, & F, & F, & F, & F, & F, & F, & F, & F, & G, & G, & G, & G, \\
G, & G, & G, & G, & G, & G, & H, & H, & H, & H, & H, & H, & H, & H, & H, & H, \\
I, & I, & I, & I, & I, & I, & I, & I, & I, & I] & & & & & &
\end{array}
$$

Comparing these sequences, it can be seen how the optimized sequence improves processing efficiency. The optimized sequence does however seem to lack any form of clear pattern.

Different datasets	**Initial time (minutes)**	**Optimized time (minutes)**	**Difference (minutes)**	**Difference (%)**
10 per product type	633.5	509.5	124	19.57%
20 per product type	1233.5	987.5	246	19.94%
50 per product type	3033.5	2427.5	606	24.96%
Red product majority	1246.5	1244.5	2	0.01%
Yellow product majority	913.5	666.5	247	27.04%
Green product majority	1053.5	907.5	146	13.85%
Randomized amount	1793.5	1355.5	483	26.93%

Table 7: The table depicts a comparative analysis of production times before and after optimization for the different datasets without any space constraints. Each dataset reflects varied production challenges, including fixed production quantities (10, 20, 50 per product), path-specific scenarios (Red, Yellow and Green product majority) and also randomized quantities. The table highlights the time differences between before and after the optimization process both in time and in percentage.

The optimization results show the scalability of the GA. As the quantity per product increases from 10 to 50, the percentage improvement in the processing time also increases, peaking at 24,96% for the 50 per product scenario. This indicates that the algorithm is effective in small and large scale operations. The results from the varied path datasets (Red, Yellow and Green) reveal that the product group has a large impact on the result. The Yellow path scenario shows the highest improvement at 27.04%, suggesting that the Yellow product group has the most potential regarding time optimization. While the Red product majority shows negligible improvement at 0.01%. Thus the red is already optimized because of the inability to find better times for the dataset. The randomized amount where each product varied between 1 and 50 resulting in a 26.93% improvement shows the algorithm's ability in handling the unpredictability and diversity for production line requirements.

Comparing the results from the experiments by the unoptimized randomly shuffled sequences seen in Table 6 shows that there is only a small improvement in the optimized route. Doing a comparison of the 10 per product dataset in optimized sequence from Table 7 and the randomized from Table 6 there is a 7.49% increase by doing the optimization and the time difference being 51.5 minutes.

7.3 Results with Space Constraints

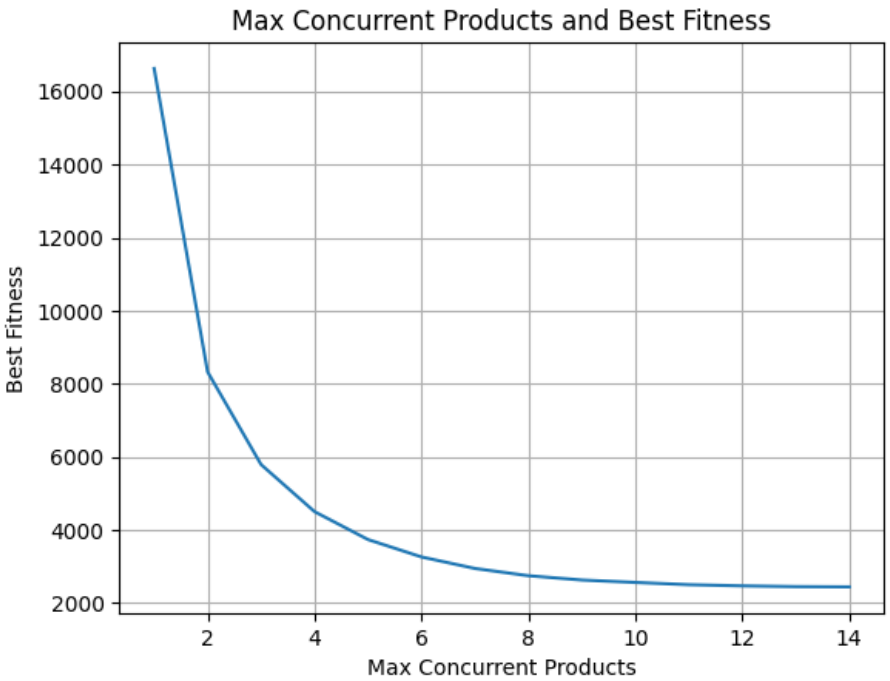

Figure 4: The graph displays the relation between the fitness, represented as minimal amount of time for a sequence and the amount of allowed concurrent products changes. The amount of products per product type during this experiment was 50. The curve shows a decrease in production time as the number of concurrent products increases. This indicates an improved production line efficiency with higher concurrency, up until a saturation point where the amount of concurrent products does not affect the production time significantly.

In the experiment with space constraints in production line optimization, one aspect explored was the impact of varying amounts of concurrent products allowed in the production line. Figure 4 shows the relationship between the production time and the maximum amount of products processed concurrently. As shown by the curve, there is a downward trend in the processing time as the number of concurrent products increases. This suggests that allowing more products to be processed concurrently has an impact on the Overall Equipment Effectiveness (OEE). This is to a point where additional increases in concurrent products have less impact than in the beginning.

Different datasets (20 concurrent products)	**Initial time (minutes)**	**Optimized time (minutes)**	**Difference (minutes)**	**Difference (%)**
10 per product type	666.0	517.5	148.5	22.30%
20 per product type	1536.0	993.5	542.5	35.32%
50 per product type	4146.0	2441.5	1704.5	41.11%
Randomized amount	2231.0	1360.5	870.5	39.01%

Table 8: This table presents the optimization outcomes with a high number of concurrent products active in the production line, allowing 20 concurrent products as a constraint on the different datasets. The table compares initial unoptimized times and optimized times, the difference between them in minutes and also percentage in increased efficiency.

Different datasets (10 concurrent products)	Initial time (minutes)	Optimized time (minutes)	Difference (minutes)	Difference (%)
10 per product type	816.0	533.0	283	34.68%
20 per product type	1686.0	1031.5	654.5	38.82%
50 per product type	4296.0	2562.5	1733.5	40.35%
Randomized amount	2381.0	1407.5	935.5	39.29%

Table 9: Displays the optimization under a moderate number of concurrent products in the production line, allowing 10 concurrent products using different datasets. The table compared between the initial time and the optimized time showing the difference in minutes and a percentage increase in processing efficiency.

Different datasets (5 concurrent products)	Initial time (minutes)	Optimized time (minutes)	Difference (minutes)	Difference (%)
10 per product type	892.0	751.5	140.5	15.75%
20 per product type	1765.5	1495.0	270.5	15.32%
50 per product type	4384.5	3745.5	639	14.57%
Randomized amount	2462.0	2048.5	413.5	16.79%

Table 10: Shows the results having a small number of concurrent products in the production line, allowing 5 at the same time by using different amounts of products. This table compares the difference in time between the unoptimized sequence and the optimized sequence.

Different datasets (1 concurrent product)	Initial time (minutes)	Optimized time (minutes)	Difference (minutes)	Difference (%)
10 per product type	3325.0	3325.0	0	0%
20 per product type	6650.0	6650.0	0	0%
50 per product type	16625.0	16625.0	0	0%
Randomized amount	9105.5	9105.5	0	0%

Table 11: This table consists of the results gotten by using a singular concurrent product throughout the production line. The data is used to show the difference between the other experiments and shows the gradual increase in efficiency when adding concurrent products.

The results using space constraints for the production line sequence optimization gave similar results regarding the percentage increase in efficiency when the amount of products increased. This can be seen in Table 8 and Table 9. Except when the amount of concurrent products decreased, then the efficiency decreased as well, as seen in Table 10. This may indicate that the GA is better at optimizing sequences when there are more variables (products) that can be rearranged, potentially because of the larger set of combinations able to be explored. The reduction in processing times is larger when allowing more concurrent products. The tables and figure 8 shows clearly that when the number of concurrent products decrease (from 20 to 1), the percentage decrease becomes significant. This means that a production line's output could be increased by having a larger amount of products processed simultaneously. Until a certain point when the constraints come closer to the production lines total capacity, where increasing concurrent products gives diminishing returns.

The Table 11. does not show any improvement, which is to be expected since there is only one product that can be in the production line at the same time. Changing the order will not change the time since all has to be processed individually. This does not provide a lot of insight other than proving that the algorithm works correctly when introducing constraints.

No test was done by combining the randomly shuffled sequences with the space constraints. This is because of the lack of time during this study. It should be noted that applying a randomized shuffle would probably impact the total time in a positive way during this scenario as well compared to using the alphabetical sequence. In the scenarios where the product amounts were randomized between 1 and 50 an improvement in processing times could be seen.

7.4 Group Specific Space Constraint

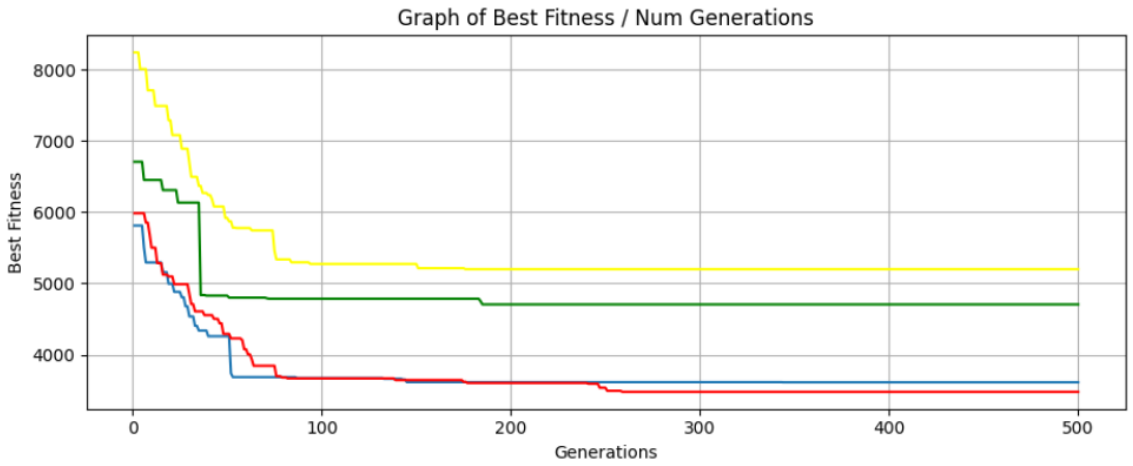

Figure 5: This graph shows the optimization throughout generation 1 to 500 by the different majority fixture groups. The red line represents the red majority group (5 red, 2 yellow, and 2 green fixtures). The yellow line represents the yellow majority group (2 red, 5 yellow, and 2 green fixtures). The green line represents the green majority group (2 red, 2 yellow, and 5 green fixtures). The blue line represents the evenly distributed group (3 fixtures per color group).

Different datasets	**Initial time (minutes)**	**Optimized time (minutes)**	**Difference (minutes)**	**Difference (%)**
Red fixture majority	5985.5	3476.5	2509	41.93%
Yellow fixture majority	8244.5	5200.0	3044.5	36.98%
Green fixture majority	6708.5	4705.0	2003.5	29.86%
Even fixture distribution	5812.5	3611.0	2201.5	37.87%

Table 12: This table summarizes the initial and optimal processing times, along with the percentage improvements. This is based on the different product group majority space constraint datasets. This data demonstrate how varying group specific constraints affect the OEE and throughput of the production line.

The results indicate that group specific space constraints influence the optimization outcomes. For instance as can be seen in Figure 5, the red products take the most time to produce, thus the extra space for the red majority dataset resulted in the fastest optimized sequence as in Table 12. While also resulting in the highest production efficiency increase. It should be noted that simply increasing the amount of available concurrent products for a specific production line path might not increase the production efficiency. If there exists a bottleneck in another path, the overall optimization will be affected negatively by waiting for the bottleneck to clear.

8. Discussion

The results achieved by the GA showed that there are a lot of optimizations that can be done in scheduling tasks, and also that AI can play a big role in this field. The model always achieved a better result than by simply doing an alphabetical order or by randomizing the sequence. Example of an alphabetical sequence:

$$[A, A, B, B, C, C, D, D, E, E, F, F, G, G, H, H, I, I]$$

In most cases the results were significantly lower than the initial starting point and would cut down production time by many hours.

8.1 Dataset Robustness

During the experimental phase we experimented with many different sizes of datasets with extra products and additional product paths to test the robustness of our general solution. The solution showed great success in being able to adapt to different paths and items that were added to the Excel sheets. This was also promising because this meant that we could add transportation times to simulate how much improvement our solution would give to the real world production, where there are actual transportation times. Although possible, our thesis aimed to create a general solution to optimize production scheduling tasks where transportation times didn't need to be considered.

It is important to emphasize the limitations of these datasets. The datasets were constructed with an ideal version of the production environments. They exclude unpredictable variables that may occur in real-world scenarios [58]. Such as malfunctions, changes in labor and supply chain disruptions. Such variables might limit the general usability of our results, as the algorithm's performance in the controlled experimental setup might not translate correctly to real-world environments although this is a common occurrence in planning and scheduling [17]. Regardless of these variables the solution can be used to plan new sequences in case of supply chain disruptions or malfunctions.

8.2 Results

We achieved varying results that were dependent on the varying product amounts as we can see in Table 3. Due to the different product categories having different paths with varying processing times where the *red group* has the longest path to take, they highly dictate how much optimization that can be done. If the majority is of the products that take the longest it is only logical that there won't be that much that can be optimized due to the other products just waiting for the *red group* to be finished. When the products are evenly distributed, the order of when the products are processed play a much larger role and the difference between good and bad solutions become much more clear, this can also be seen in Table 7.

In the first experiments there were no space constraints added, the space was only limited by the actual space in the production line, this showcased an ideal production line where the space was maximized and gave the most optimal results. As that is not a realistic setting for a production line we also experimented by adding space constraints to see how big of a difference it made.

When comparing the results achieved with the GA and the results that methods like Graph Algorithms that previous related work has used the results are very similar. Lity et al [35] received an improvement of 40.1% while our implementation of the GA achieved at most 41.93%. This result can be a bit deceiving because the production lines are not the same with the same constraints, but it still shows the importance of optimizing the sequence. As seen in Table 12 the 41.93% improvement led to an optimization of almost 42 hours in processing times. It is worth to note that just a randomized sequence of the same products yielded an improvement of around 20%. Because of this it can be argued that the actual optimization is only 21.93% since the original alphabetical sequence is unrealistic.

It can also be seen in Table 6 that a randomized sequence will be better than running an alphabetical order of products due to the products *A, B, C* being the products with the longest processing times, which shows the inefficiencies in simplistic sequencing methods.

As previously mentioned, Fawas Alharbi and Qian Wang optimized their assembly line and improved the efficiency from 570 seconds to 500 seconds in just 5 generations [37]. Our GA generally found good solutions in early generations as seen in Figure 3. However, achieving the optimal solution required the GA to run for additional iterations to converge. While an improved solution might be identified in early iterations a true optimal solution often needs further exploration, which shows that the early improvements act as a foundation towards the most efficient solution possible.

Something worth noting is that our solution initially took some time to run on the setup used (with the largest dataset taking about 60 seconds to run). However when tested later on a more powerful GPU and computer it achieved a significantly faster runtime. The solution was originally tested on a computer with a weaker GPU to ensure that it didn't require unreasonable computing power and could be deployed on commodity hardware.

8.3 Constraints

When product specific constraints were introduced a large difference in overall fitness can be seen in Figure 4. This pinpointed the optimal amount of products for simultaneous production in the line. The graph showed the importance of identifying the optimal number of fixtures, especially considering the potentially high cost associated with these fixtures. By adding the space constraints it became evident that having an optimized sequence was important because bad solutions became even worse. For instance, if the production line was jammed with the product categories with the longest production time, all the products with faster production times were waiting and the OEE was lowered.

8.4 Limitations

Throughout this thesis there have been limitations that have been accounted for, the first one being transportation time as mentioned earlier. By adding transportation time to the equation we might get a different result, but as mentioned the thesis tries to showcase the usage of AI in a general way that can be adapted to specific needs and conditions that different production lines may have.

Secondly, the optimization framework is a generalized solution that focuses on versatility and adaptability and might not capture the distinct features of a specific production environment. Although the proposed solution shows great promise in its adaptability and ease of use it needs to be adapted to the specific needs and structure of the production line.

Since the solution has not been tested on a real-life production line the results achieved are an assumption that everything functions correctly and follows the specified rules and constraints. While the constraints were tested on a smaller batch of products in a more controlled environment with only four processes, where the optimal time could be manually calculated and verified. The optimal time of the full system is yet to be verified by running it in an actual production line. Because of these limitations, it is not possible to verify the solutions actual usability and impact in a real-world manufacturing environment. But time discrepancies between planning and reality is a common issue in production planning and not specific for this thesis, as mentioned in section 1.2.

9. Conclusion

This thesis has experimented with different types of GAs to try and find answers to two research questions. Our first research question was (1) *Can Genetic Algorithms be used to optimize a sequence of products in a production line?* The answer to this research question is that GAs can be used to optimize the sequence, achieving the primary objective of the thesis. But because we did not have any benchmark to compare our optimized results with, we can't know for sure that GAs are the best way to solve this problem. By continuing to explore different AI methods to solve these types of optimization problems we can find what solutions that work best for the different types of problems. Now that there also is a baseline of what the GA achieved, the other AI algorithms can now be compared to these results to see if they improve the efficiency.

The second research question was (2) *How effectively can Genetic Algorithms optimize the sequencing of production tasks in diverse production lines to minimize total order processing times?*. The answer to the research question is that GAs can reduce the total time taken up to almost 42% for the best case, although it is important to note that the improvement is dependent on the amount of different products that will be processed. This improvement is in line with the third objective of minimizing order processing time. This shows that AI can be used in this type of optimization task. But also that with the right representation of the data there are no issues with expanding the amount of products, different processing steps, processing times and capacities in the actual production line. This was tested by adding fictional products with different processing times, amounts and paths and resulted in no issues. By making the implementation adaptable, the second objective is also achieved.

An important consideration is that the proposed solution has not actually been tested in a real world environment. It has only been tested and evaluated in simulations that have perfect conditions without any form of disruptions such as malfunctions, human error, transportation times or any other factors that might impact the algorithm's performance. With that being said, this thesis has highlighted the importance of the representation of the data. For this algorithm to have any real world use, it needs to be easily adapted to the specifications of the production- or manufacturing line. Besides the representation of data, the solution should also have a well designed UI that is user friendly and can be incorporated in existing systems for the proposed method to be feasible. While the bonus objective of developing a UI was not prioritized during this study, its importance for accessibility is acknowledged.

10. Future Works

The study has shown a lot of promises in the field of planning and scheduling with the usage of AI. While this thesis has had primary focus on the application of GAs, there are many more AI methods that could be researched to potentially find even more optimal solutions. Future research in this field could involve the investigation of alternative methods to complete planning and scheduling tasks such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Hill Climb Search, Reinforcement Learning (RL). These are different AI algorithms that could show potential in enhancing the scheduling efficiency. By comparing the results achieved from the other AI methodologies with the results achieved by the GA, benchmarks can be established and algorithms that perform well can be identified. Which would help in the selection of the most suitable approach for different types of scheduling tasks. A major factor to consider is to actually test the achieved solution in an actual production environment to verify its accuracy compared to the results achieved in the simulation.

In addition to exploring different algorithms there might be even more effective ways of representing the scheduling data which could lead to enhanced effectiveness potentially improving the performance of the scheduling algorithms. If the data is represented in a different way it could be possible to go for a hybrid approach, combining different AI techniques. By using the strengths of multiple AI algorithms the hybrid models may be superior in more complex scheduling tasks.

Lastly, as previously mentioned there should also be focus on a user friendly interface so that the AI based scheduling solutions can easily be incorporated in existing systems and workflows. This could be that the user can input the scheduling parameters, visualize the results or output results in a format that is needed. These things will increase the usability and acceptance in real world contexts.

References

[1] J. D. Rodríguez-García, J. Moreno-León, M. Román-González and G. Robles, 'Introducing artificial intelligence fundamentals with learningml: Artificial intelligence made easy,' in *Eighth International Conference on Technological Ecosystems for Enhancing Multiculturality*, ser. TEEM'20, Salamanca, Spain: Association for Computing Machinery, 2021, pp. 18–20, ISBN: 9781450388504. DOI: 10.1145/3434780.3436705. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/3434780.3436705.

[2] J. Bughin, E. Hazan, P. Sree Ramaswamy, W. DC, M. Chu *et al.*, 'Artificial intelligence the next digital frontier,' 2017.

[3] M. Zhang, 'Artificial intelligence and application in finance,' ser. IC4E '20, Osaka, Japan: Association for Computing Machinery, 2020, pp. 317–322, ISBN: 9781450372947. DOI: 10.1145/3377571.3379441. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/3377571.3379441.

[4] D. Comaniciu, 'Artificial intelligence for healthcare,' Virtual Event, CA, USA: Association for Computing Machinery, 2020, p. 3603, ISBN: 9781450379984. DOI: 10.1145/3394486.3409551. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/3394486.3409551.

[5] D. Lee and S. N. Yoon, 'Application of artificial intelligence-based technologies in the healthcare industry: Opportunities and challenges,' *International Journal of Environmental Research and Public Health*, vol. 18, no. 1, 2021, ISSN: 1660-4601. DOI: 10.3390/ijerph18010271. [Online]. Available: https://www.mdpi.com/1660-4601/18/1/271.

[6] C. Mühlroth, 'Artificial intelligence as innovation accelerator,' New York, NY, USA: Association for Computing Machinery, 2020, ISBN: 9781450371308. DOI: 10.1145/3378539.3393869. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/3378539.3393869.

[7] J. M. Anderson, S. P. Sithungu and E. M. Ehlers, 'Route optimization for sailing vessels using artificial intelligence techniques,' ser. CIIS '22, Quzhou, China: Association for Computing Machinery, 2023, pp. 60–66, ISBN: 9781450397612. DOI: 10.1145/3581792.3581803. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/3581792.3581803.

[8] X. Du, 'Research on the artificial intelligence applied in logistics warehousing,' ser. AIAM2020, Manchester, United Kingdom: Association for Computing Machinery, 2020, pp. 140–144, ISBN: 9781450375535. DOI: 10.1145/3421766.3421798. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/3421766.3421798.

[9] I. Lee, 'Artificial intelligence search methods for multi-machine two-stage scheduling,' in *Proceedings of the 1999 ACM Symposium on Applied Computing*, ser. SAC '99, San Antonio, Texas, USA: Association for Computing Machinery, 1999, pp. 31–35, ISBN: 1581130864. DOI: 10.1145/298151.298174. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/298151.298174.

[10] M. Del Gallo, G. Mazzuto, F. E. Ciarapica and M. Bevilacqua, 'Artificial intelligence to solve production scheduling problems in real industrial settings: Systematic literature review,' *Electronics*, vol. 12, no. 23, 2023, ISSN: 2079-9292. DOI: 10.3390/electronics12234732. [Online]. Available: https://www.mdpi.com/2079-9292/12/23/4732.

[11] *Visual Components - 3D manufacturing simulation software — visualcomponents.com*, https://www.visualcomponents.com/, [Accessed 15-04-2024].

[12] G. Immerman, *Production and Process Optimization in Manufacturing | MachineMetrics — machinemetrics.com*, https://www.machinemetrics.com/blog/process-optimization-manufacturing, [Accessed 17-05-2024].

[13] X. Li, G. Cong, A. Sun and Y. Cheng, 'Learning travel time distributions with deep generative model,' New York, NY, USA: Association for Computing Machinery, 2019, ISBN: 9781450366748. DOI: 10.1145/3308558.3313418. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/3308558.3313418.

[14] L. Dixon, *An Ultimate Guide to Production Line Efficiency Improvement — news.lineview.com*, https://news.lineview.com/an-ultimate-guide-to-production-line-efficiency-improvement, [Accessed 17-05-2024].

[15] G. Immerman, *How to Reduce Machine Downtime in Manufacturing - MachineMetrics — machinemetrics.com*, https://www.machinemetrics.com/blog/reduce-downtime-manufacturing, [Accessed 17-05-2024].

[16] E. Nobil, L. E. Cárdenas-Barrón, D. Garza-Núñez *et al.*, 'Machine downtime effect on the warm-up period in an economic production quantity problem,' *Mathematics*, vol. 11, no. 7, 2023, ISSN: 2227-7390. DOI: 10.3390/math11071740. [Online]. Available: https://www.mdpi.com/2227-7390/11/7/1740.

[17] P. Almström and M. Winroth, 'Why is there a mismatch between operation times in the planning systems and the times in reality,' *Proceedings of APMS2010*, 2010.

[18] R. Miñón, L. Moreno, P. Martínez and J. Abascal, 'An approach to the integration of accessibility requirements into a user interface development method,' *Science of Computer Programming*, vol. 86, pp. 58–73, 2014.

[19] F. Gerges, G. Zouein and D. Azar, 'Genetic algorithms with local optima handling to solve sudoku puzzles,' *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, pp. 19–22, 2018. DOI: 10.1145/3194452.3194463. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/3194452.3194463.

[20] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg, 2015, p. 30, ISBN: 9783662448748. DOI: 10.1007/978-3-662-44874-8. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-44874-8.

[21] Y. Pyrih, M. Klymash, M. Kaidan and B. Strykhalvuk, 'Investigating the efficiency of tournament selection operator in genetic algorithm for solving tsp,' in *2023 IEEE 5th International Conference on Advanced Information and Communication Technologies (AICT)*, 2023, pp. 170–173. DOI: 10.1109/AICT61584.2023.10452423.

[22] A. Bala and A. K. Sharma, 'A comparative study of modified crossover operators,' in *2015 Third International Conference on Image Information Processing (ICIIP)*, 2015, pp. 281–284. DOI: 10.1109/ICIIP.2015.7414781.

[23] A. Basak, 'A rank based adaptive mutation in genetic algorithm,' *International Journal of Computer Applications*, vol. 175, no. 10, pp. 49–55, Aug. 2020, ISSN: 0975-8887. DOI: 10.5120/ijca2020920572. [Online]. Available: http://dx.doi.org/10.5120/ijca2020920572.

[24] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg, 2015, p. 89, ISBN: 9783662448748. DOI: 10.1007/978-3-662-44874-8. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-44874-8.

[25] G. Immerman, *Production and Process Optimization in Manufacturing | MachineMetrics — machinemetrics.com*, https://www.machinemetrics.com/blog/process-optimization-manufacturing, [Accessed 17-05-2024].

[26] A. Salah, D. Çağlar and K. Zoubi, 'The impact of production and operations management practices in improving organizational performance: The mediating role of supply chain integration,' *Sustainability*, vol. 15, no. 20, 2023, ISSN: 2071-1050. DOI: 10.3390/su152015140. [Online]. Available: https://www.mdpi.com/2071-1050/15/20/15140.

[27] C. C. Okpala and E. O. C., 'The design and need for jigs and fixtures in manufacturing,' *Science Research*, vol. 3, no. 4, pp. 213–219, 2015. DOI: 10.11648/j.sr.20150304.19. eprint: https://article.sciencepublishinggroup.com/pdf/10.11648.j.sr.20150304.19. [Online]. Available: https://doi.org/10.11648/j.sr.20150304.19.

[28] B. Stupalo, *Assembly Line Fixturing - MAGNUM Automation Inc. — magnum-inc.com*, https://www.magnum-inc.com/products/tooling-fixturing/assembly-line-fixturing/, [Accessed 17-05-2024].

[29] *Fixturing for Robotic Welding — lincolnelectric.com*, https://www.lincolnelectric.com/en/Welding-and-Cutting-Resource-Center/Process-and-Theory/Fixturing-for-Robotic-Welding, [Accessed 21-05-2024].

[30] S. Mattsson, P. Gullander and A. Davidsson, 'Method for measuring production complexity,' in *28th international manufacturing conference*, 2011.

[31] G. Schuh, S. Rudolf, M. Riesener, C. Dölle and S. Schloesser, 'Product production complexity research: Developments and opportunities,' *Procedia CIRP*, vol. 60, pp. 344–349, 2017, Complex Systems Engineering and Development Proceedings of the 27th CIRP Design Conference Cranfield University, UK 10th – 12th May 2017, ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2017.01.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827117300070.

[32] J. Quevedo, M. Abdelatti, F. Imani and M. Sodhi, 'Using reinforcement learning for tuning genetic algorithms,' in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '21, Lille, France: Association for Computing Machinery, 2021, pp. 1503–1507, ISBN: 9781450383516. DOI: 10.1145/3449726.3463203. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/3449726.3463203.

[33] B. L. Maccarthy and J. Liu, 'Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling,' *The International Journal of Production Research*, vol. 31, no. 1, pp. 59–79, 1993.

[34] A. Giret, D. Trentesaux and V. Prabhu, 'Sustainability in manufacturing operations scheduling: A state of the art review,' *Journal of Manufacturing Systems*, vol. 37, pp. 126–140, 2015, ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2015.08.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612515000606.

[35] S. Lity, M. Al-Hajjaji, T. Thüm and I. Schaefer, 'Optimizing product orders using graph algorithms for improving incremental product-line analysis,' in *Proceedings of the 11th International Workshop on Variability Modelling of Software-Intensive Systems*, ser. VaMoS '17, Eindhoven, Netherlands: Association for Computing Machinery, 2017, pp. 60–67, ISBN: 9781450348119. DOI: 10.1145/3023956.3023961. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/3023956.3023961.

[36] U. Junior Mele, L. Maria Gambardella and R. Montemanni, 'Machine learning approaches for the traveling salesman problem: A survey,' ser. ICIEA 2021-Europe, Barcelona, Spain: Association for Computing Machinery, 2021, pp. 182–186, ISBN: 9781450389921. DOI: 10.1145/3463858.3463869. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/3463858.3463869.

[37] F. Alharbi and Q. Wang, 'Solving an assembly sequence optimisation problem using the genetic algorithm,' in *2018 International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*, 2018, pp. 1–5. DOI: 10.1109/ICECOCS.2018.8610519.

[38] H. A. Bukhori, T. Widyatmoko, S. Sunarti *et al.*, 'Optimizing tabu list in tabu search algorithm for efficient and optimal scheduling,' in *2023 Sixth International Conference on Vocational Education and Electrical Engineering (ICVEE)*, 2023, pp. 73–76. DOI: 10.1109/ICVEE59738.2023.10348342.

[39] *What is TABU Search? - GeeksforGeeks — geeksforgeeks.org*, https://www.geeksforgeeks.org/what-is-tabu-search/, [Accessed 17-05-2024].

[40] S. Nahar, S. Sahni and E. Shragowitz, 'Simulated annealing and combinatorial optimization,' in *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, ser. DAC '86, Las Vegas, Nevada, USA: IEEE Press, 1986, pp. 293–299, ISBN: 0818607025.

[41] Microsoft Corporation, *Microsoft excel*, version 2019 (16.0), 24th Sep. 2018. [Online]. Available: https://office.microsoft.com/excel.

[42] C. Bierwirth and D. C. Mattfeld, 'Production scheduling and rescheduling with genetic algorithms,' vol. 7, no. 1, 1999, ISSN: 1063-6560. DOI: 10.1162/evco.1999.7.1.1. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1162/evco.1999.7.1.1.

[43] S. Al-hayali, O. Ucan and O. Bayat, 'Genetic algorithm for finding shortest paths problem,' in *Proceedings of the Fourth International Conference on Engineering & MIS 2018*, ser. ICEMIS '18, Istanbul, Turkey: Association for Computing Machinery, 2018, ISBN: 9781450363921. DOI: 10.1145/3234698.3234725. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/3234698.3234725.

[44] C. W. Ahn, S. Oh and R. S. Ramakrishna, 'On the practical genetic algorithms,' in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '05, Washington DC, USA: Association for Computing Machinery, 2005, pp. 1583–1584, ISBN: 1595930108. DOI: 10.1145/1068009.1068270. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/1068009.1068270.

[45] U. Ehsan, P. Wintersberger, Q. V. Liao *et al.*, 'Human-centered explainable ai (hcxai): Beyond opening the black-box of ai,' in *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA: Association for Computing Machinery, 2022, ISBN: 9781450391566. DOI: 10.1145/3491101.3503727. [Online]. Available: https://doi.org/10.1145/3491101.3503727.

[46] D. Rivas, R. Quiza, M. Rivas and R. E. Haber, 'Towards sustainability of manufacturing processes by multiobjective optimization: A case study on a submerged arc welding process,' *IEEE Access*, vol. 8, pp. 212 904–212 916, 2020. DOI: 10.1109/ACCESS.2020.3040196.

[47] A. A. Bruzzone, D. Anghinolfi, M. Paolucci and F. Tonelli, 'Energy-aware scheduling for improving manufacturing process sustainability: A mathematical model for flexible flow shops,' *CIRP annals*, vol. 61, no. 1, pp. 459–462, 2012.

[48] A. Lira-Aquino, E. Miranda-Poccori, E. Altamirano-Flores and L. Cardenas-Rengifo, 'Improving production process efficiencies at a peruvian company through a lean manufacturing implementation model,' in *Proceedings of the 7th International Conference on Industrial and Business Engineering*, ser. ICIBE '21, Macau, China: Association for Computing Machinery, 2022, pp. 117–122, ISBN: 9781450390644. DOI: 10.1145/3494583.3494631. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/3494583.3494631.

[49] J. Wang, H. Xu, W. Wu *et al.*, 'Optimization of curtain wall production line balance based on improved genetic algorithm,' *Mathematics*, vol. 11, no. 21, 2023, ISSN: 2227-7390. DOI: 10.3390/math11214433. [Online]. Available: https://www.mdpi.com/2227-7390/11/21/4433.

[50] M. W. Allen, D. J. Armstrong, M. F. Reid and C. K. Riemenschneider, 'It employee retention: Employee expectations and workplace environments,' in *Proceedings of the Special Interest Group on Management Information System's 47th Annual Conference on Computer Personnel Research*, ser. SIGMIS CPR '09, Limerick, Ireland: Association for Computing Machinery, 2009, pp. 95–100, ISBN: 9781605584270. DOI: 10.1145/1542130.1542148. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/1542130.1542148.

[51] Microsoft, *Visual studio code*, Accessed: 2024-05-21, 2024. [Online]. Available: https://code.visualstudio.com/.

[52] Python Software Foundation, *Python 3.10*, Accessed: 2024-05-21, 2021. [Online]. Available: https://www.python.org/downloads/release/python-3100/.

[53] W. McKinney, 'Data Structures for Statistical Computing in Python,' in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[54] *RANDBETWEEN function - Microsoft Support — support.microsoft.com*, https://support.microsoft.com/en-us/office/randbetween-function-4cc7f0d1-87dc-4eb7-987f-a469ab381685, [Accessed 17-05-2024].

[55] J. L. Wilkerson and D. R. Tauritz, 'A guide for fitness function design,' in *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, ser. GECCO '11, Dublin, Ireland: Association for Computing Machinery, 2011, pp. 123–124, ISBN: 9781450306904. DOI: 10.1145/2001858.2001929. [Online]. Available: https://doi-org.ep.bib.mdh.se/10.1145/2001858.2001929.

[56] K. Singh and A. S. Pillai, 'Schedule length optimization by elite-genetic algorithm using rank based selection for multiprocessor systems,' in *2014 International Conference on Embedded Systems (ICES)*, 2014, pp. 86–911. DOI: 10.1109/EmbeddedSys.2014.6953096.

[57] S. Tian, J. Zhang, L. Chen, H. Liu and Y. Wang, 'Random sampling-arithmetic mean: A simple method of meteorological data quality control based on random observation thought,' *IEEE Access*, vol. 8, pp. 226 999–227 013, 2020. DOI: 10.1109/ACCESS.2020.3045434.

[58] K. Efthymiou, A. Pagoropoulos, N. Papakostas, D. Mourtzis and G. Chryssolouris, 'Manufacturing systems complexity: An assessment of manufacturing performance indicators unpredictability,' *CIRP Journal of Manufacturing Science and Technology*, vol. 7, no. 4, pp. 324–334, 2014.