TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

SAHAND FEIZIAZAR

GENETIC ALGORITHMS FOR FLOW-SHOP SCHEDULING OPTIMIZATION OF AN AUTOMATED ASSEMBLY LINE

Master of Science thesis

Examiner: Prof. José L. Martínez Lastra
Examiner and topic approved by the Faculty Council of the Faculty of Engineering Sciences
on 6th May 2015

ABSTRACT

Feiziazar, Sahand: Genetic Algorithms for Flow-shop Scheduling Optimization of an Automated Assembly Line
Tampere University of technology
Master of Science Thesis, 70 pages, 5 Appendix pages
May 2015
Master's Degree Programme in Machine Automation
Major: Factory Automation
Examiner: Professor Dr. Josè L. Martinez Lastra
Supervisor: Dr. Andrei Lobov
Keywords: Genetic Algorithm, assembly line, simulation, optimization, artificial intelligence, machine learning

Manufacturing process is a process of producing and creating a product with the use of technologies and machinery resources. In manufacturing process there are three dimensions, which are important in improving the system. These are cost, quality, and speed that can be considered as basics of every process. In this thesis speed of the manufacturing process is enhanced, which leads to reduction in cost as well.

Assembly lines are the part of manufacturing process to convert raw materials into finished products. Considering optimization problems in assembly lines, applying genetic algorithms to the established model could lead to efficient manufacturing. Genetic algorithm is a programming search technique for maximizing productivity, minimizing inefficiency and reducing production time.

This work presents an approach for developing simulation models used for optimization of production lines. The results are demonstrated using the assembly line which is located in FAST-Lab. at Tampere University of Technology.

The simulation of the line is created to assess cycle times and utilization of workstations using MATLAB and SimEvents library. The optimization, in the context of presented work, is the process of locating and scheduling the products in the line achieving best timing to fulfil production orders. The workstations can be first balanced for better performance and then products are scheduled based on reduction of the production time.

PREFACE

This thesis is made in Factory Automation Systems and Technology (FASTory) laboratory at Tampere University of Technology (TUT). The purpose of this thesis is to simulate, balance and optimize the automated assembly line for faster and better production process.

I would like to thank all my teachers, here in TUT for their help in this path to this master degree. I would like to thank the head of the department, Prof. Dr. Jose Luis Martinez Lastra for providing this opportunity for me to study in this university. Then I want to thank my supervisor, Dr. Andrei Lobov, for his help and advises in this thesis work.

Finally I want to thank the best parents in the world, my father, Hasan Feiziazar, and my mother, Shahnaz Ardi. Thank you for your patience, encouragement, and support during my studies in Finland. And thanks to my sisters for their love and care from distance.

Tampere, September, 2015

Sahand Feiziazar

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ABC	Artificial Bee Colony algorithm
ACO	Ant Colony Optimization algorithm
AI	Artificial Intelligence
AIS	Artificial Immune System
ANN	Artificial Neural Network
BFO	Bacteria Foraging Optimization algorithm
CC license	Creative Commons license
COA	Chaotic Optimization Algorithm
CRO	Coral Reef Optimization algorithm
CS	Cuckoo Search algorithm
DE	Differential Evolution algorithm
DEDS	Discrete Event Dynamic System
EA	Evolutionary Algorithm
EAs	Evolution based Algorithm
EP	Evolutionary Programming
ERP	Enterprise Resource Planning
ES	Evolutionary Strategy
FA	Firefly Algorithm
GA	Genetic Algorithm
GA	Genetic Algorithm
GPS	General Problem Solving
GSA	Gravitational Search Algorithm
GUI	Graphical User Interface
HS	Harmony Search algorithm
ICA	Imperialistic Competition Algorithm
IWD	Intelligent Water Drops algorithm
KNN	K-Nearest Neighbours
MDP	Markov Decision Process
ML	Machine Learning
MOA	Magnetic Optimization Algorithm
MPM	Manufacturing Process Management
OM	Operation Management
PCA	Principal Component Analysis
PIO	Pigeon Inspired Optimization
PSO	Particle Swarm Optimization
RWS	Roulette Wheel Selection
SA	Simulated Annealing
SFLA	Shuffled Frog Learning Algorithm
SM	Scientific Management
SOM	Self-Organizing Map
SVM	Support Vector Machines
TLBO	Teaching-Learning Based Optimization algorithm
TSA	Tabu Search Algorithm

1. INTRODUCTION

The purpose of this chapter is to expose a background on the thesis work and the definitions that have been used to help the reader understand the objectives, tasks, and the results of this Master of Science thesis better and give a proper foundation for future discussions and works.

Section 1.1 constitutes a background for the thesis topic and gives a first overview to where the thesis topic originates, section 1.2 defines the thesis problem and justifies the work, and section 1.3 describes the work that was done in the thesis and the introduction to objectives and methodologies. Finally section 1.4 outlines the structure of the manuscript.

1.1. Background

Automation aims at applying the machines to perform the tasks that were originally performed by human beings for helping them in dangerous environments such as, space, volcanoes, underwater, and fire. One of the important areas in automation technology is manufacturing. Manufacturing process is growing in speed due to increasing demand on fast delivery of components, technologies, plants and facilities. Manufacturing processes are applicable in all areas of individual lives that makes people ignore or not to think of them. All the basic consumer goods many humans encounter each days, e.g. from TVs, computers to the cars been driven and the factories for production, are in the area of manufacturing process. The most important goals in each manufacturing process are to satisfy the performance requirements, and to decrease the cost of production and meeting the deadlines according to the customers demand[1].

For couple of years developments in computer science have had an important influence on automation technology, and artificial intelligence is one of the newest fields in this area. With spending more times for implementing different algorithms for manufacturing systems, the engineers invent new methods and algorithms to use in industry. One of the algorithms used in factories and production lines is genetic algorithm, which was inspired by natural selection and genetics.

The genetic algorithm is a programming search technique for maximizing productivity, minimizing inefficiency and reducing production time. In manufacturing, reducing time and increasing efficiency can be complex, because there are multiple inputs and steps. For solving the production scheduling problems in factory floor, genetic algorithms are well suited. Chromosomes are the representation form of genetic algorithm in scheduling problems. Each chromosome is a solution for the problem, ranked based on their fitness

function. Finding the best chromosome with the best fitness function is the process of optimization in manufacturing line.

1.2. Problem Definition

1.2.1. Justification of work

Simulation and optimization in manufacturing process is one of the most important parts in automation at the factory floor. Simulating the production line for data acquisition makes the optimization easier, as the line works on the basis of unreal timing which makes the process faster.

Therefore there is a need for analysing the algorithms, which are well suited on the specific automated line, and choosing the best for easy and fast implementation of the system. This implementation helps the user to understand the basics of optimization and find the suitable algorithms in this field. Implementing optimization algorithms on the assembly line helps the line for fast production and customer satisfaction.

1.2.2. Problem Statement

The problem statement for this thesis work can be formulated as follows:

"*How to simulate, balance, and optimize an assembly line for faster and better production process. By simulating the line with the MATLAB and then applying the Genetic Algorithm for optimization to the line, the completion time for the production process can be decreased by 10% to 12%.*"

1.3. Work Description

1.3.1. Objectives

The main objective of this thesis is to create an optimization algorithm to reduce the production cycle time. To make this happen, automated assembly line is simulated and balanced, and then different algorithms in artificial intelligence field were studied for finding the best method. After gathering information about machine learning methods and choosing genetic algorithms, which is in the field of reinforcement learning, the algorithm for the production line was implemented. There are some steps in genetic algorithms that lead to the best implementation for this work with good efficiency and results which can give 10% to 12% reduction in production time.

1.3.2. Methodology

In order to achieve the best implementation, following steps are taken:

1. Study of literature in domain of manufacturing process, and learning the different job shop problems.
2. Review of the programming languages for simulating the assembly line, and selecting MATLAB for this thesis work.
3. Simulating the assembly line with workstations and ordering system which is user friendly, with MATLAB.
4. Balancing the line for high utilization of workstations with equal distribution of parameters.
5. Study of literature in domain of artificial intelligence and machine learning, for understanding the use of optimization algorithms in manufacturing process.
6. Implementing genetic algorithm for optimization of automated assembly line and reducing the cycle time.
7. Evaluating and comparing the performance.

1.4. Thesis Outline

The rest of the thesis is structured as follows: Chapter 2 gives the theoretical background of factory physics, simulation and optimization with artificial intelligence algorithms. Chapter 3 explains the approach of the thesis and the techniques and tools used for implementing the algorithms. Chapter 4 includes the implementation of assembly line, simulation and optimization of genetic algorithms on simulated line. Chapter 5 presents the results of implementation part. Chapter 6 provides the conclusion and future work.

2. THEORETICAL BACKGROUND

2.1. Principle of Manufacturing Systems

Factory physics describes the behaviour of the manufacturing systems. Understanding factory physics enables the engineers and managers to identify the chances to improve systems and design more effective systems[2]. In the manufacturing systems three dimensions are important:

Cost: For reducing the cost in manufacturing process, the utilization of labour, material, and equipment should be efficient.

Quality: To keep the manufactured products competitive with other companies, the quality of the product should be high.

Speed: The speed of manufacturing process is always an important factor, and can affect the other two dimensions, that is why the managers and engineers are working on this dimension for improving the process in the way that the factory can compete with others in the final product.

The other characteristics which manufacturing systems possess are, modularity, integrability, customized flexibility, scalability, convertibility, and diagnosability. These characteristics should be applied for designing a manufacturing systems[2] [3].

2.1.1. Manufacturing Process

Manufacturing environments vary according to their process structure. The flow line of a manufacturing process can be categorized in four different categories in the manner of which the material moves through the plant. Job shops, disconnected flow lines, connected flow lines and continues flow processes are the four categories. Job shops are the structure, which has high variety of routings and the small lots, can choose the routings to get the process done. An example of this process structure is for commercial printers, where each job has unique requirements. The majority of manufacturing systems in industry resembles the category of disconnected flow lines. In this system the product is manufactured on a limited number of routings and the stations are not connected by the paced material handling system. The difference between the connected flow lines and disconnected ones is that the connected lines are paced according to the material handling systems. In the last category which is continues flow process, the product flows in the fixed routing system[2]. Manufacturing process concerns about the changes in dimensions of the product and it does not include the transportation, handling and storage of the product[4].

2.1.2. Manufacturing Management

Manufacturing process management (MPM) is different from Enterprise Resource Planning (ERP) and is the collection of technologies and methods which is used to define the products to be produced. MPM is playing the key role in the integration of the tools and activities which leads to reducing the production time in assembly lines and reducing the work in process and allowing the system response faster to product changes.

Scientific management (SM) is the basic of operation management (OM) and made it possible. Various OM areas are inventory control, scheduling which is used in this thesis, capacity planning, forecasting, equipment maintenance and quality control. Inventory control is one of the operation management's sub disciplines, which is spawned mathematical models to factory management. The inventory control models are one of the oldest results of OM field and are still used and cited widely. Inventory plays the main role in logistical behaviour of manufacturing systems, and the classical inventory models are the basic of more modern manufacturing systems[2].

2.1.3. Production Process

Production process which is called scheduling as well is the process of arranging, controlling and optimizing the work and workloads. Scheduling is the science of allocating the plant and machinery resources. In manufacturing processes scheduling has the key role to minimize the production time and cost, by arranging and controlling the facilities.

In the production process the raw materials and semi-finished products are converting to the finished products. Production is the art of converting raw and un-finished materials into finished products with applying of tools, equipment and manufacturing processes. There are three main types of production systems, which are: Job production, Batch production and Mass production. In job production each operator works on a single job and it cannot proceed before finishing the current operation. The job production requires fixed type of layout for developing products and the production requirement is low. Batch production is the manufacturing of the products with similar parts and small variation in size and shape. Functional and process layout is need for this kind of production. Mass production is the production of large amount of products, and it requires line layout which is highly rigid and involves automation and big amount of investment to increase the production[4].

Process planning is the selection of production machines and tools, finding the efficient sequence for operation and calculation of the machining time which will lead to minimize material handling and will ensure the reduction in cost and in enlarging the productivity[4].

After knowing about process planning, process characterization will come up, which is an activity to find the inputs and outputs of the process and collect the data on their behaviour in the operation. Estimating steady state of the conditions and building models according to the relationships are the steps for characterizing the process. These would help to monitor the production process and improve it with the mathematical models. Production process is the three step activity, screening step, mapping step and finally passive step. The first step which is screening step begins with identifying all the inputs and outputs and after conducting screening experiments the key inputs and outputs will be selected. The experiments also help us to understand and model the relationships between the inputs and outputs. Mapping step is about mapping the behaviour of selected inputs and outputs over their operations. The final step will show how the model is running and showing the process stability and capability[5].

Production systems has been an important design problem in industry and it began to become more important after manufacturing technologies has progressed. They are known by their cycle times, level of automation and modern production lines. These characteristics created more problems and needs in designing production lines. The design of the production lines are in relation with machines existence and manufacturing equipment. By selection of the pieces and balancing of workstations and dimensioning storage areas and transportation systems, the production lines can be modelled[6].

2.1.4. Assembly line

Assembly line is a system consisted of complex disperse events which considers time sequence relation and parallel and competitive relations. The relationship among working procedures are determined by assembly process which has used the product assembly techniques. Assembly line is a typical Discrete Event Dynamic System (DEDS) in the domain of manufacturing[7].

Nowadays assembly lines are playing important role in manufacturing, and specially the part of electronic products. Considering optimization problems, establishing a model and applying genetic algorithms could help to acquire efficient operation planning. The assembly line planning process consists of wide range of optimization problem, such as line body balancing, scheduling problems, and optimization of operating strategy[8].

Designing an assembly line is complicated as the sequence of operation systems or workstations can influence the planning of the product, which should be completed by the end of the sequence. Static planning of an assembly line consists of analysis of products, planning of sub-assembly sequence, planning the layout and process. The process is based on assembly order, operations time, which should confirm the sequence and distribute the operations to work floors to make the working hours of the task equal to balance the work as well. After defining the logistic relationship between assembly operations, equipment and tools, the main task would be to manage the furniture of all kinds of equipment and

tools to use the limited space effectively and reduce the costs[8]. The important goal of the assembly lines designers is to improve the efficiency of the line by increasing throughput of the workstations. Assembly line's performance determines the final products and delivery time. Therefore the way of designing the line will control and improve the efficiency and quality[6].

Being flow oriented production systems, assembly lines consists of some workstations, with conveyors for connecting the stations, or some other equipment for handling the materials in the system. The jobs which are the work pieces are moving along the conveyors between the workstations and each station operating specific function according to the cycle time. There are different types of assembly line based on what the line is operation on the work pieces. For example paced assembly lines, has the fixed production rate and there is not any buffer for checking the pieces. The other type of line is buffered assembly line, each work station has a stop or wait before it for checking the piece and the next station. Single mode lines are the lines with one assembled product and mixed model lines are the lines with different models for the products[9]. In the next chapter there will be more description about the assembly line which is simulated in this thesis.

2.2. Simulation

Simulation is the tool for analysing complex processes or systems, and can help to design, plan and control the real systems without running them. It can be sometimes costly and time consuming to run a system and acquire the outputs. Simulation is a way to model the systems and mimic the response of the actual system over time. According to (Shannon 1992), "Simulation is the process of designing a model of a real system and conducting experiments with the model for the purpose of either understanding the behaviour of the system and/or evaluating various strategies for the operation of the system." For studying the problem of a system, simulation will be considered to express the construction and experimental use of the model. In this thesis simulation has used for describing the behaviour of the system and extracting some theories and hypothesis from the observed behaviour and then predicting the future behaviour which can be studied by changing the system and inputs[10].

Simulation can be used in different cases, such as simulating the production line and showing how the products are moving on the conveyors or simulation of the physical systems such as a ship's flow on the water. In computer systems such as hardware components and software systems, are the applications of simulation. In manufacturing systems, the material handling systems, and inventory control systems can be a good representative of simulation. And a lot more in business, government, ecology, and even environmental situations the track of simulation can be seen[10], [11].

Simulation is input-output based and incapable of generating the optimal solution by its own. Using simulation has some advantages and disadvantages, considering these, engineers and designers can use it in the system they want to model. Simulation is an extending tool and it cannot interrupt the system when it is running, therefore, less energy and resources are needed for the process. It is a good testing tool as well, for evaluating the system before committing into the real world and the theories can be tested for feasibility and error diagnosis. Controlling time is the other benefit of simulating the system, which can help the designer to run the model in shorter time and get the same response. So in this case simulation makes the monitoring of the system faster, hence the engineer or tester can find the errors faster and solve them even without paying loads of money for doing the same on the actual system. All in all simulation helps to acquire a good insight of the system and experiment some changes and improvements without taking the experiment into the real commitment[10].

Each simulation can have some drawbacks despite the benefits it has. Every modelling requires some knowledge, which the modeller should have for a good quality model and analysis. Sometimes it is hard or time consuming to interpret the results although the information has been gathered from the production process[10]. Every simulation has its own process but there is a general process for almost all of the simulations, which is applicable for them. They begin with problem definition that helps to understand the steps of simulation and end with documentation, which is the part that puts the results to use in real systems. The figure below is showing an example of simulation process:

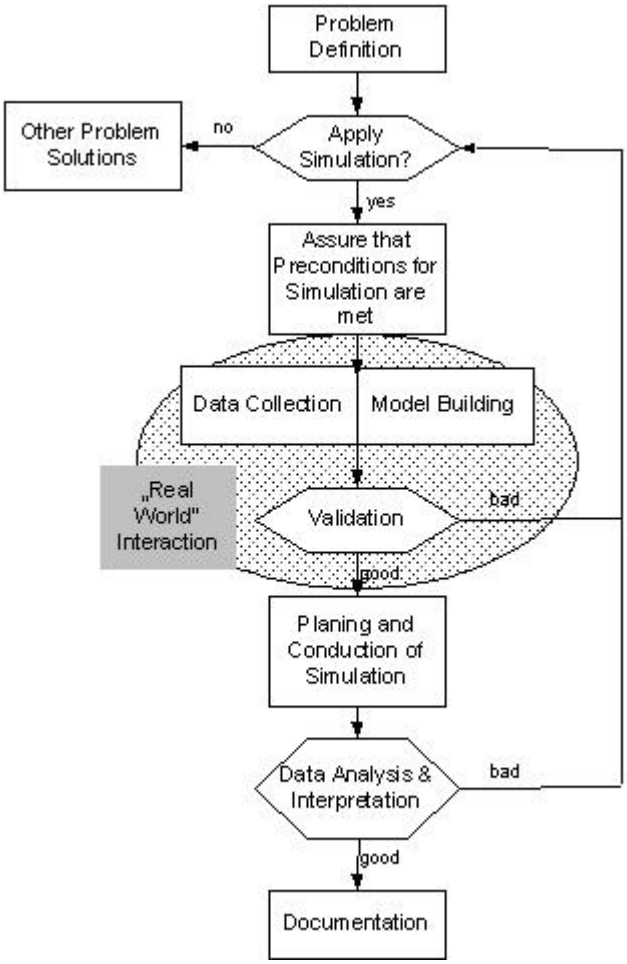


Figure 1. *Simulation process*[12]

2.2.1. Manufacturing simulation

After explaining simulation and the process it consists of, simulation in manufacturing is the hot topic for some decades. It is important because manufacturing systems are one of the largest application areas for simulation modelling and they are growing and becoming more complex.

Simulation in manufacturing addresses some specific issues, such as the quantity of equipment, like conveyors, machines and product volume. The other issue is the valuation of performance which is possible through the analysis of throughputs, and time in system. Operational procedures should be evaluated as well, by scheduling the production and controlling the strategies and analysing the reliability[13]. Simulation of manufacturing is focusing on modelling and monitoring the manufacturing organizations, processes, and systems. Important example of this simulation is the modelling of discrete and continuous manufacturing processes, such as, offline programming of robots and layout planning and assembly line planning[14].

Each simulation can estimate some measurements of the system. some of the basic performance measurements, estimated in manufacturing processes are; throughput, time in system for parts, times parts spend in queues, timeliness of deliveries, and one of the important measurement which can be used for improving the usability of the system is utilization of equipment. Simulation in the organization is being done with the commercial simulation software rather than programming languages and the criteria that the organizations have are the flexibility of the model and using the software easily. For handling the material and manufacturing, the modelling construct should be manufacturing oriented simulation language. These languages can reduce the time of simulation due to constructs for equipment. The managers and engineers in this field are looking for the software which can reduce the amount of programming and the orientation is toward manufacturing[13].

2.2.2. Simulation with MATLAB

MATLAB is a high level language, which lets the engineers and scientists to explore and visualize ideas. It is the fourth generation programming language, developed by MathWorks. MATLAB has a numerical computing environment and allows matrix manipulations, plotting of functions and data, generating algorithms and communicating with other languages such as C, Java, and Python. MATLAB is built on the MATLAB language and at first it was using for calculating numerical computations, but then they added some other features, such as Command Window for writing the code and executing the codes. It gives the possibility to write a powerful program in a few lines.

Nowadays MATLAB is a tool, which is used, in scientific and technical computing. It has a lot of different application areas, such as; financial mathematics, neural networks, control theory, optimization, and modelling production lines. MATLAB became popular because of its user friendliness, and it has been used as a teaching tool in classrooms. It has graphical capability which makes it good tool for analysing data. Simulation of the systems with MATLAB became popular as the engineers and simulators could model the complex systems with scripts and toolboxes[15].

2.3. MATLAB

MATLAB is fourth generation of programming languages, which represents multi paradigm numerical computing environment. It is useful for plotting functions and implementing algorithms, analysing data, and creating models and interfaces. MATLAB can be linked with some other languages, such as, C, C++, Java, and Python. MATLAB is useful in some applications including, control systems, test and measurement, signal processing, computer finance, and computational biology[16]. At first MATLAB was for numerical computing but then some toolboxes were added to allow the access to symbolic

capabilities. One of first packages added was Simulink, which was added as a library, allowing MATLAB to simulate systems in graphical multi domain.

2.3.1. Simulink

Simulink, developed by MathWorks, is a block diagram environment for designing, simulating and analysing multi-domain systems. It supports simulation, code generation, and verification of systems and consists of some customizable block libraries. Simulink is integrated tightly with MATLAB and enables the user to merge algorithms into models and export results to MATLAB[17].

Simulink library has wide range of blocks inside which some of them used in this thesis. The Three important blocks are; Subsystem, To Workspace which is called simout as well, and Scope. There can be found a brief description of these blocks in the next section.

Subsystem

This block can be found in Ports & Subsystems sub-library. There are different types of Subsystems, such as, Atomic Subsystem, Nonvirtual Subsystem, CodeReuse Subsystem, but the one used in this work is the normal Subsystem. A Subsystem block can represent a block for containing other blocks or codes with a model or system[17].

To Workspace

This block can be found in Sinks sub-library, and it is for writing signal data to MATLAB workspace. During the simulation, and internal buffer saves the data and when the simulation is completed the data is written to workspace[17].

Figure 2. *Simulink block: To Workspace*[17]

The icon inside the block is the variable which the data is written. Data can be saved in four different formats, a MATLAB object, an array, structure, and structure with time[17].

Scope

Scope is in the sub-library, Sinks, like To Workspace. This block is for monitoring the output signal which comes in three types, Target, Host, and File[17].

2.3.2. SimEvents

SimEvents is the library in MATLAB which is used mostly for simulating the assembly line. MathWorks developed this discrete event tool by adding a library of graphical building blocks to model queuing systems and add event based simulation engine in Simulink environment. Process and logistic simulation is one of the uses of SimEvents, for example by capacity and production planning. It can model the performance of a system, and provide the characteristics of a system such as, throughput, pocket loss, and utilization. It also provides libraries of entity generators, queues, servers and statistic reporting tasks. SimEvenets and Simulink can be used in the same model with time based and event based components in the same time, and this capability is used in this thesis work. There will be a brief description of the blocks, used in the simulation of assembly line in the next few lines.

Time-Based Function-Call Generator

This block can be found in a sub-library called, Generators. The mission for this block is to generate function call events, in the time which can be set to two different modes, one at the start of simulation using an integration period and second using a signal with connecting to the input port. The time interval between two generation events is integration[18].

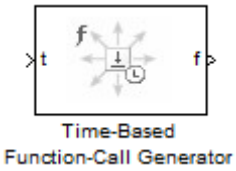


Figure 3. *SimEvents Block: Time-Based Function-Call Generator*[18]

Entity Departure Counter

This block is used for counting the number of entities, and it is located in Entity Management sub-library. By writing the numbers to a signal output or attribute of each entities, it makes it possible for the scopes to plot them with the index of Count.

Figure 4. *SimEvents Block: Entity Departure Counter*[18]

Set Attribute

This block is located in Attribute sub-library and it is one of the most important blocks in SimEvents as it is organizing the attributes. The block accepts an entity and after assigning data to it, outputs it. Data is stored in entity attributes and each attribute has specific name and value. It can take up to 32 attributes in the same time[18].

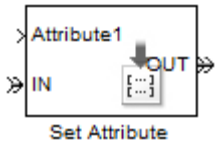


Figure 5. *SimEvents Block: Set Attribute*[18]

FIFO Queue

Queuing system in SimEvents plays an important role as it can play the role of buffer or storage. This block belongs to Queues library, where there are some other block for queuing as well. FIFO Queue can store entities with certain capacity which the programmer defines it. This block has the capability to store the entity for certain time if the next block or port is blocked or busy. The type of this storage is first-in first-out, and that is the difference between this block and the other blocks in the sub-library[18].

Figure 6. *SimEvents Block: FIFO Queue*[18]

Output Switch

This block from Routing sub-library, has different usages in the simulation. After receiving the entity the block decides which port or route to use for outputting, and the port can be changed during the simulation. The output port can be selected by these criteria; blockage of the ports, based on attributes, random selection, and from signal port[18].

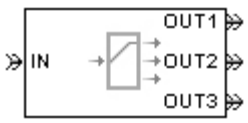


Figure 7. *SimEvents Block: Output Switch*[18]

Single Server

The block from Servers library is playing the role of work stations in this line. It serves one entity for a period of time and if the output port was not blocked, it outputs the entity,

if it was blocked, the entity can stay there until the port becomes free. Service time is the time specified by a parameter and specifies the time entity needs to be ready.

Figure 8. *SimEvents Block: Single Server*[18]

These blocks are basic blocks for simulating an assembly line in MATLAB. The descriptions of each block gives an idea for the reader how the simulation works and operates.

2.3.3. GUIDE

GUIDE is the Toolbox in MATLAB which allows programmers to use Graphical User Interface (GUI) in their programs. GUI provides a way to share code between nonprogrammers by removing end users from the command line of MATLAB. By using special compilers, GUI functionality connects to mathematical ability of MATLAB. In simple explanation, using GUI in MATLAB makes it easier and reducing time and complexity of programming. For example the code which takes one month to implement in C++, would take just couple of hours in MATLAB with using GUI. GUIDE has three basics as follows; lay out control, wire up call backs, data gathering from the controls.

GUI is a display in windows containing controls, called components, enable user to perform tasks. Components of GUI include, menus, toolbars, buttons, boxes, and sliders. GUIs perform computation, read and write data, communicate, and display data as plots. The figure below shows the GUIDE's tools and describe them briefly[19].

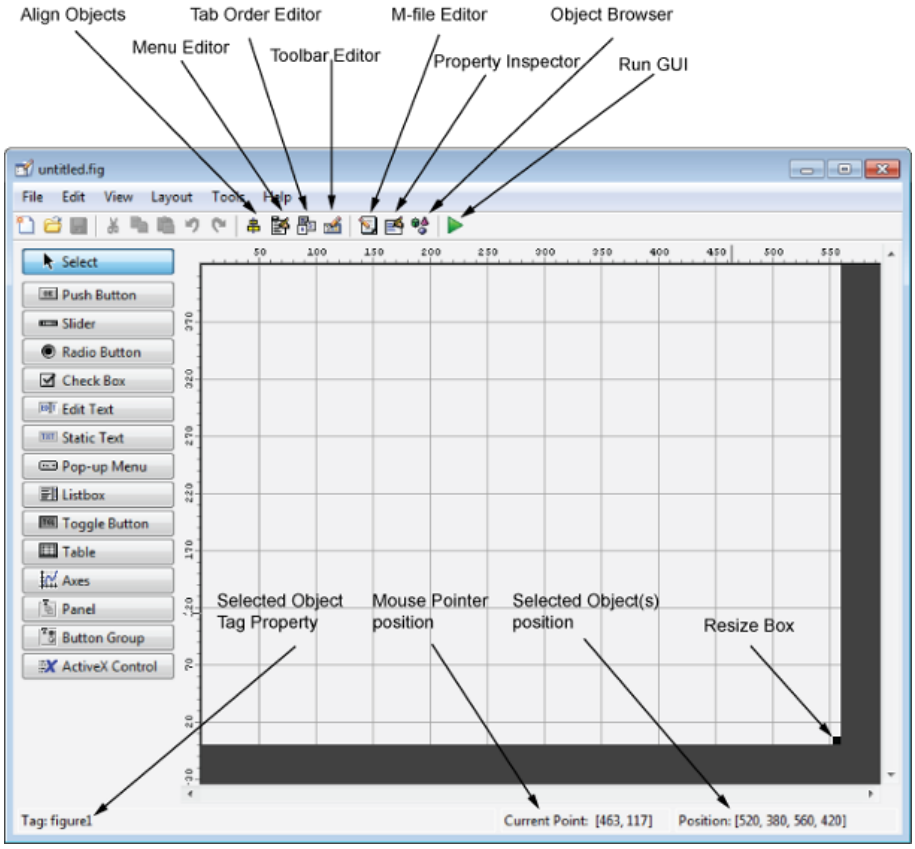


Figure 9. *Layout Editor with GUIDE tools description*[19]

2.3.4. Script

Scripts are simplest kind of program and collections of multiple sequential MATLAB commands stored in text files. They can be executed by calling their names. Scripts are .m files like functions but the difference between script and function is that, the function needs input and output parameters but script can operate on the hard-coded variables which are located in their m-file[20].

One of the challenges in this work was to connect Simulink to GUI, and by means of scripts the connection implemented. Script is the connection between GUI and Simulink, where GUI is the interface and enables the user to send the inputs to the system, and Simulink simulates the assembly line which processes the inputs. Scripts handles the data from GUI to Simulink at the start of the simulation and at the end it gets the data from Simulink and displays it to the user.

2.4. Optimization

Optimization is the science of maximizing or minimizing a real function by selecting of the best input from a set of different values. Optimization theory or technique finds the

best available value from the various values, and this value is one of the best possible values but it does not mean that it is the best one, but it guarantees, it is one of the best ones. Optimization problem is the problem of finding the maximum or minimum value from the possible solutions. There are different applications for optimization such as mechanics, economics, control engineering, and so on.

Most of the problems can be optimized; therefore the engineers and designers try to find the optimization strategies for an efficient and systematic decision making approaches. To find the best solution for an optimization task in practical standpoint, there should be some elements, such as, an objective function, which can be the system's profit or cost, then a predictive model which defines the system's behaviour, in practice this is a set of equations. Next element is variable, which comes in the predictive model to satisfy the constraints[21].

Optimization is a most common applied task in engineering, but in many cases the tasks are done by trial and error, and it cannot guarantee the best solution, for this reason a systematic determination for all optimal solutions should be taken. Research in optimization can be observed in different levels which different communities consider. For example mathematical programming level, is focusing on understanding basic and main properties of optimization algorithms. In the level of scientific computing, optimization method is implemented for efficient and practical use. Level of operation research is about how to formulize the problems and develop the strategies. And last but not least, the engineering level, is defining and challenging the real world problems and it relies on efficiency and reliability of methods and diagnosis of failure in solution methods[21].

For developing a successful optimization strategy, a working knowledge of levels is needed. For example in mathematical programming level, it is important to develop the right algorithm, in the engineering level it is more important to solve the right problem formulation. Therefore developing and solving the optimization algorithm not only requires a knowledge of existing software but also needs a knowledge of algorithmic principles[21].

2.4.1. Classification of optimization problems

Classification of optimization problems can be done according to the type of constraints, design variables, structure of the problem, involved equations, and number of objective functions. Based on existence of constraints, there are two types of classification, constraints optimization problems, which are about constraints and unconstraint optimization problem, with no constraints involved. Based on the nature of equations, the optimization problem can be classified as linear, nonlinear, quadratic, and geometric. In linear programming problem all the constraints are linear and in nonlinear one there is at least one nonlinear function, in geometric problem the functions are polynomial, and quadratic problem is a programming problem in which the objective functions are quadratic and it

has linear constraints and it is concave. The classification is important as if the engineers want to design optimization systems, they should select the computational method according to the classification of the problem. The other way of classification is based on acceptable values of the variables. According to the accepted values, optimization can be classified as deterministic or stochastic and integer or real valued. Based on the physical structure of the problem, classification can be divided to two groups of optimal control and non-optimal control problems[22], [23].

2.4.2. Optimization algorithms

There are many optimization algorithms known in computer and mathematical science and it is impossible to find one particular algorithm that is suitable for all the problems. It is possible to divide the optimization algorithms into three different categories based on the under laying principles; biology based algorithm, physics based algorithm, and geography based algorithm. The first category that will be discussed in this part is biology based category, and this category is divided into two subcategories itself, Evolution based Algorithm (EAs) and Swarm based Algorithm. Evolution based Algorithms are methods for mimicking the process of biological evolution and the behaviour of species with stochastic search methods. Evolution based Algorithms are divided to some other subcategories as well. Genetic Algorithm (GA) is one of the subcategories for evolution based algorithm and this thesis is based on that. Genetic Algorithm is a search technique with the mechanism of natural selection and it begins the search between some chromosomes which are solutions of the problem and generating other chromosomes for improving the solution until it reaches a point that seems a good and optimal solution. The basic steps in Genetic Algorithm are selection, crossover and mutation[24].

The other category in the Evolution-based Algorithm is Evolutionary Programming (EP), it starts with some random solutions and evolves over some generations and iterations. The main steps are initialization, mutation, competition and selection[24]. The following figure is showing some important classifications and algorithms of optimization, and in the rest of this part some of them will be explained briefly.

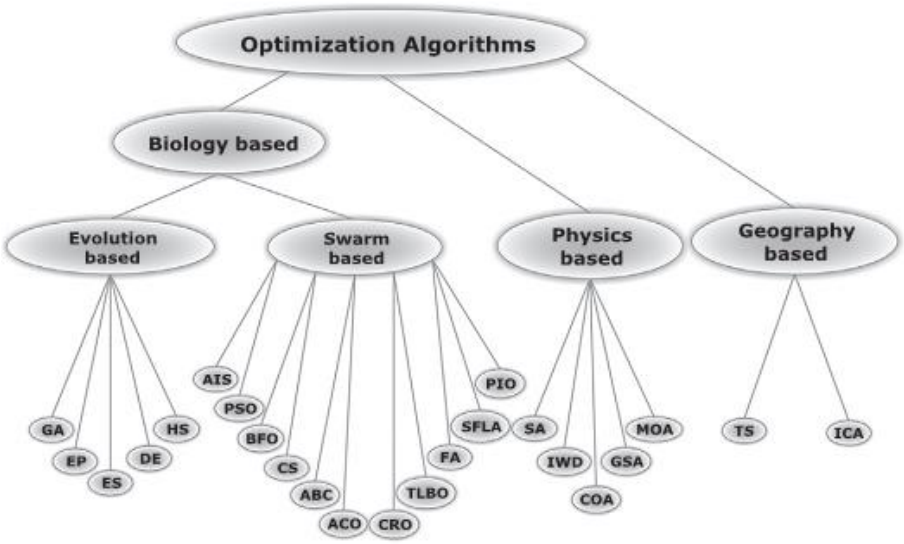


Figure 10. *Classification of optimization algorithms*[24]

The Other subcategory in Evolution-based Algorithm is Evolutionary Strategy (ES), which is inspired from biological evolution's principles. The parents are selected from a population that are the presentation of mating. By duplication and recombination of the parents, the new offspring are generated. After applying mutation to new offspring and changing it according to specified principles the new members are generated. Differential Evolution algorithm (DE) is classifying under the evolution based category as it has a promising heuristic algorithm in numeric problems. The difference between DE and GA is that DE uses real coding of floating point numbers instead of binary coding for representation parameters. Harmony Search algorithm (HS) is the last classification from the category of evolution based algorithms which is explained in this part. This algorithm mimics musician's behaviour to get a state of harmony. It contains three operations such as; random search, memory consideration, and pitch adjustment[24].

Swarm based algorithm is a biology based optimization algorithm, and is a family of nature-inspired, population based algorithms. A swarm is a number of agents working and interacting in the environment without any main control for supervising the group's behaviour. They can generate low cost, and fast solutions to problems, although they are simple themselves but together in the group they can accomplish complex tasks. They behave based on social nature behaviour such as, ant colonies, honeybees, and bird flocks. Artificial Immune Systems (AIS) are classified in these algorithms. These systems mimic biological basics of clone generation, maturation, and proliferation. Antibodies and affinities are considered as the possible solutions[24].

The other algorithms which are classified in the swarm based algorithms and can be seen in the previous figure are as follows; Particle Swarm Optimization (PSO), Bacteria Foraging Optimization algorithm (BFO), Cuckoo Search algorithm (CS), Artificial Bee Col-

ony algorithm (ABC), Ant Colony Optimization algorithm (ACO), Coral Reef Optimization algorithm (CRO), Teaching-Learning Based Optimization algorithm (TLBO), Firefly Algorithm (FA), Shuffled Frog Learning Algorithm (SFLA), and Pigeon Inspired Optimization (PIO)[24].

After explaining biology based algorithms and subcategories under the category, physics based algorithms are the algorithms to be explained. These algorithms are mimicking the physical behaviour and properties of the matter. For example Simulated Annealing (SA), which is classified in this group, is a technique used for crystallization a physical process in metals for hardening of the material. The other algorithm classified here is Gravitational Search Algorithm (GSA), which is working based on gravitational force. The other algorithms based on physical behaviour are as follows; Chaotic Optimization Algorithm (COA), Intelligent Water Drops algorithm (IWD), and Magnetic Optimization Algorithm (MOA)[24].

The last category in the optimization algorithms are geography based algorithms. These optimization algorithms are generating the solution in the geographic space. For example Tabu Search Algorithm (TSA) uses local search to explore the search space for acceptable solutions by sequences of moves. Imperialistic Competition Algorithm (ICA), is the other algorithm in this category which has the population based on colonies and imperialists[24].

2.4.3. Flow-Shop Scheduling

Scheduling in the production and manufacturing process is the process of arranging, controlling and optimizing of workloads, it is allocating plant and machinery resources, and planning production processes. In manufacturing and engineering, it minimizes the production time and costs and has impact on productivity process by organizing the staff and equipment and timing of the production, this leads to increasing of efficiency of the system.

The problems in the field of shop scheduling belong to the bigger problem class, called multi-stage scheduling, where each job includes some different operations. There are three basic categories among the shop scheduling problems, such as; flow-shop, job-shop, and open-shop. In a flow-shop problem, there is exactly the same amount of operations for each job, and the route through the machines that each job should pass is the same. In a job-shop problem, each job has specific route to pass, and the number of operations for each job can be more or less or even equal to the number of machines. In an open-shop problem there is not any specific route to be defined for the jobs and each job can be processed in any machine[25].

In flow-shop problem, the sequence of the machines is the same for all the jobs, and the problem is to find the best sequence of orders. Therefore the jobs should pass the machines in the same order according to the technological constraints. There are two important decisions that the scheduling problems focus on. First is the sequence for orders of the jobs which should be processed by two or more machines, second is the schedule of machine loading which defines the start and finish times on each machine for different jobs. Engineers and managers usually prefer to take care of the job sequence and machine loading schedules, like flow time, and utilization[26].

2.5. Artificial Intelligence

Artificial Intelligence (AI) is the field of study in which the researchers and scientists look for the ways to produce and create machines and computers with intelligent behaviour. Artificial intelligence is a method that simulates the operation principles of human brain. It was proposed at first by a group of neurophysiologists in the 1950s, and it was mostly about how human brain works. In the history of human development, the purpose of artificial intelligence was to free people from labour with machines. The advances of different sciences and technologies such as mathematical logic, information theory, computer science, and psychology led to development of AI in theoretical and ideological way. In real world, some of the problems are quite complex, even if they have calculation methods, they are NP problems. Scientists are introducing heuristic knowledge for solving such problems, but not all the time they are optimal solutions. This kind of problems led to introducing and birth of AI. Since then many progress has been made for developing the disciplines of AI, such as natural language processing, pattern recognition, robotics and image processing[27].

In the 1950's mainly the research in AI was focused on game playing, Arthur Samuel wrote the first game program with learning ability. Then in 1960's it changed and went toward developing search algorithms and general problem solving (GPS), Allen Newall and his colleagues release the general problem solver, which was more powerful than the other solvers of the time. In 1970's, the focus was on natural language understanding and knowledge representation. In this time Edward Feigenbaum stated that knowledge engineering is the tools of AI research to solve the difficult problems. In 1980's, AI developed successfully and the expert systems were used widely, and industrial AI prospered. Like other disciplines there are some obstacles in AI history, such as being accused to be too optimistic. Some theories of AI still need to improve, such as Machine Learning, knowledge representation and reasoning[27].

In this field the engineers should have a proper knowledge of machines and intelligent agents to make the systems perceive the environment. There are some tools used in AI, including search and mathematical optimization, logic, and probability. Long-term goals are social intelligence, creativity and general intelligence. AI has a lot of different application in industry nowadays. These applications are used in computer science, finance,

hospitals, transportation, toys and games, music, and aviation. Two field of the AI that are used in this thesis will be explained in the following sections.

2.5.1. Machine Learning

Machine learning (ML) is a subfield of AI that evolved from integration of two fields, pattern recognition and computational learning theory. It is a field of study which gives the machines and computers the ability to learn without being programmed and it is used for optimizing performance of the machine using example data or past experience. The difference between traditional programming and machine learning is that, in traditional programming the inputs to the computer are data and the program and the computer will give out the output but in machine learning data and outputs are the input of the machine, and the computer will give out the program according to the data and output. ML field has three main components; representation, evaluation, and optimization. The representation section deals with the model to represent the problem, after selecting the appropriate model for representing problem and solution. The selected algorithm should be evaluated to see how good it can produce the target function. Then the optimization part will optimize the candidate to get better solution and output.

After development of AI, over past decades ML has become one of the most important foundations of information technology. With increasing in amount of data available in industry, the analysis methods of this data was needed for technological progress. ML has different applications deal with different types of data, and after generating new prototypes for the application according to the data, it will be easier to guarantee the good solution for the problems without reinventing new programs. Some of new applications of ML are as follows; web page ranking, which is used vastly in search engines, which helps the user to find the pages they need. Other application is collaborative filtering, and Internet stores such as Amazon and Netflix use this in their search engine to filter the pages and goods according to the past searching history of the user. Automatic translation of the documents is the other application, in which the translator uses the other defined documents for translating the sentences correctly. Speech recognition is the last application that will be mentioned in this part, and has the similar learning algorithm as hand writing recognition and pattern and video recognition. These applications use the defined documents for predicting the future data[28].

In ML field, data will be classified in different classes, for introducing to the machine. For example one way to define data is a vector which is the most basic entity in computer science. The other data types are lists, sets, matrices, images, videos, trees and graphs, strings, and compound structures. The range of learning problems is large, that is why it is better to classify the type of the problems as well. The most frequent problem in ML is binary classification, which has led to some important developments over the past years. Multiclass classification is the extension of binary classification, and the difference with binary is that one variable can present range of different values. Regression is another

type of problem, in which the goal is to estimate a variable with giving a pattern. In novelty detection problems, the goal is to find an unusual pattern or variable with defining a set of past measurements[28].

In this section some of the learning method will be explained, and use case and related techniques will be mentioned.

Supervised Learning

Supervised learning is the most commonly used type of machine learning. In this type of learning there are some training data with label and the machine will produce a program for predicting the labels of new data. Supervised learning splits into two broad categories; classification which has just a few known values, such as 'true' or 'false' and regression which are for the responses with real numbers. Supervised learning has some steps for producing the model for a system:

1. Preparing data
2. Choosing an algorithm
3. Fitting a model
4. Choosing a validation method
5. Examining and updating the model until it is satisfied
6. Using fitted model for prediction

There are some different algorithms in this learning method, which use different hypothesis and cost functions. The important and most commonly used ones are as follows:

- Linear Regression
- Logistic Regression
- Artificial Neural Network (ANN)
- K Nearest Neighbours (KNN)
- Naïve Bayes
- Bayesian Network
- Decision Trees
- Support Vector Machines (SVM)

Unsupervised Learning

In contrast with supervised learning, in unsupervised learning there is no any label for data and it will group the unlabelled data according to their similarity. Unsupervised learning will find the hidden structure in unlabelled data, which is the reason for not having any error or reward signal. The most important algorithms in this learning method are as follows;

- Clustering

- Gaussian Mixture Models
- Self-Organizing Map (SOM)
- Principle Component Analysis (PCA)
- Hidden Markov Models

Semi-Supervised Learning

This learning method is a class of supervised learning tasks and techniques that also make use of unlabelled data for training, the difference between the other two learning methods and semi-supervised learning is that, in this learning method not all the data is labelled or unlabelled, small amount of data is labelled and larger amount is unlabelled. The algorithm will generate a program according to the labelled data and test it on unlabelled data for finding the error and improvement. The important algorithms in this category are as follows;

- Self-Training
- Generative Models
- Semi-Supervised Support Vector Machines
- Graph Based Algorithms
- Multi-View Algorithms

Reinforcement Learning

Reinforcement learning mostly deals with such situations where an agent should sense and act in its environment and choose the optimal action. Control of mobile robot and optimization operation in factories are the applications of this learning. The idea behind this learning is that the trainer will provide the feedback to the agent and according to the feedback, the agent will improve its action. Genetic algorithm, which is used here in this thesis for improvement of the factories production time is working based on this learning method. Some of important algorithms used in this learning are as follows:

- Q-learning
- Monte-Carlo Methods
- Temporal difference methods
- SARSA
- Markov Decision Process (MDP)
- Policy Gradient Algorithms

The other algorithms in machine learning, which are less important or not common in industry, are as follows: Deep Learning, Active Learning, and Multi-Task Learning.

One of the hardest parts of solving a machine-learning problem is finding the right algorithm for the job. Different algorithms are better suited for different types of data and problems.

In machine learning field there are a lot of different types of problems. Supervised learning is predicting labels from attributes, unsupervised learning discovers structure in data without labelling, semi supervised learning improves performance of predicting with using both label and unlabelled data, reinforcement learning uses feedback to improve the agents action and so on[29].

2.6. Genetic Algorithms

Genetic algorithms (GA) are stochastic search techniques mainly based on the mechanics and behaviour of natural selection and natural genetics. GA is used to generate solution to optimization and search problems. GA belongs to the other class called evolutionary algorithms (EA), which uses techniques inspired by natural evolution such as mutation, selection, and crossover. They are used in artificial intelligence for searching a space of potential solutions for finding the optimal one. GA is classified in machine learning category, as it is the working based on learning the machine's behaviour. Reinforcement learning is the category that is close to GA in machine learning. GA uses the feedback from the system to improve the solution, similar to what reinforcement learning does. There are three reasons why genetic algorithms are important in machine learning. First they can work on discrete spaces where gradient-based methods do not work. Second, they are used in some situations where there is only information about performance's measurement that is why they are classified as reinforcement learning algorithms. Finally, they work like multi agent systems, and instead of working on a single entity, they involve a population and a group of entities[30].

John Holland first proposed GA in his book "Adaptation in Natural and Artificial Systems". He guessed the feature that distinguished natural adaptive systems from artificial systems was that biological systems are based on an effective solution. He tried to generate an artificial procedure to simulate this, and one of the components called a genetic algorithm with crossover[30].

Genetic algorithms and the other genetic programming, which are the extension of those, are the most important searching tools in machine learning. The chromosomes which are in natural genetic systems consist of genes, which have value and position. Genetic prescription is forming from combining the chromosomes for the construction and operation of organism. For simulating chromosomes and genes, strings and characters are used which are corresponding to the natural genetic systems[31].

A simple genetic algorithm has some elements; representation is one of the elements used in GA. In order to solve a problem with GA, the problem should be encoded to strings of

characters which machine can understand, these strings are called chromosomes. Fitness function is the other element in GA, which defines how good the chromosomes fitted in algorithm, and it guarantees to increase the fitness of the strings. The string represented by genotype and it produces the phenotype, but these two are not so different. Selecting population of strings is called population dynamics. There are three operations that take the population and makes new population. The first one is selection, which happens as the first operation, and function of this operator is to select the strings that better fit to the fitness function. In selection part the new strings are not introduced but just reducing the number of strings for further operations. The next one is crossover, which takes two strings as parents and produces two or more offspring by combing parts of the parents. Mutation is the last operation that has less influence in producing new strings. This operation randomly changes some characters in offspring and makes small changes in them[30].

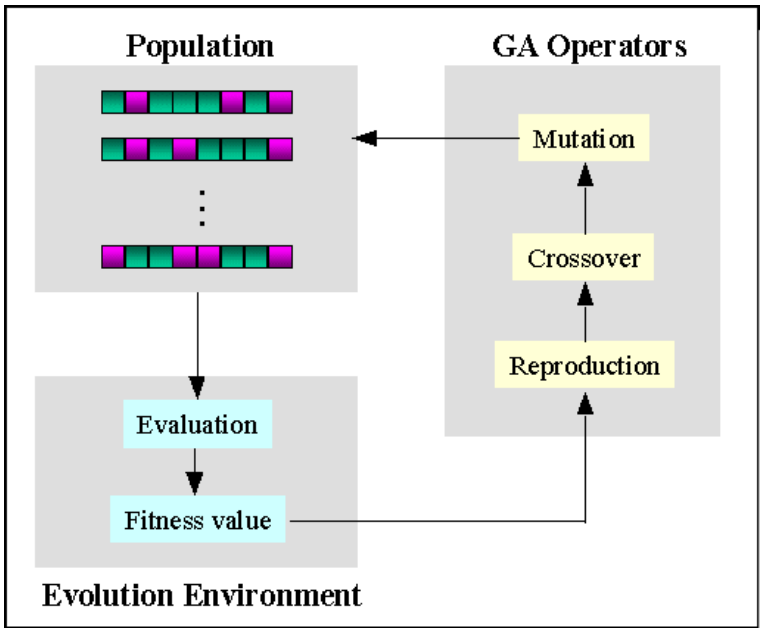


Figure 11. *Evolution flow of Genetic Algorithm*[32]

To apply GA in scheduling problems, first chromosomes should be represented, and in the next chapter it will be discussed in more details. Different representations are possible in case of different types of inputs. The next operation is to select some of chromosomes which are more fitted to fitness function, this selection mechanism can be done with different algorithms. Ranking the chromosomes and selecting according to the rank of them is one way to do the selection. Crossover and mutation have the responsibility to generate new chromosomes from selected ones. Different methods are applicable based on how the chromosomes look like. Mutation is for reducing the rate of convergence by flipping the elements in chromosomes. Two types of mutations are applicable, first exchanging the elements and second shifting them. After generating new chromosomes the loop will reach to the initial point which is selection, and this will continue until the iterations are completed[33].

For solving optimization problems, many mathematical programming methods have been developed, but there has not been a single completely efficient method for all optimization problems in different engineering fields. In design problems, the variables and mathematical point of view, are different from each other, and mostly continues variables are focused by most of mathematical optimization applications[34].

The calculus based techniques and numerical methods have some shortcomings, which makes the random methods popular among engineers. The random search methods are known as evolutionary algorithms, and they are robust optimization methods. The approaches based on population, which use selection and variation to create new solutions, can be called as evolutionary techniques. Genetic Algorithms is such search method which uses random selection for optimization, and they behave successfully in robust searches of complex spaces. That's the reason why GA are broadly used in scientific applications as business and engineering circles[34].

In this thesis work GA has been used for optimizing the simulated assembly line. There are some steps for designing GA for a system, and each step has different methods with different efficiency. In the following couple of lines these steps will be explained, and methods exists for the steps will be mentioned.

2.6.1. Encoding

The first step in solving a problem is encoding which depends directly on the problem. Encoding is a process of representation of genes, which can be performed by using bits, numbers, trees, arrays, and list. Encoding can be classified into 1-demsional and 2-dimensional depending on the structure. For 1-dimensional encoding, Binary, permutation, value encoding, and for 2-dimensional, Tree encoding can be mentioned[35].

1. Binary encoding

One of the most common representation methods is binary encoding. Chromosomes are represented by strings as {0,1} in this encoding. For example following table represents two chromosome in binary encoding:

Table 1. *Example of chromosomes with Binary Encoding*

Chromosome A	1011010001100101
Chromosome B	1000100111100110

With just two bits, binary encoding can generate many possible chromosomes, but sometimes it needs corrections if it was not natural for the problems.

2. Permutation Encoding

Permutation encoding consists of representing the chromosomes by strings of numbers which is in sequence. This encoding method can be used in ordering problems, such as task ordering problem and travelling salesman problem[36]. The table below shows two chromosomes represented with permutation encoding:

Table 2. *Example of chromosomes with Permutation Encoding*

Chromosome A	12345678
Chromosome B	32578416

This encoding is mostly useful for the problems with specific order.

3. Value Encoding

In the problems where using binary encoding is difficult, value encoding is preferred. Value encoding is the method of representation of chromosomes with strings of values. Values can be from numbers, real numbers or chars depending on the problem. The table below shows three different kind of value encoding:

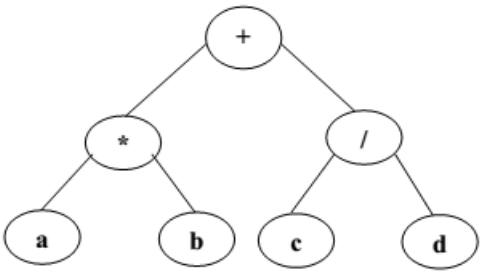
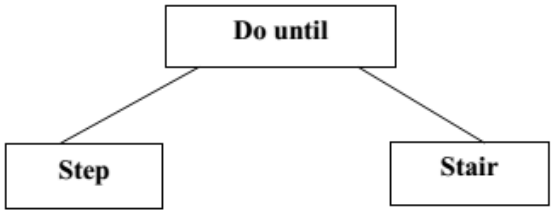
Table 3. *Example of chromosomes with Value Encoding*

Chromosome A	1.234 2.012 0.209 6.765
Chromosome B	ABYTFDSERYJMNFCD
Chromosome C	(back),(right),(left),(back)

4. Tree Encoding

This encoding method is used mainly for evolving programs and expressions, in genetic programming. Tree encoding represents every chromosome with a tree of objects, such as commands or functions[35]. Table below represents two chromosomes with tree encoding:

Table 4. *Example of chromosomes with Tree Encoding*[35]

Chromosome A	Chromosome B
	
$(+(*\ ab)/(cd))$	Do until step stair

In this thesis work, permutation encoding is selected for representing the chromosomes, as the problem is ordering problem. Therefore each product gets one number from one to the number of products needed. This is done by giving an attribute and permutation values with Set Attribute block in SimEvents.

2.6.2. Initial population

After selecting a proper encoding technique for the products and entities, generation of an initial population is needed. Initial population is the population which is generated randomly by changing the order of the bits in chromosome. For example the first chromosome is the sequence of numbers from one to number of products. Then for generating the second and next chromosomes, the program shuffles the numbers in random order and then creates new chromosomes. This can continue until there is enough number of population, which is demanded by the user or customer.

2.6.3. Selection

After generating initial population, some chromosomes should be selected to be the parents of new chromosomes. There are some different ways to select the best chromosomes from the population, but not all of them guarantee to select the best ones and eliminate the poor ones. Some of the common selection methods are listed below:

1. Roulette Wheel Selection

This selection method works based on the fitness of the chromosomes. This means the better chromosomes have better chance of selection, and the poor ones with low fitness rate has less chance. This method originates from a game called roulette, which has slices in the wheel and fit slices occupy more space on the wheel and the ones with low fitness occupy less space. After spinning the wheel each time a slice will be selected by a marble, and logically the slices with more spaces have more chance to be selected[36]. The figure below shows the division of the roulette wheel:

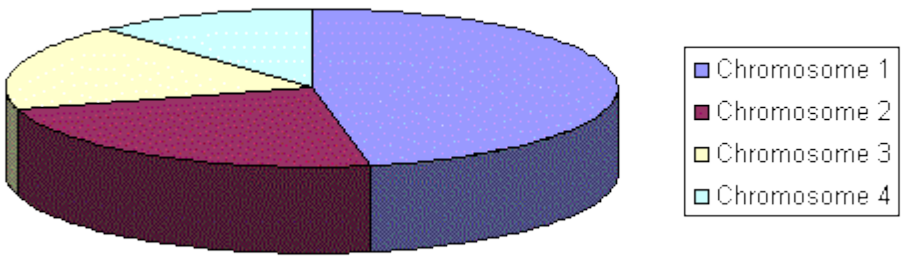


Figure 12. *Graphical representation of Roulette Wheel Selection*[36]

2. Rank Selection

If the fitness differs a lot Roulette Wheel Selection will have some problems that is the reason for inventing Rank selection. In this method the population gets the ranks, and then every chromosome receives fitness from ranking. For example the worst one will get fitness 1, and the best one N. This method can reduce the population's genetic diversity and will find acceptable solution[36]. Next figure shows how the situation changes after using Rank selection:

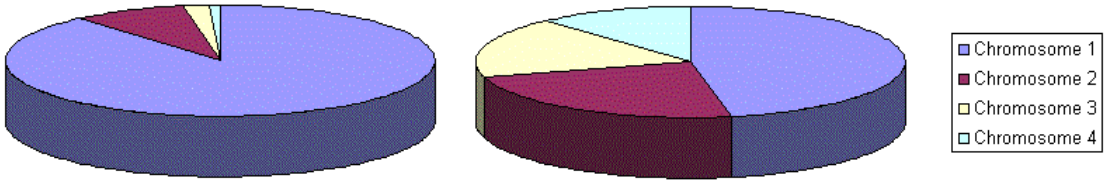


Figure 13. *Graphical representation of Rank Selection,1) The picture in left shows the selection before ranking 2) The picture in right shows the selection with ranking*[36]

3. Elitism

When the chromosomes are selected, there is a big chance to lose the best chromosomes, Elitism chooses the best chromosomes according to the fitness rate. This method prevents losing the best solutions and can lead to increase performance of GA[36].

The selection methods used in this thesis work are Roulette Wheel Selection and Elitism. These two selection methods are used for comparing the results of two different selection methods in GA and getting better results for improving the optimized automated line.

2.6.4. Crossover

Crossover in GA is a genetic operator to change the programming of a chromosome. Crossover merges the genetic information of two chromosomes which are the parents and generates new chromosome which is the offspring. Choosing the Crossover method depends on the encoding technique. In the next few lines, there will be some information about crossover in binary encoding method and then permutation encoding will be discussed and after that short explanation about other encoding methods.

Binary Encoding

1. Single point crossover

This method is simple as it has only one crossover point in the chromosome. The new chromosome is generated from copying the binary string of first parent from beginning to the crossover point and from crossover point to the end of the second parent's string. The following figure shows the simple copying of the strings.

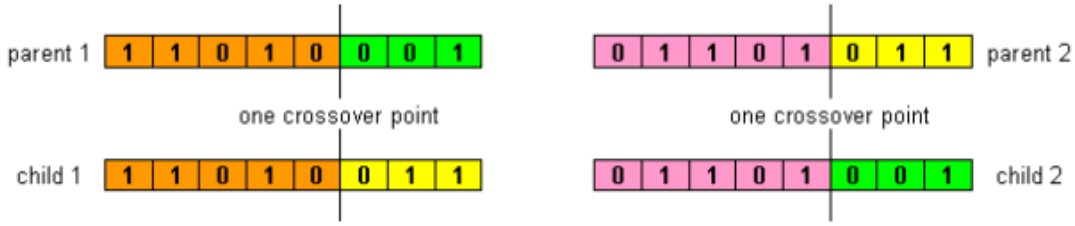


Figure 14. *Single point crossover in binary encoding*[37]

2. Two point crossover

Two point crossover is similar to single point crossover except that there are two cut points instead of one point. The offspring is generated from copying the strings from outside of two cut points of first parent and string which is copied from inside of two cut points of the second parent. The figure shows how this process is done.

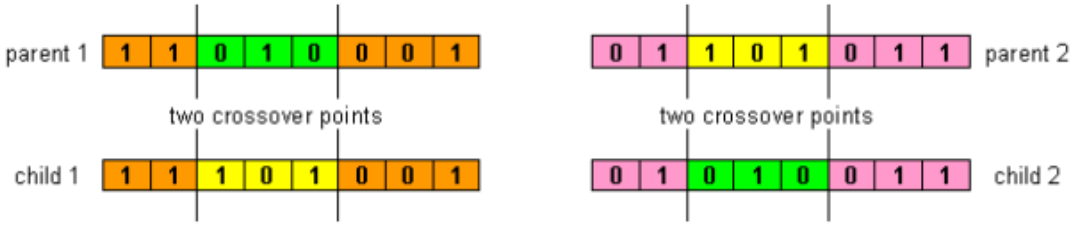


Figure 15. *Two point crossover in binary encoding*[37]

Permutation Encoding

In permutation crossover the methods which are used in binary encoding can be used but with some differences. For example in this work the ordering method is implemented. Ordering method in single point crossover is beginning with selecting a cut point on the two parents, then for the first offspring the bits from the first parents are copied to the cut point and for the second offspring, the bits are copied from the second parent, from the cut point to the end. The rest of the bits for the offspring are copying from the other parent. The difference between these methods with the other one explained before is the string for second half of the offspring is copied from the bits of the other parent in order of bits located in the strings. This example is shown in the following line.

(**1 2 3 4** 5 6 7 8) + (4 7 2 3 **8 5 1 6**) = (**1 2 3 4** 7 8 5 6) & (2 3 4 7 **8 5 1 6**)

In the other encoding techniques these crossover methods can be used but with some changes according to the ordering of the strings.

2.6.5. Mutation

The mutation is an operator applied to create different new chromosomes and to prevent the population from stalking in local optimal solution. The probability of mutation should

be low to maintain the parent's genes, and if the probability is set to high, changing of the chromosomes turns into a random search. A common method for implementing mutation in a chromosomes is to generate a random variable in the sequence. For example in this work the changing of the place for bits in permutation encoding has been implemented. The implementation has been done in two ways to show the difference between having a mutation in the algorithm and without mutation, which is proven that mutation is preventing from having a local minima by preventing the population from becoming too similar to each other. For example the next line shows a bit changing in permutation encoding which is used randomly in this thesis.

(1 2 3 4 5 6 7 8) => (1 7 3 4 5 6 2 8)

2.7. Genetic Algorithm in Manufacturing

In manufacturing, to be productive, there are some steps to be taken such as, optimizing the resources, reducing the waste and increasing efficiency. Genetic algorithms are one of the best solutions for production scheduling problems. Production lines are important in the industrial production of high quantity products and recently even in low volume production of customized products. Lack of knowledge in management of company resources and especially in production line can lead to high cost. Therefore the challenge in production line is to reduce the cycle time which leads to reduction of cost. Scheduling is an important field in manufacturing optimization. Optimal solutions in manufacturing scheduling are being classified in combinatorial optimization problems and most of them are in the class of NP-hard[38], [39].

In scheduling problems there is not a specific final answer, so the solution is to resort for searching a good answer. In production scheduling the population of solutions consists of many answers which may have different objectives. For instance, one solution can lead to optimization of a production process with minimal amount of time, and in the other one with a minimal amount of defects. Genetic algorithms are good for the types of problems which has large search space and small number of feasible solutions.

To apply genetic algorithm in production line for scheduling problems, representation is the first step. It should be represented as a genome. To represent a scheduling genome, a sequence of tasks should be defined. With making random sequences, then the initial population will be generated. In this initial population the best fitted genomes will be selected. The offspring selected based on the fitness function will generate the next generation of the population and this will continue until the best genome is selected.

2.8. Utilization

After simulating the line and studying the differences between balanced models and optimized ones, the most considerable parameter which has the most effect on the simulation

and optimization is utilization of each workstations. This parameter is defined with the fraction of time the work cell is not idle for lack of parts. This includes the fraction of time the work cell is working on parts or has parts waiting to work on them. The utilization can be computed as:

$$Utilization = \frac{Arrival\ rate}{Effective\ production\ rate}$$

Where the effective production rate is defined as the maximum average rate at which work cell can process parts. Arrival rate is called throughput rate as well. The average output of a production process (machine, workstation, line, plant) per unit time (e.g., parts per hour) is defined as the systems throughput, or sometimes throughput rate. In the other words throughput is defined as the rate of the parts produced by the line that are used. Ideally, this should exactly match demand. Too little production, and the sales would be lost and too much will build up unnecessary finished goods inventory[2].

3. APPROACH

In this chapter the approach for simulating and optimizing a production line will be explained. In the first part the method and steps needed for simulating a production line with MATLAB is explained and the second part shows how GA is created for the modelled line.

3.1. Simulation with MATLAB

For simulating a production line, there are some steps to take. The first step is to understand and study the line and how it works. Learning about production line is done by studying the instruction of the line and visiting the line and knowing how it works. Then after choosing the software and the program for modelling the line, it comes to choosing the tools needed. In the previous chapter, MATLAB and the tools needed for simulation has been introduced. In the following figure the flowchart for simulating a production line is depicted.

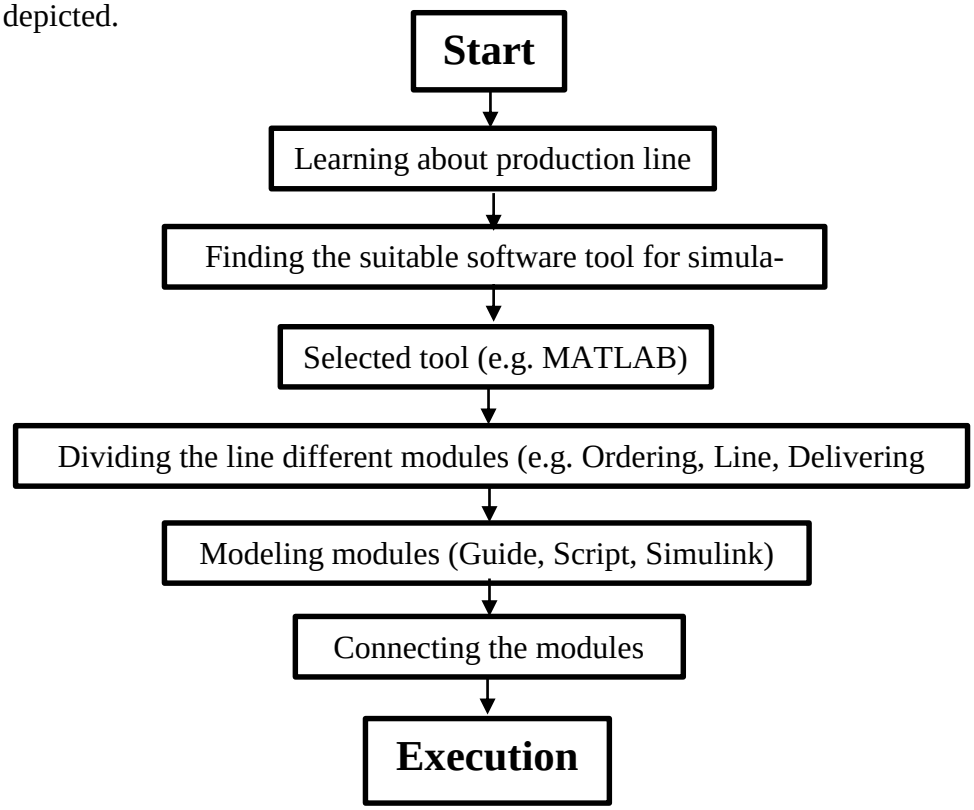


Figure 16. *Flowchart for simulating a production line*

After learning about production line and finding the suitable software for the line which is MATLAB in this case, the modelling and programming of the line will be started.

Every production line can be divided with three modules. These modules represent three different functions which can make the production line work. In this work the first module is the ordering module, which creates the possibility for the user to set the orders or products user want. This module consists of two parts itself. One part generates products and the part for the routing and sequencing the produced entities. Second module is the line itself which the orders and products are made. This part is the important part of the line. In this module the products and orders which are ordered in the first section by the customer will be assembled. This module represents the topology of the production line. Like it will be shown in the following chapter, a particular implementation for the module consists of ten workstations and each workstation has a specific responsibility for assembling one part of the product. At the end of the workstations chain there is checking section. In this part the products are getting checked to be known as ready to go for the next section or the ones which are not complete will be sent back to the chain to get completed. Last module is responsible for delivering the completed products. This section is connection between the production line and the customer. The completed products will be stored here and ready to be sent to the customer. The first module, which is the ordering module is the interface of the system, where the user can set the features of the products. This section can be modelled using GUIDE and then writing MATLAB code in Script. Line and delivering parts can be programmed by Simulink. After creating all three sections, and connecting them in the sequence, system can be ready for execution.

3.2. Optimization with GA

For implementing GA on modelled system, there are some steps, and each step has different methods with different efficiency. The first step is encoding which should be selected between different encoding methods, such as, Binary, Permutation, Value, and Tree. Following figure shows the flowchart for implementing a GA for a production line.

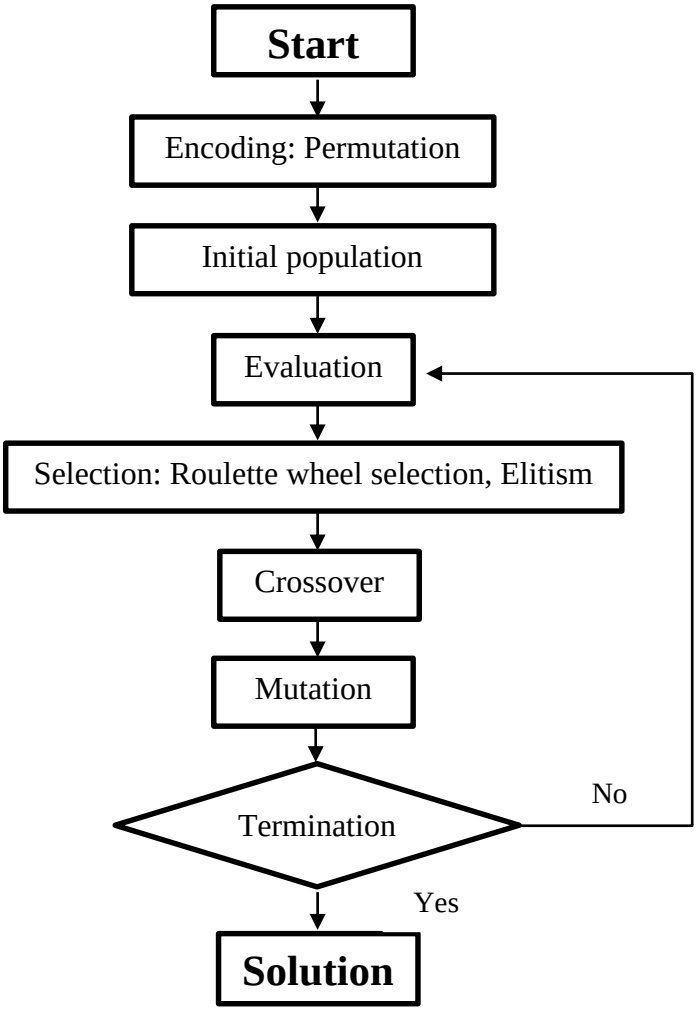


Figure 17. *Flowchart for implementing a GA for a production line*

After choosing the right encoding method which is Permutation in this work, the initial population should be generated. Initial population is generated randomly with changing the sequences in chromosomes. The number of initial population is variable which can be selected by the designer. In the evaluation part each chromosome in initial population gets a fitness rate. This fitness rate would help the algorithm to choose the chromosomes for next generation.

Selection is one of the most important steps in designing a GA system. According to the fitness rate which each chromosomes has, selection would be executed. There are different selection methods, such as Roulette wheel selection, Rank selection, and Elitism. These selection methods has been explained completely in previous chapter. In this work two selection methods has been implemented for checking the differences. These two selection methods are RWS and Elitism.

Crossover and then Mutation are for generating new offspring from the parents. With crossover each two parents can generate two offspring and then mutation will change some elements in chromosomes. After generating new chromosomes the criteria of termination will be checked, for example the number of iterations can be a criteria. The other criteria can be that, the system will check the results and if the results stalk in the optima, the system will terminate the system. If the system does not meet the criteria the system will continue the execution and will go back to generate new population and select among them. The result of this system will be the chromosome which fits the best and gives the best result.

4. IMPLEMENTATION

This chapter gives overview and details for the line which is used for simulation and optimization. In the first part of this chapter an overview of FASTory line which is an automated line is represented. Then the implementation part will be begun, with the simulation of the automated line. After simulation in the third section the line will be balanced with different implementations. The last section which is the fourth section, the automated line will be optimized.

4.1. FASTory line overview

FASTory line is an assembly line with 10 workstations, which is located in Factory Automation Systems and Technology (FAST) laboratory in Tampere University of Technology. In this line, each workstation has a robot with specific function. The overall and top view of this line is shown in the next figures.

Figure 18. *Overall view of FASTory in FAST lab*[40]

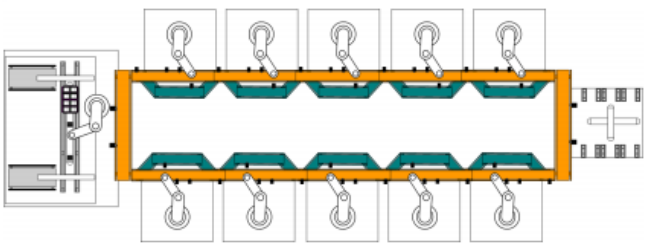


Figure 19. *Top view of FASTory in FAST lab*[40]

In the workstations, the robots assemble the cell phones with different features. Three main parts which are assembled in this line are, screens, frames, and keyboards. These parts can be produced in three different shapes, thin line rectangle, thick line rectangle,

and oval. Each shape can be painted in three different colours, red, green, and blue. Following figure shows an example of one cell phone with different features.

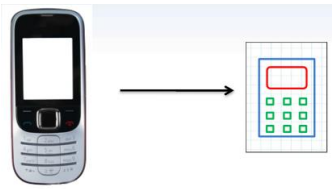


Figure 20. *Cell phone with different features produced in FASTory*[40]

For calculating the number of variety of products, which this assembly line can produce, one cellphone with one shape and colour for frame part will be considered. The figure below shows the variety of products for one specific shape and colour of frame which is thin line rectangle and blue.

Figure 21. *Variety of products for specific frame colour and shape*[40]

Previous figure shows that for one specific frame model in one colour there are 81 product variants. There are 9 different frame models and colours, which leads to have 729 product variants in total. Therefore the FASTory line assembles 729 different cell phones in different orders. The number of products and orders are defined by the customer and the line assembles these in the same sequence that the customer wants. To make the process faster there should be different sequences which is the purpose of this thesis work.

4.2. Simulation of FASTory line with MATLAB

To simulate the line with MATLAB, first it should be divided to three sections, with specific function for each section. The first section is the ordering system, which is called Production Order here in this work. Production Order has the responsibility of taking the orders from customer or user. Second section is the assembly line with the work stations inside. This section is called Line and there are ten workstations inside this section. The last section is delivery section which is responsible for delivering the completed products as outputs. This section is called Deliver and it shows the products which are ready to go out of the system. The figure below shows the complete model of the system in Simulink.

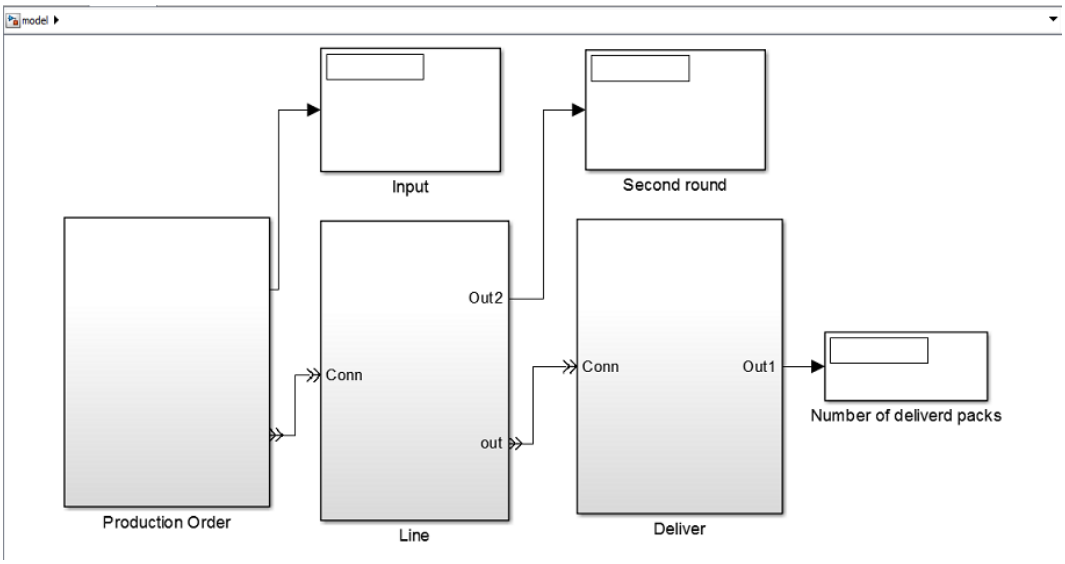


Figure 22. *Overview of the model in Simulink*

There are three different displays in the model to monitor the number of entities in each section. The first one which is called Input, is connected to Production Order section to display the number of entities generated in the ordering system. Next display block is to monitor the number of entities which are sent to the next cycles and are not satisfied or completed with one cycle of the system. This display is connected to the Line section and called Second round. The last display is connected to Deliver section is called Number of delivered pack. This display shows the number of products which are ready to deliver.

Each section of the system is defined with a subsystem which is consist of some other blocks for making the job operable. These subsystems are explained in more details in the following sections.

4.2.1. Production Order

In the first part of the system, the entities which present the products are generated and routed to the next section. This part is divided to two sections, one section is for generating the products with specific attributes which represent the specification of the products, and the other section is for routing and sequencing the generated products to the next block.

1. Product

In Simulink, it is possible to create subsystems and then load them wherever they are needed. Therefore, the product is created and saved in a subsystem, with some blocks which makes it possible to set some attributes to the generated product. Following figure presents a product with the blocks inside.

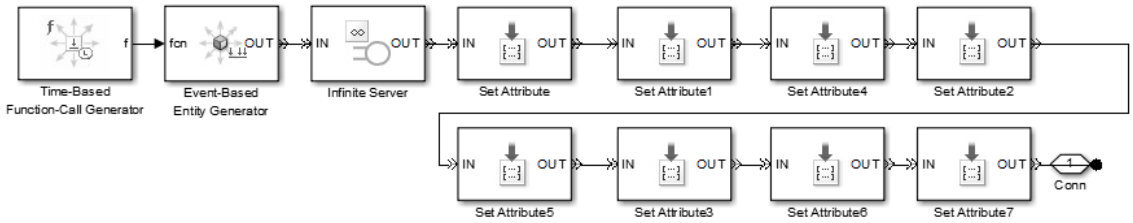


Figure 23. *Product subsystem with blocks for generating entities and setting attributes*

The first block is Time-Based Function-Call Generator block which is used to set the number of the products needed from one specific type. After defining the number of entities which needed, the entity itself should be created, this is done by Event-Based Entity Generator. Infinite Server block is needed to set the timing of the generation of the entity. Now that the entity is created, it needs some attributes which represent the features of the product which is the cell phone. By these attributes, the entities will have identities and they know which workstations are needed for operation. There are 8 Set Attribute blocks, 6 of them are for specifying the colour and shape of the cell phone parts and one is to set the order number and the other one is to change the sequences.

2. Routing and Sequencing

This subsystem routes the generated entities to the next section. The other operation in this part sequences the entities according to one specific attribute. Sequencing is done with the block named Priority Queue. This block sequences the entities according to the attribute which is defined in the Product subsystem. The complete view of this subsystem and the connections are shown the next figure.

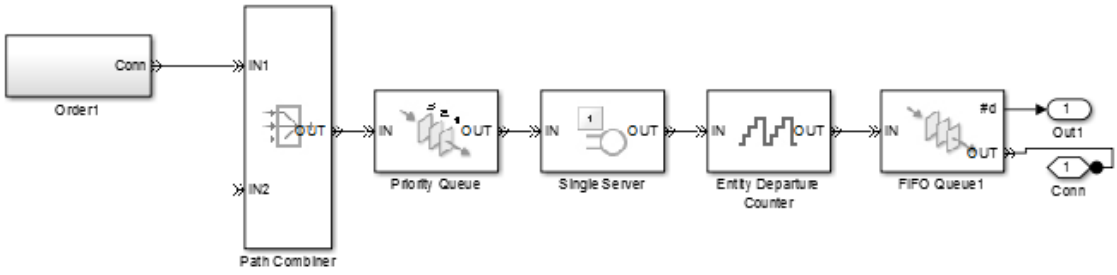


Figure 24. *Production Order subsystem for routing and sequencing*

4.2.2. Line

The second part of the system is the part that the workstations line one after the other. This part consists of 10 work cells and each work cell is shown by one block and the last block is to check the products if they are ready to go to delivery part or not, if they were not ready they will be sent to the first work cell for the next cycle. Next figure shows the overview of the work cells and how they locate in the system.

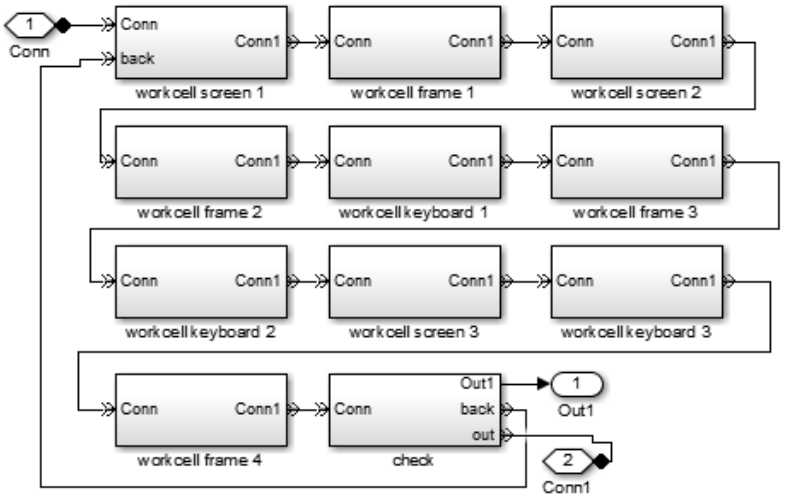


Figure 25. *Line subsystem consists of 10 workstations*

Each work cell has specific name which shows the operations for the work cell. In the next few lines two of the blocks will be explained. One block which assembles the screen models, and then the check block will be explained at last. There are some differences between the frame and screen and keyboard workstations, as every product should get the frame first and then it will be ready for other parts.

Workstation

The first workstation is for assembling the screens with different shapes and colours. When the products come inside this cell, first they get checked if they have the frame part, as it is the first part which is assembled. Then if the products have the frame part already they go through some other checking. Following figure shows the blocks used in this work cell for implementing the screen assembly.

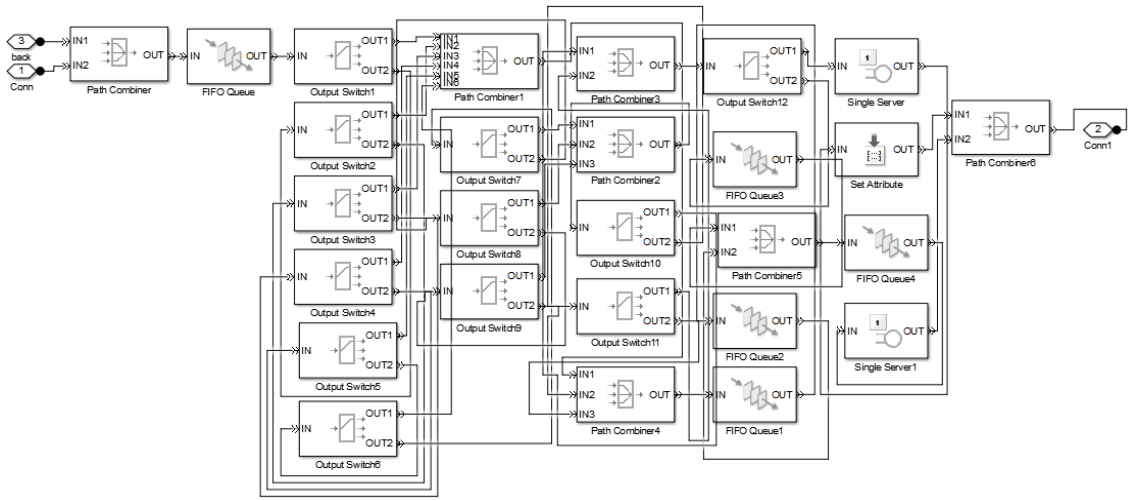


Figure 26. *Workstation with screen operation*

The Output Switches in this workstation check the attributes of the entities to route them through the bypass or the service block in the work cell. If the product's screen has been assembled before, it will go through bypass which takes 5 seconds to reach the other work cell, but if the product needs to get the screen, it will go through the service block, to get the screen, and it will take 25 seconds for the robot to assemble the screen on the product. As it is mentioned before the difference between the work cells for frame and the other work cells is that for assembling frames, there is no need to check if the product has it or not, but as it is represented before for the other work cells, this should be checked, as it is the first part which should be assembled on the part.

Check

In the checking part there are 18 blocks to check the attributes as it is shown in the following figure.

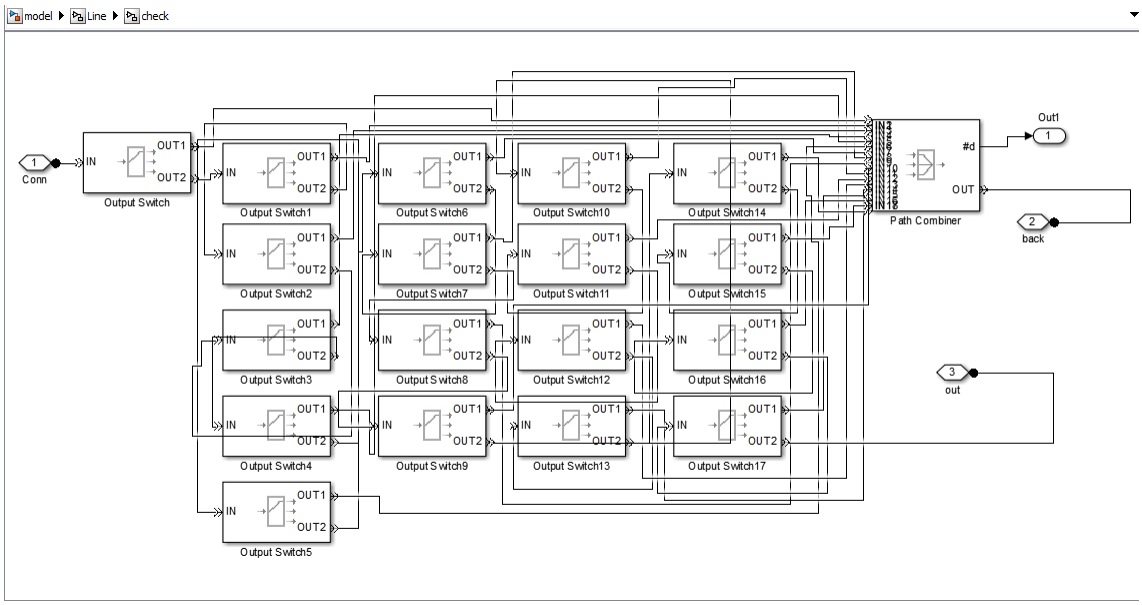


Figure 27. *Block for checking the assembled parts*

The blocks here check if the product is ready. Each block is for checking the attributes of the entity which is showing one of the specifications of the product. If the product was ready, it will go to the delivery section and if there was any deficiency, this checking block will send the product to next cycle in the work stations to get all the parts done.

4.2.3. Deliver

This section is the last one in this system and it delivers the products as demanded. In this section some scopes are implemented for showing the conditions of the entities and the products are delivered to the customer. Each scope is for monitoring different timing of the entities. Following figure is the overview of delivery section with the blocks used inside.

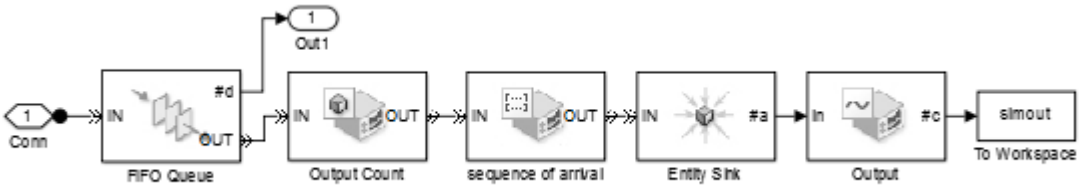


Figure 28. *Deliver block for completed products*

4.3. Connecting GUI to Simulink

After simulating the assembly line in MATLAB, in this part the connection between GUI and Simulink will be explained. When the GUI is created, it needs to communicate with Simulink for sending and receiving data. In this work, two different methods for ordering products has been implemented. The first one is the manual way, which gives the user or customer this chance to choose the products manually. The second way is the automatic way, in which the system produces a number of products randomly, but the amount of products is changeable by the user.

4.3.1. Manual ordering

When the user runs the manual program in MATLAB, a window will be open, which is for loading, ordering and running the model. The window is shown in the next figure.

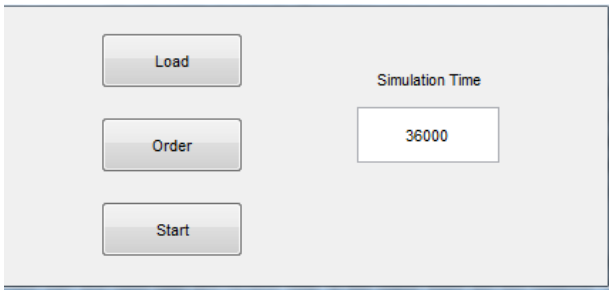


Figure 29. *GUI Window for configuring Manual ordering*

This window is known for GUI window and it is configured from a blank GUI. To create this window, three Push Buttons and one Edit Text are needed. The commands used for connecting this window to Simulink is given in appendix A. The Load button is for loading the model and Order button will open another window for choosing the properties of the products and at the end Start button will start the simulation.

To order the products with different properties, the button Order is built. This button opens another window for choosing the properties and number of products and orders. The figure below shows the window with the buttons and sliders.

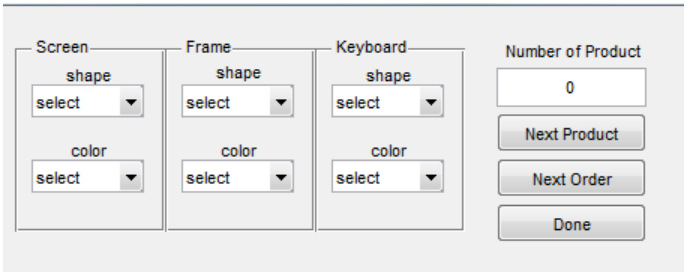


Figure 30. *GUI Window for choosing the features of products*

There are some sliders and push buttons for configuring the properties of the products. The call backs used in this window to connect it to Simulink model is shown in appendix B.

4.3.2. Automatic Ordering

In automatic ordering, the system requests the number of demanded products and orders, and will generate the products according to those numbers. The properties of the products are selected randomly to make different sets of products in one sequence. Following figure shows the window for connecting the GUI window to Simulink model and choose the numbers.

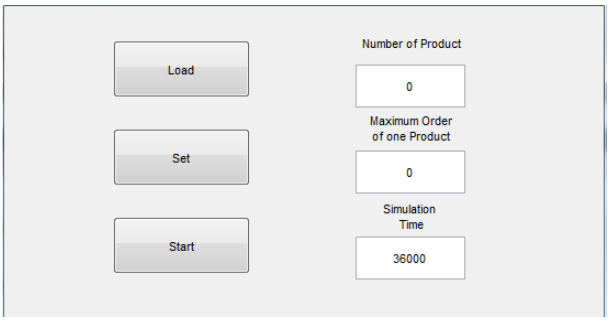


Figure 31. *GUI window for configuring Automatic ordering*

This window consists of three Push Buttons and Three Edit Text button. The push buttons are for loading model, setting the parameters, and starting the simulation. Edit boxes are for setting the number of products, orders, and for setting the simulation stop time.

4.4. Balancing the line according to Data acquisition

After simulating the automated line with MATLAB and connecting GUI to Simulink for ordering the products, some different implementations are examined for balancing the Line. The implementations were begun with changing the operations of each workstation and bypass time of the line. The implementations are explained briefly in following few lines.

4.4.1. One operation and two colours workstations with 10 seconds bypass

In this model there are 2 colours assigned for each work cell instead of three to see the difference in arrival times of the products. For example the work cell can produce the cellphone with red and blue frame, and the green frames are not allowed in this one. The model checks the attributes for two colours and then it will allow the products with these two colours inside the work cell. The following figure is the overview of this model.

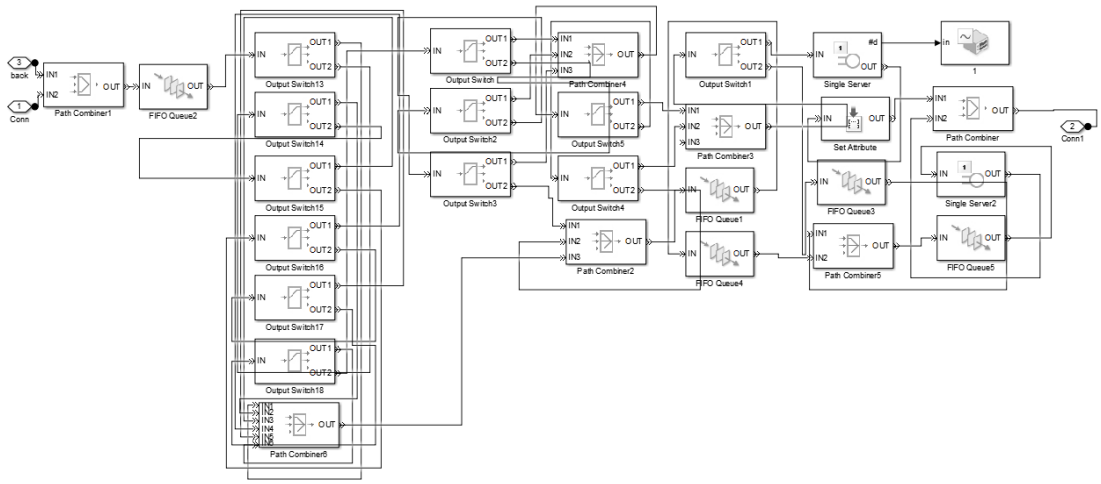


Figure 32. *Simulink model of screen work cell for one operation and two colours*

The results and outputs of these models will be discussed in the next chapter which consists of the results of the thesis work.

4.4.2. One operation workstations with 10 seconds bypass

In this implementation, each work cell assigned with one operation and three colours. For example the work cell can assemble the frames with blue, green and red colours. The bypass time is 10 seconds and operation time in work cells is 25 seconds. The next figure shows the overview of one work cell in Simulink.

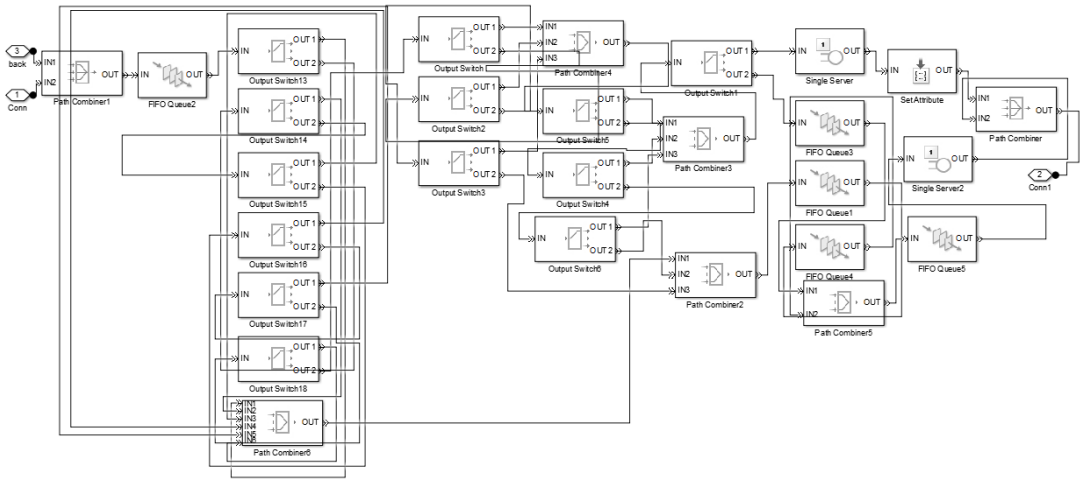


Figure 33. *Simulink model of screen work cell for one operation*

The specification of this model is to check one operation at a time and assembles one part of the cellphone with the bypass time of 10 second which is assigned in Single Server2.

4.4.3. One operations workstations with 5 seconds bypass

In this model, there is one operation and three colours per each work cell as well. The only difference is that the bypass time in this model is 5 seconds to reduce the cycle time. Reducing cycle time in a way that bypass times become less than the operation times will change the behaviour of the products in one cycle and waiting times in queues.

4.4.4. Two operations workstations with 5 seconds bypass

In this part of implementation, the work cells are capable of operating two different functions, for instance one can assemble frames and the other one screens. Next figure shows a work cell with the capability of assembling of screen and keyboard. In the first part of the model, the entities are checked for having the frame part as it is needed in the first place. Then the entities are checked for other specifications, then the routing will take the entities to the operation part or bypass part according to the attributes assigned. The figure below is the overview of this work cell.

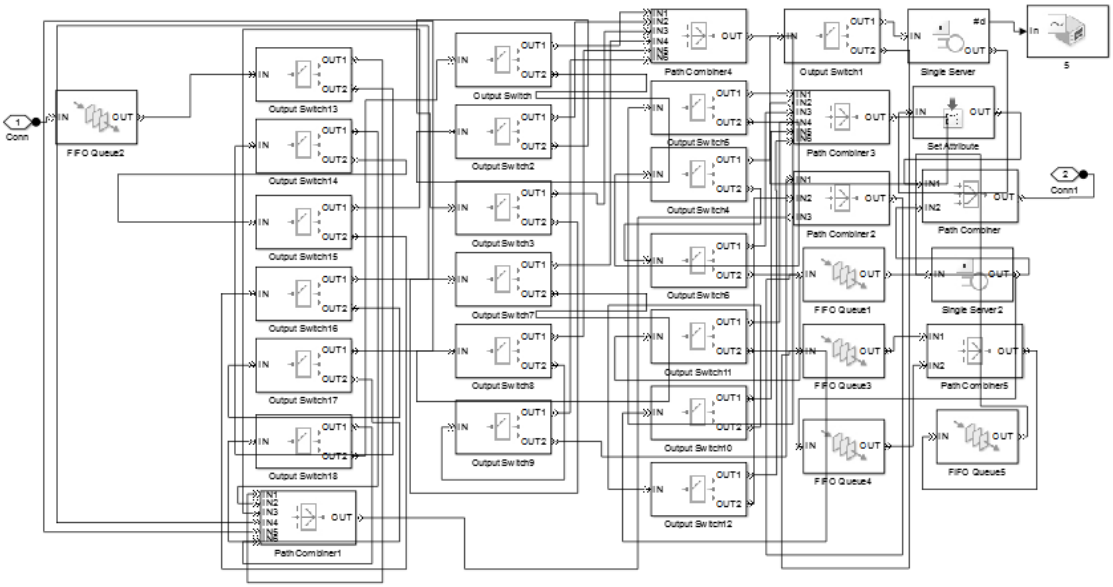


Figure 34. *Simulink model of screen and keyboard work cell with bypass of 5 seconds*

4.4.5. Balancing line with two colour distribution

Balancing a production line is an important part of optimization, which leads to cycle time reduction. The other aim of balancing in this work is to increase utilization of workstations. Therefore with proper distribution of the colours in work cells, the utilization would increase and cycle time will decrease.

The random distribution of colours in work cells causes abnormal behaviour in different situations. For instance, for some sequences it can cause low cycle time, and for other ones higher cycle time. Balancing the colour distribution will normalize the cycle times, which leads to normal behaviour of the line. Following table shows the random distribution of colours in each workstation.

Table 5. *Distribution of colours in unbalanced line*

Workstation	Colour distribution
WS1	Blue, Green
WS2	Green, Red
WS3	Red, Blue
WS4	Blue, Green
WS5	Green, Red
WS6	Red, Green
WS7	Blue, Green
WS8	Green, Blue
WS9	Blue, Red
WS10	Blue, Green

After testing some different distributions and calculating the cycle times, the balanced line, which showed good behaviour, implemented. The table below represents the distribution of this balanced line.

Table 6. *Distribution of colours in balanced line*

Workstation	Colour distribution
WS1	Blue, Green
WS2	Green, Red
WS3	Blue, Red
WS4	Blue, Green
WS5	Green, Red
WS6	Blue, Red
WS7	Blue, Green
WS8	Green, Red
WS9	Blue, Red
WS10	Red, Green

4.5. Optimization with Genetic Algorithms

Optimization has begun after implementing the line and balancing it with distribution of colours in workstations. Adding a button for optimization on the GUI window is the first step to generate the system with optimization. Following figure shows the same line with optimization button added.

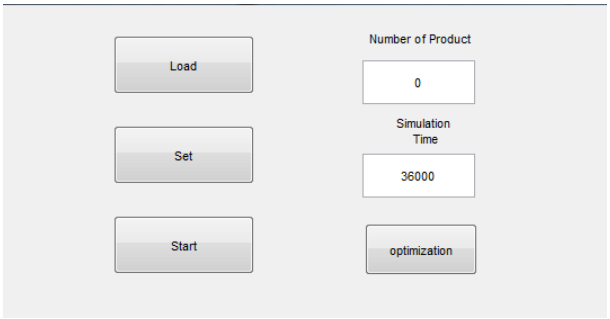


Figure 35. *GUI window for configuring the Ordering with optimization button*

To give the system the capability of GA configuration, the optimization button opens another window with capability of getting population and iteration numbers. Following figure is the overview of GA configuration.

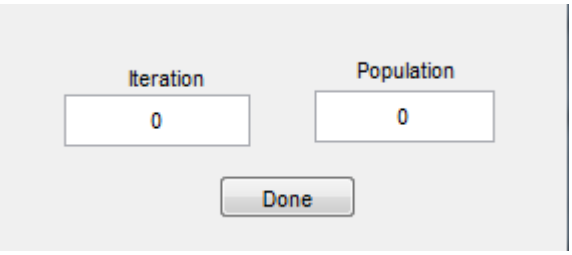


Figure 36. *GUI window for configuring GA parameters*

For implementing GA on a system there are some steps which should be considered. These steps should be implemented in a certain order, to result a good GA and consequence with good optimization.

4.5.1. Encoding

The first step is choosing encoding method, Permutation is selected here, based on the problem definition. In this method each product or entity gets a number as an attribute for making the orders in sequence. This number or value defined with Set Attribute block and will give each product the attribute to give them capability of sequencing.

4.5.2. Initial population

The second step is generating initial population with the products in the order. Choosing a population by the user will lead to have the number of data generations. After changing the sequences of the products in each generation, there will be some data which is equal to the number of population. The script for generating initial population can be seen in appendix C.

4.5.3. Selection

After generating the initial population, selection methods, will select some of the best sequences, based on their fitness. Selection is an important part here as the wrong method in selection leads to bad results. But with good selection the results can improve.

In this thesis work 2 different methods of selection have been implemented. Roulette Wheel selection and Elitism are the selection methods, used here to show and compare the results of GA with different selection methods.

1. Roulette Wheel Selection

Roulette Wheel Selection (RWS) is one of the most common methods in selection. This method uses a selection method based on the fitness rate of the data. At first it will give a fitness rate to each member of population. Then it normalizes the rates to make the rates between 0, and 1. If the population is 3 and fitness rates according to the results are [2, 3,

5], after normalizing the fitness rates will be [0.2, 0.3, 0.5]. For these fitness rates there will be a probability rate [0.2, 0.5, 1], which defines how the members are selected. At the end, the individuals are selected according to these probabilities. The following table shows how these individuals are selected.

Table 7. *Selecting individuals according to the probabilities*

Select	Probability
1st	x < 0.2
2nd	0.2 < x < 0.5
3rd	0.5 < x < 1

This method selects the parents for reproduction of next generation. The script of this method is given in appendix C.

2. Elitism

To implement Elitism selection in GA, first the chromosomes should be sorted based on their fitness, and then according to how many parents are needed, the best fit ones will be selected. Script of this method is given in appendix C.

4.5.4. Crossover

After selecting some parents which is 6 parents here, the offspring should be generated. In this part the parents mate in a random order together and will reproduce the new population, with the help of crossover and then mutation. By crossover the parents will send some of the genes to the offspring and by using mutation the offspring will be changed for letting the chromosomes pass the local optimal.

The first step is to select parents to mate with each other, which is done randomly in this thesis in each iteration. After choosing two parent for mating, with using single point crossover, the chromosome will be divided to two parts. The first part of the first parent will be chosen for the first offspring and the second part of the second parent will be chosen for the second offspring. The other bits of the offspring will be chosen with respect to sequence of the bits in the parents. The script of single point crossover in MATLAB is given in appendix C.

4.5.5. Mutation

Mutation is a way to change the chromosomes and give the offspring, capability of skipping the local optima. In this part every offspring changes with mutation. Mutation rate in this thesis work is low as the purpose of mating is to keep the basic genes of the parents in every iteration. Therefore two bits in each offspring will be changed. This changing

will be random and for each offspring, mutation will select two random bits for flipping. The script for mutation in GA for this system is given in appendix C.

5. RESULTS

In this chapter the results of different systems is documented. Different implementation of the system has different results which is represented in following sections. The first section represents the results of first implementation with operations and colours distribution. The second section represents the results and comparison of Genetic Algorithms in the system. The last section, represents the result of utilization of the workstations with different configuration.

5.1. Simulation of the assembly line

In this section the results of different systems which were implemented for balancing the line will be represented, and explained. In the implementation chapter, 4 different models were designed for balancing the line. The results of the different models varies, in the way it shows improvement. The results are explained for 100 products in 4 different models.

5.1.1. One-operation and two colours workstations with 10 seconds bypass

In this implementation, each workstation works with one operation duty and can assemble one part with two colours and three different shapes. For example the operation for the first workstation can be assembling frames on the cellphones. The bypass time is 10 seconds and the time for the operations are 25 seconds. The number of products which are in the order is 100, which are selected randomly with different features. The next figure shows the arrival time of products in the deliver section.

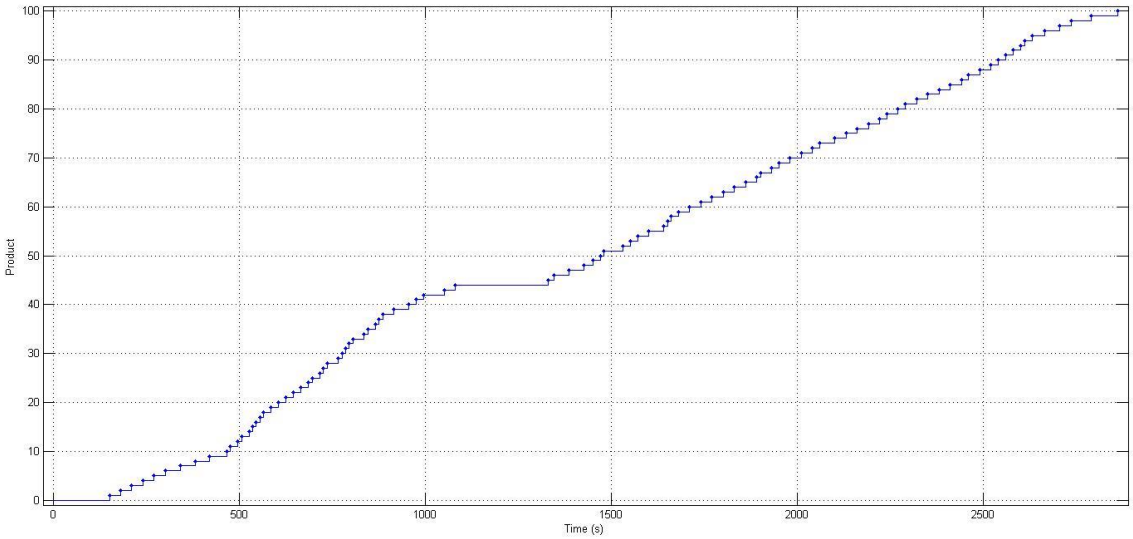


Figure 37. *Arrival time of the products in deliver section: One operation and two colours and bypass of 10*

This diagram shows that the arrival time of the products is not linear and this system should be balanced and after balancing the optimization can be implemented on the system. Arrival time of the first product (the production of the first product is completed) is 145 seconds and the last one arrives in 2860 seconds.

5.1.2. One-operation workstations with 10 seconds bypass

Changing the system in the way that each work cell assemble the products in three colours will lead to some changes in the behaviour of the products in the system, which will be represented and explained in the few lines. The next figure is the result of this system for 100 products.

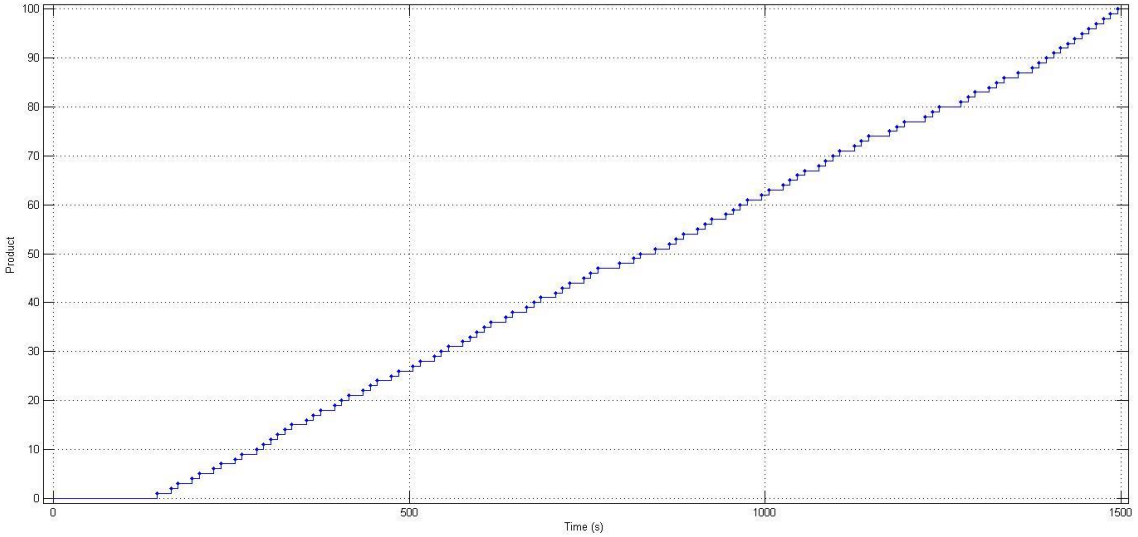


Figure 38. *Arrival time of the products in deliver section: One operation and bypass of 10*

This result shows that the first product will be delivered at 145 seconds and the last product will be arrived at 1495 seconds, which will be the time of completion of the assembly production. This figure shows that the arrival of the products are almost linear. Next figure represents the sequence of arrivals for each product, which is implemented according to the attributes of counter block.

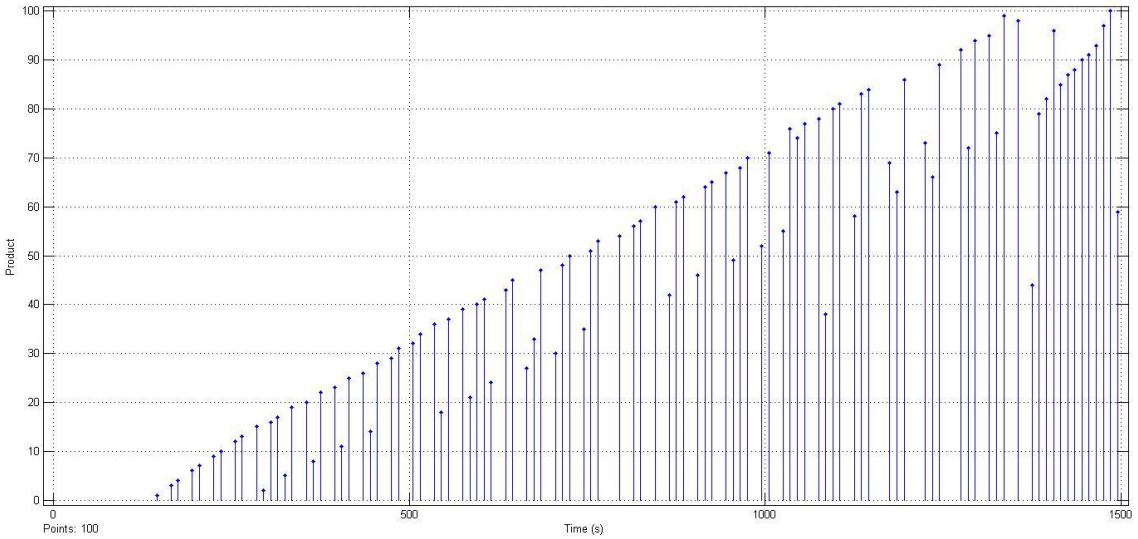


Figure 39. *Sequence of arrival for each product*

This figure shows how the products will arrive based on the sequence that they have in the ordering section. For example, the first product arrives first and third one arrives in the second place, and the time for arrival of the second product is 295 seconds from the starting time. The figure shows that the arrival of the products based on the sequence is not linear. This diagram can be used for the time that the customer needs the products in order.

5.1.3. One-operation workstations with 5 seconds bypass

The work cells in this section assemble one part, but the difference is that the bypass has reduced to 5 seconds. This reduction can help the line to assemble the products faster. Following figure is the result of this model on 100 products which are selected randomly.

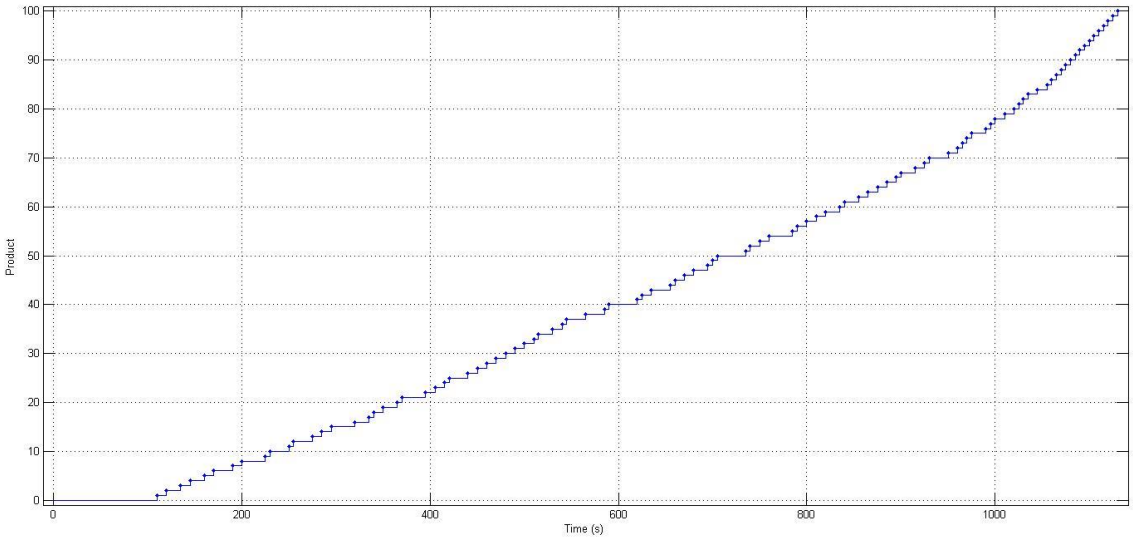


Figure 40. *Arrival time of the products in deliver section: One operation with bypass of 5 seconds*

The understandable theory of this diagram is that the arrival time is linear and the system behaves well for different order of products. The first completed product arrives at 110 seconds which shows that the system assembles faster. The last product arrives at 1130 seconds which shows an improvement in the completion time.

5.1.4. Two-operation workstations with 5 seconds bypass

The last model which is implemented for the line is the workstations with two operations and 5 seconds bypass. This model should be faster than the other models as it has the capability of assembling two parts in one workstation and has the bypass of 5 seconds which makes the waiting time less than previous models. Following figure shows the result of this system for 100 random products.

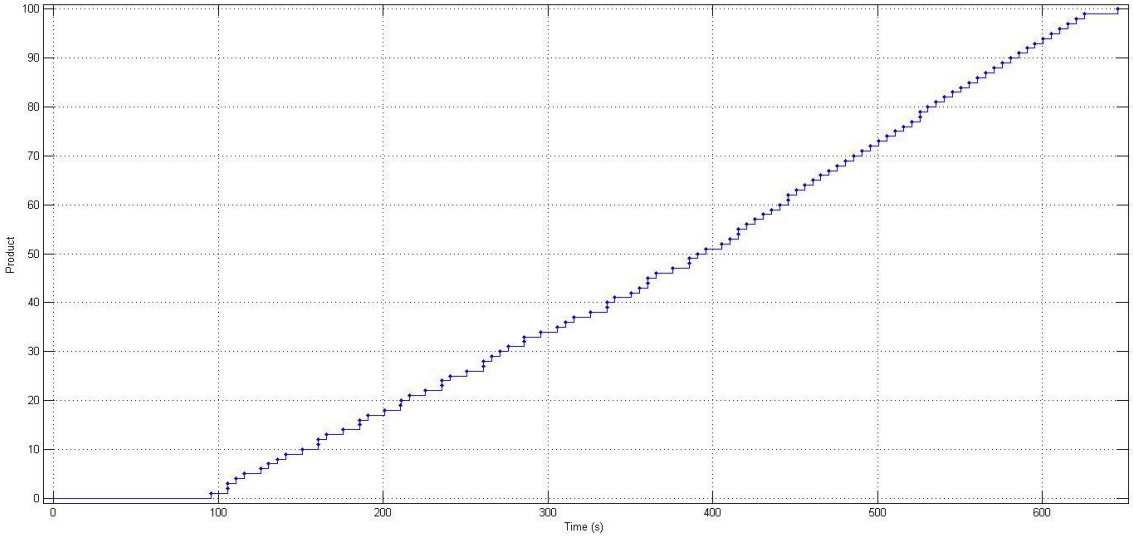


Figure 41. *Arrival time of products in deliver section: Two operations and bypass of 5 seconds*

Previous diagram shows that the model has a linear behaviour with random products. Linear behaviour is the sign of having a stable model for different products and optimization does not change the results. As it is perceivable from the diagram, the first product arrives in 95 seconds and last one in 630 seconds.

5.1.5. Comparison of different models

In this section the results in different models will be compared to show the improvements in the completion times and other parameters. Following table is the comparison of the models in different criteria.

Table 8. *Comparison of 4 different implemented models*

Model	First product arrival	Completion time	Linearity	Total cycles
One operation two colours bypass of 10	145	2860	non-linear	266
One operation three colours bypass of 10	145	1495	linear	141
One operation three colours bypass of 5	110	1130	linear	222
Two operations three colours bypass of 5	95	630	linear	118

This table shows the improvements of the models in completion time which is important in this thesis work and for implementing the Genetic Algorithms for optimization of completion time, the first model is selected as it shows non-linearity in the results.

5.2. Optimization

In this thesis work 4 different methods of GA has been implemented for optimizing the assembly line. These methods have the same basics but the difference is that, in each method one part of GA has been changed. GA with different methods show different behaviour in optimization and they are optimizing the line in different ways and with speed. There are two selection methods used for the implementation, one with Roulette Wheel Selection, and the other with Elitism. In one case mutation is implemented and in the other case mutation is neglected to represent the difference in local optimal. In the next few lines these 4 implementations will be explained briefly.

The system which is selected for this experiment as it has shown a non-linear behaviour is the model with one operation and two colours per each work cell and bypass of 5 seconds. The result for the model without optimization and 100 products is shown in the next figure.

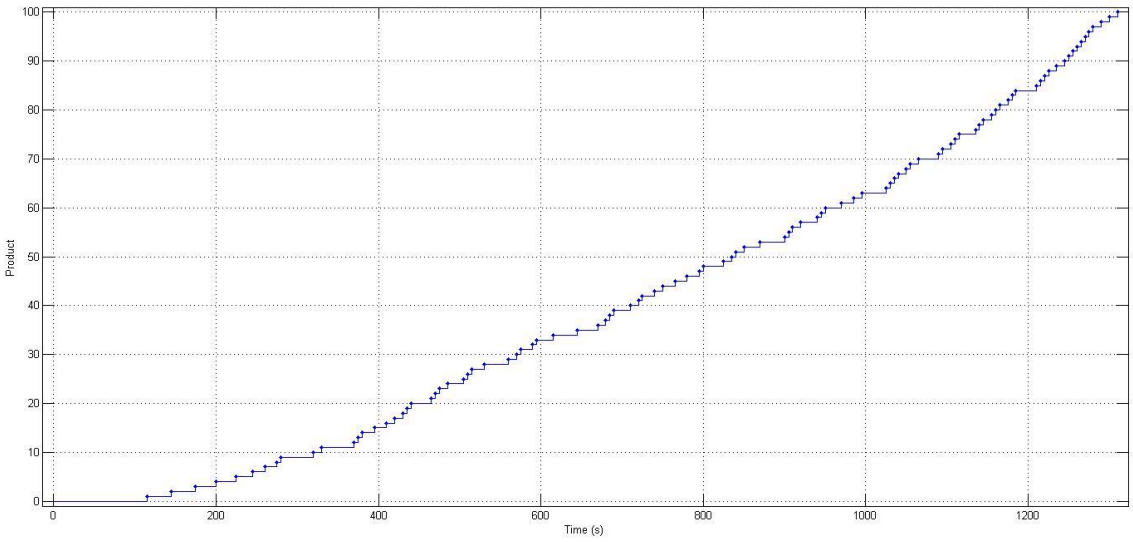


Figure 42. *Arrival time of the products in deliver section: One operation, two colours and bypass of 5 seconds*

The diagram shows that the first products arrives in 110 seconds and the last one arrives in 1310 seconds. This timing should be reduced with optimization which is described in the next section with different methods. The model for optimization is configured with these parameters; the population number is 20 and iteration of 10.

5.2.1. Genetic Algorithm with Roulette Wheel Selection

In this Genetic Algorithm, Roulette Wheel Selection method has been selected, and mutation ratio is set to 0 percent. After running the optimization on the model for 100 predefined products, the diagrams show that after 4 iteration the results represent in a stable way and after 8 iteration the GA finds the best solution and it stalks there. Following figure shows the first iteration which is the initial population.

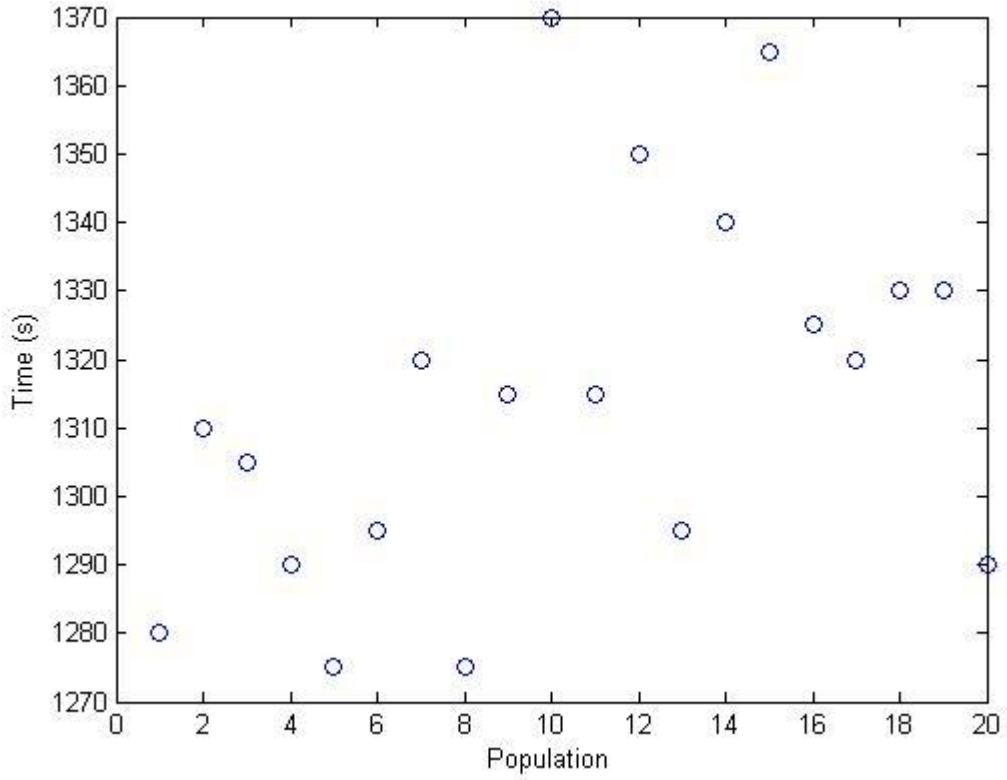


Figure 43. *First iteration for RWS without mutation*

The diagram shows the range of completion time for each chromosome. The maximum completion time is 1370 seconds and minimum is 1275 seconds. This diagram shows that the population produced in a wide range which is good in this case. Next figure is the 10^{th} iteration where the optimization has completed.

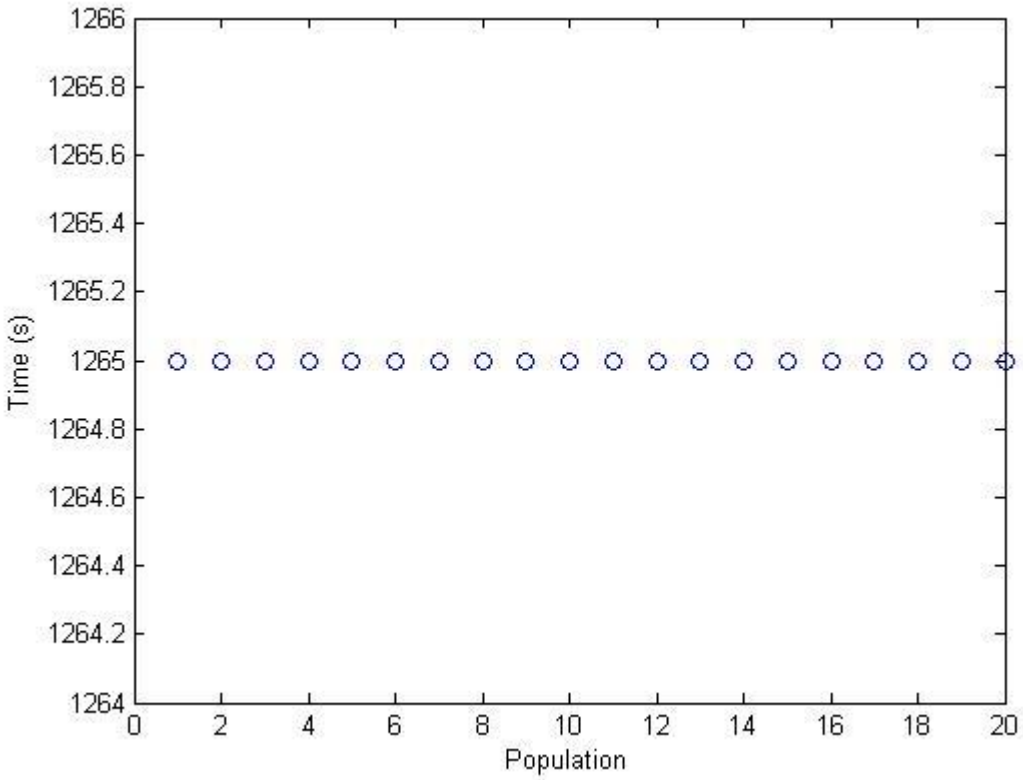


Figure 44. *10th iteration for RWS without mutation*

The 10th iteration shows that the population is stalked in 1265 seconds which is the local optima. The population gets stalk in this time from 8th iteration which represents lack of mutation. And in the other hand it is good that it finds a stable completion time for the population.

5.2.2. Genetic Algorithm with Roulette Wheel Selection with Mutation

In this part like previous section the selection method is RWS but this time mutation is considered with probability of 10 percent, which is set to low to keep the genes of chromosomes of each generation. Next figure shows the 10th iteration of this optimization.

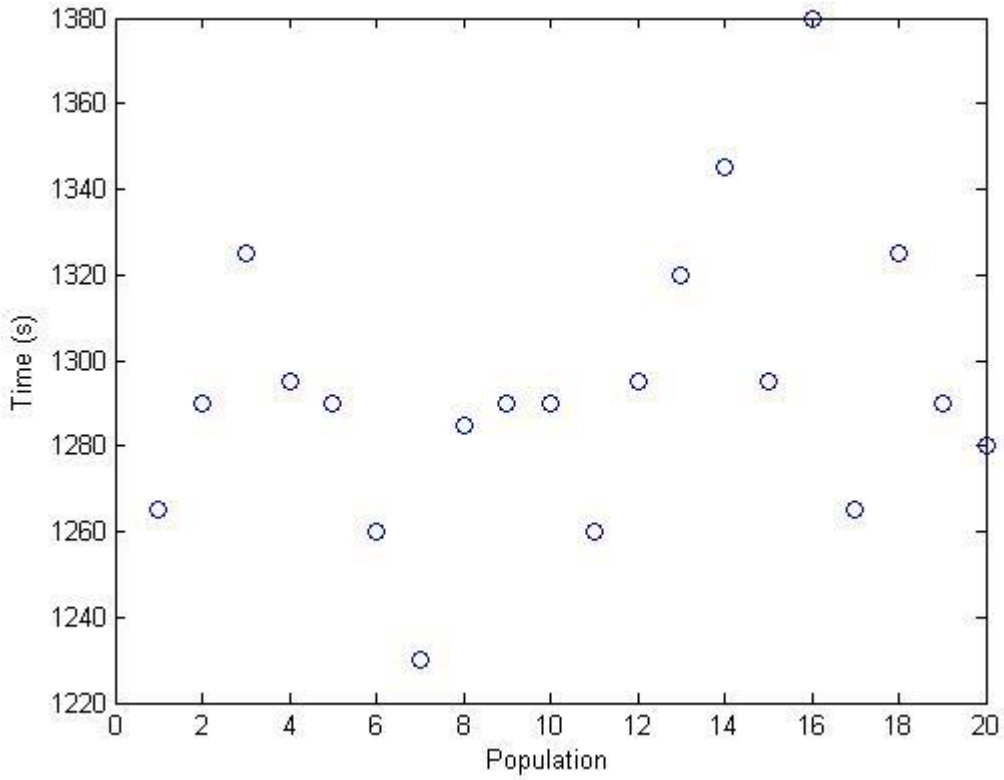


Figure 45. *10th iteration for RWS with mutation*

The conceivable information from this diagram is that, mutation does not let the population to stalk in one point. But there is a range of different completion time for the chromosomes in the population in each iteration. The last iteration shows that the range of completion time is from 1230 seconds to 1380 seconds and the mean is on 1290 seconds. This optimization method represents the chromosome with completion time of 1230 as the best one in the last iteration.

5.2.3. Genetic Algorithm with Elitism Selection

Genetic Algorithm with Elitism selection method is not as common as RWS but it has better probability of representing optima. Mutation is neglected in this section to see the influence of it in the algorithms. Following figure represents the 10th iteration of this method on the same 100 products which were chosen in the first part.

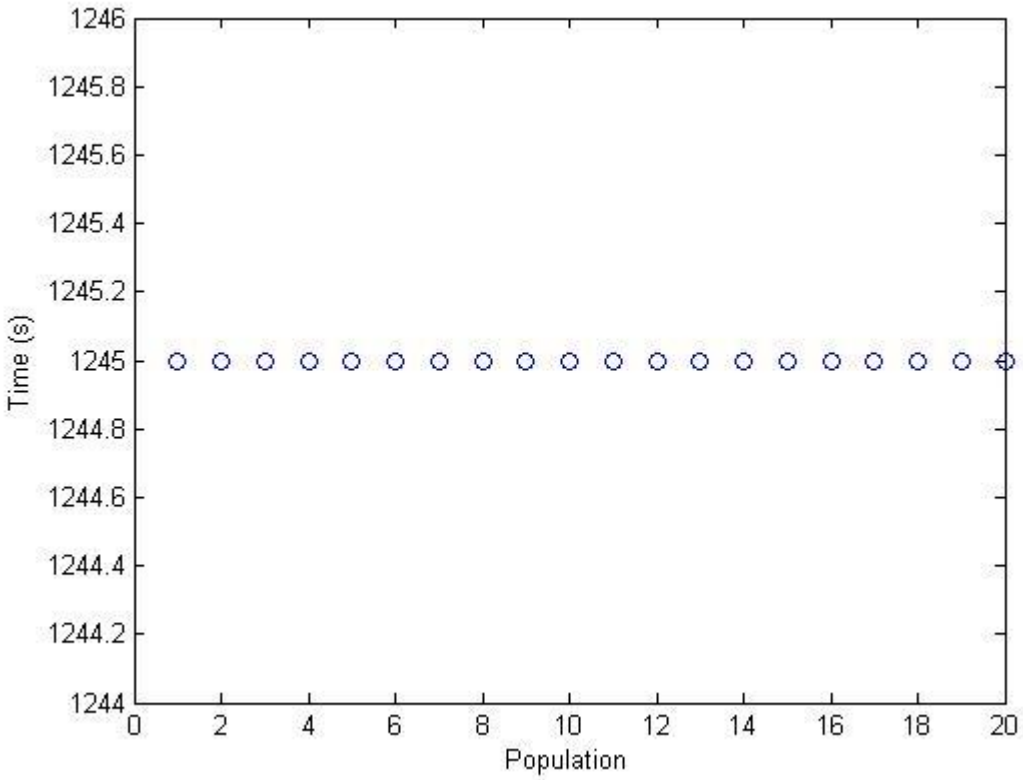


Figure 46. *10th iteration for Elitism selection without mutation*

This diagram is a good example of finding an optima which is not in the range of initial population. The last population has the completion time of 1245 which is not in the range but it stalks there and finds a chromosome which has good genes. In this method the completion time stalks in this time from the 5th iteration and it shows that this system is fast and better in this case.

5.2.4. Genetic Algorithm with Elitism Selection with Mutation

The last algorithm which is implemented is with Elitism selection and mutation, to see the influence of mutation on this method. The mutation probability is 10 percent to keep the inheritance of the good genes. Following figure shows the 10th iteration of this optimization method on the same predefined 100 products.

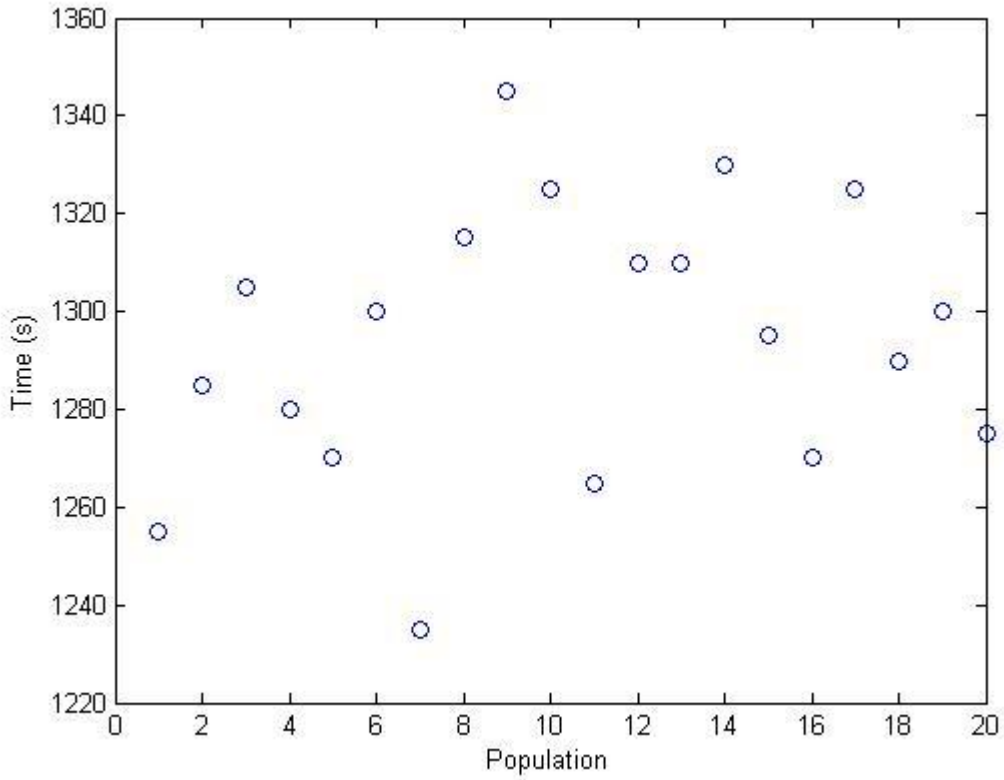


Figure 47. *10th iteration for Elitism selection with mutation*

Again the mutation makes the population to have different optima. The range here is from 1235 seconds to 1345 seconds, and the best chromosome has the completion time of 1235 seconds. The mean in this diagram is 1300 seconds.

5.2.5. Comparison of 4 different optimization methods

In this section, a table for comparing the 4 different methods is represented. The table shows the speed of convergence and stability of the methods. Next table compares the 4 models in 1st iteration, 5th iteration, 10th iteration, and the best time they find in the last iteration.

Table 9. *Comparing 4 models in different iterations (in seconds, in percent)*

Model	1st iteration	5th iteration	10th iteration	Best	Improvement
RWS	1275 - 1370	1265 - 1280	1265 - 1265	1265	10%
RWS with mutation	1270 - 1370	1255 - 1345	1230 - 1380	1230	12%
Elitism	1270 - 1390	1245 - 1245	1245 - 1245	1245	11%
Elitism with mutation	1245 - 1340	1240 - 1360	1235 - 1345	1235	12%

In this table each row shows the algorithm used for optimization. The second and third and fourth columns are for showing the range of the completion times in the 1st and 5th and 10th iterations. The fifth column shows the best completion time among the range. In the last column the improvement has been calculated. The improvement has been calculated from subtracting the best completion time from the worst case scenario which is made in the first iteration, and then dividing to the time of the completion.

As it is conceivable from the table, Elitism is the fastest method which gets stable in the 5th iteration. And the best result is for RWS with mutation but the problem with this method is even 10 percent mutation probability is changing the chromosomes a lot, in a way that they do not keep the genes of the parents. In the improvement column, the percentage of improvements are calculated. For example in the first row which is for RWS, can be seen, in the worst case scenario, if the order would be just started to be manufactured, it can take 1370 seconds to complete the order. The application of genetic algorithm already on the first iteration can give a significant improvement of 8%. Furthermore after 10 iteration the total improvement comparing to the worst case scenario is 10%. In this simulation the longest time to complete the production is 1390 seconds and the best time is 1230 seconds, which gives the tolerance of 160 seconds. This parameter represents that in 100 products this optimization method can improve the completion time up to 12 percent.

5.3. Utilization

In this part utilization of workstations with different models and algorithms are represented and compared. Instead of effective production rate, in the line the bottle neck rate can be used. The bottleneck rate is the rate of the work cell having the highest long term utilization. To calculate the bottleneck of the work cell, the capacity of the work cell in one hour should be calculated. For example, in the work cells of the model which is implemented in this work, the bottle neck rate can be calculated as:

$$Bottleneck\ rate = \frac{60 \times 60}{25} = 144$$

Where the numerator of the fraction is the seconds in one hour and denominator is the time every product needs to be in the work cell, as it is 25 seconds here.

For calculating the utilization of a work cell, each one should have a scope to show the arrival of the products. To calculate the throughputs of each work cell, the simulation will run for one hour and then the scope counts the number of products which go through the work cell.

In this section the utilization of the work cells of some models will be represented. As the optimization algorithm is implemented for the model with one operation and two colours

with bypass of 5, therefore the utilization is calculated before optimization and after optimization for that model. Next figure shows the throughput of a balanced work cell before optimization.

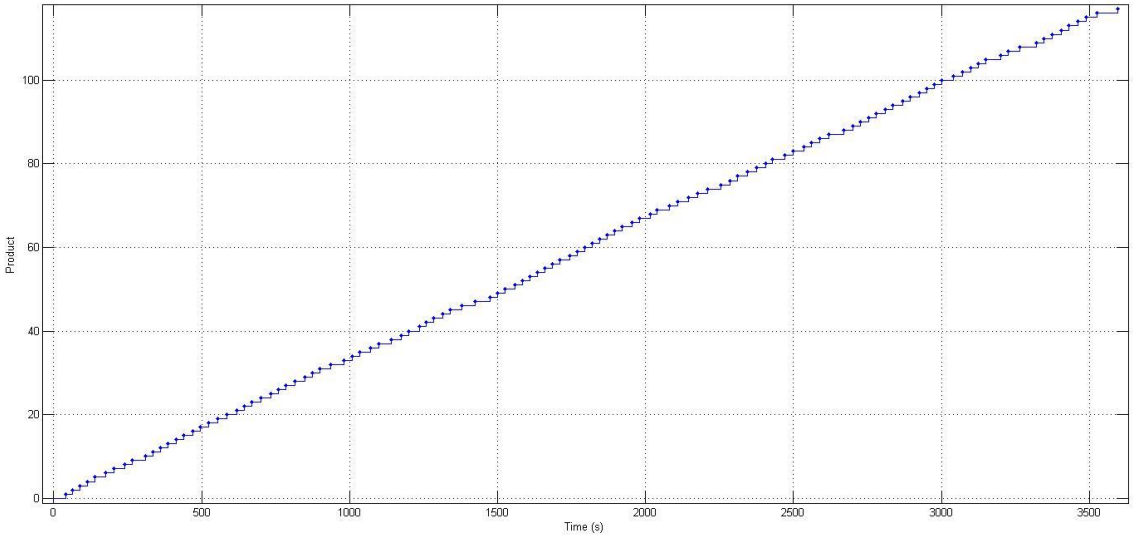


Figure 48. *Throughput of a work cell in one hour*

The utilization of this work cell is calculated in one hour, and as the figure shows that the throughput for the work cell is 118. Therefore the utilization will be;

$$Utilization = \frac{118}{144} = 81.9\%$$

After checking the scopes and counting the throughputs of each work cell, utilization is calculated. The following table shows the utilization of 10 work cells of this system, and the comparison of three different systems.

Table 10. *Throughput and utilization of work cells*

	Unbalanced Model		Balanced Model			
			Without Optimization		With Optimization	
Cell	Throughput	Utilization	Throughput	Utilization	Throughput	Utilization
1	92	63.9%	108	75%	115	79.9%
2	120	83.3%	118	81.9%	120	83.3%
3	92	63.9%	104	72.2%	108	75%
4	119	82.6%	112	77.8%	115	79.9%
5	99	68.8%	101	70.1%	105	72.9%
6	94	65.3%	96	66.7%	102	70.8%
7	105	72.9%	102	70.8%	108	75%
8	110	76.4%	105	72.9%	110	76.4%
9	79	54.8%	85	59%	98	68.1%
10	86	59.7%	79	54.8%	90	62.5%
Total	996	69.2%	1010	70.1%	1071	74.4%

Previous table represents some improvements in the work cells utilization after using optimization. Utilizations of the work cells in this table for the system before optimization is the representation of the system with mean completion time. This means the system can have worse completion time, and utilization than what mentioned in the table.

One of the best systems with good utilization of the work cells is the system with two operations for each work cell. This pushes the work cells to be busy with production most of the times. Following table shows the utilization of the balanced system with two operations per each work cell and after implementing the optimization.

Table 11. *Utilization and throughput of the model with two operations and bypass of 5*

Work cell	Throughput	Utilization
1	144	100%
2	143	99.3%
3	143	99.3%
4	143	99.3%
5	142	98.6%
6	142	98.6%
7	142	98.6%
8	141	97.9%
9	141	97.9%
10	140	97.2%
Total	1421	98.7%

This last table concludes that the system with two operations have better results as idle time of the work cells is close to zero and they are almost busy all the time for better and faster production.

6. CONCLUSION

This section discusses the conclusion of the implementation and results for this thesis work. In the second section, future work will be explained, due to possibilities of continuing this work in different aspects in future.

6.1. Conclusion of implementation and results

Simulation and optimization in manufacturing process are the part of automating the factory floor. Simulation of the production line in this thesis work is done with MATLAB, for data acquisition, such as, cycle time, and utilization of each workstation. MATLAB makes the simulation and programming of the line faster and easier. Monitoring the data and studying them, leads to implementation of models with different properties. In this thesis four models has been represented, with improvements in completion times for the products. The model with two operations for one workstation has the best results, but the model with two colours for each workstation has the non-linear results. Non-linearity in this case is the reason for choosing the model for optimization.

Implementing an optimization system on simulated manufacturing line is the intention of engineers, since it can decrease product delivery time. In this work four algorithms are implemented on the non-linear model for optimizing the production time. Genetic algorithm has been chosen for this work, because this algorithm is a method for searching the best solution among the solutions provided by the line. The results generated by the algorithms show that mutation in this case is not a good choice, if the user wants the convergence of the results. Mutation makes the algorithm slower with wide range of chromosomes. In the different crossover techniques used in this work, Elitism, finds the optima faster but in some cases, neglects better chromosomes with lower fitness. RWS which is more common than Elitism is better choice, although it is slower, but it considers the fitness of the chromosomes.

6.2. Future work

There are some studies and implementations, for continuing this thesis work. Future work for this work can be classified in following implementations:

- Balancing the workstations, with consideration of parts, colours, and shapes.
- Optimizing the system by considering the orders deadlines.

REFERENCES

[1] "Manufacturing Process." [Online]. Available: http://www.thelibraryofmanufacturing.com/. [Accessed: 20-Apr-2015].

[2] W. J. Hopp and M. L. Spearman, *Factory Physics: foundation of manufacturing management*, 2nd ed. New York: McGraw-Hill/Irwin, 2000.

[3] T. Ceccoli, "Quality, Speed, and Cost – How to Achieve All Three with Innovation - Viewpoints on Innovation." Kalypso.

[4] R. Singh, *Introduction to Basic Manufacturing Process and Workshop Technology*. 2006.

[5] "3.1.1. What is PPC?" [Online]. Available: http://www.itl.nist.gov/div898/handbook/ppc/section1/ppc11.htm. [Accessed: 22-Apr-2015].

[6] B. Rekiek, A. Dolgui, A. Delchambre, and A. Bratcu, "State of art of optimization methods for assembly line design," *Annu. Rev. Control*, vol. 26 II, pp. 163–174, 2002.

[7] L. I. Guiqin, Y. A. O. Zhiliang, Y. Qingfeng, and F. Minglun, "OPTIMIZATION METHODS OF THE PRODUCT ASSEMBLY LINE SYSTEM," pp. 19–22.

[8] S. Zhang, "Planning and Optimization of Assembly Line Simulation."

[9] C. Becker and A. Scholl, "A survey on problems and methods in generalized assembly line balancing," *Eur. J. Oper. Res.*, vol. 168, no. 3, pp. 694–715, 2006.

[10] R. C. C, J. R. Wilson, and E. Shannon, "Proceedings of the 1992 λProlog Workshop," 1992.

[11] F. Seila, "INTRODUCTION TO SIMULATION WHAT IS SIMULATION ? methodology and What lated ? Types of Systems can be Simu- USE SIMULATION ? as a Tool for Decision-Making," pp. 7–15, 1995.

[12] S. Krüger, *Simulation: Grundlagen, Techniken, Anwendungen*. Berlin, Newyork: de Gruyter, 1975.

[13] A. M. Law, M. G. Mccomas, A. M. Law, and P. O. Box, "No Title," pp. 49–52, 1998.

[14] S. J. E. Taylor, "Proceedings of the 2001 Winter Simulation Conference B.," no. 2, pp. 1605–1612, 2001.

[15] R. Choy and A. Edelman, "Parallel MATLAB: Doing it right," *Proc. IEEE*, vol. 93, no. 2, pp. 331–341, 2005.

[16] “MATLAB - The Language of Technical Computing - MathWorks Nordic.” [Online]. Available: http://se.mathworks.com/products/matlab/?refresh=true. [Accessed: 29-Apr-2015].

[17] “Simulink - Simulation and Model-Based Design - MathWorks Nordic.” [Online]. Available: http://se.mathworks.com/products/simulink/. [Accessed: 29-Apr-2015].

[18] “Discrete Event Simulation Software - SimEvents - Simulink - MathWorks Nordic.” [Online]. Available: http://se.mathworks.com/products/simevents/. [Accessed: 29-Apr-2015].

[19] The MathWorks Inc., *MATLAB - Creating Graphical User Interfaces, Version 7*. US, 2004.

[20] “MATLAB Scripts.” [Online]. Available: http://web.cecs.pdx.edu/~gerry/MATLAB/programming/scripts.html. [Accessed: 30-Apr-2015].

[21] L. T. Biegler, *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. Philadelphia: Society for industrial and applied mathematics, 2010.

[22] A. Astolfi, “Optimization An introduction,” *J. Anal. Psychol.*, vol. 55, no. 5, pp. 617–635, 2010.

[23] D. N. Kumar, “Introduction and Basic Concepts (iii) Classification of Optimization Problems,” *Optimization*, no. iii.

[24] S. Behera, S. Sahoo, and B. B. Pati, “A review on optimization algorithms and application to wind energy integration to grid,” *Renew. Sustain. Energy Rev.*, vol. 48, pp. 214–227, 2015.

[25] F. Werner, “Genetic algorithms for shop scheduling problems: A survey,” *Preprint*, 2011.

[26] P. Schedules and B. Algorithm, “Flow Shop Scheduling,” .

[27] Z. Shi, *Advanced Artificial Intelligence*. River Edge: WSPC, 2011.

[28] Y. Baştanlar and M. Özuysal, “Introduction to machine learning,” *Methods Mol. Biol.*, vol. 1107, pp. 105–128, 2014.

[29] M. Metcalf, “A first encounter with f90,” *ACM SIGPLAN Fortran Forum*, vol. 11, no. 1, pp. 24–32, 1992.

[30] J. Shapiro, “Genetic algorithms in machine learning,” *Mach. Learn. Its Appl.*, pp. 146–168, 2001.

[31] S. Ghoshray and K. K. Yen, “More efficient genetic algorithm for solving optimization problems,” *1995 IEEE Int. Conf. Syst. Man Cybern. Intell. Syst. 21st Century*, vol. 5, 1995.

[32] Y.-H. Liao and C.-T. Sun, “An Educational Genetic Algorithms Learning Tool,” *Educ. IEEE Trans.*, vol. 44, no. 2, 2001.

[33] C. R. Reeves, “A genetic algorithm for flowshop sequencing,” *Comput. Oper. Res.*, vol. 22, no. 1, pp. 5–13, 1995.

[34] R. L. Y. M. A. Rangel-Merino, J. L. López-Bonilla, “Optimization Method based on Genetic Algorithms,” *Apeiron, Vol. 12, No. 4*, vol. 12, no. 4, pp. 393–408, 2005.

[35] C. Science and S. Engineering, “A Review Paper on Different Encoding Schemes used in Genetic Algorithms,” vol. 4, no. 1, pp. 596–600, 2014.

[36] “Main page - Introduction to Genetic Algorithms - Tutorial with Interactive Java Applets.” [Online]. Available: http://www.obitko.com/tutorials/genetic-algorithms/index.php. [Accessed: 30-Apr-2015].

[37] “Genetic Algorithms Demystified - CodeProject.” [Online]. Available: http://www.codeproject.com/Articles/707505/Genetic-Algorithms-Demystified. [Accessed: 02-May-2015].

[38] M. Gen and L. Lin, “Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey,” *J. Intell. Manuf.*, pp. 1–18, 2013.

[39] P. G. Naik, “Application of Genetic Algorithm to Mass Production Line for Productivity Improvement,” pp. 125–131, 2013.

[40] A. Nieto, “FASTory FAST,” Tampere, 2014.

APPENDIX A: MANUAL ORDERING MATLAB CODE

```
% --- Executes on button press in order.
function order_Callback(hObject, eventdata, handles)
% hObject    handle to order (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA);

GUI_2_handle=simple2;
load_system('simeventslib');
add_block('order/Order','model/Production Order/Order1','Position',[30
30 60 60]);
add_line('model/Production Order','Order1/RConn1','Path Com-
biner/LConn1');
add_block('production/Production','model/Production Order/Order1/Pro-
duction1','Position',[30 30 60 60]);
add_line('model/Production Order/Order1','Production1/RConn1','Path
Combiner/LConn1');
set_param('model/Production Order/Order1/Production1/Infinite Serv-
er','ServiceTime','0');

% --- Executes on button press in start.
function start_Callback(hObject, eventdata, handles)
% hObject    handle to start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
sim('model');



% --- Executes on button press in load.
function load_Callback(hObject, eventdata, handles)
% hObject    handle to load (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
open_system('model');



function simulation_Callback(hObject, eventdata, handles)
% hObject    handle to simulation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of simulation as text
%        str2double(get(hObject,'String')) returns contents of simula-
tion as a double
Val = get(hObject,'String');
Val=num2str(Val);
set_param('model','StopTime',Val);
```

APPENDIX B: ORDER CONFIGURATION MATLAB CODE

```
% --- Executes on selection change in screenshape.
function screenshape_Callback(hObject, eventdata, handles)
% hObject    handle to screenshape (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns screenshape
contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
screenshape
% Determine the selected data set.
strScreenShape = get(hObject, 'String');
valScreenShape = get(hObject,'Value');
global p;
global t;
productionScreenShape = sprintf('model/Production Order/Order%i/Produc-
tion%i/Set Attribute1',p,t);
% Set current data to the selected data set.
switch strScreenShape{valScreenShape};
case 'rec' % User selects peaks.
   set_param(productionScreenShape,'AttributeValue','1|2|2');
case 'oval' % User selects membrane.
   set_param(productionScreenShape,'AttributeValue','2|1|2');
case 'curvrec' % User selects sinc.
   set_param(productionScreenShape,'AttributeValue','2|2|1');
end
guidata(hObject,handles)
% Save the handles structure.
```

Command line for choosing the number of each product:

```
function numberOfProduction_Callback(hObject, eventdata, handles)
% hObject    handle to numberOfProduction (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
NewVal = get(hObject,'String');
NewVal=num2str(NewVal);
global p;
global t;
productionTime    =    sprintf('model/Production    Order/Order%i/Produc-
tion%i/Time-Based Function-Call Generator',p,t);
productionService   =   sprintf('model/Production   Order/Order%i/Produc-
tion%i/Infinite Server',p,t);
 set_param(productionTime,'NumberOfEventsPerPeriod',NewVal);
 set_param(productionService,'ServiceTime','0');
```

The Done button will close the window:

```
% --- Executes on button press in done.
function done_Callback(hObject, eventdata, handles)
% hObject    handle to done (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(simple2);
```

APPENDIX C: GENETIC ALGORITHM MATLAB CODE

Initial population

```
for m=1:Pop;
Rand=randperm(NumOfProd-1,NumOfProd-1);
Rand=Rand+1;
set_param('model1/Production Order/Order1/Production1/Set At-
tribute','AttributeValue','1');
for i=2:NumOfProd;
    r=sprintf('model1/Production Order/Order1/Production%i/Set At-
tribute',i);
    t=Rand(1,i-1);
    t=num2str(t);
set_param(r,'AttributeValue',t);
end
sim('model1');
Rand=[1 Rand];
time=simout.time(NumOfProd+1);
times(m,:)=time;
plot(m,time,'o')
hold on
tot=[Rand time];
total(m,:)=tot;
end
```

Roulette Wheel Selection

```
b=1;
Minimum = min(times);
for p=1:Pop;
    if total(p,NumOfProd+1)==Minimum;
        minim(b,:)=total(p,:);
        b=b+1;
    end
end
Sum=0;
for z=1:Pop;
   Sum=times(z,1)+Sum;
end
sorted=sortrows(total,NumOfProd+1);
maximum=max(times)+1;
Sum=0;
for v=1:Pop;
    fitness(v,1)=maximum-sorted(v,NumOfProd+1);
    Sum=fitness(v,1)+Sum;
end
cum=0;
for v=1:Pop;
    fitRate(v,1)=fitness(v,1)/Sum;
    cum=fitRate(v,1)+cum;
    fitCum(v,1)=cum;
end
n=1;
for v=1:5;
    random=rand;
    for i=1:Pop;
```

```
        if random<fitCum(i,:)
            selected(n,:)=sorted(i,:);
            n=n+1;
            random=1;
        end
    end
end
sizeMinim=size(minim,1);
if sizeMinim>1
randomMin=floor((sizeMinim-1).*rand(1)+1);
selected(6,:)=minim(randomMin,:);
else
  selected(6,:)=minim(1,:);
end
```

Elitism Selection

```
sorted=sortrows(total,NumOfProd+1);
selected=sorted(1:6,1:NumOfProd);
```

Crossover

```
for i=1:6;
line(i,:)=selected(i,(1:NumOfProd));
end
%choosing parents
for d=1:Pop/2;
    matrix(d,:)=randperm(6,2);
end
for q=1:Pop/2;
selected1(q,(1:round(NumOfProd/2)+1))=line(ma-
trix(q,1),(1:round(NumOfProd/2)+1));
b=round(NumOfProd/2)+2;
for i=1:NumOfProd;
    if any(line(matrix(q,2),i)==line(ma-
trix(q,1),(1:round(NumOfProd/2)+1)))==0
     selected1(q,b)=line(matrix(q,2),i);
      b=b+1;
    end
end
end
for w=Pop/2+1:Pop;
k=1;
selected1(w,(round(NumOfProd/2):NumOfProd))=line(matrix(w-
Pop/2,2),(round(NumOfProd/2):NumOfProd));
for j=1:NumOfProd;
    if any(line(matrix(w-Pop/2,1),j)==line(matrix(w-
Pop/2,2),(round(NumOfProd/2):NumOfProd)))==0
        selected1(w,k)=line(matrix(w-Pop/2,1),j);
        k=k+1;
    end
end
end
```

Mutation

```
mutNum=floor((NumOfProd-1).*rand(Pop,2)+2);
for e=1:Pop;
    test2=selected1(e,:);
    test2([mutNum(e,1),mutNum(e,2)])=test2([mutNum(e,2),mutNum(e,1)]);
    selected1(e,:)=test2;
end
```