AUTOMATIC SUBASSEMBLY SEQUENCING

FOR OFFSHORE RIGS

ABBOTT ERNEST LESLIE SIDNEY

(B.Sc.(Special), London University)

(B.A.(Hons) Thames Polytechnic)

(M.Th. London University)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2015

This page is intentionally left blank

.

DECLARATION

I hereby declare that this thesis is my original work

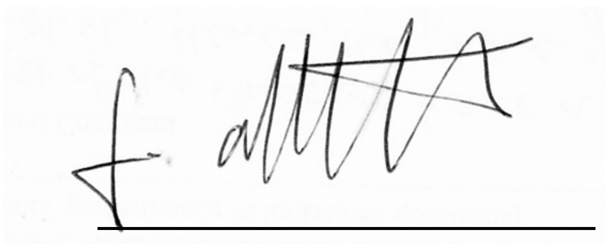
and it has been written by me in its entirely.

I have duly acknowledged all the sources of information

which have been used in the thesis.

This thesis has also not been submitted for any degree

in any university previously.



ABBOTT ERNEST LESLIE SIDNEY

12 February 2015

This page is intentionally left blank.

ACKNOWLEDGEMENTS

Heading the list of acknowledgements is my dear wife, *Kee Tip*. If it were not for her continuous support, encouragement and love during this PhD journey this work would never had got started. I dedicate this thesis to her.

Next are my two daughters, *Rebekah* and *Hannah*, who while I was engaged in this work were involved in their 'A' and 'O' level exams. They may have some understanding of what daddy was going through, when he was also taking exams.

A/Prof *David Chua* first suggested to me to think about doing a PhD. Initially I was very reluctant, but his persistence won the day. He has provided help, guidance and encouragement throughout this project help me to stay on the right path when I was seeming going astray.

Two people from Jurong Shipyard have provided valuable advice and assistance at stages of this research. Firstly, Mr *Juhari*, who willingly and joyfully shared with me his experience in the shipbuilding industry that goes back more than 40 years. He patiently explained how block assembly takes place, showing me all the difficulties and challenges that he has encountered. Secondly, Mr *Dermond Tan*, engineer and section leader of Jurong Integrated Services, who validated the welding length calculations produced by one of my programs. Mr Tan also taught me the seemingly endless possibilities of part connections, combinations and orientations.

My office colleagues of the past, A/Prof. *Song YuanBin*, Dr. *Liu Zhou*, Dr. *Shen LiJun*, Dr. *Yeoh Kerwei*, Dr. *Alireza Khalili*, Dr. *Meghdad Attarazah* and of the present, Dr. *Nguyen Thi Qui,* Ms. *Zhu Lei* are to be thanked for their support and encouragement and Mr. *Ahmad Tashrif* for reading the thesis.

I record my thanks to A/Prof *Meng Qiang*¸ for always being available for discussion when I wanted to "bounce ideas off him". He was always encouraging in our conversations.

Finally, I am grateful to the department of civil and environmental engineering for accepting my application for the PhD when many of my contemporaries had either retired or were contemplating retiring.

This page is intentionally left blank.

TABLE OF CONTENTS

SUMMARY

Offshore rigs are composed of *blocks* which are of steel construction and the basic building component of offshore rigs. Blocks vary in size, weight and number of parts. The basic component of a block is a *plate.* Plates vary in shape and size, depending on the design of the block. Plates will have other parts welded to them, such as *stiffeners, flanges* and *brackets.* When this happens the whole construction is referred to as a *subassembly*. Subassemblies welded together are termed *sub-blocks.*

The major work component in constructing blocks for offshore rigs is welding. Welding in different orientations requires a different amount of time. The welding of parts is in one of 3 positions, horizontal, vertical and overhead. Welding is either automated or manual. Automated is only for horizontal welding. Manual welding is in any 3 positions. For the same unit length, vertical welding is half the speed of horizontal welding and overhead welding is one-third the speed of horizontal welding. Automated welding is twice the speed of manual horizontal welding.

Subassemblies, after construction, are assembled on a base panel, or *on-block*. On-bloc sub-block construction requires a vertical weld, which is more time consuming than off-block horizontal welding. Having as much welding as possible off-bloc reduces the work content of the block and speed up the construction process.

This research focuses on optimizing the subassembly sequencing such that as many as possible subassemblies are combined with other subassemblies off-block in order to construct as many sub-blocks as possible. This leads to a reduction in the welding on-block and thus improves the construction time by eliminating unnecessary work.

The motivation behind this work is Lean principles; this can be seen in the reduction of work content and reducing any double handling of parts.

This work employs Polychromatic set theory (PST) as its knowledge base. The knowledge is used by a reasoning engine that identifies part connectivity and part relationship. The knowledge is employed in the sequencing engine to generate the plate assembly sequence. The sequencing engine, working with the basic unit of the plate, automatically identifies the correct construction sequence of the plates for the block construction. An inference engine, using the knowledge from the sequencing engine infers the plate usage, such as those that compose a panel. It also, identifies the parts, such as stiffeners, that are connected to a plate to compose a subassembly. Further, the engine calculates the contact length of parts and the welding length.

The optimizing engine uses the knowledge from the inference engine to optimize the combination of subassemblies into sub-blocks. The optimization is constrained by length, width, height and weight of the sub-block. Each subassembly and sub-block differs in work content, and is required at a different time at the assembly point (where subassemblies/sub-blocks are fitted to a base panel). The optimizing engine optimizes the welding work sequence using a Genetic Algorithm in such a manner that the work (subassemblies &/or sub-blocks) arrive at the required assembly point as near to Just-in-Time as possible.

Keywords: Offshore Rigs; Construction Sequences; Polychromatic Sets; Lean in Rig Construction.

LIST OF TABLES

LIST OF FIGURES

This page is intentionally left blank

Chapter 1. Introduction

1.1. Research Motivations and Background

The construction of offshore drilling rigs is a complex and labour intensive process. All successful construction processes require the implementation and execution of a proper construction schedule in order that progress may be monitored and managed effectively. While it is possible, at a macro level, to determine a broad-brush schedule based on estimated man power availability and estimated resource required to construct blocks, (Liu et al. 2010; Liu et al. 2011), this is not the case for micro level scheduling of individual sections of an offshore rig. The individual sections of an offshore rig are not uniform, differing in size and construction complexity. Unlike components in a production environment, where repetition is the order of the day, and assembly times of individual components are well known, the construction times of individual section is unknown. The uniqueness or non-uniformity of the individual sections is the main reason for this. Offshore drilling rigs are constructed from steel sections, which are commonly referred to as *blocks.* Blocks, (sometimes referred to as *ship's block,* a carry-over from the time shipyards only constructed ships), vary in all aspects of size, shape, weight and construction complexity. The weight of the block can be from a few tonnes to over a hundred tonnes. The upper limit is determined by the shipyard's crane lifting capacity, transportation capacity and space for construction.

A typical block is composed of hundreds of parts, nearly all differing in shape and size. All parts are welded together to construct a block. The process of constructing a block is a complex one; it is nothing like a production line, where operations are repeated in a fixed cycle. Blocks for offshore rigs are really one-off operations. Unique blocks with an offshore rig are commonplace.

The one-off nature of the block's construction presents challenges in scheduling and estimating the amount of work required for its construction. The welders and fitters working on the block will always start at the bottom of the learning curve for the particular construction, since each section to be welded would be different from the previous one welded. The only learning carried forward is that of general welding skills.

(The remainder of this page is intentionally left blank for formatting purposes.)

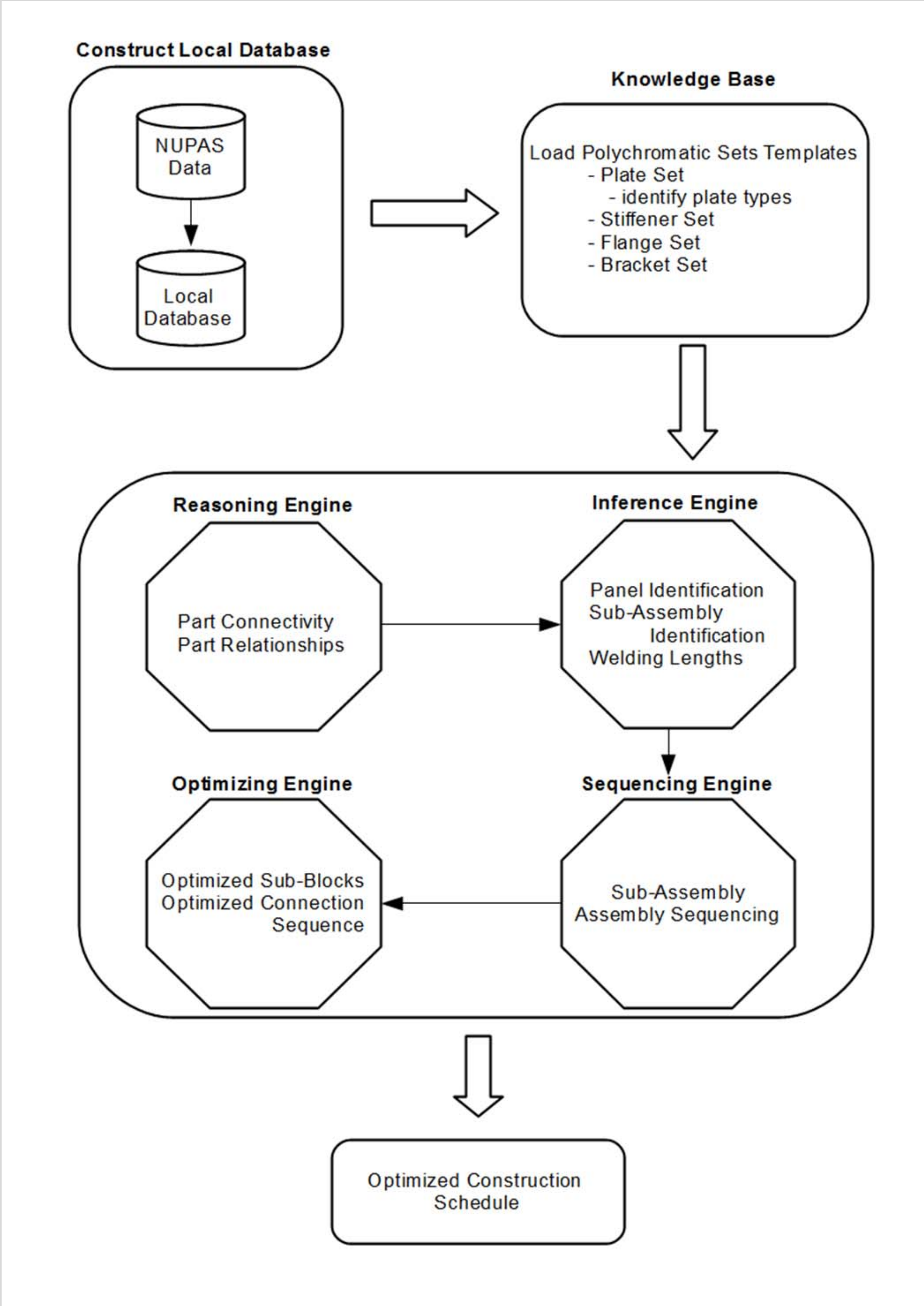


Figure 1-1: System Architecture

Developing an optimum assembly schedule for a block, which in practice will be a non-repeatable series of operations, is a daunting task. One cannot be carried out manually within a reasonable timescale. It clearly calls for a computer-based methodology to be developed from the bottom up as it were; a methodology that begins with the assembling of small parts to the assembling of the block itself. The work presented here, whose architecture is illustrated in Figure 1-1, proposes using Polychromatic Set Theory (PST) as the mathematical underpinning for part relationships. The relationships facilitate the use of a reasoning engine to identify connectivity and part relationships and builds upon the relationships. The product of the reasoning engine is used by an inference engine to derive the composition of panels and subassemblies as well as the calculation of the welding length. It is the calculation of the welding length that enables a far more accurate estimation of the total work involved in the construction of parts. A sequencing engine identifies the fundamental on-block assembly sequence of subassemblies. The optimizing engine optimizes the combination of subassemblies into sub-blocks that is then used in calculating the optimum construction sequence of the sub-blocks/subassemblies. An optimized construction schedule for the block that uses the result from the sequencing engine, together with the available labour resource, incorporating Lean principles where possible, produces a makespan for the block construction

1.2. Block and Part Relationships

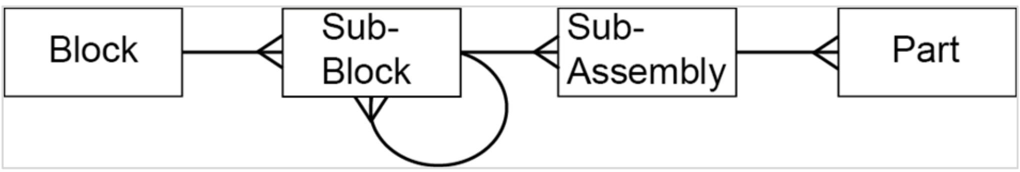


Figure 1-2: Block Part Relationships

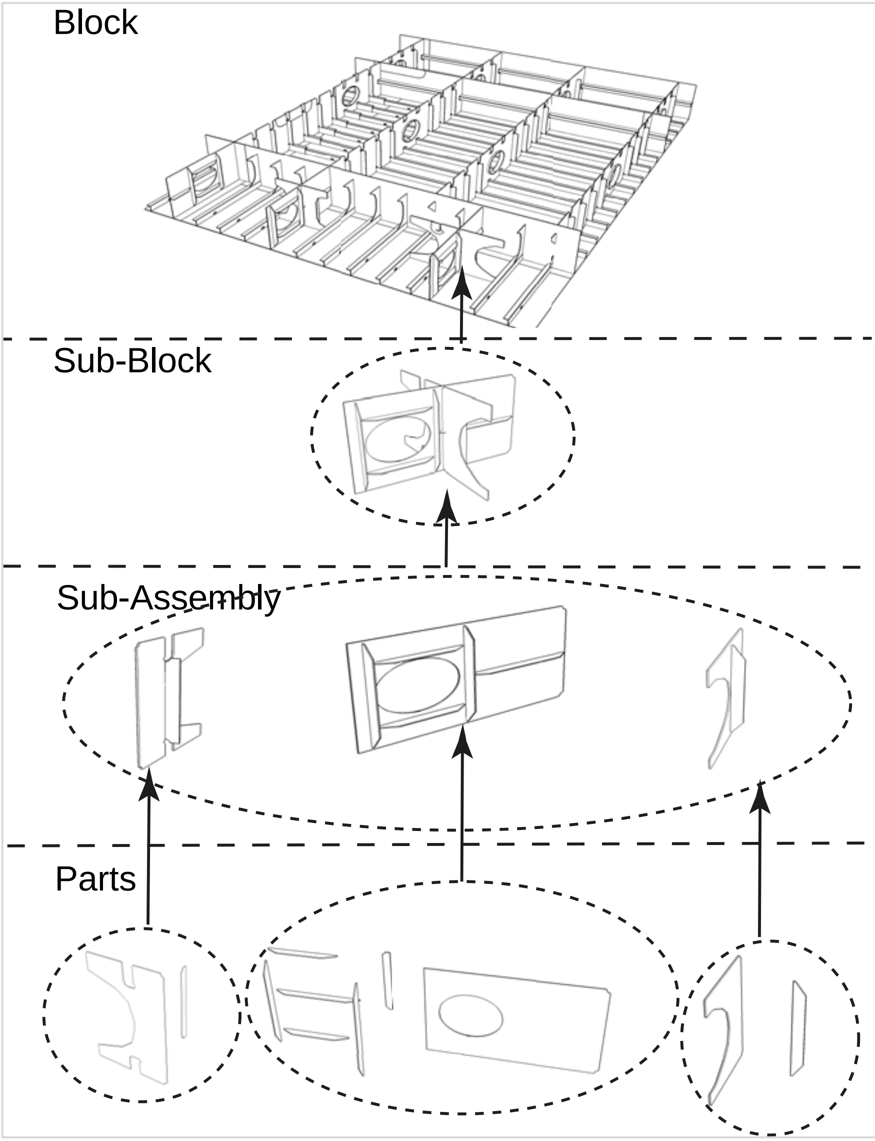


Figure 1-3: Use of Parts That Compose a Block.

Figure 1-2 illustrates the entity relationship between the different parts that compose a block. Figure 1-3 gives a graphical illustration of different parts that compose a block. The terms used in this figure will be the terms used throughout this thesis.

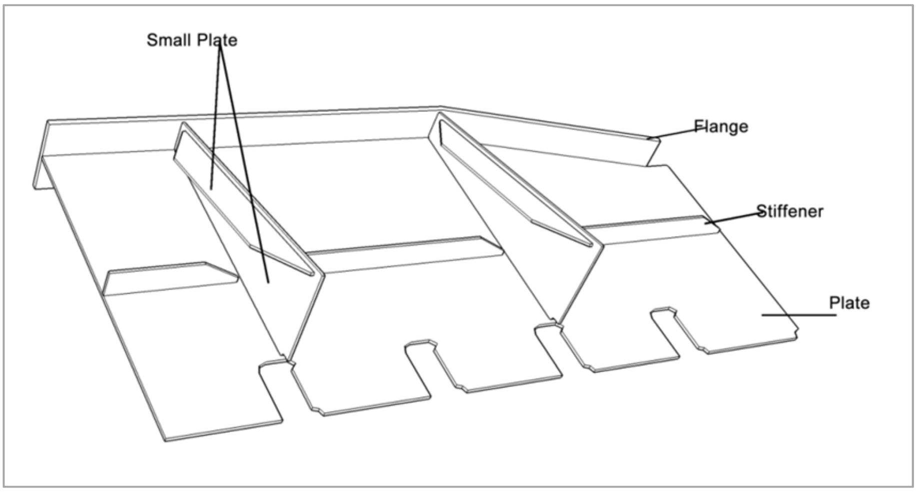


Figure 1-4: Illustration of Some Parts

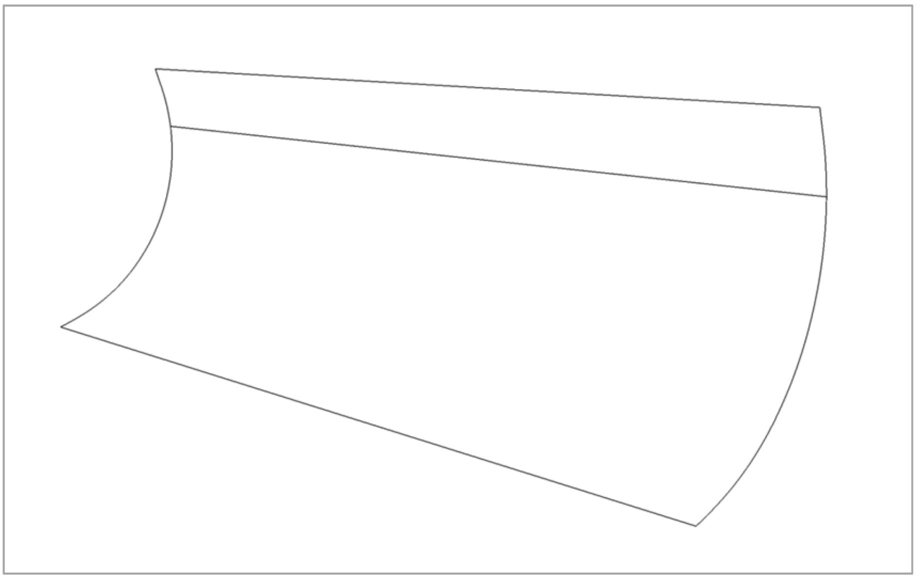


Figure 1-5: Illustration of Curved Plates

The parts that compose a block are plates, stiffeners, brackets, flanges and shell plates. Figure 1-4 shows a sub-block consisting of a larger plate with a flange, three stiffeners and four small plates. Figure 1-5 illustrates curved plates that are also known as shell plates. A sub-block is a plate with at least one other part welded to it. For example, a plate, as shown in Figure 1-4, may have just a flange welded to it. This would then be defined as a sub-block.

Figure 1-4 has several smaller parts welded to the 'base' plate and this is also referred to as a subassembly. A subassembly welded to one or more subassembly is known as a sub-block. Sub-blocks together with subassemblies compose a block.

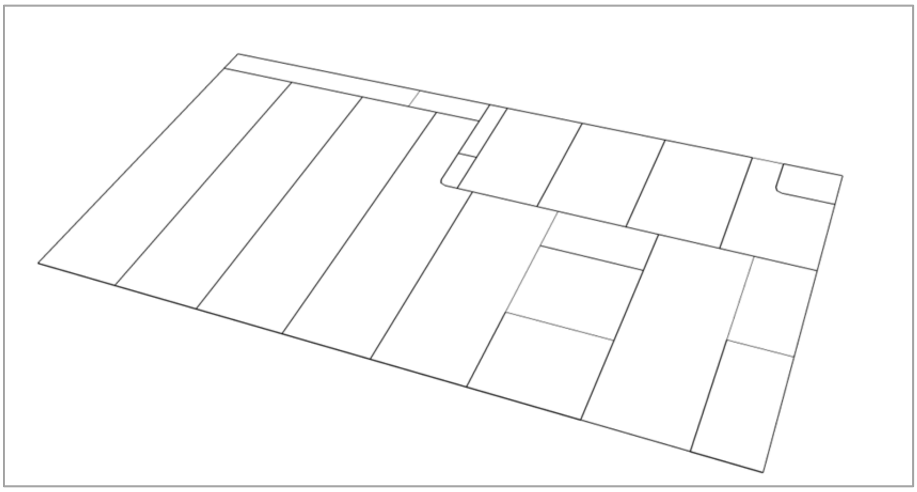


Figure 1-6: Plates Joined to Make a Large Panel

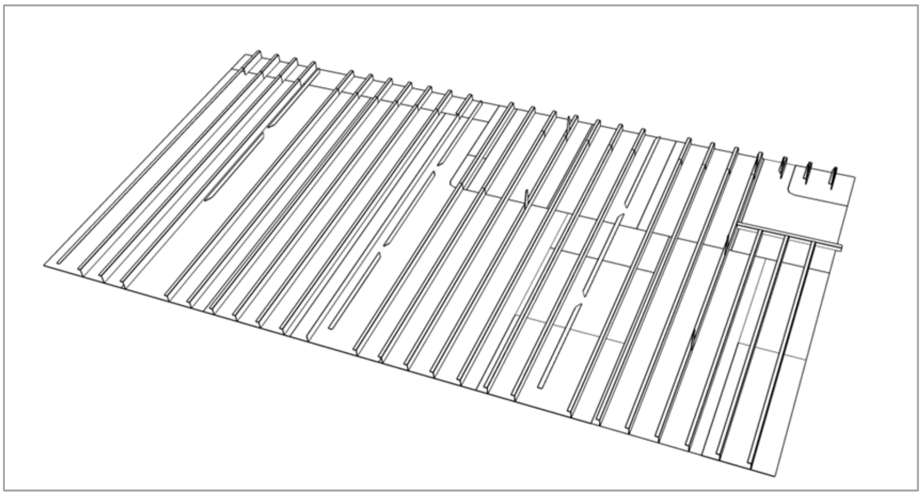


Figure 1-7: Panel with Stiffeners

In the block assembly process, a set of plates are joined to form a large panel that will be the 'base' of the block in the assembly process. This is illustrated in Figure 1-6. The next process is to weld the stiffeners into place, which is illustrated in Figure 1-7. Finally, and here the actual sequence is not shown, only the final result is shown, other sub-blocks and subassemblies are fitted and welded into place. This is illustrated in Figure 1-8.

There are two main types of block, open hull, which is illustrated in Figure 1-8 and double bottom which is illustrated in Figure 1-9

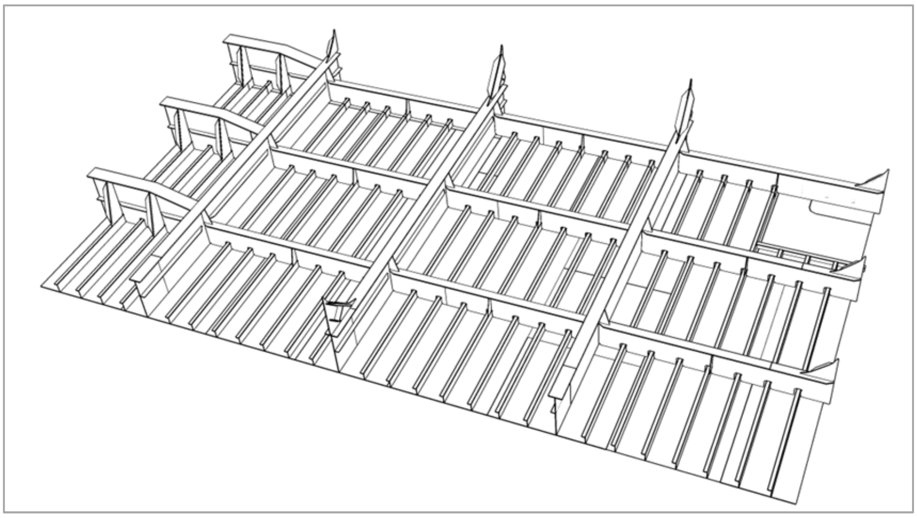


Figure 1-8: Final Assembled 'Open Hull' Block

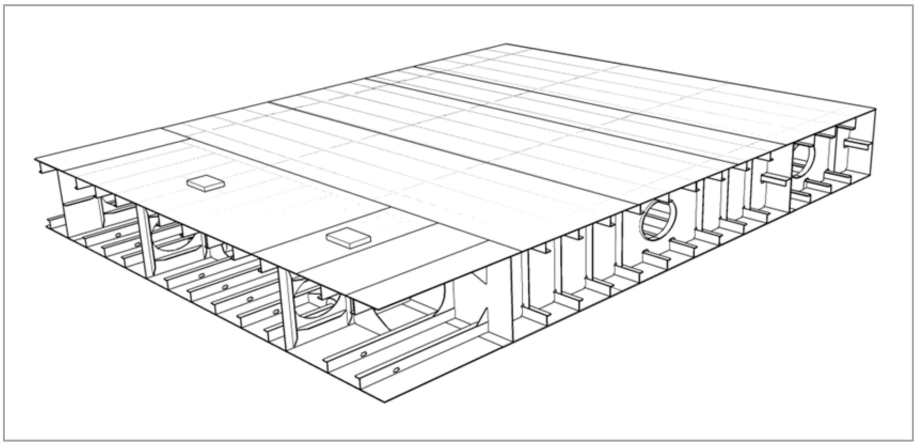


Figure 1-9: Example of Double Bottom Block

1.3. Current Assembly Processes

The current process for block assembly in the candidate shipyard used in this research is a *push* system. *Push* is a term borrowed from *Lean Production* (Tapping et al. 2002) In Lean Production, two terms are used to identify the type of production process: *push* and *pull*. A *push* system is where items are produced without regards to the time they are actually required. In a *push* system, each stage of the production process is a silo, producing items without any knowledge of the down-stream requirement. *Push* systems are obviously inefficient as they will build up inventory, whether it is required or not. This increases the possibility of misplaced parts and also leads to the multiple handling of parts. Parts are moved from one stage of the production process to the storage point and, at best, will then be moved from the storage point to the next stage of the production process almost immediately. If the warehouse is badly organized or it requires some reorganizing due to space constraints, then the parts can easily be handled several times. The excessive inventory has to be warehoused, which increases the stock holding cost. Further, excessive inventory takes up valuable space.

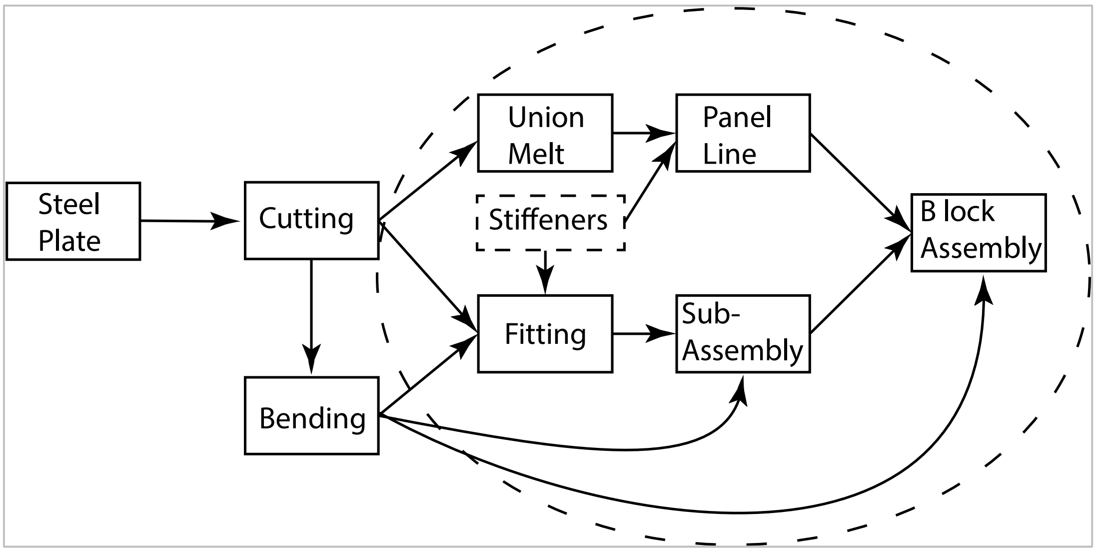


Figure 1-10: Paths to the Block Assembly Point

Push systems have the systemic failure of producing parts that are not synchronized with their timely requirement at the subsequent stage of production, which will either lead to stock holding or a delay in production.

Pull systems are more efficient. In place of starting with autonomous construction sections, each production section is integrated into one continuous workflow and the demand from any section is driven by the final assembly stage.

Steel plate passes through a number of stages to the block assembly, some of which have already been mentioned. A full picture of the construction process is illustrated in Figure 1-10. As may be seen, the block assembly makes a natural *pulling* point, which is part of the work of this research, to ensure that, as far as possible, sub-blocks/subassemblies arrive at the block assembly point when they are required.

All parts for the block are cut from a large steel plate. The cutting process is by a high-precision laser-cutting machine that is numerically controlled. The cutting process is a rapid process, in comparison with that of subsequent processes.

The bending process is the one that produces the shell plates and other parts that require bent profiles. The union melting process is the process whereby plates are welded edge-to-edge (this is known as a butt join) to produce larger sections, often called panels. Figure 1-6 is an example of a number of small plates having been joined to produce a base panel for a block. The size of these panels may be 8 by 4 meters. Stiffeners are welded to the panel. An example of the result is in Figure 1-7. Stiffeners or profiles, as they are sometimes referred to, are not produced in the shipyard. They are bought-in as they have a special profile. The panel with the stiffeners welded forms the base of the block and goes to the block assembly area.

The fitting process is where individual plates have parts welded to them and are transformed into sub-blocks. The subassembly process is where one or more sub-blocks are combined to produce a larger unit. This is not always the case; a sub-block may go directly to the block assembly stage.

The research focusses on the area enclosed in the dotted ellipse in Figure 1-10.

1.4. Research Opportunities

This section describes the gaps and research opportunities that have been identified in the construction of offshore rigs.

1.4.1. Fragmented Approach

A review of the available literature has shown that there has been no holistic approach to the construction of offshore rigs. Beginning with the fundamental work of assembly sequencing of the block it has been found that the previous work has been limited, to say the least. The approach adopted and adapted in this research finds its roots in Work Measurement used in a production environment. Two aspects of Work Measurement have been used: timings and estimating. Observations were carried out on the time a welder takes to fit and weld parts. Estimating the time for the total job from the time parts arrive for sub-block/subassembly to the assembly point is based on the timings used. The amount of work involved is derived from the welding lengths required to complete a sub-block/subassembly. This coupled with the time it takes to complete the welds gives a good estimate of the work content of the sub-block/subassembly. This allows a good schedule and order of work to be calculated to ensure that the required sub-blocks/subassemblies arrive at the assembly point when they are required.

The holistic approach for optimizing block assembly is as follows:-

- The automated inference of part topological relationships
- The automatic identification of part assembly sequence
- The calculation of welding length for sub-blocks
  - The calculation of manpower required for the assembly of the sub-blocks and weight of sub-blocks
- The optimization of sub-block/subassembly for the assembly sequence
- Using Lean principles for the sub-block/subassembly construction process.

The holistic approach facilitates the planning and scheduling process immediately a block has been designed. The requirement of manual input virtually ends once the block design process has been completed. The current practice is for engineers to identify manually the routing and group of the parts. Some are being routed to the union-melt process and panel line, others to teams of welders/fitters to construct sub-blocks/subassemblies.

1.4.1.1. Automatic Assembly Sequencing

Clearly, the correct sequencing of construction is essential for the efficient construction of a block for an offshore rig. One study by Kim et al. (2003) used case-based reasoning to establish the assembly sequence. This clearly has limitations, as it cannot operate in an information vacuum. If there are no previous cases, the system will not function, or at least will not function without some human intervention. The computational cost can be high, as there is the necessity to search for similar cases and then decide which case best fits the existing case.

Another study by Jin et al. (2010) on assembly sequence optimization for ship's structural components, such as blocks, which are identical in process terms to offshore rig blocks, used an ant colony algorithm to establish an assembly sequence. This study adopts the assembly-by-disassembly approach. This approach assumes that the assembly sequence of a block is the inverse of the disassembly sequence which in the case of ship's blocks, it is not the case. The study conflates the assembly sequence with the final welding sequence. During the assembly process, plates are positioned and tack welded. The assembly sequence begins at one edge and moves across and along the block, whereas the welding sequence begins in the centre of the block and moves outwards, thus reducing any distortion in the plates caused by the welding process. The ant colony process may be computationally heavy due to the iterative nature of the algorithm.

The importance of assembly sequencing is not confined to a construction environment; it is extremely important in a production environment. Since the seminal work by Bourjault (1984), researchers have proposed numerous methods of automatically or at least semi-automatically generating assembly sequences, or assembly plans. These methods use neural networks, knowledge-based reasoning, genetic algorithms, precedence relationships and PST. Details are discussed in the literature review.

1.4.1.2. The Optimization of Sub-block/Subassembly for Assembly Sequence

A literature review has resulted in no reference to this. The focus here is to reduce the on-block welding by welding as much as possible off-block prior to the block assembly.

1.4.1.3. Lean Principles Applied to Block Assembly

A literature review has resulted in no reference to this in relation to offshore rig blocks. However, research has been carried out into Lean processes for shipbuilding and in the layout of shipyards, in general to facilitate Lean processes. The candidate shipyard used in this research was established in the early 1960s and the layout may not be so easily amiable to reorganizing to facilitate a modern Lean construction site. What has been adopted from the research on Lean construction is to minimize the makespan for a block's construction concomitant with Lean principles, where possible.

1.5. Research Objectives

The immediate objective of this research is to improve the productivity of the construction of blocks for offshore rigs with the minimum of manual intervention. The thesis will present the necessary automated tools to achieve this objective.

The research objects are as follows:-

1. To use Polychromatic Set Theory as the tool for identifying the necessary topological relationships of the parts of the block.
2. To use Polychromatic Set Theory as the foundation knowledge base for using various system engines.
3. To build on the sets of relationships established to calculate the welding length and hence, the work required to assemble and weld sub-blocks

4. To optimize the assembly sequence of the sub-blocks/subassemblies for their final assembly on the block. This step builds on steps 1. and 2.
5. To derive an algorithm for the makespan for the construction of the sub-blocks/subassemblies so that the constructed components arrive at the block assembly location as near to Just-In-Time as possible, taking into account double handling of the constructed sub-blocks/subassemblies.

These research objectives form the foundation of a system that has powerful Boolean reasoning capabilities which takes a ship-designer's engineering drawings in digital form, and from the inherent properties of the parts, by use of various engines, is able to produce an optimized construction schedule for the block. The schedule is such that it minimizes the on-block work by combining subassemblies into suitable sub-blocks that are welded off-block. The schedule for construction seeks to ensure that the sub-blocks arrive at the on-block assembly Just-in-Time, or near to Just-in-Time. The work has the capability of being extended to cover the whole rig together with a complete construction schedule for the rig.

1.6. Research Scope

The process of constructing an offshore rig block involves many processes as may be seen in Figure 1-10. This takes place in an environment that is totally unlike a production line. Hundred ton blocks do not move along on a construction line. In a shipyard, there are designated areas for designated processes. The cutting of parts from large steel plates, for example, will be done by a precision laser steel cutter. The cutting process is quite rapid when compared to the welding process or bending process. Once the parts are cut, they are placed in a storage area. This research does not take into account the time for the cutting or bending of parts. As mentioned, these processes work at a much faster rate than the welding process.

This research makes the assumptions that all parts such as plates, flanges, brackets etc., are ready when required. This is a reasonable assumption, as the shipyard will cut parts of several blocks ahead of requirement. It also assumes that stiffeners, which are sourced from outside the shipyard, are also ready when required.

The time taken for the fitting and welding processes are those derived from observation or from using the speed of welding as supplied by the manufacturers of the automated welding machines. Then setting up time is taken from observations. No account has been taken for the transportation time of parts from one section of the shipyard to another. This depends on the layout of a yard and location of where the processes are carried out, as well the location of the block assembly area. All these variables are considered outside the scope of the research.

1.7. Research Methodology

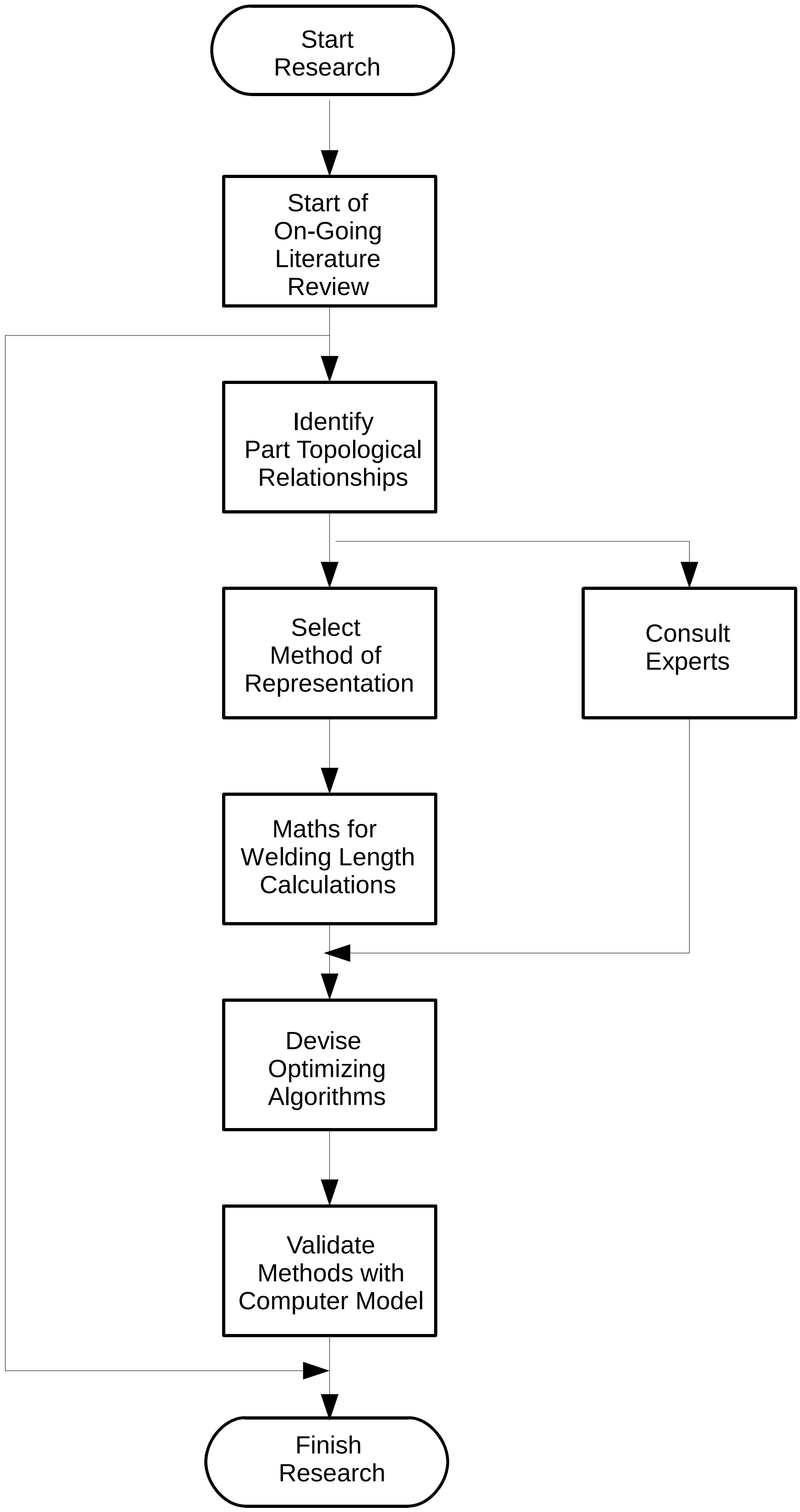


Figure 1-11: Flow of Research Methodology

The research methodology adopted for this thesis is illustrated in Figure 1-11. The research methodology has the following steps:

- A literature review of research in part sequencing methods, Polychromatic Set theory, parallel machine shop flow processing and Lean processes in the production and construction industries. This review is through the life cycle of the research.
- To understand parts that compose a block and the possible types of connectivity between these parts.
- The timings required to construct the subassemblies is required to enable a more accurate calculation of the welding duration of sub-blocks/subassemblies.
- Using expert input, the devising of some algorithms to automate the sequencing of sub-blocks/subassemblies for the block assembly. With the automated assembly sequence, an algorithm for optimizing the sub-blocks is devised.
- To validate the research, several computer programs were written to create a database, calculate welding lengths, identify sub-block/subassembly sequencing and optimizing the off-block construction, and also to produce a construction schedule to minimize the makespan.

1.8. Organization of Thesis

This thesis is organized into 8 chapters, the remaining 7 chapters are organized as follows:-

Chapter 2 presents the review of current literatrue on topics related to the research which are: -

- The use of Polychromatic Set Theory and its diverse applications is discussed.
- The various sequencing methods used in the production industry with a review of their possible use in a production environment.
- Numerous makespan algorithms are available, some of which are reviewed.
- A review of Lean applications in industry and its appliableness to shipyard.

Chapter 3 introduces and illustrates Polychromatic Set Theory, which is the foundation for building the knowledge base for the system. The knowledge base is used in the reasoning and inference engines.

Chapter 4 covers the use of knowledge base, which is built upon polychormatic sets, and by way of a reasoning engine, it reasons from the knowledge base to derive a parts topological relationships and connectivity. It then automatically sequences the parts into their correct assembly sequence using the sequencing engine.

Chapter 5 uses Polychromatic sets and, by inference, identifies panels and subassemblies, and calculates welding lengths of the connections of the subassemblies. The calculation of the welding length is based on data collected during the research period to which a linear regression analysis was applied in order to derive an algorithm.

Chapter 6 covers the optimizing engine, which optimizes the combination of subassemblies to form sub-blocks such that the off-block welding is maximized and consequenctly, the on-block welding minimized. The construction sequence of the sub-blocks/subassemblies is obtained by a developed set of algorithms. Four Greedy based algorithms and one Genetic Algorithm to mimimize the makespan for the construction of the sub-blocks/subassemblies were developed and compared.

Chapter 7 presents a case study bringing together the research. The case study uses a block from a project of the candidate’s shipyard. The case study covers all engines that are illustrated in Figure 1-1 as well as the construction of the local database.

Chapter 8 presents the conclusion of the thesis and some recommendations of how the research can be extended into the complete rig construction with optimizing the use of labour and minimizes the makespan for the whole rigs construction.

Chapter 2. Literature Review

2.1. Introduction

This literature review covers the four main areas of the research. It begins with an introduction to the variety of uses to which Polychromatic Set Theory (PST) has been deployed; a set of examples on the use of PST is given in chapter 3. The correct sequencing of parts in any assembly process is paramount for an efficient and effective operation. A review of the various techniques for assembly sequence is reviewed. Lean principles are introduced and discussed, as this research uses Lean principles, where possible, for the assembly construction sequencing of the sub-blocks/subassembly that compose a block for offshore rigs. Usually, there are a few teams of workers used to construct the sub-blocks/subassemblies required for a given block for an offshore rig. The work of these teams may be viewed as parallel machines, working with a single purpose. Parallel machine processing is discussed as the final section in this literature review.

Many of the approaches used focus on a single objective, such as obtaining a good or relatively good assembly sequence. None of the approaches found in the literature review begin with the digital output from the engineer's drawing, then proceed to construct a knowledge base as a foundation to identify, automatically and intelligently, the various topological relationships of the parts. None has automatically combined parts to form subassemblies. Neither have any optimized the combination of subassemblies into sub-blocks in order to reduce unnecessary work in line with Lean principles. The optimization of the construction schedule for component parts of a block to minimize the makespan for the block assembly, taking into account the number of teams available for work, seems absent. To this extent, this research is a holistic approach to the construction of a block for an offshore rig.

2.2. Polychromatic Set Theory Applications

This section will cover the usages of Polychromatic Set Theory, showing that the relatively new theory has been applied in a wide and diverse range of industrial problems.

Polychromatic set theory is a relatively new theory proposed by V.V. Pavlov (2001)[1] . It extends Cantor's set theory with the introduction of properties of members of a set, or colours as they are referred to. Polychromatic set theory (PST) has been successfully applied to a wide number of industrial and non-industrial applications.

PST has been applied in a number of ways in construction sequence planning. Xu et al. (2012) have applied PST in dynamic assembly modelling for assembly sequence planning. They formalize the dynamic assembly model's incidence relationship as a polychromatic matrix. Further, the locating and collision relations of the parts are expressed as polychromatic matrices.

Yan et al. (2006) use PST for developing a disassembly model for manufacturing systems. Their approach of using PST simplifies the whole process, reduces the complexity of the algorithms involved as compared to other methods, and is extendible to more complex systems than the case study presented.

---

[1] Pavlov's work is in Russian, no English translation seems to be available. However, his work is quoted in all papers that refer to Polychromatic Set Theory.

Chen et al. (2011) have applied PST to the design of product data (PDM) management multi-version management model. They use PST to represent related aspects of PDM the document version and the document's relationship information.

PST has been used in the design or manufacturing of car panels, (Zong et al. 2009), as well as the assembly of car body parts with their special constraints, (Zhao et al. 2007). This enables the locating, constraint and interference matrices to be generated.

PST has been used for the assembly planning of aircraft parts, (Wang et al. 2010), Wang et al focus on the necessary assembly equipment, the selection of location mode, assembly sequence planning and finally, the assembly process planning. It has been applied to the automatic synthesis tolerance for complex assemblies (Zhang et al. 2011). Li et al. (2006) summarises the application of PST in the conceptual design, the product assembly modelling, work flow modelling and tolerance modelling.

Zhao and Li (2008) use PST in sequence planning of 3D objects as well as using PST to derive a formal method for reasoning of assembly sequences. They also use PST for assembly sequencing (Zhao and Li 2008). The usual locating matrix, collision matrix and interference matrix are replaced by polychromatic matrices. In so doing, the solution space is reduced. Li et al. (2009) combine a constraint model based on PST with a Genetic Algorithm for sequencing machining steps in a workshop.

PST has been applied to a number of other research problems, such as the conceptual design of a product using CAD, (Gao and Li 2006), the simulating of accessory surface processing of a metal product, (Li and Xu 2003) and the extending of a graph structure (Gao and Wang 2013). (Guo and Lan 2008) have used PST coupled with the Unified Model Language (UML) state diagram of systems analysis to give a more robust formulation for the semi-formal UML language. PST has found a place in research involving wireless network modelling, (Vikram; 2013) as well as scalable routing modelling (Wang and Li 2013) and in a dynamic spectrum access method for cognitive radio networks (Li and Wang 2012)

As may be seen from this short summary, PST has found uses in wide and diverse fields of research, and as such, has a firm foundation to be used in the current research as may be seen in Figure 1-1, PST is used in the knowledge base, reasoning engine, sequencing engine, and inference engine.

2.3. Assembly Sequencing

Having a correct assembly sequence is fundamental to any efficient construction process. It does not matter what the process is, it can be anything from a suit of clothes to a shopping complex. Incorrect or poor assembly or construction sequences inevitably lead to inefficiencies. Proper assembly sequence planning of a block for offshore rigs is essential for any automatic or computer based generation of an assembly sequence (Bai et al. 2005). The generation of an assembly sequence for any product can be quite complex. Several methods have been proposed in an attempt to reduce this complexity. These methods may be broadly classified into the following categories: dis-assembly based reasoning, knowledge-based reasoning, genetic algorithms and neural networks, precedence relations, miscellaneous methods and polychromatic sets. A literature review of research into assembly sequencing for offshore rigs indicates that the research is sparse. The assembly sequencing covered in the review is mainly from the production industries, not the construction industries.

Dis-assembly based or geometric reasoning comes in various forms. Dini and Santochi (1992), use three types of matrices: interference, contact and connection to derive the dis-assembly sequence, whereas Homem de Mello and Sanderson (1991) algorithm use the relational model coupled with a geometric model of the assembly and geometric reasoning to produce a dis-assembly sequence. Lai and Huang (2004), generate an assembly sequence using assembly precedence relations created by an engineer using computer based liaison graphs and matrices. Jin et al. (2010), generate a dis-assembly interference matrix which is based on the model's component interference matrix. This is input to an ant colony algorithm for the purposes of optimization.

Knowledge-based reasoning has also received the attention of researchers. Chakrabarty and Wolter (1997), propose that the assembly sequence planner uses a structured hierarchy both as a framework for structured-dependent definition of a *good plan*, and also a tool for locating *good plans* quickly by using high-level expert advice. Tianyang et al. (2007) and Dong et al. (2007) propose a method that uses both the geometric and non-geometric knowledge and produces sequences that required less computation time and led to sequences that are more practical.

Genetic algorithms and neural networks are widely used for *NP*-hard problems, as well as in sequencing methodologies. Bonneville et al. (1995) use a GA to generate and evaluate assembly plans. The initial population is the set of valid assembly plans produced by an expert of the product in question. This approach rapidly generates a set of good assembly plans. Chen and Liu (2001), not relying upon experts, propose *an adaptive genetic algorithm* to locate global-optimal or near global optimal sequence. Sinanoglu and Borklu (2005) use a neural network approach to develop assembly sequence plans.

The seminal work of Bourjault (1984) uses precedence relations to construct an assembly tree. His work was refined by De Fazio and Whitney (1987) and later by Wilson (1993). Minzu et al. (1999) obtain feasible assembly sequences by merging assembly sequences.

Graphical methods are the main stay of many of the methods already mentioned. Henrioud et al. (2002) extends the work of Minzu et al. (1999) by using hypergraphs for assembly system design. Lee and Gossard (1985) use a hierarchical data structure to represent assemblies in conjunction with a database that holds topological and geometric information and is supported by a data structure that identifies how all components in an assembly are connected.

Guo et al. (2010) use a hierarchical constraint assembly model for assembly sequence planning where a model is composed into small constructible components; this process is recursive, with constraint being formed between the 'layers'. Lai and Huang (2004) use liaison and precedence matrices along with precedence relations to obtain the optimized assembly sequence.

Petri nets, with an algorithm, is the method Zhang (1989) uses to generate assembly plans. In a similar vein, Lu et al. (1993) propose an assembly sequence algorithm that is based on quotient space model, which produces multiple possible assembly sequences that are represented by a Hidag tree. They translate the Hidag tree into a series of linear sequences.

All these methods mentioned have some shortcomings. Dis-assembly sequencing assumes that the assembly is the reverse of the dis-assembly. Although this seems reasonable, there is no guarantee that it is always the case. Knowledge base systems require expert input, and this, at best, is subjective knowledge. It also suffers from a time lag. Sufficient expert knowledge has to be accumulated before a system can be used and of course, by its very nature, it has human input at various stages in the process. Genetic Algorithms can be computationally heavy, and are really not suitable for ad-hoc work such as blocks for offshore rigs, unless, as in this research, they are constrained to solving a problem in a specific area. Precedence relationships can be complex and convoluted to use.

Polychromatic Set Theory (PST), is a relatively new theory, which has received some attention in recent years and has been applied to a wide variety of industrial type problems. PST has the advantage of being simple to apply; it is not computationally heavy as in the case of some methods. It is readily extendable should additional constraints or requirements come to light.

PST has been used in simulation modelling, product life cycle simulation, product conceptual design, concurrent engineering and process modelling. Li and Xu (2003) have applied PST to product assembly modelling, work flow modelling and tolerance modelling. Gao and Li (2006) have applied PST to the conceptual design in a product development. PST is widely used in the Russian aviation industry and astronavigation (Zhao et al. 2006), Zhao et al. (2007) use PST for car body assembly. Liu et al. (2010) use PST in the process model for body-in-white welding[2] assembly. They analyse the connection and interference relationship with an algorithm to generate a welding assembly sequence. Their Polychromatic Set consists of the vehicle's body parts with their associated interference and connection relations. Li and Wang (2012) have applied PST to scalable routing modelling for wireless ad hoc networks as well as spectrum access in cognitive radios.

This brief survey of the literature for PST shows its flexible and diverse use. PST does not require any specialised knowledge per se; it is straightforward to use and is easily extendible.

PST has been chosen for this research for its suitability. While there are other competing and suitable methods, this method has been selected for its simplicity. Ockham's principle[3], that when there are competing solutions to a problem, the simplest solution is the best solution. Polychromatic sets also have the advantage of flexibility and extendibility. If, at any time, additional properties or knowledge is required for the reasoning processes, they can easily be added. Further, if some properties or knowledge embedded in the set are deemed to be redundant, they can easily be deleted from the set. Having embedded knowledge as a part of the set allows for reasoning, sequencing and inference of the set data, which are a powerful tool in the research.

---

[2] Body-in-White welding is the stage in car production when the car's sheet metal body parts have been welded together, prior to the adding of any moving parts.

[3] Better known as 'Ockham's razor' from William of Ockham (c. 1287 – 1347). It states in its simple form, "Plurality should not be posited without necessity." Which is taken to mean that when dealing with competing hypotheses, the hypothesis with the fewest assumptions should be selected. http://global.britannica.com/EBchecked/topic/424706/Occams-razor

2.4. Lean Production and Lean Construction

The term *Lean* was first coined in an article by John F. Krafcik,(1998). In his article, Krafcik compares and contrasts different approaches to car production. One is where there is a company culture that prosecutes a drive to improve efficiency, reduce costs at every point of its operations and to improve quality. The other is where the automated equipment is the same but the culture is seen as virtually antithetical to that of the first. Both car production companies had essentially the same equipment; the essential difference was the *weltanschauung*[4] of the management and workers. The point that can be drawn from Krafcik is that Lean is more than an equipment; it is an equipment plus a culture that prizes quality and productivity, seeing them as opposite sides of the same coin.

The vehicle manufacturer, Toyota, is credited with developing the modern Lean production approach. The Toyota Production System (TPS), which is the archetypical Lean production system, has a number characteristics: standardization of parts, continuous supply of parts, minimal stock levels and stock buffer, worker flexibility and a high degree of team work. The characteristic that is more commonly recognised is the continuous supply of parts, which is better known as Just-in-Time. This approach differs from the traditional method of supplying parts in a continuous production process, where parts are produced at one stage of the process for the next, irrespective of the requirement of the next stage. This is called a *push* system and leads to high levels of inventory (Womack et al. 1991).

---

[4] A German word that is used in the academic discipline of philosophy and epistemology that means "a comprehensive conception or apprehension of the world especially from a specific standpoint". http://www.merriam-webster.com/dictionary/weltanschauung visited 11-Nov-2014

Lean production, with its emphasis on efficiency, aims to eliminate waste at all stages of the production process. A process in the production of a product will either add value to the product or not. The aim of Lean production is to add value to the product at each stage of the production process. Waste, in Lean manufacturing, increases the time of production or increases the cost of production, but does not add value to the product (Liker and Lamb 2002).

Lean production or manufacturing has, as its underlying philosophy, a shortening of time between the placement of an order and the delivery of the order by eradicating sources of waste in the process. Toyota identified seven areas of waste in manufacturing: overproduction, defective product, high inventories, non-value action, unnecessary or superfluous processes, transportation of materials and idle time for material and people (Lang et al. 2001). Toyota's drive to eliminate waste at every point in the manufacturing process of the product has clearly been successful, as their ideas of Lean production have permeated other sections of industry.

Lean production is not an end in itself (Womack and Jones 1994), or an event, but part of a process that stretches the full length of the value chain (Zimmer et al. 2008). In civil construction projects, the value stream involves a multiplicity of sub-contractors not just the main contractor. In offshore rig construction, the value stream is mainly an internal one. Each of the construction steps are considered as a separate supplier to the next process. Although not considered in this research, following the block construction there are several other process that the block has to undergo, such as sandblasting, painting, combined with other blocks to make a super block, assembly in the final position, piping work, electrical work etc. There is a minimum of outside suppliers, for the actual block construction; these supply steel plate and special profiles (stiffeners).

Womack and Jones cite the example of Lucas PLC, who successfully implemented Lean technologies in 1983 under the guidance of Professor John Parnaby, and saw great gains in productivity with orders delivered on time, increasing from a derisory 25% to a highly credible 98%. However, these successes began to unravel as the organizations that Lucas supplied had not, at that stage, implemented Lean methodologies. The lesson is that Lean needs to be implemented from head to tail of the value chain. In the selection of a supplier, Cagliano et al. (2004) show that both Lean and Agile strategies for supplies models work equally well. They both out-perform other supplier strategies such as price and operational performance. Having said this, the number of suppliers is also an important factor, (Warnecke and Huser 1995), it was typical for European car manufacturers to have 2.6 times more parts suppliers and for the American car manufacturers to have 2.99 times more parts suppliers than their Japanese counter parts. Japanese car manufacturers would have a single supplier responsible for something like car seats, whereas General Motors would purchase all components for a car seat from a range of suppliers and then assemble the seat in-house.

Lean principles and practices from the Japanese car industry may not always transfer 100% to general manufacturing companies. As Herron and Hicks (2008) show, the basic principles are easily transferrable, but the cultural and work practice baggage of the recipient company makes a 100% transfer of the Lean ecosystem difficult.

Lean principles have even been applied to knowledge-based work, in particular software programming. In spite of the totally different nature of the work; software development is dynamic with possible changes at any stage of its production, whereas, the production of a product on an assembly does not change (Staats et al. 2011), it would seem that it is possible to apply lean methodologies to knowledge work. As mentioned earlier, it is the *weltanschauung* of the whole organization that is fundamental to the success or otherwise of the implementation and execution of Lean methodologies. In the knowledge-based industry, the implementation of Lean methodologies led to the way an organization learns by solving problems using hypothesis-driven techniques[5], improved channels of communications and the simplification of the process architecture, which in effect removed unnecessary software changes or development.

The application of Lean principles to both the production sector and knowledge work leads to the possibility of the application of Lean principles to the construction sector. There is probably more similarity between the production and construction sectors than between the production section and knowledge work (Paez et al. 2005).

The adoption of Lean production principles used in the car industry by the ship building industry has received some attention in recent years. The differences between these two industries are clear. The car industry deals with high production volumes and short customer order to customer delivery times, whereas the shipbuilding industry deals with low production volumes and long customer order to customer delivery times.

The difference between production and construction is also an important factor (Salem et al. 2005), certainly in the construction of offshore rigs. Offshore rig

---

[5] In software development, a well-defined hypothesis of how the system should function is tested. The hypothesis is tested by allowing the clients/customers to add new features or use new features.

construction faces many of the same issues that have been articulated for the construction industry when it comes to Lean, such as, one-of-a-kind-production, owner-product definition and owner modification (Salem et al. 2006). Production depends on fixed operation cycle times. In the ship building industry, in particular in the offshore sector of the industry, processes do not have fixed cycle times. As has been highlighted by Liker and Lamb (2002) takt[6] time for cars is in minutes but for ships, it is in multiple of months and further, there can be a wide variation in takt time between different ships. However, this has not prevented research being carried out into the use of Lean principles in the shipbuilding/offshore rig construction industry.

Storch and Lim (1999), focus on the general flow of processes in the shipyard. They make the point that inter-stream and intra-stream communications need to be efficient and for this to be most effective, successive work processes in the value stream need to be located, if not adjacent certainly in juxtaposition. Further, Lean production methodologies require a high degree of flexibility in the utilization of resources in order to respond rapidly to production changes. The way traditional shipyards are organized does not support Lean methodologies. For Lean production to succeed in a shipyard, all the various stages (hull, outfitting, painting etc:), need to be closely integrated. As with a production line, the workflow between values stream process should be continuous and uniform.

Lang et al. (2001), recognising that Japanese and Korean shipbuilders were more than twice as efficient as the America shipbuilders, carried out an academic exercise of adopting Lean manufacturing principles in a traditional American shipyard. In their hypothetical model, they saw that is was essential, as did Storch and Lim

[6] The classic calculation is: Takt time = Available Minutes of Production/ Required Units of Production

(1999), to have the shipyard organized such that movement between value stream processes was minimized. Further, it was necessary to have a JIT material procurement system coupled with Quality Management principles. In their academic exercise, they saw significant improvements in productivity.

Liker and Lamb (2002), also propose a reorganization of the shipyard based on a product-flow process, rather than the traditional function-batch process. They also propose a levelling of the process time for the block, which is effectively Just-in-Time for a grand block construction. (A grand block is a composed of a number of smaller blocks.)

Liker and Lamb, in their detailed paper, adumbrate the way the Japanese shipbuilding industry improved their productivity by 150% in the period 1965 to 1995. They adopted and improved the U.S. and European system of structural block construction and pre-outfitting. The industry was developing their own flavour of Lean principles to apply to ship construction at the same time that the more famous Toyota were developing lean methodologies for car production. As with Toyota, and others who use Lean methodologies, the employees in the ship building industry were fully involved in the continuous improvement effort and were multi-skilled, which enabled them to be deployed more effectively.

Liker and Lamb advocate that a shipyard be organized by "product line." This is where similar parts or families of parts are processed in the same process line. Their vision for a Lean shipyard is one where steel plate that is cut in the morning will be used and assembled into blocks in the afternoon. This may be the ideal for Lean production, but it is also in conflict with the idea of reducing waste. Modern plate cutting software, such as *nestix*, is designed to reduce waste in the cutting process.

Reduction of waste means that parts cut today, may not be required immediately. They are more likely to be required in the near future but not immediately. This gives rise to the necessity of storing the cut parts, which builds up inventory. Unfortunately, since cutting software is not tied to any construction schedule, this may be a necessary evil.

Kolich et al. (2011)[7](2012) suggest that that there is a dearth of a Lean manufacturing methodology for shipyards. In their study, they model a panel-block assembly before and after the implementation of Lean construction using Monte Carlo simulation. The study shows that with the implementation of Lean principles, there can be a reduction of up to 60% in man-hours required for construction.

Liu et al. (2011) recognize the inherent challenges of having long production lead times for a product. They use aggregate production planning in the area of workforce levelling as well as inventory usage. They achieve their goal by a multi-objective genetic algorithm. Their work deals with smoothing the workflow of block construction, and as with Liker and Lamb, their object is to construct a grand block just-in-time.

Lean production in the shipyard is an ideal, but one that may not be so easy to achieve. The ethos of the company is very important. The point made by John F. Krafcik,(1998) is important, it is not a matter of equipment or infrastructures alone, the right *weltanschauung* of the workers is essential. Having said, this, it is not a case of giving up on Lean production in the construction of offshore rigs. Where it can be done, it should be done. All steps, no matter how small, that move in the direction of Lean production are worth taking. Similar sentiments have been aptly stated by Liker and Lamb (2002) “Lean manufacturing is a philosophy and what is most important is

[7] Author has variant in spelling of his name, Kolic, Kolich.

the *process* of involving associates in reducing the production flow by eliminating waste."

2.5. Construction and Parallel Machines

The research so far mentioned generally focuses on the overall processes of the shipyard with a number dealing with grand block construction. The research that deals with blocks sees the block as the smallest unit of construction, and treats the block more or less in the same way as a production line would treat a component part. What is missing here is that a block is not in itself a simple component. It is made up of possibly hundreds of parts and it requires assembly in a particular order. Lean principles have not been applied to blocks as such. Abbott and Chua (2013) treat the block as a series of sub-blocks/subassemblies for assembly sequence purposes. This is a necessary move towards using Lean principles, where possible, for the block assembly.

This research focuses on the implementation of Lean principles for the assembly of a block for offshore rigs. This work does not deal with the much larger issue of Lean principles for the whole construction process of offshore rigs. Indeed, as Åhlström (1998) has pointed out, always working with the core principles of Lean, a case for the sequential implementation of Lean can be made. The research here is examining the possibility of introducing Lean principles at one part of the total construction process. The work is aware of the point made by Zimmer et al (2008) that various business units in the value stream act with a silo mentality; that is no consideration of the downstream effects on their decision is considered, a comment reinforcing the work of Professor John Parnaby (1986; 1988) in regards to Lucas' ventures into Lean. However, as a part of moving the construction process towards Lean, a start has to be made somewhere, and the extension to the whole rig construction process is for future work.

Block construction is decomposed into the construction of smaller units, (sub-blocks and subassemblies). The sub-blocks/subassemblies are not processed in a production line fashion; as mentioned earlier, construction is not production, as each sub-block/subassembly differs from one another. The work will be carried out such that several sub-blocks/subassemblies are worked at by different groups or teams of workers at the same time. The issue here is how to allocate the work to the groups or teams of workers such that there is a smooth flow of work at the final assembly point.

The teams of workers may be considered as working in parallel and the research of scheduling with identical parallel machines is of importance here. For the purpose of this research, each team of workers is treated as being equally competent with identical skills. The work on the sub-block/subassembly is treated as being continuous, in that the team of workers will not stop working part of the way through on one sub-block/subassembly and move to another. *Prima facia*, the scheduling of work among the parallel identical machines is without pre-emption. It has been pointed out (Blazewicz et al. 1983; Chen et al. 1988; Karger et al. 2010) that the scheduling with parallel identical machines without pre-emption is an *NP*-hard problem. It has been suggested by Mosheiov (2001), however, that flow-time minimization on parallel identical machines, albeit only two parallel identical machines, is polynomial solvable. The solution is too problem-specific for this research, since it posits a learning effect, which may be possible in construction environments that deal with repetitious units, which is not the case in the construction of the offshore rig. Further, the computation resource required to resolve this for only two parallel machines is $O(n^4)$.

The algorithm adopted will depend on the objective. A primary objective is to minimize the makespan for whatever happens in the construction process, the overall completion of the schedule must not suffer. Researchers have proposed several algorithms for the *n* jobs, *m* machine problem. The work on parallel machines is not limited to the machine shop environment but encompasses computer job scheduling within the computer operating system (Horowitz and Sahni 1976). Machines in the computer context are processors.

If, when considering the schedule makespan, the order in which the jobs have to be processed is of no importance, then the longest processing time (LPT) algorithm provides an approximation and has been shown to be a 4/3 approximation algorithm for $P || C_{max}$ [8]. (Karger et al. 2010).

Scheduling algorithms for minimizing the total tardiness using identical parallel machines have been proposed by a number of researchers (Biskup et al. 2008; Yalaoui and Chengbin 2002). However, the research here focuses on flow-shop environment, with its expected due date. They organized their work either using the shortest processing time algorithm (SPT) or the longest processing time algorithm (LPT) and reply upon NEH[9] algorithm, which is an algorithm for m-machines with n-jobs for flow-shop sequencing, or a modified version of it. The present research does not have due dates as such; it is dealing with a predefined assembly sequence, and in

---

[8] P machines in parallel, no preemption, minimize the makespan. Notation as per Graham et al (1979)

[9] NEH algorithm named after its developers. Nawaz, M., Enscore Jr, E. E., and Ham, I. (1983). "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem." *Omega*, 11(1), 91-95.

that sense is constrained. Tardiness, in the context of this research, can be measured as the difference between the finishing times of one job and that of the subsequent job. Since, the time to fit a completed sub-block/subassembly into its required position is considered small compared with that of the total construction time.

Viewing the teams of workers as parallel machines and having a predefined output sequence of the work allows for a much simpler heuristic, whether a Greedy heuristic or a Genetic algorithm.

Chapter 3. Polychromatic Set Theory

3.1. Introduction

In this chapter, the Polychromatic Set Theory (PST) is introduced and discussed. PST has a mathematical basis and forms a core part of this research. PST allows the embedding of knowledge that allows for reasoning and inference.

3.2. Introduction to Polychromatic Set Theory

Polychromatic Set Theory was invented and developed by the Russian professor V. V. Pavlov of Moscow State University as an extension to Cantor's standard set theory (Pavlov et al. 2001).

In conventional set theory, a set is defined as

$$A = (a_1, \cdots, a_i, \cdots, a_n) \tag{3.1}$$

where A is the set and a_i is an element of the set which contains n members. If element a_i belongs to the set A, this is stated as $a_i \in A$. However, if a_i does not belong to the set A, this is stated as $a_i \notin A$.

In conventional set theory, a set is composed of similar objects. These can be anything such as cars, computers, athletes of particular discipline, people of a given age range, etc. Conventional set theory does not give any information about the properties of the individual members of the set. For example, the set of all male athletes who compete in an Olympics 100 meters heats, the country the athletes represents, their height, weight, marital status, shoe size and their sponsor are not reflected in conventional set theory. PST builds on conventional set theory by defining not only the elements of the set, but also the properties of each element in the set as well as the properties of the whole set. PST gets its name from the usage of colours to represent different properties; hence, each element in the set has a different colour, or a multiple of colours, depending on the number of properties a given element of the set possesses. The set itself can also have a colour or colours.

The colour set in PST is defined as F, the colour for an element $a_i \in A$ is given by

$$F(a_i) = (F_1(a_i), \cdots, F_j(a_i), \cdots, F_m(a_i)) \tag{3.2}$$

where the total number of properties of a_i is *m*. This colour is referred as an *individual colour*. Not all members in the set will have the same set of properties. A day-to-day example is the set of cars. Cars differ wildly in their individual properties. One car may have a sunroof, whereas another may not. One may have 5 seats and another 6 seats. The different properties are considered different colours, or individual colours.

By way of example, let $A = (a_1, a_2, ...a_i, ...a_n)$ be a set of notebook computers owned by a class of students where a_1 is compter 1, a_2 is computer 2, ……, a_n is computer n. Let some properties of the computers, set $A = (a_1, a_2, ...a_i, ...a_n)$, be screen size and memory capacity, where the screen sizes are, for example, *12.1”, 13.5” ,15”* and *17”*, and the memory capacities are, *4GB, 8GB, 16GB* and *32GB*.

Let F_1=12.1”, F_2=13.5”, F_3=15”, F_4=17”, F_5 = 4GB, F_6=8GB, F_7=16GB and F_8= 32GB, then, $F_1(A) = 12.1"$, $F_2(A) = 13.5"$, $F_3(A) = 15"$, $F_4(A) = 17"$,

$F_5(A) = 4GB$, $F_6(A) = 8GB$, $F_7(A) = 16GB$, $F_8(A) = 32GB$. If computer a_i has a 17" screen with a memory capacity of 32GB, then the set $F(a_i) = \{F_4(a_i), F_8(a_i)\}$

In a similar vein, not only the elements of a set can be coloured, but the whole set can be coloured. The whole set colouring is referred as *unified colour* of the set and is given by:-

$$F(A) = (F_1(A), F_2(A), \cdots, F_k(A), \cdots, F_n(A)) \quad (3.3)$$

where *n* is the total number of properties or colours for the set *A*. The collection of all individual colours or properties of the members of a set constitute the unified colour. The existence of the unified colour set $F_j(A)$ is defined by the existence of the individual colours of the set member a_j.

In equations (3.2) and (3.3), the colours represented by $F_1, \cdots, F_m / F_n$ are different.

The colours of sets $F(a_i)$ and $F(A)$ are called pigmentations. The colouring of all elements is given by the set *F(a)*, then

$$F(a) = \cup_{i=1}^{n} F(a_i) \quad (3.4)$$

The colour set *F* includes the colour set *F(A)* of the entire set *A* and the colour set *F(a)* of the individual elements in set *A*. Using set notation, this becomes:-

$$F \supseteq F(A), F(a) \quad (3.5)$$

In formal terms, $F_m(A)$ and $F_j(a_i)$ are referred to as unified colours and individual colours.

All elements that have the same colouring or property are given by the set

$$A(F_l) = \{a_{i1}, \cdots, a_{ij}, \cdots, a_{ip}\} \tag{3.6}$$

This means that for any colour, or property, F_l, there are p elements that have the same colour where $p \le n$. Using the computer notebook example, $A(F_6)$ will be the set of computers that have 8GB of memory.

The Cartesian cross product, $A \times F(a)$ gives the relationship between all the elements of the set A and the colouring of the individual elements $F(a)$, which may be seen in the matrix equation (3.7)

$$\left\| c_{i(j)} \right\|_{A,F(a)} = \left[A \times F(a) \right] = \begin{array}{c} \begin{matrix} F_1 & \cdots & F_j & \cdots & F_m \end{matrix} \\ \begin{pmatrix} c_{1(1)} & \cdots & c_{1(j)} & \cdots & c_{1(m)} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ c_{i(1)} & \cdots & c_{i(j)} & \cdots & c_{i(m)} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ c_{n(1)} & \cdots & c_{n(j)} & \cdots & c_{n(m)} \end{pmatrix} \begin{matrix} a_1 \\ \vdots \\ a_i \\ \vdots \\ a_n \end{matrix} \end{array} \tag{3.7}$$

where $c_{i(j)} = \begin{cases} 1. \text{ if } F_j \in F(a_i) \\ 0. \text{ if } F_j \notin F(a_i) \end{cases}$

$$[A\times F(a)]=\begin{array}{c} \begin{matrix} F_1 & F_2 & F_3 & F_4 & F_5 & F_6 & F_7 & F_8 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{matrix} \end{array} \tag{3.8}$$

Returning to the computer notebook example for just 8 computers, $[A\times F(a)]$ is shown in matrix equation (3.8). From the matrix (3.8), computer a_5 has colours or properties, (F_3, F_7) i.e. a 15” computer with 16 GB of memory.

$A\times F(A)$ is the unified pigmentation of all elements of the polychromatic set which is the relationship between the elements of set *A* and set *A*’s individual colours or properties. More formally, this is represented as

$$\| c_{i(j)} \| = [A\times F(A)] = \begin{array}{c} \begin{matrix} F_1 & \cdots & F_j & \cdots & F_m \end{matrix} \\ \begin{pmatrix} c_{1(1)} & \cdots & c_{1(j)} & \cdots & c_{1(m)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{i(1)} & \cdots & c_{i(j)} & \cdots & c_{i(m)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{k(1)} & \cdots & c_{k(j)} & \cdots & c_{k(m)} \end{pmatrix} \begin{matrix} a_1 \\ \cdots \\ a_i \\ \cdots \\ a_k \end{matrix} \end{array} \tag{3.9}$$

The unified set *F(A)* is the colour set, or property set, of the whole polychromatic set. The unified colour set, which is the relationship between the set *A* and the set of all elements with the same colour, can be formally represented as

$$\left\| c_{i(x)} \right\| = [A\times A(F)] \tag{3.10}$$

The complete polychromatic set is composed of the following components, not all of which are necessarily required when using PST.

$$PS = (A, F(a),\ F(A), [A \times F(a)], [A \times F(A)], [A \times A(F)]). \tag{3.11}$$

To summarise,

$\left[A \times F(a)\right]$ = Individual pigmentation of all the elements of a polychromatic set

$\left[A \times F(A)\right]$ = Relationship between the unified colours of a polychromatic set and the individual colour of element $a_i \in A$

$\left[A \times A(F)\right]$ = The constituents of the elements of the system entity that guarantees the existence of all unified colours of the polychromatic set.

$A(F)$ = Composition of all system entities of the unified colours.

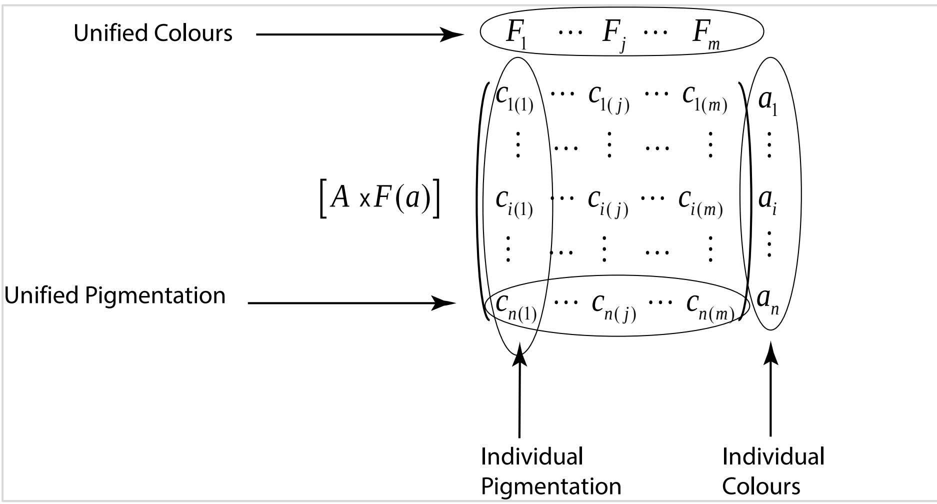


Figure 3-1: Summary or Rows & Columns for Polychromatic Set

3.3. Polychromatic Sets and their Operations

Polychromatic sets support both numerical and logical operations. An object's colour or property may be taken as a Boolean value. This being the case, the logical operations of conjunction and disjunction are supported.

Consider the colour set $F = (F_1, F_2, F_3, F_4, F_5, F_6)$ and the subset F_A where $F_A \subseteq F$ is composed of the values of the sets $F(a_i), F(a_j)$.

If $F(a_i) = (F_1, F_5)$ and $F(a_j) = (F_3, F_5, F_6)$ then

$$F(a_i), F(a_j) \subset F_A = (F_1, F_2, F_3, F_4, F_5, F_6) \tag{3.12}$$

The sets $F(a_i), F(a_j)$, which is the pigmentation, may be represented in terms of Boolean vectors, where 1 indicates the existence of the pigmentation and 0 indicates the non-existence of the pigmentation.

$$F(a_i) = (1,0,0,0,1,0) \tag{3.13}$$

$$F(a_j) = (0,0,1,0,1,1) \tag{3.14}$$

The logical conjunction of sets in (3.13) and (3.14) is given in equation (3.15) and the logical disjunction of the sets in (3.13) and (3.14) is given in equation (3.16).

$$F(a_i) \wedge F(a_j) = (1,0,0,0,1,0) \wedge (0,0,1,0,1,1) = (0,0,0,0,1,0) \tag{3.15}$$

$$F(a_i) \vee F(a_j) = (1,0,0,0,1,0) \vee (0,0,1,0,1,1) = (1,0,1,0,1,1) \tag{3.16}$$

3.4. Summary

The chapter has given an introduction of Polychromatic Set theory, showing it to be a profitable extension of the Cantor's set theory. Using the concept of colours or properties of elements of the set with Boolean vector manipulation, it has been shown that a wide variety of industrial problem may be solved using PST. It is the use of the inter-relationship between the properties of the elements of a set that makes PST very useful, powerful and, at the same time, simple to use. How this is applied to the current research of identifying sequencing of sub-blocks is the subject of the next chapter.

Chapter 4. Polychromatic Sets, Knowledge Base, Reasoning Engine and Sequencing Engine

The previous chapter outlined Polychromatic set theory with examples of its usage. This chapter illustrates PST's application in the construction of a knowledge base and the role it plays in the systems reasoning engine and system sequencing engine.

Figure 4-1 illustrated the part of the overall system architecture that this chapter focuses on. The local database's input is an XML formatted file exported from the NUPAS marine drawing software. The file contains, inter alia, all the vertices of a part's perimeter, or profile, the thickness of the part, the extrusion vector of the part, the part's weight, the part's total length and the part type, i.e., bracket, flange, plate, stiffener. This information is used to populate the local database. The local database holds separate database tables for each part type and these tables may be thought of as sets.

In order to have a fuller understanding of what is involved, this chapter continues by introducing the terminology used and then moves onto how the Polychromatic sets are involved in the knowledge base, reasoning engine and sequencing engine.

4.1. Constructing a Local Database

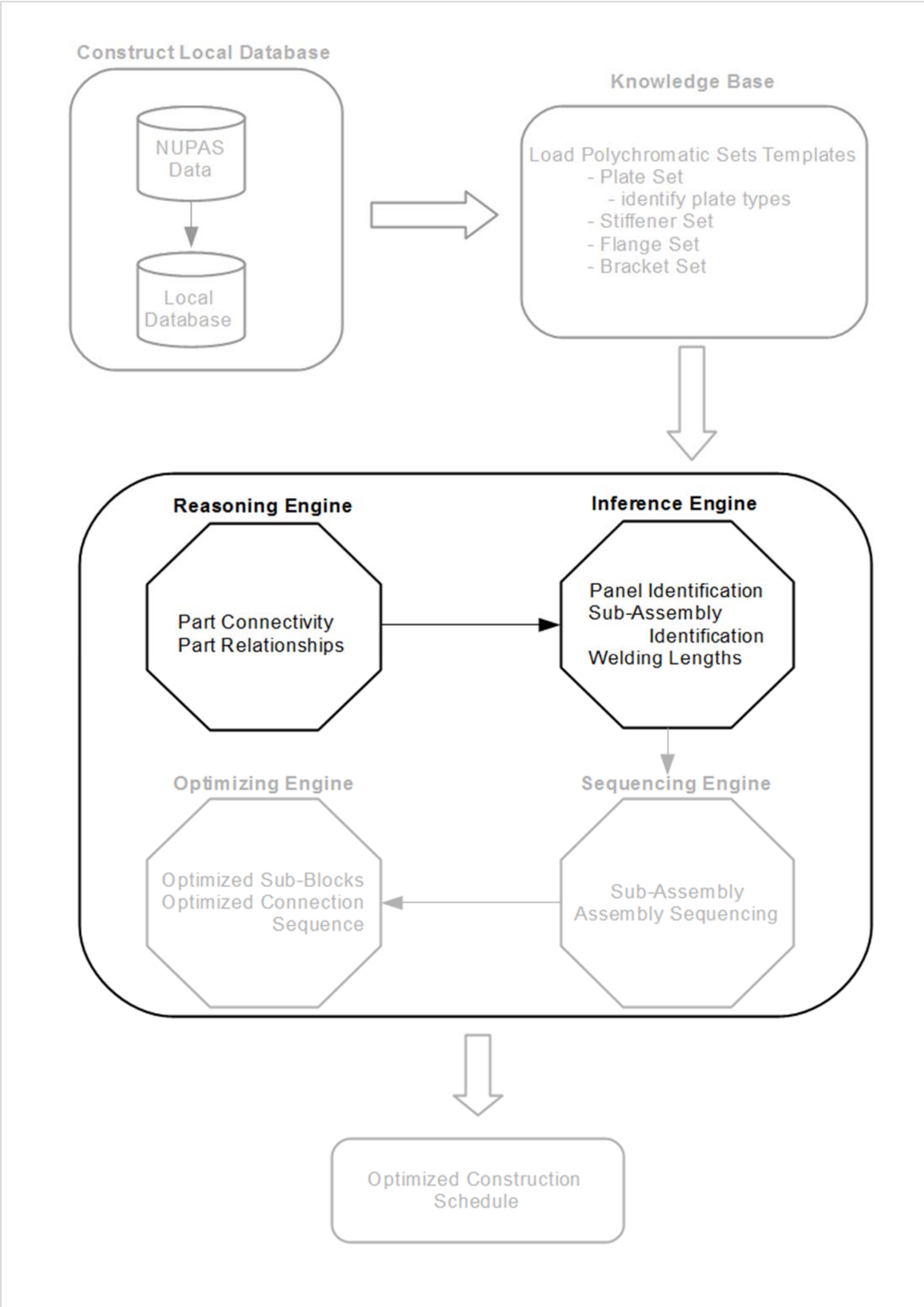


Figure 4-1: Partial System's Architecture

4.2. Terminology Used

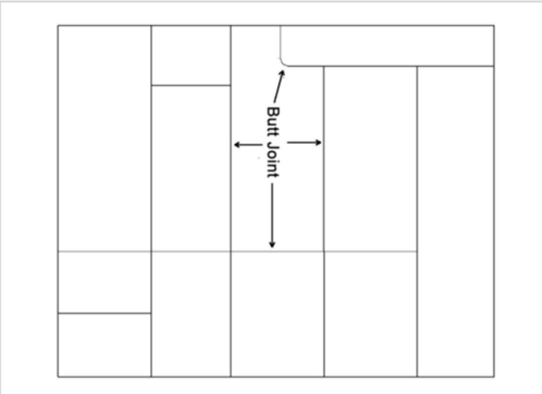


Figure 4-2: Example of Several Plates Welded to Form a Panel

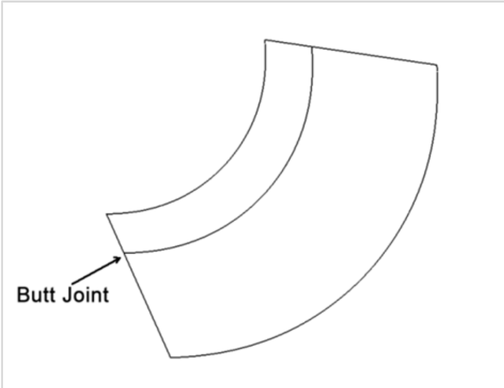


Figure 4-3: Example of 2 Shell Plates Making a Shell Panel

An offshore rig, as mentioned earlier, is composed of a number of different part types. In this research, the basic 'base' is that of a *plate.* A plate, as its name suggests, is a steel plate. Plates vary in shape, size and thickness. Plates are mostly flat or planar. However, plates may be curved. Curved plates are also referred to as *shell plates*. Plates are connected to other plates in two ways: butt connection, which is an edge-to-edge connection, and fillet connection, which is an edge-to-surface connection. A collection of plates that are butt welded is referred to as a panel. Panels can vary in size, depending on their function in a block. An example of a panel is in Figure 4-2 and an example of a shell plate is shown in Figure 4-3.

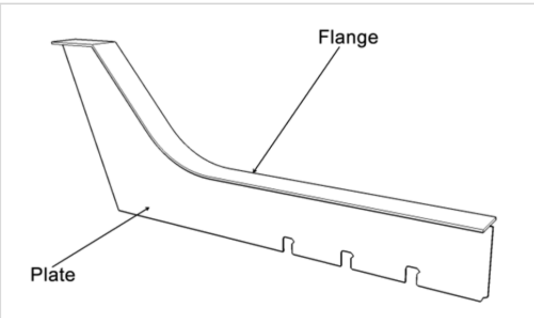


Figure 4-4: Example of an Exterior Flange

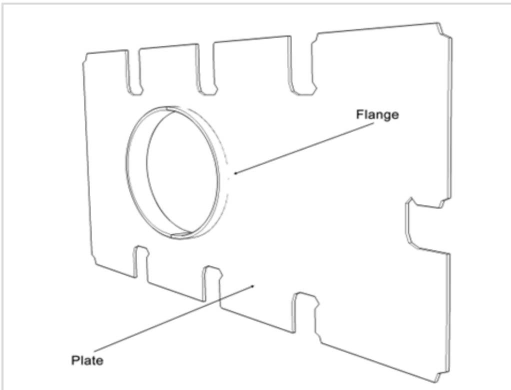


Figure 4-5: Example of an Interior Flange

Flanges are fillet connections of smaller plates that form a '*T-connection*' with larger plates. Figure 4-4 illustrates a flange connected to an edge of a plate. From the figure, it can be seen that the edge is on the plate's perimeter. However, flanges are not only joined to perimeter edges, but may also be joined to interior edges as is illustrated in Figure 4-5.

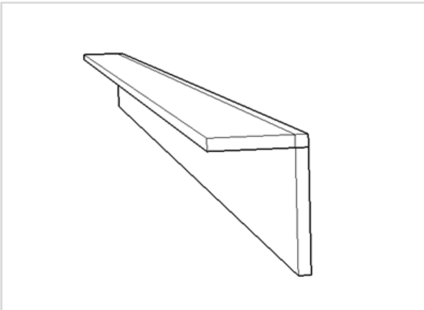


Figure 4-6: Example of a Stiffener

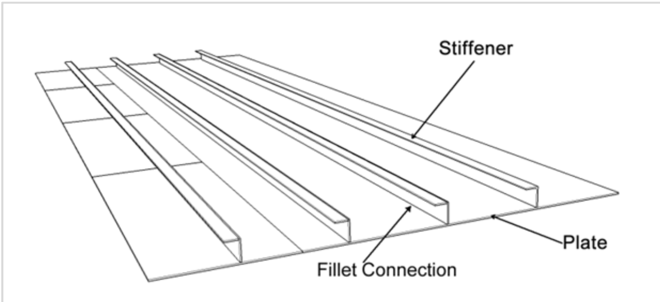


Figure 4-7: Example of Stiffeners on a Panel

Stiffeners have particular profiles and make fillet connections with a plate. Stiffeners can make butt or fillet connections with other stiffeners. Stiffeners vary in length and thickness. Figure 4-6 shows an example of a stiffener. This is just one example of the many and varied profiles that are available for stiffeners. Figure 4-7 shows a number of stiffeners on a panel.

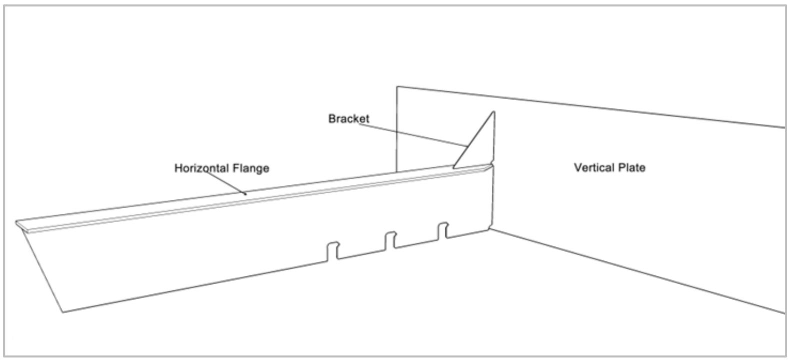


Figure 4-8: Bracket between Plate and Flange

Brackets are used to add rigidity to the block construction and are often welded to upright and horizontal members. Figure 4-8 shows a bracket between a vertical plate and a horizontal flange. Brackets make fillet connections with plate, stiffeners and flanges.

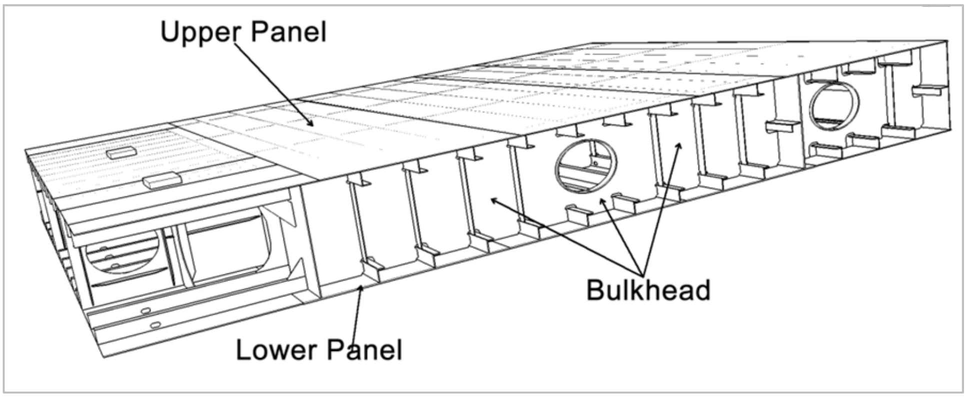


Figure 4-9: Bulkheads between Upper and Lower Panels

Bulkheads are vertical plates relative to the horizontal base of the block. Bulkheads in effect go from the base of block to the top of the block. This is illustrated in Figure 4-9. Note here the terms *vertical* and *horizontal* are used in relationship to the illustrations. In practice, the vertical plane may be, for example, horizontal.

Bespoke plates – sometimes it is necessary for the design engineer to design a non-standard part, such as a special bracket. Not only in NUPAS, but also in other ship design software, brackets, for example, are standard parts with a predefined relationship between edge and curves, for examples. The same can be said for stiffeners. As mentioned, stiffeners are bought-in parts that have a well-defined profile. However, sometimes the design requires a non-standard stiffener, in which case, instead of being bought-in, it would be fabricated at the shipyard. Such special parts are drawn as plates, but have a different function to that of a plate. Further, these *bespoke parts* are identified as ‘plates’ by NUPAS, the ship’s drawing package. The real distinguishing point between these *bespoke plates* that are identified as plates, but in fact serve a different function to that of a standard plate, is their size. So as not to confuse these *bespoke parts* as plates, this work designates them as *bespoke plates*.

A *subassembly* is a plate with stiffeners and/or flanges attached to it, but with no other plates attached. A subassembly is the simplest multipart construction. The example shown in Figure 4-10 is a plate and a stiffener.

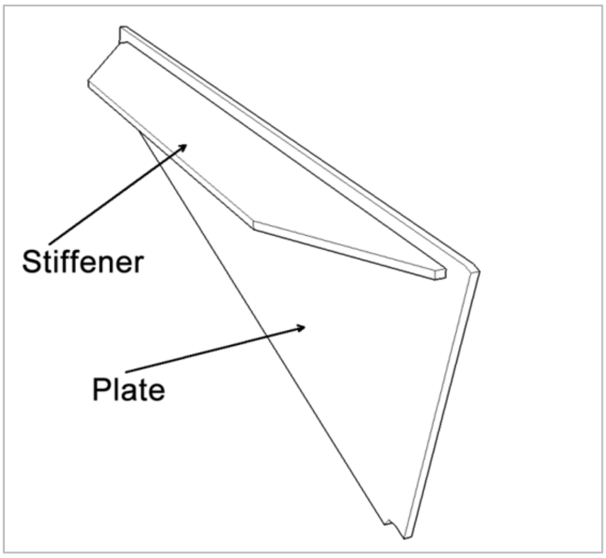


Figure 4-10: A Simple Subassembly.

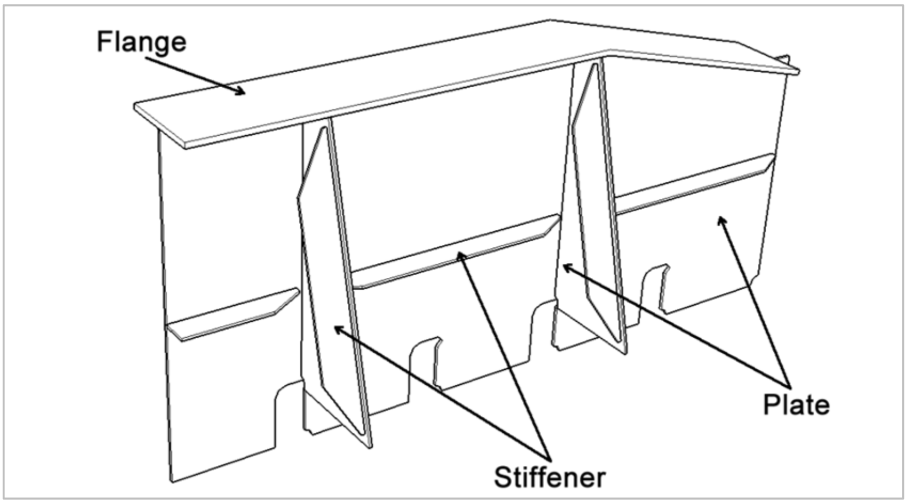


Figure 4-11: Sub-block Example

A sub-block, as illustrated in Figure 4-11, is a plate with other parts *connected to* or *welded to*[1], it. These other parts may make a butt or fillet connection.

X-direction, Y-direction and *Z-direction* - this notation is used to identify one of the directions in standard 3D space, using the standard 3D coordinate reference. This notation will be used when referring to vertices of parts. All parts, in this work, are considered as objects in 3D space. Part information is exported by NUPAS in XML format contains, among other data, a part's vertices relative to an origin defined by the design engineer. The origin is defined for the whole rig by the design engineer. The use of the term *part* in the title of this section may need some explaining. A *part* here can refer to a plate, a sub-block, a subassembly, a bracket, a stiffener or a flange. The context dictates the usage of the term.

Ship's Design software is software that is used by a ship designer. It is to the ship designer to what *Revit* is to an architect or *Autocad* is to a mechanical engineer. The same software that is used to design ships is also used to design offshore rigs. As there are many different drawing packages for the architect, so there are many different drawing packages for the ship design engineer. As with packages such as *Revit* and *Autodesk*, ship's design software internal file structure is proprietary. However, many packages have a scripting language that gives the user some limited facility to create files with a bespoke structure. *Tribon*, for example has a scripting language, *Vitesse*, which is a dialect of *Python*. The package AVEVA Marine has its own Program Macro Language (PML) for users use.

The candidate shipyard used in this research uses Nupas-Cadmatic. This package does not have a scripting or programming language. It does however, have a

[1] The terms 'connected to' and 'welded to' are used interchangeably in this thesis.

program that produces, in XML format, a file containing topological and physical data. An example of the XML format is reserved for the case study, Figure 7-2

4.3. Polychromatic Set Theory Used in Reasoning

Polychromatic Set Theory has been employed in developing the algorithm for the assembly sequencing of the plates. In this section, the various sets that will be used are identified. Following this section, the heuristic will be given in terms of the sets developed here.

The first step in using PST for part sequencing is to identify the different parts and their topological relationship with each other. This relationship is obtained by the system-reasoning engine, which uses the created sets. It is through the reasoning engine that the topological relationships of sub-blocks/subassemblies can be automatically identified, which is the primary stage in optimizing their assembly sequences.

In the following discussion, reference is made to *vertical* and *horizontal*. This is relative to the base of the block under consideration.

Table 4-1: Connectivity Table

	Plate	Stiffener	Flange	Bracket
Plate	*			
Stiffener	*	*	*	
Flange	*		*	
Bracket	*	*	*	*

The four basic parts that compose a block: plates, stiffeners, flanges and brackets have a topological relationship to each other. Table 4-1, which is read from left to right and top to bottom, shows the order in which connectivity is dealt with in this research.

Plates can be connected to other plates. Since the plate is the fundamental part, the other parts are considered as connected to the plate, not the plate connected to the other parts, i.e. connectivity is from the perspective of the plate. Although it is true that a part connected to plate is also a plate connected to the part. In functional processing terms, the small part is considered as connected to the larger part.

The discussion that follows describes the use of PST to identify the relationship and function of plates in a block. As all smaller parts are connected to a plate or plates, the use and topological relationship of plates play an important part in the sequencing of parts for the assembly process. If the assembly sequence of the plates can be sequenced correctly, then the work of optimizing the construction of the sub-blocks/subassemblies follows.

The part identification used on the local database, and originating from NUPAS, has some exceptions. Parts identified as plates are not always true plates; sometimes parts identified as plates are bespoke brackets. Such exceptions cannot easily be identified and, in such cases, are treated as plates.

The reasoning engine, using PST, does the following:-

- Identifies top and bottom plates of a double bottom hull or the bottom plates of an open hull.
- Identifies bulkhead plates
- Identifies connectivity for brackets, stiffeners, flanges, bespoke plates.

The following discussion describes how PST is used in the reasoning engine. Table 4-1 shows the Polychromatic sets of connectivity that are necessary for the processing of assembly sequencing. These 'sets' are derived sets, as will be seen as the discussion moves on.

Let $A=(a_1,\cdots,a_i,\cdots,a_n)$ be the set of all plates in a block. Let $F(A)=(F_1(A),F_2(A),F_3(A),F_4(A))$ represent the colours of the plate set. Where

F_1 = Plate orientation is horizontal

F_2 = Plate orientation is vertical

F_3 = Plate is at the lower boundary of the block.

F_4 = Plate is at the upper boundary of the block

$[A\times F(A)]$ is the Cartesian cross product of the set of all plates in a block with properties, or colours as defined above in F_1,F_2,F_3 and F_4, the results of which can be seen in the matrix equation (4.1). This, as for all following matrices, is the general form that is used. It does not apply to any specific block.

$$\left[A\times F(A)\right]=\begin{matrix} (a_1) \\ (a_2) \\ \vdots \\ (a_j) \\ \vdots \\ (a_n) \end{matrix}\overset{\begin{matrix} F_1 & F_2 & F_3 & F_4 \end{matrix}}{\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 \end{pmatrix}} \tag{4.1}$$

From the set $A\times F(A)$, a number of other sets are derived. Firstly, the set of plates that is in the horizontal plane and at the upper boundary of the block. Secondly, the set of plates that is in the horizontal plane and at the lower boundary of the block. Thirdly, the set of plates that are bulkheads. Fourthly, the set of residual plates, that is, plates that are in none of the first three sets identified. Bulkheads, as may be seen in Figure

4-9, are full height plates, having a connection with one or more top plates, if this block is a double bottom and with one or more plates for both double bottom and open hull blocks. Bulkheads are identified since they form the initial plate for the assembly sequencing.

Let the set of plates at the lower boundary of the block be represented by the set *L*, where

$$L = \{l_1, l_2, l_3, \cdots, l_i \cdots l_n\} \quad (4.2)$$

Let the set of plates at the upper boundary of the block be represented by the set *U*, where

$$U = \{u_1, u_2, u_3, \cdots, u_i \cdots u_n\} \quad (4.3)$$

Let the set of plates that are not in set *L* or *U* be represented by the set *O*, where

$$O = \{o_1, o_2, o_3, \cdots, o_i \cdots o_n\} \quad (4.4)$$

The sets *L*, *U* and *O* make up the set $[A \times F(A)]$. From these sets, all plate connectivity can be obtained. Not all blocks have both top and bottom panels. The block in Figure 4-13 only has a bottom panel, whereas the block in Figure 4-12 has both top and bottom panels. If the block does not have a top panel, which is composed of several plates, then the set *U* will be an empty set.

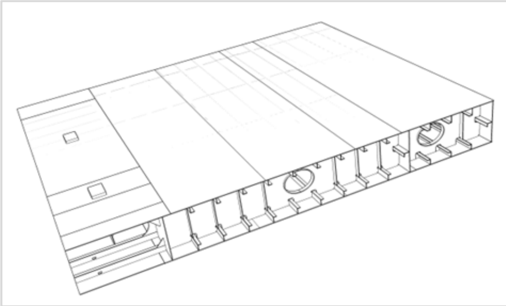


Figure 4-12: Double Bottom Block

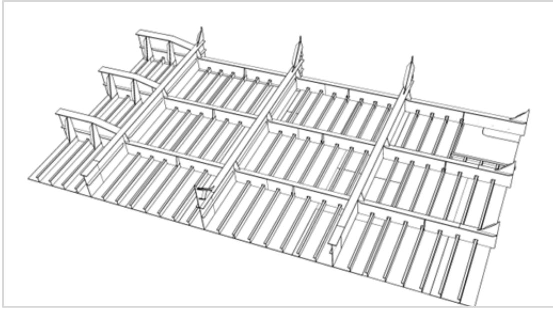


Figure 4-13: Open Hull Block

Consider the set $L \times F(L)$ where the colour F_1 is the connection of the plate in the *X-direction* and the colour F_2 is the connection of the plate in the *Y-direction*, as illustrated in equation (4.5) . As mentioned, these matrices are the general form used, they do not apply to a specific block or set of parts found in this discussion.

$$\left\| l_{i(j)} \right\|_{L,F(L)} = \left[L \times F(L) \right] = \begin{matrix} \\ (l_1,l_2) \\ (l_1,l_3) \\ \vdots \\ (l_i,l_{i+1}) \\ \vdots \\ (l_i,l_n) \\ \vdots \\ (l_{n-1},l_n) \end{matrix} \begin{matrix} \begin{matrix} F_1 & F_2 \end{matrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ \vdots & \vdots \\ 1 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{pmatrix} \end{matrix} \tag{4.5}$$

Where $l_{i(j)} = \begin{cases} 1, F_j \in F(l_i) \\ 0, otherwise \end{cases}$

To ensure that not only the connectivity is identified but also that the connectivity is in the required order for the assembling of the bottom set of plates necessary to create a panel, the resultant set $L \times F(L)$ is ordered into ascending *X-direction,* then ascending *Y-direction*. The same procedure is performed for the Cartesian cross product $U \times F(U)$. Figure 4-22 illustrates this for a panel. The sets establish the fact of connectivity as well as the direction of connectivity. For example, plate 3 connects to plate 4 in the X-direction and connects to plate 6 in the Y-direction.

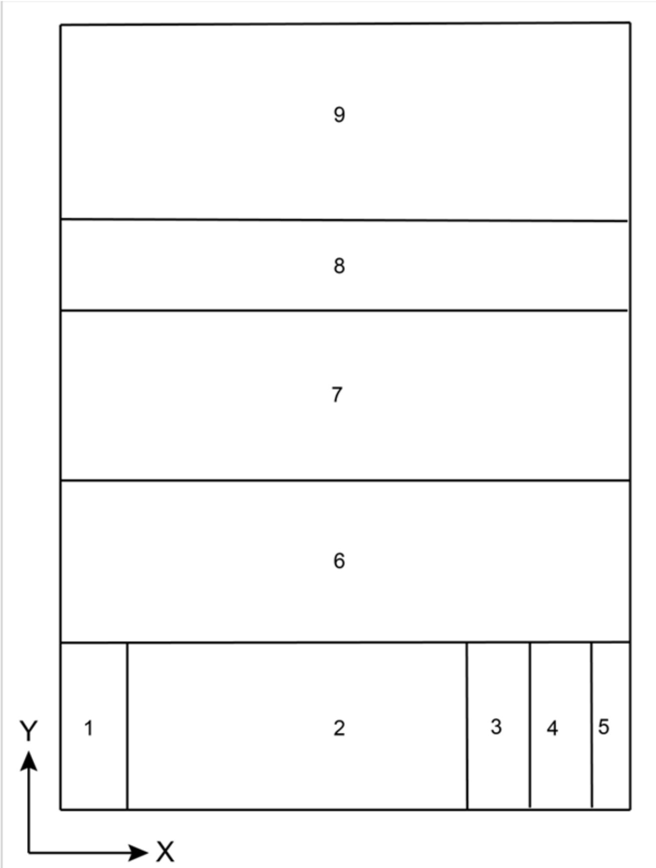


Figure 4-14: Panel with Butt Connected Plates

The set $O = \{o_1, o_2, o_3, \cdots, o_i \cdots o_n\}$ is the set of parts that is neither in the top or bottom set of plates; this set contains bulkheads and residual plates. Let the set $L' = [L \times F(L)]$. Let the set $U' = [U \times F(U)]$. The set $[O \times F(L')]$, represented by the matrix equation (4.8), is the set in which all the plates are connected to the base panel or lower panel. The set $[O \times F(U')]$ represented by the matrix equation given in equation (4.9), is the set of all plates connected to the upper panel. Plates that are in both sets, $[O \times F(L')]$ and $[O \times F(U')]$, are bulkheads, and form the bulkhead set *SB*. The bulkhead set is composed of plates that are connected to both the top and bottom panels.

$$SB = [O \times F(L')] \cup [O \times F(U')] \tag{4.6}$$

Open hull blocks will have plates only in the $[O \times F(L')]$ set with the set $[O \times F(U')]$ being an empty set. The bulkheads for open hulls consist of plates connected to the lower panel only.

$$SB = [O \times F(L')] \tag{4.7}$$

A Double bottom hull may have plates in either set $[O \times F(L')]$ or set $[O \times F(U')]$, but not both. These plates are often flange plates. For an open hull block, flange plates will be identified only by their size.

The initial plate for the assembly sequencing will be a bulkhead, if the block is a double bottom block or its equivalent for an open hull block. If the block is a double bottom block, the initial plate or candidates for the initial plate are found in the union of sets from the matrix equation (4.8) and matrix equation (4.9), and is given in equation (4.10).

$$\left\| o_{i(j)} \right\|_{O,F(L')} = \left[O \times F(L') \right] = \begin{matrix} \\ (o_1, l_1) \\ (o_1, l_2) \\ \vdots \\ (o_i, l_n) \\ \vdots \\ (o_n, l_n) \end{matrix} \begin{matrix} \begin{matrix} F_1 & F_2 \end{matrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \end{pmatrix} \end{matrix} \tag{4.8}$$

Where $o_{i(j)} = \begin{cases} 1, F_j \in F(o_i) \\ 0, otherwise \end{cases}$

$$\left\| o_{i(j)} \right\|_{O,F(U')} = \left[O \times F(U') \right] = \begin{matrix} \\ (o_1,u_1) \\ (o_1,u_2) \\ \vdots \\ (o_i,u_n) \\ \vdots \\ (o_n,u_n) \end{matrix} \begin{matrix} F_1 & F_2 \\ \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \end{pmatrix} \end{matrix} \tag{4.9}$$

Where $o_{i(j)} = \begin{cases} 1, F_j \in F(o_i) \\ 0, otherwise \end{cases}$

The resultant set of the union is given in equation

$$P = \left\| o_{i(j)} \right\|_{O,F(U')} \cup \left\| o_{i(j)} \right\|_{O,F(L')} \tag{4.10}$$

In Polychromatic set represented by matrix equations (4.8) and (4.9) , the colour $F_{1=}$ connection in the –Z direction and the colour $F_{2=}$ connection in the +Z direction. Equation (4.10) is the set of bulkheads.

The set $O = \{o_1, o_2, o_3, \cdots, o_i \cdots o_n\}$ of other plates have a butt, fillet or no relationship with each other. This is shown in matrix equation (4.11). In this equation, the colour F_1 is a butt connection and the colour F_2 is a fillet connection.

$$\left[O \times F(O) \right] = \begin{matrix} \\ (o_1,o_2) \\ (o_1,o_3) \\ \vdots \\ (o_i,o_j) \\ \vdots \\ (o_{n-1},o_n) \end{matrix} \begin{matrix} F_1 & F_2 \\ \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \end{pmatrix} \end{matrix} \tag{4.11}$$

Where $o_{i(j)} = \begin{cases} 1, F_j \in F(o_i) \\ 0, otherwise \end{cases}$

All connectivity shown in Table 4-1 is derived from Polychromatic sets using the part's topology. Since the part type (bracket, stiffener, flange or plate) is known, the various set connectivity is reasoned out by applying the appropriate Cartesian cross product. If P is the set of all plates, S is the set of all stiffeners, B is the set of all brackets and FL is the set of all flanges, then the required sets are $[P \times F(S)]$, $[P \times F(B)]$, $[P \times F(FL)]$, $[S \times F(S)]$, $[S \times F(B)]$, $[S \times F(FL)]$, $[B \times F(B)]$, $[B \times F(FL)]$, $[FL \times F(FL)]$. The reasoning here is to find what parts connect to what other parts and how they are connected. Again, these are general equations, not specific to any block. The archetypical Polychromatic set Cartesian Cross product of these sets is in matrix equation (4.12) where *N* is the set *P* or *S* or *B* or *FL* and *F(M)* is the set *F(S)* or *F(B)* or *F(FL).*

$$[N \times F(M)] = \begin{matrix} & F_1 \quad F_2 \\ \begin{matrix} (n_1, m_1) \\ (n_1, m_2) \\ \vdots \\ (n_i, m_n) \\ \vdots \\ (n_n, m_n) \end{matrix} & \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ \vdots & \vdots \\ 0 & 1 \end{pmatrix} \end{matrix} \tag{4.12}$$

In matrix equation (4.12), the colour F_1 is a butt connection and the colour F_2 is a fillet connection. In this general equation, a 1 in the F_1 column indicates that the connection is a butt connection and a 1 in the F_2 column indicates that the connection is a fillet connection.

Consider Figure 4-14 by way of example. This is a panel composed of 9 plates with butt connections. Applying matrix equation (4.12) to this specific case and for illustration purposes, the resultant matrix (4.13) is a snippet showing only the connections for plates 1, 2, 3, 4, 5 and 6.

$$
[P \times F(P)] = \begin{array}{c} \\ (1,2) \\ (1,3) \\ (1,4) \\ (1,5) \\ (1,6) \\ \cdots \\ (2,3) \\ (2,4) \\ (2,5) \\ (2,6) \\ \cdots \\ (3,4) \\ (3,5) \\ (3,6) \\ \cdots \\ (4,5) \\ (4,6) \\ \cdots \\ (5,6) \\ \cdots \end{array} \begin{array}{c} \begin{array}{cc} F_1 & F_2 \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ \vdots & \vdots \end{pmatrix} \end{array} \tag{4.13}
$$

It may be observed that the resultant Polychromatic set matrices are sparse. This is not a problem in using them, and the methods used in programmatic terms on dealing with the searching of Polychromatic set matrices is discussed in Chapter 7, where details of the model used to demonstrate the theory is given.

The number of comparisons using polychromatic sets for plate part-pair connection is initially of the order of $O(n^2)$, where n is the total number of plates. If this is extended to the whole block, the initial part-pair comparison is m^2 where m is the total number of parts for the block.

When comparing the elements of the set for sequencing, the set is traversed just once, thus computational time is minimal. The number of comparisons is of the order $O(n^2)$. Having identified the starting point of the assembly sequence, the subsequent part in the sequence is implicit; there is no need to iterate through the whole set in order to identify it. For example, consider the part-pairs $(o_i,o_j),(o_j,o_k),(o_k,o_m)$. Having identified (o_i,o_j) as a part-pair for the first two parts of an assembly sequence, the next part in the sequence is o_k which is obtained from the part-pair (o_j,o_k) and the subsequent part is o_m which is obtained from the part-pair (o_k,o_m) . Having identified the first link in a chain, subsequent links are readily identified.

Parts of the set *O*, that are not in the set $\left[O \times F(O)\right]$, are plates that are connected directly to the plates of the top or bottom or connected to existing plates and not bulkheads. Bulkheads are connected to both the top and bottom plates, whereas the parts in set *O*, that are not in the set $\left[O \times F(O)\right]$, are only connected to either a top plate or a bottom plate.

4.4. Developing the Sequencing Engine

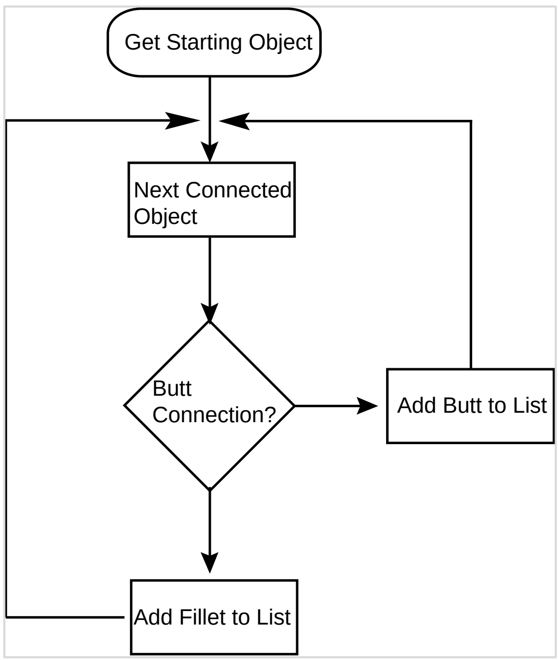


Figure 4-15: Basic Flow for Object Selection

The following discusses the development of the sequencing engine necessary for the assembly sequencing of plates; a practical example is described in section 4.6. The engine's focus is on the plates, since, as may be seen in Table 4-1, everything connects to a plate. The first step is to identify a starting object. This will be by plate that will, in practice, be a subassembly that has the lowest *Z-direction*> lowest *Y-direction*> lowest *X-direction* for its bounding box.

Subsequent selection of next plate/subassembly requires that if there is more than one connection, then a butt connection has precedence over a fillet connection. The butt connection extends the existing plate of a subassembly or plates if the object is a sub-block. The plate is the primary part in sequencing, as has been mentioned. The selection precedence is illustrated in Figure 4-15. This is not a complete flowchart of the system by any means; it omits many things including returning to an object that has more than one subassembly connected it. The purpose of this snippet of a flowchart is to show the order of selection. The pseudo code of the full flow is in Table 4-2.

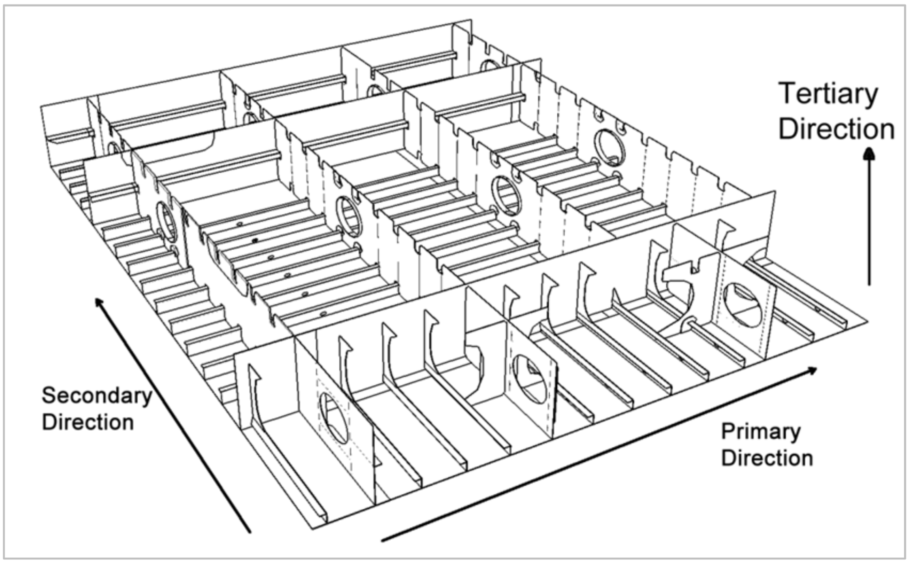


Figure 4-16: General Principle for Assembling Parts on a Block

As may be seen from Figure 4-15, every connection has either a butt or a fillet connection as its next connection. The additional connections follow the same precedence of butt connection before fillet connection. The processing to deal with this correctly, which is not illustrated in Figure 4-15 but is illustrated in Table 4-2, is a recursive process.

The sequencing focuses only on plates, which are the base for subassemblies. Once the plate assembly sequence has been identified, it is possible to optimize the combination of the plates to make sub-blocks, but that is running ahead a little.

$$\left\| p_{i(j)} \right\|_{\mathrm{P},F(\mathrm{P})} = \left[P \times F(P) \right] = \begin{array}{c} \\ (p_1, p_2) \\ (p_1, p_3) \\ \vdots \\ (p_{i-1}, p_n) \\ (p_i, p_n) \\ (p_{i+1}, p_n) \\ \vdots \\ (p_{n-1}, p_n) \end{array} \begin{array}{c} \begin{array}{cc} F_1 & F_2 \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{pmatrix} \end{array} \tag{4.14}$$

The plate part set $P = \{p_1, p_2, p_3, \cdots, p_i \cdots p_n\}$ is an ordered set of plates that are ordered into lowest *Z-direction*> lowest *Y-direction*> lowest *X-direction*. These are the vertical plates that compose the block; they are not the plates that are part of the bottom panel, and if it is a double bottom hull, the top panel. The lowest Z, Y, X direction values are taken from the vertices that describe the part in 3D space. Ordering the set of parts this way ensures that parts are dealt with from the base panel upward, and the precedence sequence illustrated in Figure 4-16 is maintained. The primary direction and secondary direction show the priority in the selection of plates. If there is a contention over which plate to select, then the one that has the lowest primary direction value (*X-direction)* is the one to choose. This will be covered in more detail in section 4.6. The set of bulkheads[2] P has been defined in equation (4.10). The Polychromatic set in equation (4.14) is the result of taking the Cartesian cross product $([P \times F(P)])$ of the set with itself, where the colour F_1 is a butt connection and the colour F_2 is a fillet connection. The resultant matrix of taking the Cartesian cross product identifies how one plate is connected to another plate: whether the plate is butt connected, fillet connected or not connected to another plate. The resultant matrix is used to reason out the assembly sequence of the vertical (bulkhead) plates. In general, from the matrix equation (4.14), it can be seen that p_1 connects to p_2 with a butt connection and p_i connects to p_n with a fillet connection since there is a 1 in the respective column but p_{n-1} has no connection with p_n.

The assembly sequence of the bulkhead plates is governed by the precedence illustrated in Figure 4-16. Matrix equation (4.6) applies if the block is a double bottom.

Matrix equation (4.7) applies if the block is an open hull and only to the plate pairs that have a 1 in the F_1 column. A set of plates that are connected by a butt connection is thus derived. The resultant bulkhead plate set is $SB = \{sb_1, sb_2, sb_3, \ldots, sb_n\}$. The individual connectivity is reasoned from the matrix equation (4.15), where the colour F_1 is 1 if the part pair (b_i, b_j) is connected by butt connection and 0 if it is not connected by a butt connection.

[2] As the plates in this example extend from the bottom panel of the block to the top panel of the block, they are referred to here as bulkheads; they could just as easily be referred to as vertical plates.

$$
[SB \times SB] = \begin{matrix} (b_1,b_2) \\ (b_1,b_3) \\ \vdots \\ (b_{i-1},b_n) \\ (b_i,b_n) \\ (b_{i+1},b_n) \\ \vdots \\ (b_{n-1},b_n) \end{matrix} \overset{F_1}{\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}} \tag{4.15}
$$

A plate can have no more butt connections than it does edges. A rectangular plate can have at most four butt connections; similarly, an octagonal plate can have at most 8-butt connections. A set for an individual plate, sb_i that has *m* edges can be reasoned using the matrix equation(4.15), which gives the set of butt connections as in (4.16), which is a subset of the matrix (4.15).

$$
BC_i = \{bc_{(i)1}, bc_{(i)2}, bc_{(i)3}, ..., bc_{(i)m}\} \tag{4.16}
$$

Let *IB* be the set of *initial search bulkheads*, which are composed of bulkheads from the set $[P \times F(P)]$ where the part is first used in a sequence of connections.

The set *IB* is iterated. For each member of the set *IB*, a set of parts that have butt connections IBB_i is built, where IBB_i (the set of initial bulkhead butt connections) is the set for the i^{th} member of the set *IB*. Butt connections have a higher priority than fillet connections. There will be at most one butt connection for each edge of the member of the set *IB*. Having built the sets IBB_i, the next step is to identify the fillet connections to each member of the set IBB_i.

$$BSA = \left[IBBi \times F(P) \right] = \begin{pmatrix} F_1 & F_2 & F_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \end{pmatrix} \tag{4.17}$$

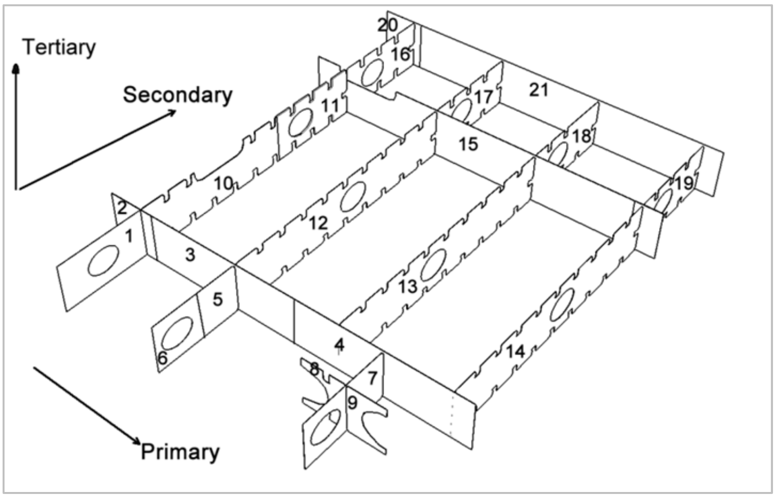


Figure 4-17: Example of Plate Assembly Sequence

Equation (4.17) is the result of taking the Cartesian cross product of the plates that have butt connections with that of all parts. F_1 is a part that has fillet connections that have a lower *Y-direction* value than the part in set IBB_i. F_2 is a part that has fillet connections that have a higher *Y-direction* value than the part in set IBB_i. F_3 is a part that does not have a fillet connection to the part in IBB_i. This is best illustrated by looking at parts 7, 8 and 9 in Figure 4-17 Part 7 has two plates that have fillet connections.

4.5. Assembly Sequencing Heuristic

Table 4-2: Pseudo-code of Assembly Sequencing Heuristic

```
Start:
  Create Bottom-Plate-Set [BS]
  Create Top-Plate-Set [TP]
  Create Vertical-Plate-Set [VP]
  Foreach member {VP_i} of [VP]
      Add {VP_i} to Assembly-List-Set [ALS]
      Build Butt-List-Set [BLS] for {VP_i}
      Add members of [BLS] to [ALS]
      Build Fillet-List-Sets for {VP_i}
      Expand Butt-List-Set [BLS],[FSH]
 End Foreach
End:

Build Fillet-List-Set for {VP_i}
  ForEach Fillet Connections to {VP_i}
                Build HigherY-Fillet-Set [FSH]
                Build LowerY-Fillet-Set [FSL]
        End ForEach
End Build Fillet-List-Set:

Build Butt-List-Set for {VP_i} :
        ForEach Butt Connections to {VP_i}
             Add to Butt-List-Set [BLS] plate {VP_i'}
      *Build Butt-List-Set [BLS_vp_i'] {VP_i'}
        End ForEach:
        Order Butt-List-Set [BLS_vp_i'] low.X, asc, low.Y,asc
End Build Butt-List-Set:

Expand Butt-List-Set [BLS], [FS] ):
       (note [FS] is either [FSH] or [FSL])
  Foreach member {BLS_i} in [BLS]
           Add member to [ALS]
        End Foreach

        Foreach member , {BLS_i} in [BLS]
     Build Fillet-List-Sets [FSL^L] and [FSH^L] for {BLS_i}
     Foreach member, {FSL_i^L} in [FSL^L]
```

```
        Add {FSL_i^L} to [ALS]
        Build Butt-List-Set [BLS^L] for {FSL_i^L}
        Build Fillet-List-Set [FSL^L2] and [FSH^L2] for {BLS_i^L}
            *Expand Butt-List-Set [BLS^L], [FSL^L2]
         End Foreach

         Foreach {FS_i} in Fillet Set [FS]
           Add {FS_i} to [ALS]
           Build Butt-List-Set [BLS^L2] for {FS_i}
           Foreach {BLS_i^L2} in [BLS^L2]
                Add {BLS_i^L2} to [ALS]
            Build Fillet-List-Set [FSL^L3] and [FSH^L3] for {BLS_i^L2}
           *Expand Butt-List-Set [BLS^L2],[FSL^L3]

         Foreach Foreach member , {BLS_i} in [BLS]
           Foreach connected part {FSH_i^L} in [FSH^L]
             Add {FSH_i^L} to [ALS]
             Build Butt-List-Set [BLS^L2] for {FSH_i^L}
             Build Fillet-List-Set [FSL^L4] and [FSH^L4] for [BLS^L2]
                 Expand Butt-List-Set-2 [BLS^L2], [FSL^L4]
              End Foreach
              *Expand Butt-List-Set [BLS^L2], [FSH^L4]
          End Foreach
```

EndExpendButtSet:

Expand Butt-List-Set-2 $[BLS]$, $[FS]$ **):**

```
        (note [FS] is either [FSH] or [FSL])
    Foreach member {BLS_i} in [BLS]
           Add member {BLS_i} to [ALS]
         End Foreach

    Foreach member {BLS_i} in [BLS]
        Build Fillet-List-Set [FSL^L] and [FSH^L] for [BLS]
            Foreach member {FSL_i^L} in [FSL^L]
                Add {FSL_i^L} to [ALS]
           Build Butt-List-Set [BLS^L] for {FSL_i^L}
           Built Fillet-List-Set [FSL^L2] and [FSH^L2] for [BLS^L]
           *Expand Butt-List-Set-2[BLS^L], [FSL^L2]
        End Foreach
    End Foreach
```

```
Foreach member {FS_i} in [FS]
        Add {FS_i} to [ALS]
        Built Butt-List-Set [BLS^L2], {FS_i}
        Foreach {BLS_i^L2} in [BLS^L2]
                Add {BLS_i^L2} to [ALS]
            Build Fillet-List-Set [FSL^L3] and [FSH^L3] for [BLS^L2]
           *Expand Butt-List-Set-2[BLS^L2], [FSL^L3]
        End Foreach
EndButtSet2:
```

The heuristic for assembly sequencing requires the use of connectivity of various parts. As the assembly sequencing process sequences only plates, the connectivity, butt or fillet and relative position in the block needs to be identified. Table 4-2 is the pseudo-code solution for the heuristic. As may be seen, lines that are marked with an asterisk (*) are recursive function. This heuristic has a nested recursive function within a recursive function.

The sets in the heuristic and the notation may need some explanation. Set names are in square parenthesis, for example $[BS]$. A general member of a set is shown in curly braces with the suffix *i*, indicating the i^{th} member of the set. For example, $\{VP_i\}$, is the i^{th} member of the set $[VP]$; the number of members of a set varies.

Statements such as "Foreach member $\{VP_i\}$ of $[VP]$" indicate an iteration for every member of the set. "End Foreach" indicates the end of the iteration. Superscripts are used for sets and set members to indicate that the set content is different for the same set type.

Several procedures or functions are used: *Build Fillet-List-Set, Build Butt-List-Set, Expand Butt-List-Set, Expand Butt-List-Set-2*. These functions are invoked or called, and have parameters passed to them, such as “Expand Butt-List-Set $[BLS]$, $[FSH]$”. In this example, the function “Expand Butt-List-Set” uses two already created sets, $[BLS]$ and $[FSH]$.

4.5.1. Function Usage in Heuristic

4.5.1.1. Build Fillet-List-Set

This function builds two sets of fillets lists. One set, designated as $[FSH]$, is the set of plates connected to the candidate plate, $\{VP_i\}$, whose lowest coordinate in the *Y-direction* is greater than the highest coordinate in the *Y-direction* of the candidate plate. The other set, $[FSL]$, is the set of plates connected to the candidate plate, $\{VP_i\}$, whose lowest coordinate in the *Y-direction*, is less than or equal to the highest coordinate in the *Y-direction* of the candidate plate.

4.5.1.2. Build Butt-List-Set

This function builds a set of plates that are connected to the candidate plate by a butt joint. This is a recursive function, as each plate that is added to the set itself becomes a candidate plate whose butt connections are also added to the set.

4.5.1.3. Expand Butt-List-Set

The parts in the parameter set are added to the *Assembly List Set*. For every plate in the parameter list set, a fillet list is built. For every member of the lower fillet list, a butt list is built.

4.5.1.4. Expand Butt-List-Set-2

This function, which is also a recursive, is called from the recursive function ‘*Expand Butt-List-Set*’. The parts in the parameter set are added to the *Assembly List*

Set. For every plate in the parameter list set, a fillet list is built. For every member in the lower fillet list, a butt list is built.

4.6. An Illustration of the General Principles of the Sequencing Engine

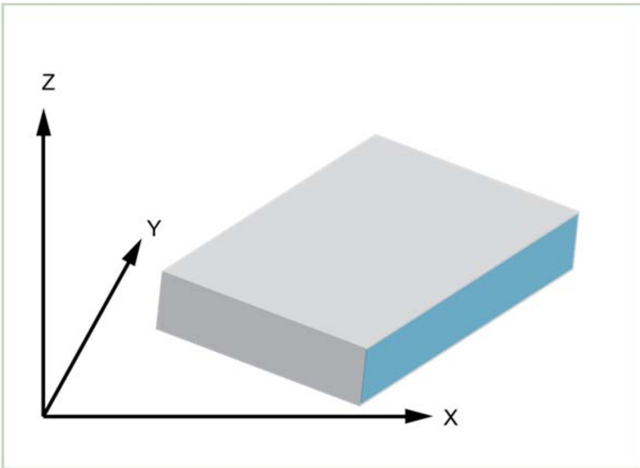


Figure 4-18: Example of a Block with Edges Not Coincident with Axes

The general principles of assembling parts on the base panel of a block were illustrated in Figure 4-16. Using the standard Cartesian notation, the *primary direction* shall be denoted as the *X-direction* and the *secondary direction* as the *Y-direction.* Blocks are not always designed to be such that they are nicely orientated along the X-Y-Z axes as shown in the various illustrations in this work, although a majority of them are. The examples used in this work are taken from a real project and are aligned on the X-Y-Z axes. For blocks that are not aligned along the X-Y-Z axes (see Figure 4-18 as an example), the application of a suitable matrix translation/rotation, the coordinates used for the parts vertices can easily be aligned for calculation purposes. Hence, the use of designations of *X-direction, Y-direction,* and *Z-direction,* assumes that any necessary translation/rotation has already taken place.

The general order for assembly has the *X-direction* as the primary direction and the *Y-direction* as the secondary direction. All part-pair connections are sorted in ascending sequence with X-coordinate > Y-coordinate > Z-coordinate. For example, Part A with its lowest vertex (6, 1, 3), Part B with its lowest vertex (5, 2, 1) and Part C with its lowest vertex (6, 0, 9) would be ordered as Part B, Part C, Part A.

The sequencing engine that has been developed as a part of this research is one that has its foundation in consulting with practitioners of the candidate company who have over 40 years of working experience in this field. The heuristic approach used here allows for a safe and efficient assembly of the sub-block/subassemblies in the final location. The safe assembly ensures that there is no congestion on the block, thus allowing sufficient workspace for the welders to work and for subsequent parts to be placed in position.

The focus on the assembly sequencing is that of the plates, since the plates are the base unit for the block and all other parts are attached to a plate. By taking the plate as the basic unit for assembly sequencing, the task then is to identify the order that the plates have to be assembled. It needs to be emphasised that the forgoing assembly sequence is not intended to imply that plates are assembled on the block one at a time, as this research is to identify optimum, sub-blocks/subassemblies sequences. Plates will be combined to optimize the sub-block/subassembly sequence. The identification of the assembly sequence of the plates is necessary in order to build the sub-blocks/subassemblies.

The assembling of plates has to take into account the use of the plate or plates, relative to the base plate, which has already been assembled; otherwise, no other assembly can take place, and the vertical plates are assembled. When a plate is assigned to a place for assembly, it is not alone; other plates are connected to it. The type of connection of these other plates determines their priority. A butt connection takes precedence over a fillet connection. This is because a series of butt connection will produce a longer plate that will possibly be a longitudinal or latitudinal plate, i.e. a long panel which makes a clear demarcation in the block.

The design engineer has standards to follow when designing a block, such as leaving sufficient working space for the welders to work. This research has assumed that all standards have been adhered to, and no checking is performed to see if the design conforms to the standards. It is assumed that, for example, in Figure 4-17 (or Figure 4-19) there is sufficient space between plates 4 and 7, 8 and 9 for a welder to work.

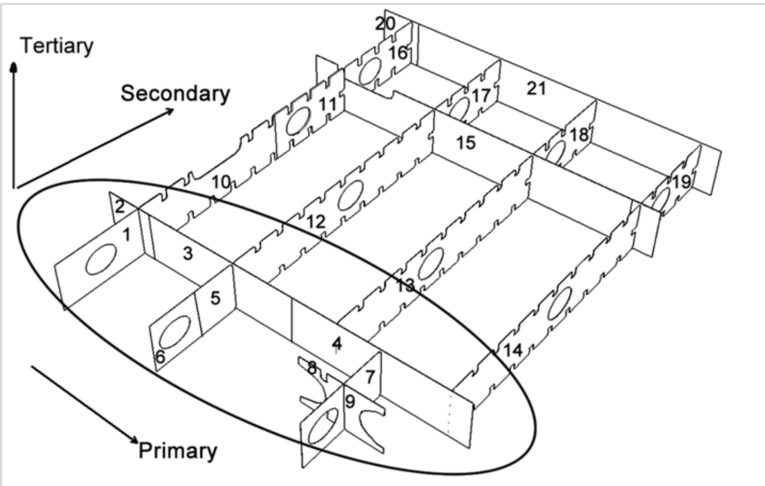


Figure 4-19: Highlighted Section of Figure 4-17

Figure 4-17 (and Figure 4-19) illustrates an example of a few plates that are numbered in their assembly sequence. This sequence is used for identification purposes in the following discussion. In this example, other plates, brackets, flanges and stiffeners are not shown for reasons of clarity. As may be seen, in the highlighted area of Figure 4-19, the first plate assembled, plate 1, has only one other plate attached to it - plate 2, with a fillet connection. Plate 2, however, has 2 plates connected to it, one as a butt joint and one as a fillet joint. In the heuristic developed, a butt connection has precedence over a fillet connection. Connections in the primary direction have precedence over connections in the secondary direction. As butt connections take precedence over fillet connections, plate 3 is the next in the sequence. Plate 3 has one plate with a butt connection, which is plate 4, and this is assembled next. Since there are no other butt connections to plate 4, the procedure returns to plate 2 and check for fillet connections. Plate 2 has two fillet connections, plates 1 and 10, and plate 1 has been dealt with. However, not all fillets have been dealt with for plates 3 and 4, and these plates have a higher priority in the primary/secondary processing sequences than plate 10, as illustrated in Figure 4-17 Plate 3 has two fillet plate connections, plate 5 and plate 12. Plate 5 is assembled next as it has a higher priority processing sequence. Plate 5 has a butt connection to plate 6, which is assembled next.

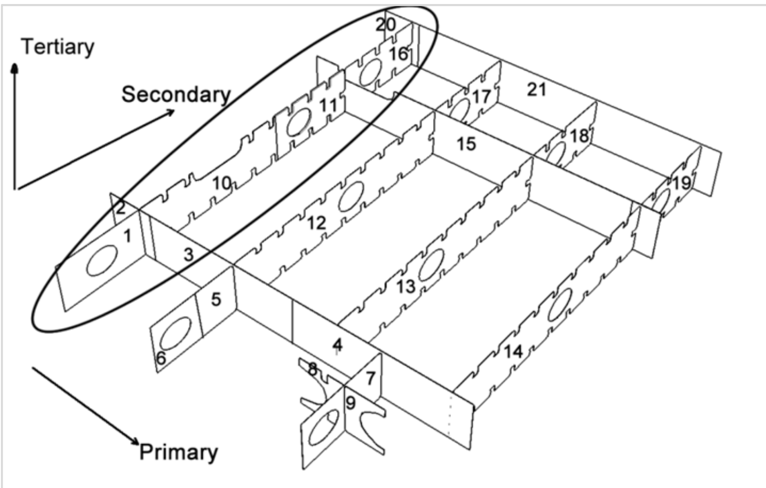


Figure 4-20: Sections Highlighted of Figure 4-17

The next plate with highest priority processing sequence is Plate 4. It is examined next for any fillet connections. Plate 7 has a fillet connection to plate 4 and is then next to be assembled. Plate 7, in itself, has two plates with fillet connections, plates 8 and 9. Plate 8 is assembled before 9 as it has a higher priority processing sequence. At this point, the processing returns to examining plate 2 that has one fillet connection, plate 10 (see Figure 4-20 highlighted section) which is the next plate in the assembly sequence.

Plate 10 has one other plate connected to it as a butt connection, plate 11. As there are no more butt connections to deal with, processing returns to plate 3 which has plate 12 as a fillet connection. Following this, plate 4 is examined for any outstanding fillet connections and these, plates 13 and 14, are next to be assembled.

Plate 11 is now reviewed. It has the highest priority order in the primary direction so it will be reviewed before plates 12, 13 and 14. Plate 11 has a fillet connection to plate 15 so plate 15 is the next plate assembled. For plates 12, 13 and 14, they also have a fillet connection to plate 15, but since plate 15 has already had an assembly assigned to it, no action is required.

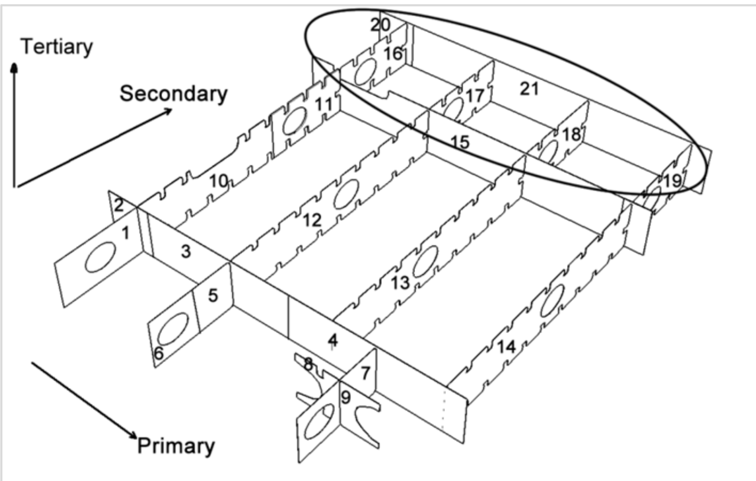


Figure 4-21: Final Assembly Parts Highlighted of Figure 4-17

Plate 15 has four fillet-connected plates (see Figure 4-21). These are assembled in the order of 16, 17, 18 and 19. Since plate 16 has a higher order in the primary direction, it is examined for any fillet connections. Plate 20 has a fillet connection and is assembled next. Plate 20 has a butt connection to plate 21 which makes plate 21 the next plate to be assembled. Since plate 21 has no further butt connections, plates 17, 18, and 19 are examined in turn for fillet connections. Plate 21 is a fillet connection to these plates, but since it has already been assigned an assembly sequence, no action is necessary.

4.7. Assembling Stiffeners on Base/Top Panels

The top and bottom panels are not treated in exactly the same way as the bulkhead plates as described in section 4.6. Whereas the sub-blocks/subassemblies are constructed by a team of workers, the base and top panels are usually constructed in Union-melt section of the shipyard. The Union-melt section is where plates are machine welded to produce a panel. The order of the assembly of the plates is almost too obvious to need defining, since they are all butt connections. However, the sequence is that as defined by the rules that are discussed later in this chapter. Following the Union-melt process, which forms a panel from several plate, the subsequent process is that of the panel line, which is where stiffeners are fitted and welded to the panel.

In the Union-melt section of the shipyard, the welding process is done by a machine which is capable of welding two plates in a single pass, provided the plates are less than or equal to 12.5mm. Thicker plates require multi-passes and have to be welded on both sides of the plate.

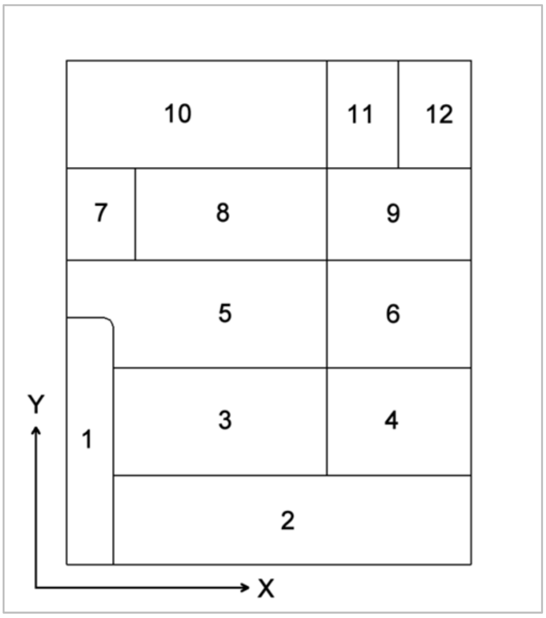


Figure 4-22: Panel/Plate Ordering

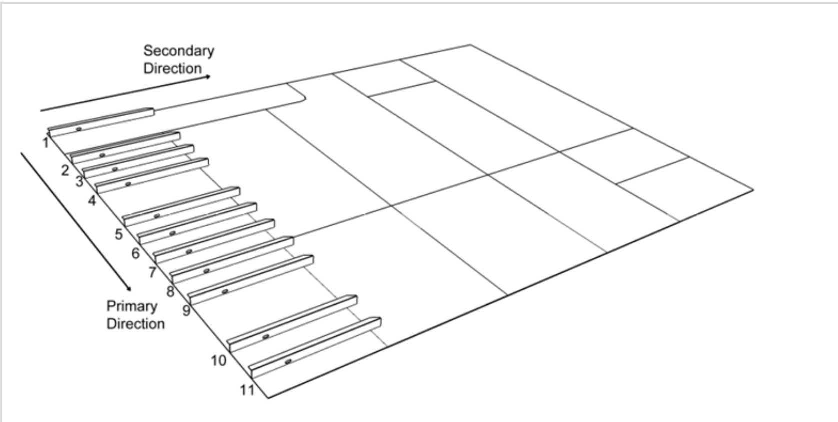


Figure 4-23: Base Panel with 1st Set of Stiffeners

Figure 4-22, illustrates the base panel with assembly sequences numbers shown. These numbers follow the heuristic that has been developed in this research and show the expected assembly sequence. Once the base panel has been completed, the next stage is to weld the stiffeners to it. The stiffeners are usually welded using automated welding machines. The stiffeners are first fitted into position by the process of tack-welding. Once tack-welded into position on the base panel, they are then welded by an automated welding machine. Figure 4-23 illustrates the order in which the stiffeners are fitted into position, prior to welding. The stiffeners follow the primary direction.

Figure 4-24 shows the base panel with all the stiffeners in place. Again, the order of the assembly of stiffeners follows the heuristic developed in section 4.6. Note that stiffeners 1 and 2 are joined with a butt join, as are stiffener pairs (3,4), (5,6), (7,8), (9,10), (11,12), (13,14), (15,16), (18,19), (20,21) and (22,23).

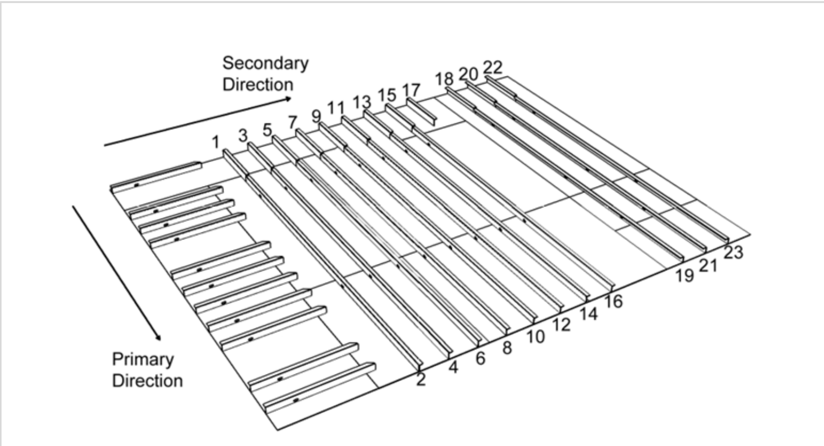


Figure 4-24: Base Panel with All Stiffeners in Place

4.8. Summary

This chapter has discussed how Polychromatic sets provide a foundation for the construction of a knowledge base that feeds into a reasoning engine. The reasoning engine provides the input for the sequencing engine, which produces an assembly sequence for the plates of a block. It is only necessary to find the assembly sequence for each plate of the block since the plate is the fundamental part. By sequencing the plates, the overall sequence is obtained. All other parts are associated with a plate.

Chapter 5. Calculating Welding Length

The calculation of welding lengths of all connections is important for knowing the work content of any subassembly/sub-block or for the whole block for that matter. Knowing the work content and the work rate, a more accurate estimate of the time involved in any welding process may be calculated. Knowing the time it takes to weld allows for a more realistic schedule for the sub-blocks/subassemblies and eventually the whole block.

5.1. General Process for Welding Length Calculation

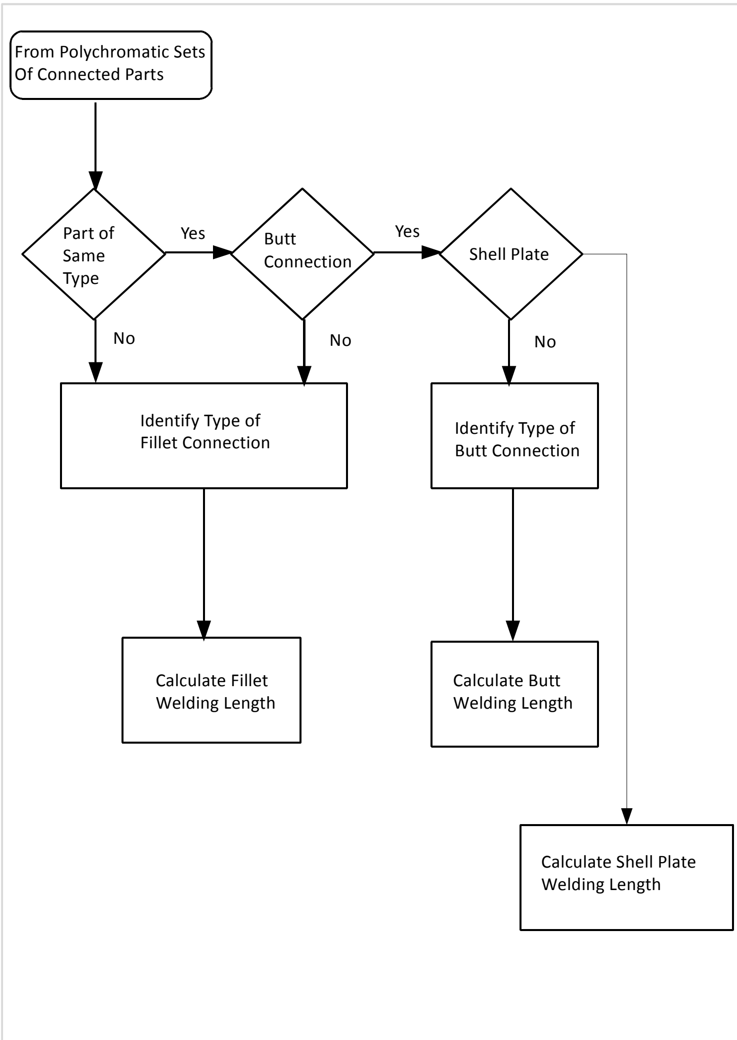


Figure 5-1: Structure for Welding Length Calculations

The welding length is not the linear connection (or contact length) of one panel or part with another. It is the total number of passes required to weld one part to another. A contact length of 1 metre will require a number of welding passes to complete the weld. The welding length is the product of the contact length and the number of welding passes, which is a function of the thickness of the plates being welded together.

The initial process of obtaining the welding length is the calculation of the contact length one part has with another part. Figure 5-1 illustrates the process that is undertaken, depending on the connection type. Butt connections will only occur with parts of the same type, e.g., bracket-to-bracket or flange-to-flange. Fillet connections are possible between the same and different part types. Butt, fillet and shell plate connections have a different processing path. Parts are of different sizes and the length of contact will be dependent on one parts relation to another. This will be discussed in section 5.2. Depending on the part shape, one part may touch another more than once along a fillet edge connection.

$$\text{length} = \sum_{\substack{\text{all contact} \\ \text{point pairs}}} \left(\sqrt{(R_{ix} - R_{i+1x})^2 + (R_{iy} - R_{i+1y})^2 + (R_{iz} - R_{i+1z})^2}\right) \tag{5.1}$$

Where

R_{ix} is the x coordinate of the point R_i. R_{i+1x} is the x coordinate of the point R_{i+1}
R_{iy} is the y coordinate of the point R_i. R_{i+1y} is the y coordinate of the point R_{i+1}
R_{iz} is the z coordinate of the point R_i. R_{i+1z} is the z coordinate of the point R_{i+1}

The contact length of any plate, whether a butt joint or a fillet joint has the same basic calculation (see equation (5.1)). It is possible for one part to have multiple contact points with another part, as edges are not always linear, as shall be explained later. The points R_i and R_{i+1} may be actual vertices or edges of intersection. The methods used to obtain R_i and R_{i+1} depend on the connection type, the plate type and the edge relationship between the plates. This will be discussed as the chapter moves on.

The remainder of this chapter discusses the various methods used to calculate the contact lengths for planar plates and shell plates with butt or fillet connections. The discussion on the connection relationships of parts explains the issues involved in using real-world engineering drawings for calculations. The discussion moves on to the various combinations of butt connections and how the contact lengths, (and subsequently the welding lengths) are calculated. The various types of fillet connects are then discussed with the methods used for calculating the contact lengths. The chapter concludes with a discussion on how to calculate welding lengths when shell plates are involved.

A detailed description of the calculations involved is covered for different plate connections. That is, planar plate-to-plate fillet connections, planar plate-to-plate butt connection, shell plate-to-plate fillet connections and shell plate-to-plate butt connections.

The methods involved for the calculating of the contact length for butt connections hold for all part types of butt connection. Similarly, the methods involved in calculating the contact length for fillet connection are common for all fillet length calculation. When it comes to shell plates, at the small detailed level, it will be seen that the butt and fillet methods of calculation can be equally applied to shell plate butt and fillet connections.

5.2. Part Connections and Relationship

Table 5-1: Connection Combinations

	Plate		Bracket		Stiffener		Flange	
	Butt	Fillet	Butt	Fillet	Butt	Fillet	Butt	Fillet
Plate	*	*						
Bracket		*	*	*		*		*
Stiffener		*			*	*		*
Flange		*					*	*

The connectivity of the parts has already been established by using polychromatic set theory and was covered in chapter 4. Although PST gives the topological relationship between parts, it does not give the lengths of connections. The connectivity in terms of butt or fillet joints of the different parts that compose an offshore rig block is given in Table 5-1

In calculating the contact length of two plates, it is necessary to know the part connections and joint type (i.e. butt or fillet). This has already been explained in chapter 2 of this thesis. Referring to Table 5-1, and taking *Bracket* as an example, the elements of the *Bracket* set are compared with *Plates, Stiffeners* and *Flanges* for the connections as shown.

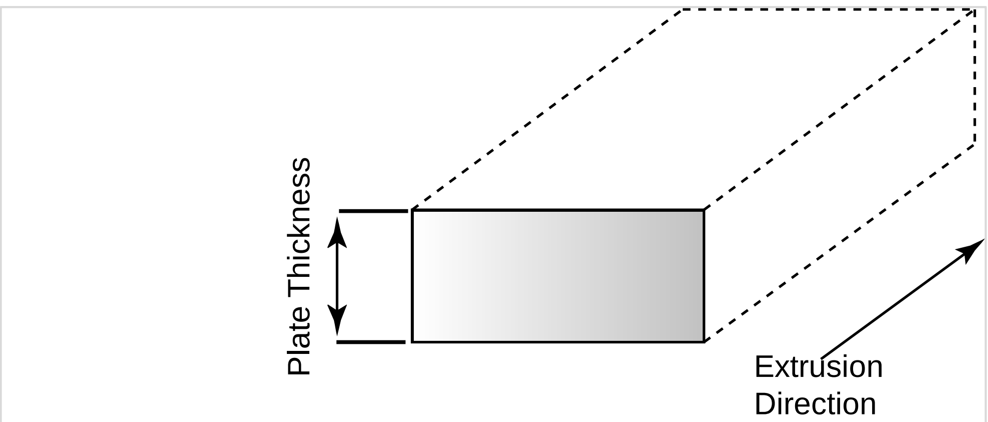


Figure 5-2: Planar Plate Extruded in ‘Long’ Direction

5.3. Connections and Engineering Drawings

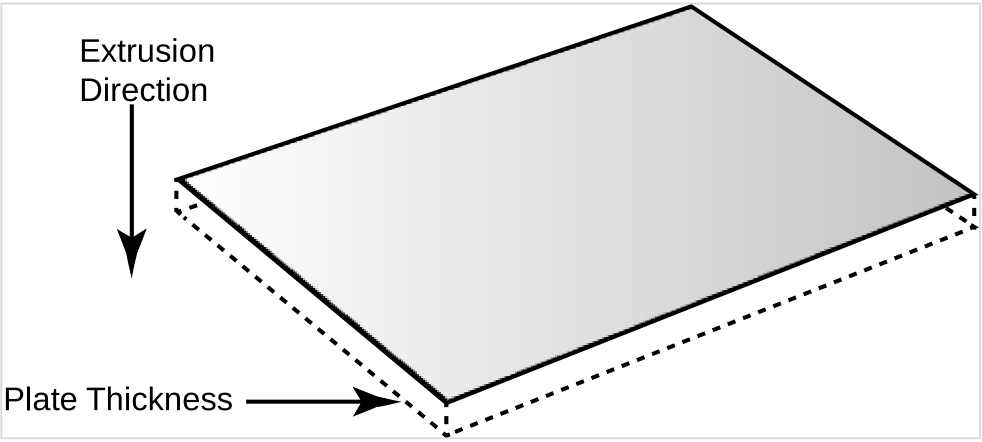


Figure 5-3: Planar Plate Showing Extrusion and Thickness

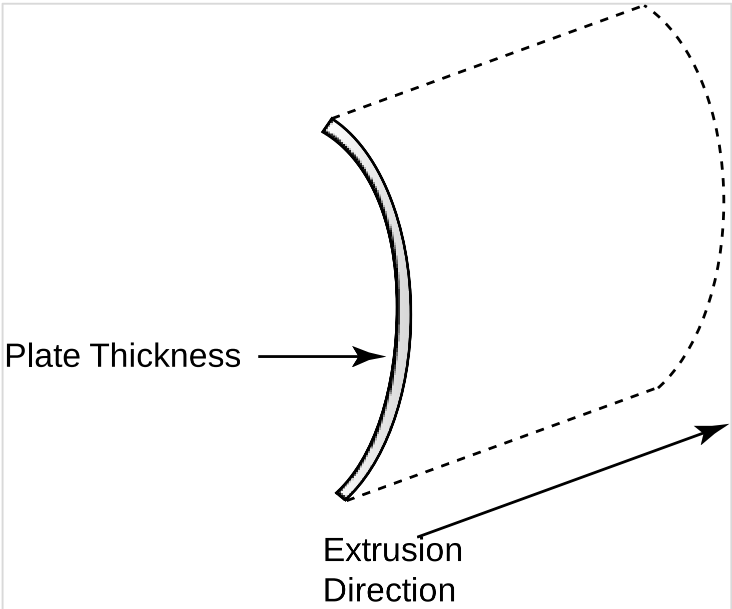


Figure 5-4: Curve Panel Showing Extrusion

The data used in the calculations that follow is derived from NUPAS, a ship design and drawing software. NUPAS holds vertices in floating point form, to 6 decimal places of a millimetre. When an engineer draws a part, such as a planar plate, he will usually draw the surface of the plate and then extrude the plate to the required thickness. NUPAS records the vertices of the plate, together with the thickness and the extrusion vector, as shown in Figure 5-3. Although this may be the normal way, it is not a proscribed way. If the engineer is drawing a long plate with a small cross-section area, he may just draw the cross section of the plate and extrude it in the length direction. NUPAS records the thickness, derived from the cross-section, as shown in Figure 5-2. The extrusion direction is, in this case, the length direction. For shell plates, the engineer will draw a profile and then extrude it along the curve, or the engineer simply imports a pre-triangulated surface from some other software, as shown in Figure 5-4.

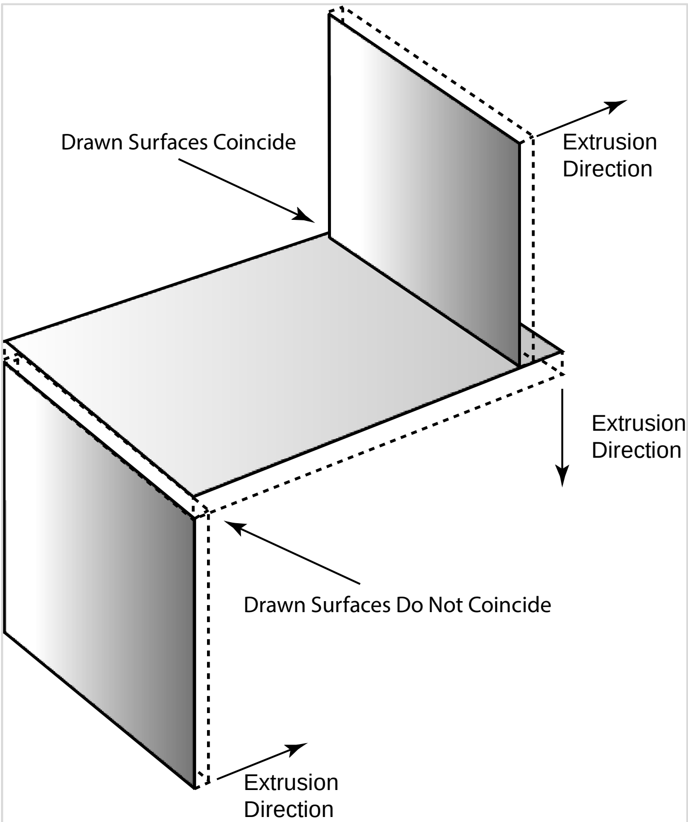


Figure 5-5: Touching Panels Possibilities.

Figure 5-5 illustrates two conditions: the first is where an engineer's drawn surfaces touch each other; the second is where the engineer's drawn surfaces do not touch, but do touch the extruded surface. NUPAS records the drawn surface's vertices and the extrusion direction and amount of extrusion. The bounding boxes of the parts intersect, since the bounding boxes take the part's thickness into account, but the actual given surfaces, as originally drawn do not touch. To detect such cases, it is necessary to perform two checks. The first one is with the vertices of the drawn surfaces; if this does not produce a result of vertices meeting on a surface, a further check is necessary using the extruded surfaces and vertices.

5.4. Calculating Welding Passes

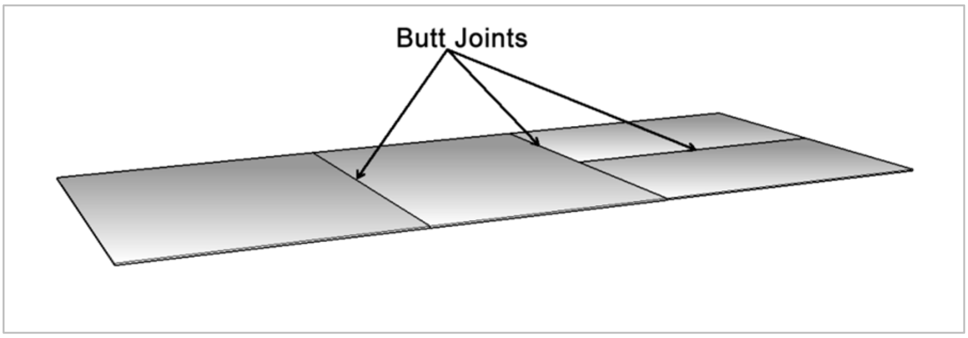


Figure 5-6: Butt Joint Example of 4 Plates

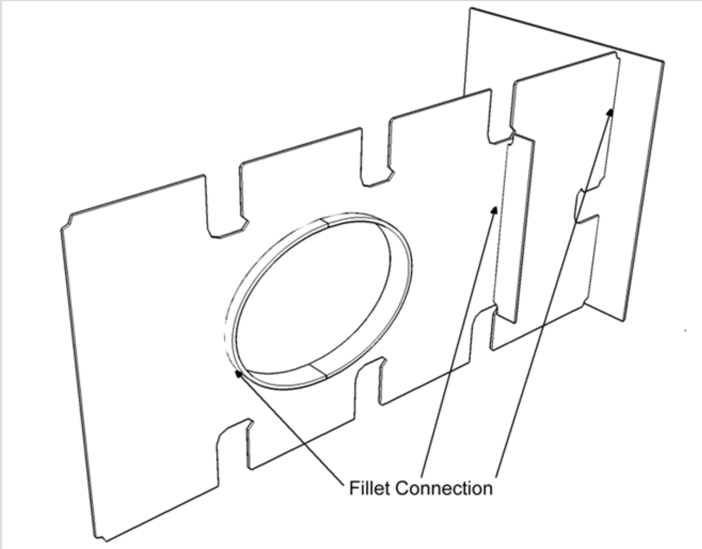


Figure 5-7: Fillet Joint Example

By way of a recap, there are two types of joints, butt joint and fillet joints, these are illustrated in Figure 5-6 and Figure 5-7. Figure 5-6 is an example of butt joints. It shows four plates joined along their edges whereas Figure 5-7 illustrates two examples of fillet joints. The first example is where the edge of one plate touches the surface of another. The second example is where the ring (composed of two semi-circular plates) is within the boundary of a flat plate.

Table 5-2: Welding Passes for Fillet Plates < 19mm

Plate Thickness	Number of Passes
$\leq 12mm$	2
$> 12mm \leq 15mm$	4
> 15 and <19	6

Plates of different thickness require a different number of welding passes. This is because each pass of the welding process deposits only a limited amount of metal and the purpose of the welding process is to fuse the metal parts together. To have a complete fusion of parts, therefore, requires different amounts of welding material that requires a different number of welding passes. Table 5-2 gives the number of welding passes for fillet joints for plates less than 19 mm thick.

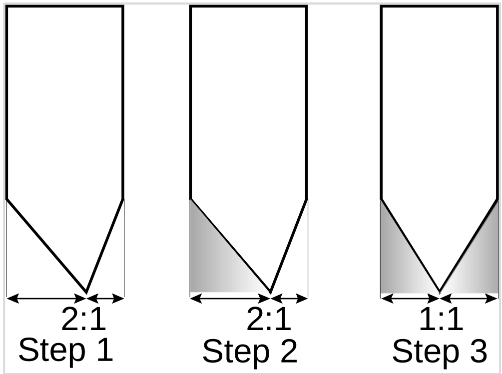


Figure 5-8: Welding For Fillet Joints >=19mm

Fillet joints whose thickness are greater than or equal to 19 mm require a different calculation as the welding process is different. The process is such that one piece of the metal is cut so that the contact point is a 'V' shape. This is not a symmetric 'V' shape. The sides are in the ratio of 2:1. In Figure 5-8, step 1, the left hand side is two-thirds of the width. This is welded, as shown in Figure 5-8, step 2. The right hand side is then gouged[1] out to 50% of the material, following which it is welded, as can be seen in Figure 5-8, step 3.

Table 5-3: Plate Thickness vs Number of Welding Passes

Thickness of Steel	LH Bevel Width	Low Passes	High Passes	RH Bevel Width	Low Passes	High Passes	Total Low Passes	Total High Passes
19	12.5	6	8	6.5	5	7	11	15
25	17	10	12	8.5	6	10	16	22
32	21	12	15	11	10	12	22	27
45	30	20	24	15	16	20	36	44
51	34	28	32	17	24	28	52	60
57	38	38	40	19	28	32	66	72

[1] Gouging is the process of burning or cutting away metal using a carbon rod and an electric arc welder. The normal welding process uses a steel wire. Gouging is a lengthy process, however, the time taken for gouging has not been included as no information as to what joints require gouging is provided in the digital model.

Research was carried out into the process and data samples were collected and analysed on the number of passes required.

Table 5-3 shows the thickness of steel and the initial sizes of the bevels. The data was collected from research at the candidate company. The low passes are the lowest number of welding passes recorded for the welding of the bevel and the high passes are the highest number of welding passes recorded for the bevel.

Table 5-4: Ratio of Welding Passes to Plate Thickness

Thickness of Steel	Low Passes Ratio	High Passes Ratio	Mean Passes Ratio
19	0.58	0.79	0.68
25	0.64	0.88	0.76
32	0.69	0.84	0.77
45	0.80	0.98	0.89
51	1.02	1.18	1.10
57	1.16	1.26	1.21

Table 5-5: Regression Coefficients

Coefficient	A	B	R^2
Low Passes	0.261	0.015	0.9173
Mean Passes	0.397	0.013	0.9080
High Passes	0.534	0.012	0.8880

The next step in the calculation of the ratio of the number of passes to the thickness of the plate is show in Table 5-4. A linear regression plot for this data is shown in Figure 5-9, Figure 5-10 and Figure 5-11. As may be seen, these ratios are almost linear.

Table 5-5 shows the regression coefficients from the linear regression analysis of the data.

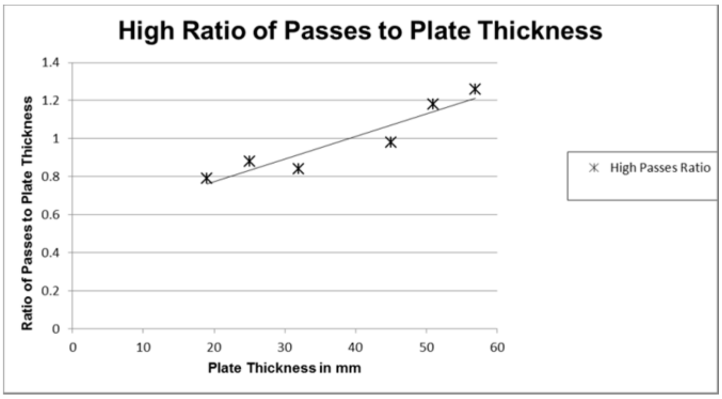


Figure 5-9: Regression Plot for High Ratio of Welding Passes to Plate Thickness

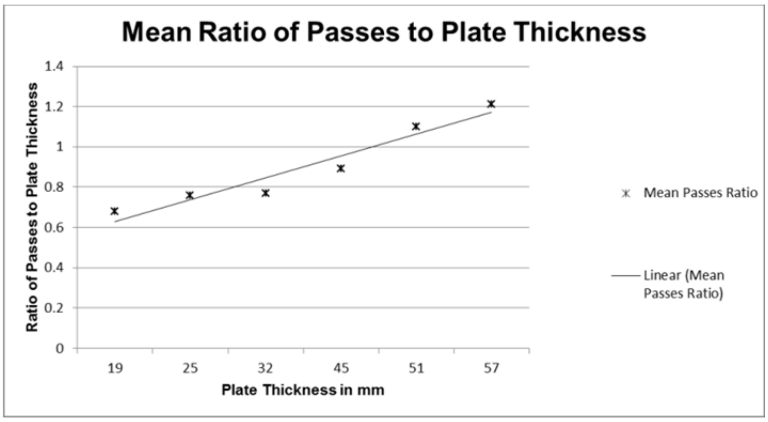


Figure 5-10: Regression Plot for Mean Ratio of Welding Passes to Plate Thickness

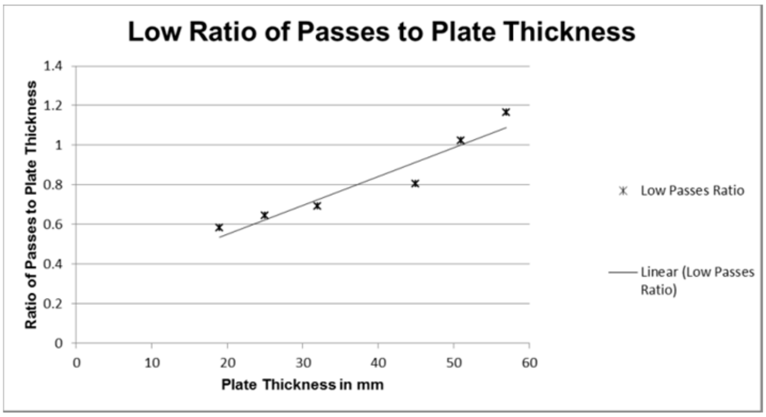


Figure 5-11: Regression Plot for Low Ratio of Welding Passes to Plate Thickness.

The regression line derived from Table 5-5 is of the form $y = A + BT$, where T is the thickness of the material. The function y, is the ratio of the welding passes to the thickness of the material. The linear regression equation can be rewritten as

$$\frac{P}{T} = A + BT \tag{5.2}$$

Where P is the number of passes and T is the thickness of the material.

Making P the subject of the equation gives:-

$$P = T(A + BT) \tag{5.3}$$

Using the values for A and B from Table 5-5 and substituting them in equation (5.3) gives rise to equations (5.4), (5.5) and (5.6)

$$P = T * (0.261 + T * 0.015) \text{ for low passes} \tag{5.4}$$

$$P = T * (0.397 + T * 0.013) \text{ for mean passes} \tag{5.5}$$

$$P = T * (0.534 + T * 0.012) \text{ for high passes} \tag{5.6}$$

All three equations are used to produce a range of welding lengths, giving the worst, best and average case. The values shown in parenthesis are the rounded values that are used in calculations.

Table 5-6: Comparison of Welding Passes for Various Plate Thicknesses

Passes / Thickness→	20mm	25mm	30mm	35mm	40mm	45mm	50mm	55mm	60mm
Low Passes	11.22 (12)	15.9 (16)	21.33 (22)	27.51 (28)	34.44 (35)	42.12 (43)	50.55 (51)	59.73 (60)	69.66 (70)
Medium Passes	13.1 (14)	18 (18)	23.55 (25)	29.75 (30)	36.6 (37)	44.1 (45)	52.25 (53)	61.05 (62)	70.5 (71)
High Passes	15.46 (16)	20.83 (21)	26.79 (27)	33.36 (34)	40.52 (41)	48.29 (49)	56.65 (57)	65.62 (66)	75.18 (76)
Medium % Difference	16.67%	12.5%	13.63%	7.14%	5.71%	4.65%	3.92%	3.33%	1.43%
High % Difference	33.33%	31.25%	22.72%	21.43%	17.14%	16.30%	11.76%	10.00%	8.57%

For butt joints, where the thinner of the two plates is less than 19mm, equation (5.7) applies. Where the thinner of the two plates is greater than or equal to 19mm, equation (5.8) applies.

$$\text{If } T < 19 \text{ mm then } P = \begin{cases} (T/24.5) + 1 \text{ if } (T/24.5) > 1 \\ (T/24.5) + 2 \text{ if } \left(T/24.5\right) \leq 1 \end{cases} \tag{5.7}$$

$$\text{If } T \geq 19 \,\text{mm then } P = (T * 0.41) + 1 \tag{5.8}$$

Where

P = number of welding passes,

T = thinner of the two plates (thickness in mm)

5.5. Statistical Analysis of Linear Regression Data

The amount of data collected to arrive at the above linear regression is limited. This data was obtained from a set of standard butt and fillet joint welding used by the candidate shipyard (see Figure A-13 to Figure A-18). The following tables show the various linear regression statistics output from Excel. The tables, Table 5-7, Table 5-8 and Table 5-15, have their values rounded to five places of decimals.

Table 5-7: Regression Statistics for Low Passes

Multiple R	0.99654
R-Squared	0.99310
Adjusted R-Squared	0.99127
Standard Error	0.02122
Observations	6

Table 5-8: Regression Statistics for Mean Passes

Multiple R	0.98462
R-Squared	0.96947
Adjusted R-Squared	0.96184
Standard Error	0.03740
Observations	6

Table 5-9: Regression Statistics for High Passes

Multiple R	0.99264
R-Squared	0.98533
Adjusted R-Squared	0.98166
Standard Error	0.02841
Observations	6

Table 5-7, Table 5-8 and Table 5-9 show the regression statics output from Microsoft Excel. All three tables give a high correlation coefficient, with the worst case being 0.98462 and the best-case being 0.99654. There is a small difference between the worst case and the best case, but it is only 0.01193, which is not significant. What can be deduced from the multiple R-values, is that there is an excellent regression line fit.

The R-squared values, or the coefficient of determination also show that in the worst case (Table 5-8), 96.95% of the values fit the regression model. In the best case (Table 5-7), 99.31% of the values fit the regression model.

The standard error is an estimate of the standard deviation of the error μ, this is small, with the worst case being 0.03740 and the best case being 0.02122.

Table 5-10: ANOVA for Low Passes Regression Analysis (Part 1)

Source	Degrees of Freedom	Sum of Squares	Mean Square	F-Statistic	Significance F
Regression	1	0.26045	0.26045	575.70725	0.00002
Error	4	0.00181	0.00045		
Total	5	0.26226			

Table 5-11: ANOVA for Low Passes Regression Analysis (Part 2)

	Coeffs.	Std. Error	t Stat	P-value	Lower 95%	Upper 95%
Intercept	0.45772	0.01720	26.60983	0.00001	0.40996	0.50548
X Var 1	0.01053	0.00044	23.99390	0.00002	0.00931	0.11749

Table 5-12: ANOVA for Mean Passes Regression Analysis (Part 1)

Source	Degrees of Freedom	Sum of Squares	Mean Square	F-Statistic	Significance F
Regression	1	0.21675	0.21675	268.63806	0.00008
Error	4	0.00323	0.00081		
Total	5	0.21998			

Table 5-13: ANOVA for Mean Passes Regression Analysis (Part 2)

	Coeffs.	Std. Error	t Stat	P-value	Lower 95%	Upper 95%
Intercept	0.55399	0.02415	22.93883	0.00002	0.48694	0.62105
X Var 1	0.00904	0.00057	16.39018	0.00008	0.007812	0.01100

Table 5-14: ANOVA for High Passes Regression Analysis (Part 1)

Source	Degrees of Freedom	Sum of Squares	Mean Square	F-Statistic	Significance F
Regression	1	0.17763	0.17763	127.02127	0.00035
Error	4	0.00559	0.00140		
Total	5	0.18322			

Table 5-15: ANOVA for High Passes Regression Analysis (Part 2)

	Coeffs.	Std. Error	t Stat	P-value	Lower 95%	Upper 95%
Intercept	0.65512	0.03328	19.68321	0.00001	0.56271	0.74752
X Var 1	0.00833	0.00074	11.27037	0.00002	0.00628	0.01039

The ANOVA from Microsoft Excel gives all the necessary data to test if there is a relationship between the number of welding passes and the thickness of the material. The null hypothesis, H_0:β_1=0. The confidence that the null hypothesis can be rejected is obtained from the P-Values, with the probably that the null hypothesis can be rejected given by *1-PValue.*

From Table 5-11, it can be seen that the null hypothesis can be rejected with a 99.999% level of confidence. From Table 5-13, it can be seen that the null hypothesis can be rejected with a 99.998% confidence. From Table 5-15 it can be seen that the null hypothesis can be reject with a 99.999% level of confidence.

5.6. Plate-to-Plate Butt Contact Length Calculation

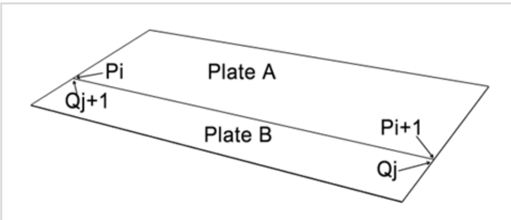


Figure 5-12: Case #1

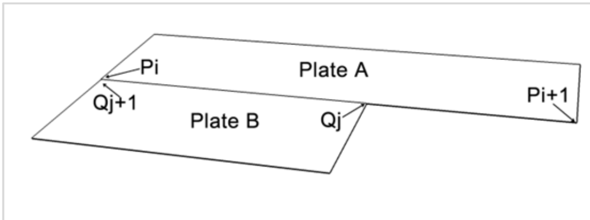


Figure 5-13: Case #2

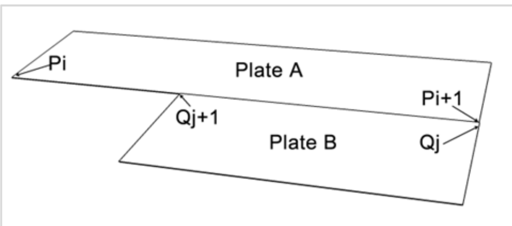


Figure 5-14: Case #3

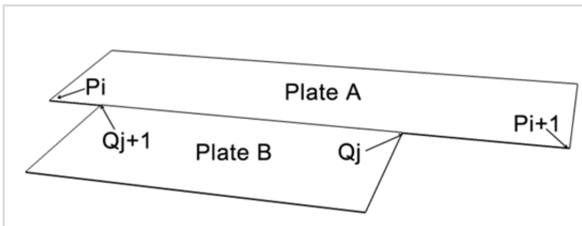


Figure 5-15: Case #4

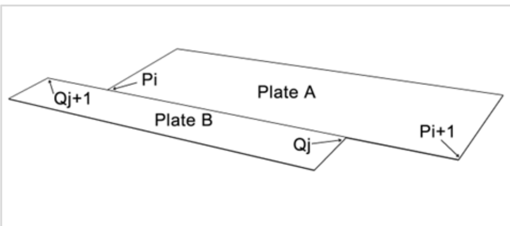


Figure 5-16: Case #5

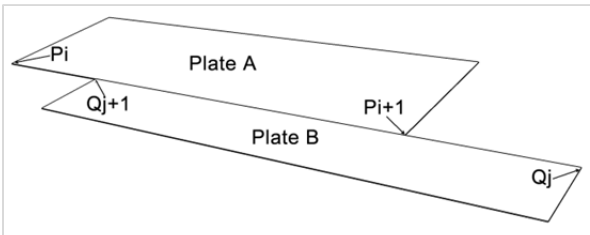


Figure 5-17: Case #6

Plate-to-Plate butt connections are along the edges of the plates. For the edges to be welded together for a butt joint, the edges have to be parallel. To calculate the butt welding length on any plate pair, how the plates touch has to be taken into account. Although one plate may touch another plate, it may not touch it along its entire edge. Figure 5-12 to Figure 5-17 illustrates the possible butt connections between plate A and plate B. One pair of vertices on each plate is represented by dots. Plate A, which for this discussion is referred to as the primary plate, is fully described by vertices $P_1, P_2, ...P_m$, where m is the number of vertices on the plate. Similarly, Plate B, which for this discussion is referred to as the secondary plate, is fully described by vertices $Q_1, Q_2, ...Q_n$, where *n* is the number of vertices on the plate. Vertex *i*, precedes vertex $i+1$, and vertex *m* on plate A precedes vertex 1 on plate A; similarly, vertex *n* on plate B precedes vertex 1 on plate B. The vertices are always treated in a cyclic manner.

The vertices shown on the plates in Figure 5-12 to Figure 5-17 are designated as P or Q. These letters represent the vertices. Each vertex has three direction components, *x, y* and *z*. The standard notation for 3D vertices is used where Px refers to the x component of the vertex *P*. Py refers to the y component of the vertex *P*. Pz refers to the z component of the vertex P. P could be just as easily written as $P(x, y, z)$, but for brevity, it is written simply as p. The writing of the vertex as *P*, implies $P(x, y, z),$. The vertex at P_i has the components Px_i, Py_i, Pz_i.

The explanations of the cases shown in Figure 5-12 to Figure 5-17 for butt edge connections are as follows: -

- Case #1: the vertices of both plates, A and B coincide.
- Case #2, only one vertex on each plate coincides, i.e. Q_{j+1} and P_i.
- Case #3 is similar to case #2, except that the coincidence of vertices is different for each plate i.e. Q_j and P_{i+1}.

- Case #4 - None of the vertices coincide, however, the vertices on plate B, Q_{j+1}, Q_j, are on the line connecting vertices P_i, P_{i+1} on plate A.
- Case #5- Only one vertex on plate B, Q_j, falls on a line connecting vertices P_i, P_{i+1} on plate A. The other vertex on plate B Q_{j+1}, is outside the line connecting vertices P_i, P_{i+1} on plate A.
- Case #6 is similar to case #5, except that the vertex on plate B that is on the line connecting vertices P_i, P_{i+1} on plate A is Q_{j+1}, and vertex that is outside the line joining the vertices P_i, P_{i+1} on plate A is Q_j.

To calculate the contact length, given the 3D coordinates of the vertices, each vertex pair on plate B (Q_j, Q_{j+1}) is compared with each vertex pair on plate A (P_i,P_{i+1}) to see if any vertex pair on plate B meet one of the conditions shown in Figure 5-12 to Figure 5-17. Since the comparisons are cyclic, when $i=n$ then $i+1=1$ and when $j=m$, then $j+1 = 1$. The total contact length is the sum of all the individual contact lengths from the comparisons.

Referring to the various cases in Figure 5-12 to Figure 5-17, Case #1 is where the vertices coincide exactly. The contact length is calculated using the general equation, (5.1), with only one set of point pairs, where $R_{ix} = Q_{jx}$, $R_{i+1x} = Q_{j+1x}$, $R_{iy} = Q_{jy}$, $R_{i+1y} = Q_{j+1y}$, $R_{iz} = Q_{jz}$, $R_{i+1z} = Q_{j+1z}$. Similarly, for Cases #2, #3, and #4, since the vertex pair Q_j, Q_{j+1} lay between the vertex pair P_i, P_{i+1}.

For Case #5, only one vertex, Q_j of the vertex pair Q_j, Q_{j+1}, lies between the vertices P_i and P_{i+1}, but vertex Q_{j+1} is outside the vertex pair P_i, P_{i+1}. The contact length is given by the general equation (5.1), where $R_{ix} = Q_{jx}$, $R_{i+1x} = P_{ix}$, $R_{iy} = Q_{jy}$, $R_{i+1y} = P_{iy}$, $R_{iz} = Q_{jz}$, $R_{i+1z} = P_{iz}$.

For Case #6, the current vertex Q_j does not lay between vertices P_i and P_{i+1}, but Q_{j+1} does. The contact length is given by the general equation in (5.1), where $R_{ix} = P_{i+1x}$, $R_{i+1x} = Q_{j+1x}$, $R_{iy} = P_{i+1y}$, $R_{i+1y} = Q_{j+1y}$, $R_{iz} = P_{i+1z}$, $R_{i+1z} = Q_{j+1z}$.

For vertex pairs to participate in the comparison process, the line joining them must be parallel. Let $\vec{r} = P_i - P_{i+1}$ and $\vec{s} = Q_{j+1} - Q_j$ For the vectors to be parallel, the cross product of the vectors must be zero, i.e. $\vec{r} \times \vec{s} = 0$. While the parallel criterion is a necessary condition, it is not a sufficient condition. Parallel vectors, but in the same direction, would indicate that either the plates are on top of each other (surface-to-surface), or that the plate has another edge running parallel with the one in question. Rectangular plates have opposite edges parallel.

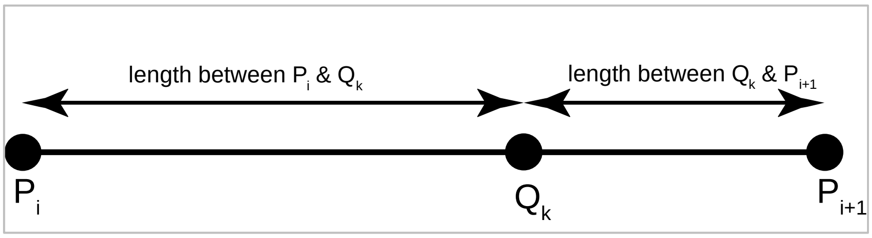


Figure 5-18: Required Vertex is Between Vertices

For a vertex, Q_k, to lie between two other vertices P_i, P_{i+1}, as in Figure 5-18, the sum of the distance P_i to Q_k and Q_k to P_{i+1} must equal the distance between vertices P_i and P_{i+1}.

5.7. Contact Length for Planar Plate-to-Plate Fillet Connections

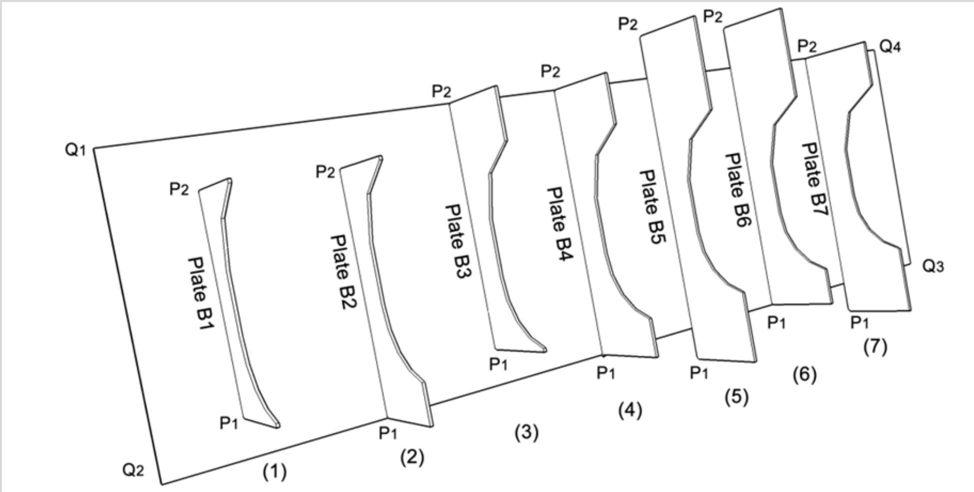


Figure 5-19: Possible Fillet Plate Connections

In calculating contact lengths for planar fillet connections, five different connection conditions have to be taken into account. Figure 5-19 illustrates these conditions. A base plate with vertices Q_1, Q_2, Q_3, Q_4 has seven fillet plates, labelled (1), (2), (3), (4), (5), (6) and (7) forming fillet connections associated with it. The vertices of the fillet plates are labelled P_1 and P_2. There are seven possible fillet connections. These are:-

- Fillet plate (1) has both vertices within the boundary of the base plate.
- Fillet plate (2) has one vertex (P_1) at an edge of the base plate and vertex P_2, within the base plate.
- Fillet plate (3) has one vertex (P_2) at an edge of the base plate and vertex P_1, within the base plate.
- Fillet plate (4) has both P_1 and P_2 at the edge of the base plate.
- Fillet plate (5) both P_1 and P_2 are outside the boundary of the plate.

- Fillet plate (6) has one vertex (*P_2*) at an edge of the base plate and vertex *P_1*, outside the boundary of the base plate.
- Fillet plate (7) has one vertex (*P_1*) at an edge of the base plate and vertex *P_2*, outside the boundary of the base plate.

In examples (1), (2), (3) and (4) the connection length is calculated using the general equation (5.1), where $R_{ix} = P_{ix}$, $R_{i+1x} = P_{i+1x}$, $R_{iy} = P_{iy}$, $R_{i+1y} = P_{i+1y}$, $R_{iz} = P_{iz}$, $R_{i+1z} = P_{i+1z}$. Examples (5), (6) and (7) have to take into account the intersection of the plates B5, B6 and B7 with the edge of the base plate.

The set $P = \{p_1, \cdots, p_i, \cdots p_n\}$, of all parts, is parsed for fillet connections. Although the set identifies a fillet connection, it does not identity the length of the connection. Further, although two plates are identified as having a fillet connection, the set does not indicate which plate edge connects to which plate surface. This is only ascertained from additional computations. The initial check is to see if a vertex of one plate lies on the plane (or surface) of the other plate. This is not a sufficient check, but a necessary check, since the plane is considered as infinite, and a vertex lying on a plane does not mean that the vertex is within the bounds of the plate. Example (5), (6) and (7) are cases in point.

In the following discussion, the plate described by vertices *Q_1, Q_2, Q_3* and *Q_4* is referred to as the 'base plate'.

The normal to the base plate can be derived from vertices *Q_1, Q_2* and *Q_4* as is shown in equations, (5.9), (5.10) and (5.11). For the vector from *Q_1* to *P_1* (see equation (5.12)) to be on the same plane as the base plate its vector dot product with the base plate's normal must be zero.

When it comes to plates, the thickness of the plate must be taken into account.

Figure 5-5 shows fillet plate intersections with the thickness shown. The solid lines are the engineer's lines and the dotted lines are the extrapolated thickness lines. If the base thickness is not taken into account then vertex Y_1 lies on the plate but vertex W_2 does not. Equation (5.13) takes the plate thickness into account.

$$\vec{V}_1 = Q_2 - Q_1 \tag{5.9}$$

$$\vec{V}_2 = Q_4 - Q_1 \tag{5.10}$$

$$\hat{n}p = V_1 \times V_2 \tag{5.11}$$

$$\vec{V}_3 = P_1 - Q_1 \tag{5.12}$$

$$\vec{V}_4 = P_1 - (Q_1 + T * \vec{E}) \tag{5.13}$$

Where T = thickness of the base plate and E = extrusion vector of the plate.

$$d_1 = \hat{n}p \bullet V_3 \tag{5.14}$$

$$d_2 = \hat{n}p \bullet V_4 \tag{5.15}$$

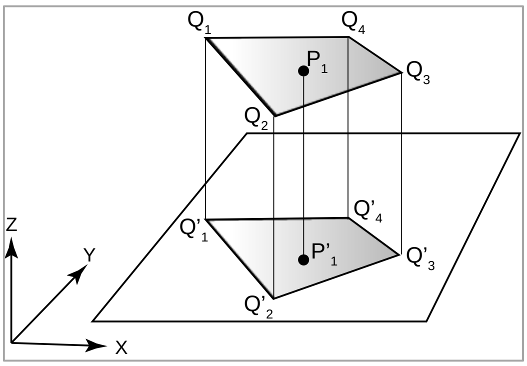


Figure 5-20: Polygon in 3D Space Projected onto a 2D Plane

If $d_1 = 0$ or $d_2 = 0$, then the vertex (P_1) is in the same plane as the surface of the base plate. To verify if a vertex lies within the bounds of a base plate, the base plate is treated as a polygon in 3D space. The vertices of the polygon in 3D space are projected onto a 2D plane, thus eliminating one of the dimensions of the base plate's vertices. The choice of plane of projection is determined by geometry. If the largest bounding box face of the plate is in the X-Y plane, the projection is on to the X-Y plane. If the largest bounding box face is in the Y-Z plane, the projection is on to the Y-Z plane. If the largest bound box face is in the Z-X plane, the projection is on to the Z-X plane. The vertex to be verified is also projected on to the same plane. The vertex, now referred to as a point, (P'_1) is then tested to see if it is within the polygon.

Figure 5-20 shows the vertices projected on to the X-Y plane, by way of illustration. The project of the polygon in 3D space onto a 2D plane in effect eliminates one of the dimensions. In the example given in Figure 5-20, the Z dimension has been removed.

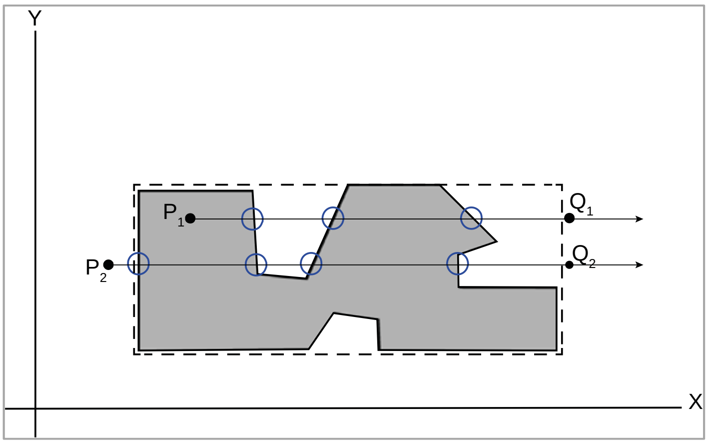


Figure 5-21: Polygon Ray Casting Example

Following the projection of the points of a polygon in 3D space, the projected point may now be checked to see if its lies within the projected 2D polygon. The method adopted here for validation is ray casting using the *Jordan Curve Theorem*, The theorem states that for a point to be inside a polygon, a ray from the point will cut an odd number of the polygon's edges.

The method used here for the testing of the point is based on the one by Heckbert (1994). From the point for testing, a ray is cast in the +X direction with the Y coordinate remaining fixed throughout, only the X coordinate changes. A count is made of the number of edges of the polygon that the ray crosses. An odd value of edges indicates that the point is inside the polygon.

Figure 5-21 illustrates this method, although the figure shows the polygon in the X-Y plane, this is for illustrative purposes only; the method works for the Y-Z and Z-X plane just as well. The bounding box of the polygon is shown as slightly larger than it really is. This is for purposes of clarity, and to make it distinctive. A ray is projected from the point to be validated parallel to the X-axis just beyond the bounding box of the polygon under consideration (these are shown in as Q_1 and Q_2). If the number of polygon sides the ray cuts is odd, the point is inside the polygon; otherwise, the point is outside the polygon. A ray, parallel to the X-axis, is cast from point $P_{1.}$ It cuts three sides of the polygon, which indicates that the point lies within the polygon. For the point P_2, rays cast parallel to the X-axis cut four sides of the polygon; this indicates the point is outside the polygon.

Although ray casting is able to identify points within a polygon, it is not able to identify points that lie on an edge. A point laying on an edge of a plate has already been dealt with above in the explanation to Figure 5-19.

Consider Figure 5-19, which shows the five possible cases of fillet connections in relation to a plate. For purposes of this description, vertices Q_1, Q_2, Q_3 and Q_4 are the vertices of a plate, Case 1, if where smaller plate is wholly within the boundary of the base plate.

Considering Figure 5-19, the contact length in examples (1), (2) and (3) are resolved using the method described and illustrated in Figure 5-20 and Figure 5-21. Examples (4), (5), (6) and (7) need additional information to identify the point where the part edges intersect in 3D space. For example (4) the edge between P_1 and P_2 intersects with the edge between Q_2 and Q_3. In the case of example (5), the edge between P_1 and P_2 intersects with the edge between Q_1 and Q_4. In example (6), P_1 is on the edge between Q_2 and Q_3. and the plate (B6) and the edge P_1 and P_2 intersect with the edge between Q_1 and Q_4. In example (7), P_2 is on the edge between Q_1 and Q_4 and the plate (B7) and the edge P_1 and P_2 intersect with the edge between Q_2 and Q_3.

The vector for the vertices of the fillet plate is given in equation (5.16) and the vector for the edge of the base plate is given in equation (5.17); the normal to these two vectors is given in equation (5.18).

$$\vec{V}_1 = P_{i+1} - P_i \tag{5.16}$$

$$\vec{V}_2 = Q_{j+1} - Q_j \tag{5.17}$$

$$\vec{V}_3 = \vec{V}_1 \times \vec{V}_2 \tag{5.18}$$

To check for the intersection of plate edges, all edges of vertex pairs P_i and P_{i+1} on one plate are compared with all vertex pairs Q_j and Q_{j+1} on the other plate. The vector direction of these vertices is given in equations (5.16) and(5.17). The cross product of these vectors is given by equation (5.18). If the cross product is zero, then the vectors are parallel and therefore cannot intersect, and are ignored.

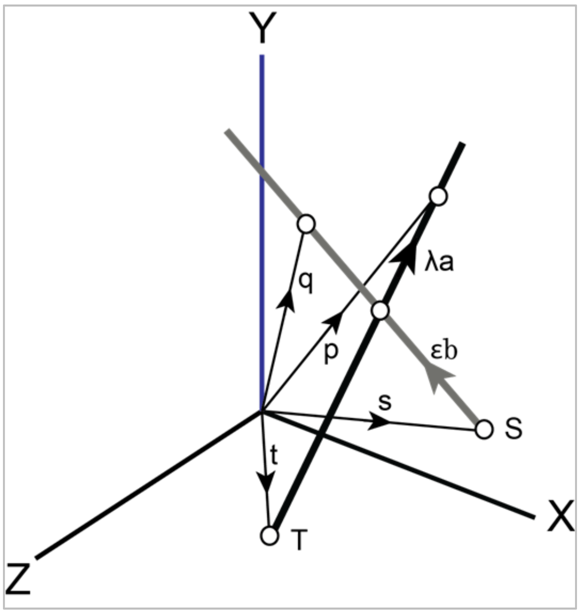


Figure 5-22: Skew Lines Intersecting in 3D Space

Referring to Figure 5-22, the distance between two skewed lines in 3D space (represented by the thickest 2 lines) is given by

$$d = \frac{\|(t-s).(a\times b)\|}{\|a\times b\|} \tag{5.19}$$

if $(t-s).(a\times b)\neq 0$, then the lines do not intersect. The lines only intersect when the shortest distance between them is zero. The lines cross when $p=q \;\; or \;\; t+\lambda a = s+\varepsilon b$.

This leads to:-

$$\lambda x_a - \varepsilon x_b = x_s - x_t \tag{5.20}$$

$$\lambda y_a - \varepsilon y_b = y_s - y_t \tag{5.21}$$

$$\lambda z_a - \varepsilon z_b = z_s - z_t \tag{5.22}$$

Let $$dv = (Q_j - P_j) \bullet (\vec{V}_1 \times \vec{V}_2) \tag{5.23}$$

Then if $dv \neq 0$, the lines do not intersect. From equations (5.20), (5.21) and (5.22) the values for ε and λ are solved. This leads to the point of intersection of the plate edges. The welding length is then readily calculated.

5.8. Contact Length for Planar Plate-to-Plate Discontinuous Fillet Connections

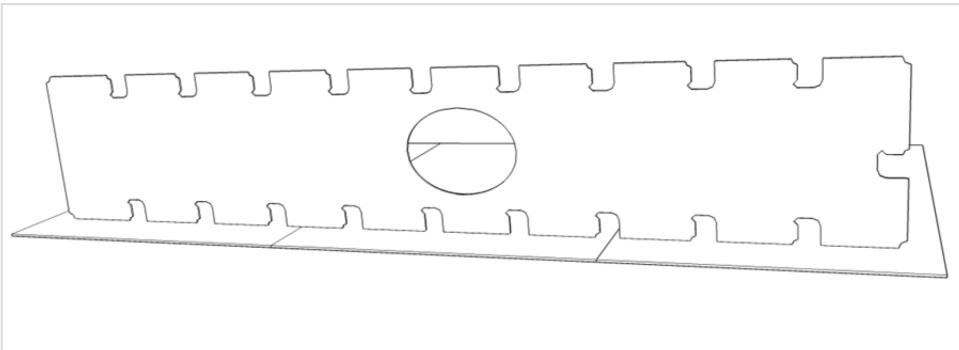


Figure 5-23:Fillet Plate with Interruptions

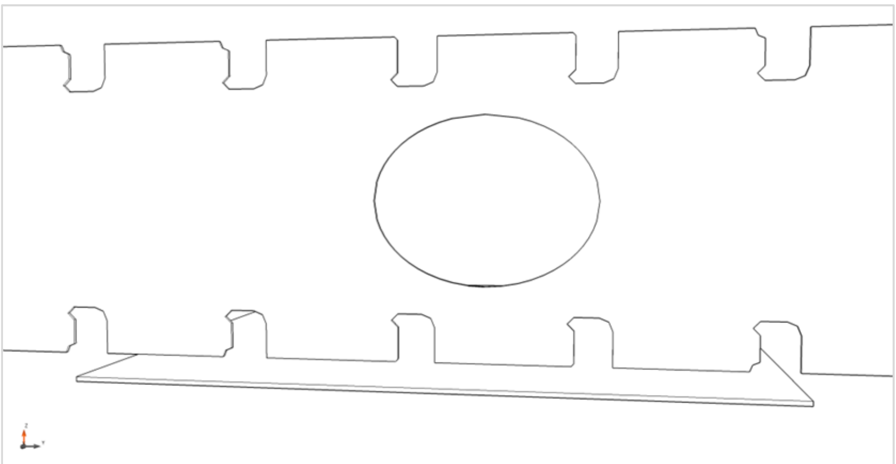


Figure 5-24: Detail of Fillet Plate with Interruptions

Having discussed the calculation methods used in continuous planar plate-to-plate fillet connections in the previous section, these methods can equally apply to plate-to-plate interrupted fillet connections. Figure 5-23 shows a vertical plate that does not have a continuous connection with the horizontal plates with which it makes a fillet connection.

Figure 5-24 shows the close-up detail of one of these plates. The methods of calculation here are not different to those discussed in the previous section. Here an iteration of all vertices pairs P_i, P_{i+1} of the vertical plate is carried out to ascertain if they lay within the horizontal plate. If they do, they contribute to the contact length. This is analogous to example (1) in Figure 5-19. The interrupted fillet connections only take place on plate-to-plate and bracket-to-plate connections.

5.9. Fillet Length of Non-Planar (Shell) Plates

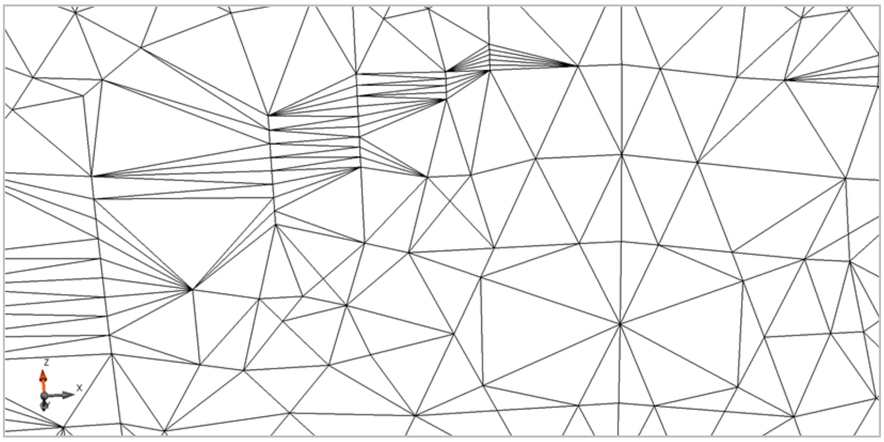


Figure 5-25: Part of a Shell Plate Showing Triangulation.

The preceding sections have dealt with planar plates. Planar plates make up the majority of the plates used in the construction of offshore rigs. Planar plate representation only requires the vertices and thickness of the plate. However, non-planar or shell-plates are represented in data terms as a collection of triangles. This is the standard way curved surfaces are represented in digital form. The process for representing curved surfaces as a series or triangles is referred to as triangulation. The resultant set of triangles cover the whole surface; the triangles vary in size, as may be seen in Figure 5-25.

Let $T\{t_1,...,t_i,...,t_n\}$ be the set of all triangle edges of a shell plate where, $t_1,...,t_i,...,t_n$ are the edges. Let *F(T)* be the colour set where F_1 are edges that are not shared with other triangles and F_2 are edges shared with other triangles. A one (1) in the row/column of the Polychromatic Set cross product, shown in (5.24), is where the condition is satisfied. A zero (0) is where the condition is not satisfied. The resultant set of edges is the boundary of the shell plate.

$$\left[T \times F(T)\right] = \begin{matrix} & \begin{matrix} F_1 & F_2 \end{matrix} & \\ & \begin{pmatrix} 1 & 0 \\ \vdots & \vdots \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{pmatrix} & \begin{matrix} t_1 \\ \vdots \\ t_i \\ \vdots \\ t_n \end{matrix} \end{matrix} \tag{5.24}$$

5.10. Shell Butt Plate Connections

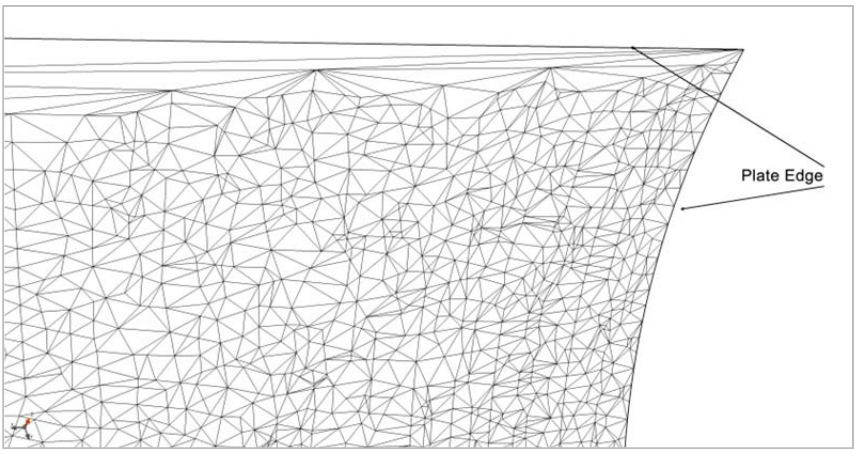


Figure 5-26: Shell Plate Boundary Edges with Triangulation

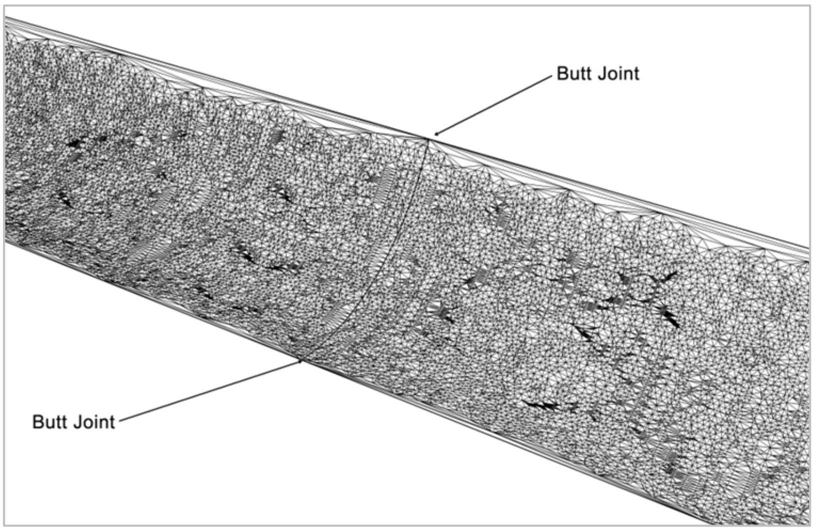


Figure 5-27: Two Shell Plates Butt Joined

The edge of every triangle on the surface of a shell plate is shared by at least one other triangle, except for the boundary edges. Boundary edges only occur in a single triangle, as may be seen in Figure 5-26. This allows identification of the boundary edge of a shell plate. Figure 5-27 shows two shell plates with butt joints.

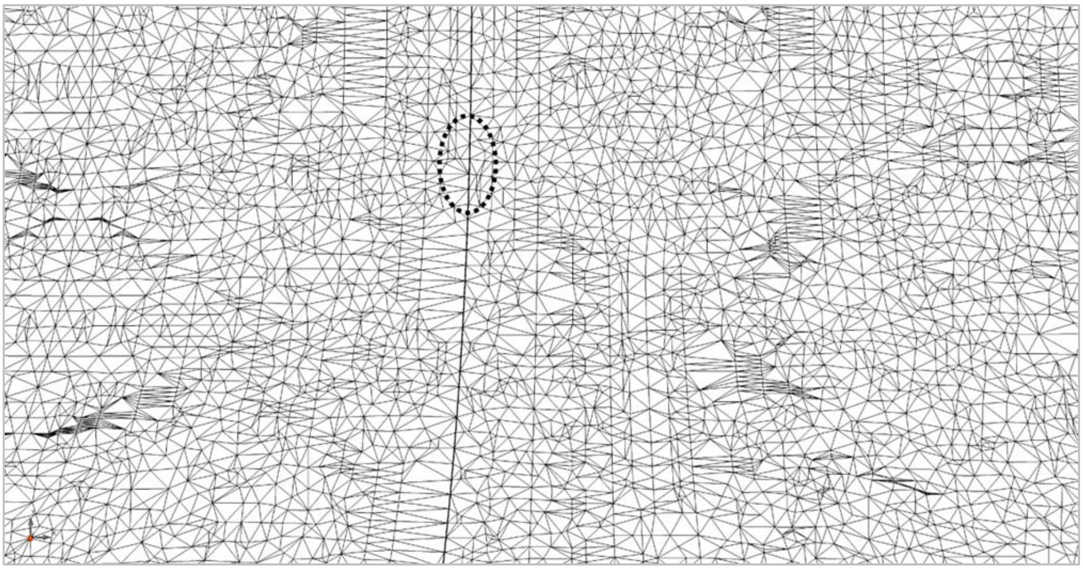


Figure 5-28: Shell-Butt Connections with Area Highlighted

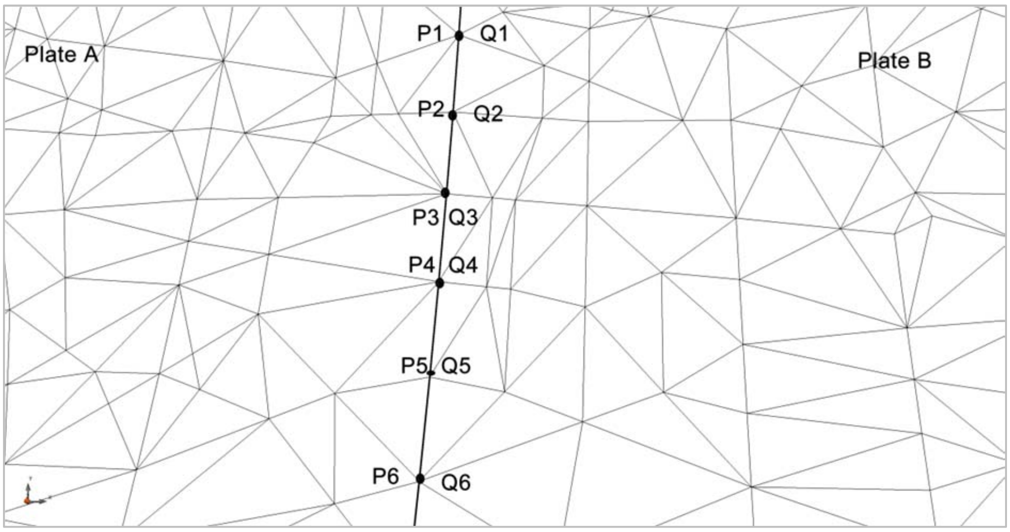


Figure 5-29: Detail of Figure 5-28

The calculation of the welding length for shell plates may be viewed as a micro version of the calculation of the welding length for planar plates. The triangle division of the surface may be treated as a planar plate. The combinations of the relationship of the vertices on one plate with those on another, as shown in Figure 5-12 to Figure 5-17 for planar plates, applies in the analogous way for triangles on the butt edge of shell plates, since each triangle may be considered as a planar plate. The process of triangulation divides the surface of the shell plate into small planar triangles. The triangles on the boundary of the plate are the ones of interest in calculating the butt welding length.

The boundary of the shell plate is formed by the edge of a triangle. This edge has two vertices. The boundary vertices for the plates that have butt connections are compared against each other. In the comparison, vertices of one plate's edge may coincide with that of the vertices of the butting plates edge, as seen in Figure 5-29, (which is the detail of the area of two shell plates with butt joints shown in Figure

5-28) in which case, the length of connection is given by the standard equation (5.1) where $R_{ix} = P_{ix}$, $R_{i+1x} = P_{i+1x}$, $R_{iy} = P_{iy}$, $R_{i+1y} = P_{i+1y}$, $R_{iz} = P_{iz}$, $R_{i+1z} = P_{i+1z}$.

This case is analogous to Case #1 as illustrated in Figure 5-12 resulting in an identical calculation. The other cases in Figure 5-13 to Figure 5-17 have their analogy in the small edges of a shell plate, with identical calculations for contact length.

5.11. Shell Plate Fillet Connections

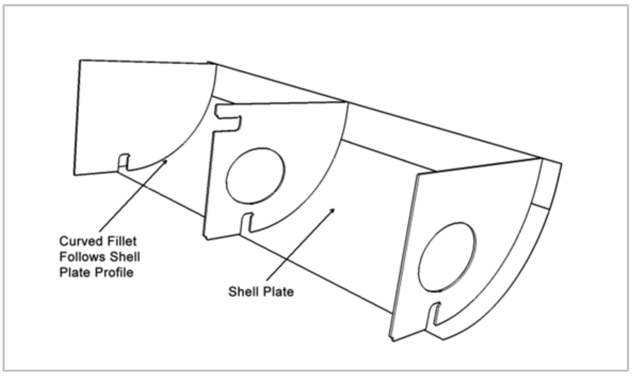


Figure 5-30: Shell Plate with Curved Fillet

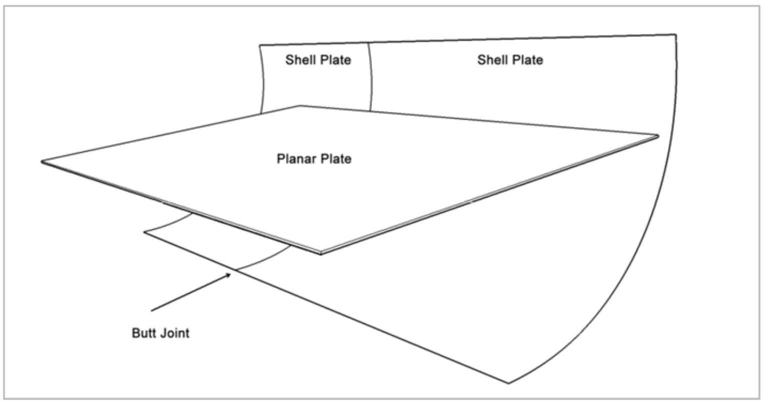


Figure 5-31: Shell Plates with Straight Fillet

Fillet connections to shell plates either follow the curve profile of the shell plate, as illustrated in Figure 5-30, or they run across the curve as illustrated in Figure 5-31.

The different methods used to calculate the contact length, and hence welding length, are discussed in the next sub-sections.

5.11.1. Shell Plate and Curve Planar Panel

The curve of a curved plate, as exported from NUPAS, is a segment of a circle, with no vertices between the start and end points. To overcome this issue, the curve is approximated to a series of very small straight lines drawn as a chord to an arc of one degree. These small lines have well defined vertices and are used in the contact length calculation. The closeness of the approximation of the curve to a series of straight lines is well within the accepted tolerance for contact length calculation.

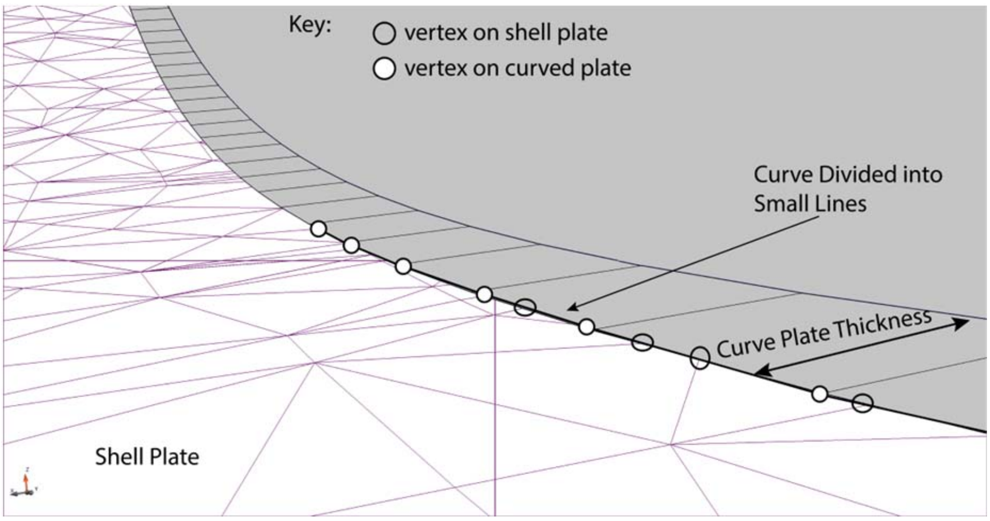


Figure 5-32: Shell Plate with Curve Plate Fillet Connection

The curve of the curve plate, as seen in Figure 5-30 is composed of a series of small straight lines that approximate very closely to the curve, as may be seen in Figure 5-32. Each line is well defined by a starting vertex and a finishing vertex. A line will touch the shell plate with two well-defined vertices. These vertices will touch the shell plate in one or more triangles. Figure 5-33, Figure 5-34 and Figure 5-35 illustrate the possible relationships between a line on the curved plate and the triangles on the shell plate. All three views are schematic and represent a plan view of the connections. The section of the shell plate is represented by the solid lines whose vertices are annotated as V1, V2, V3, V4, V5 and V6. The fillet plate is represented by the dotted line with vertices P_1 and P_2.

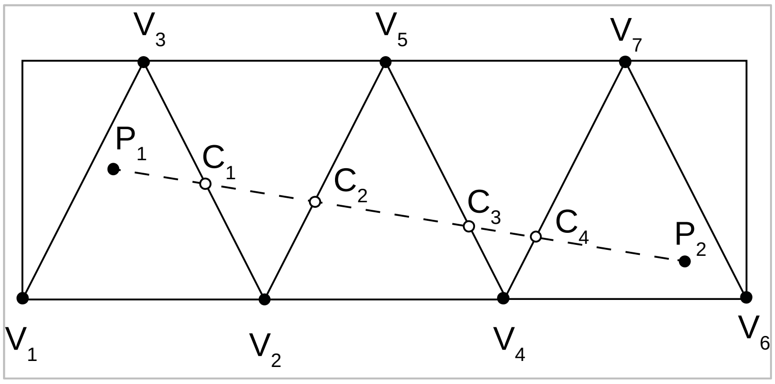


Figure 5-33: Fillet Plate/Shell Connection

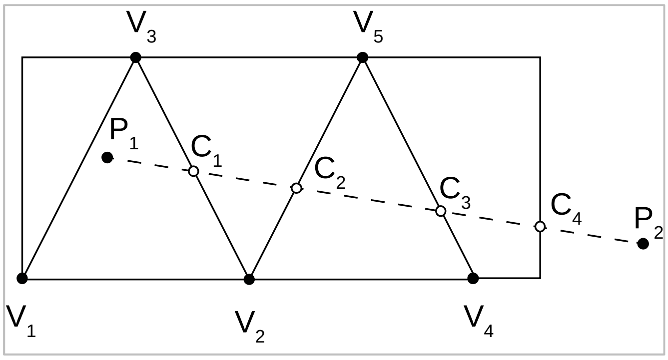


Figure 5-34: Fillet Plate Ends outside Shell Plate.

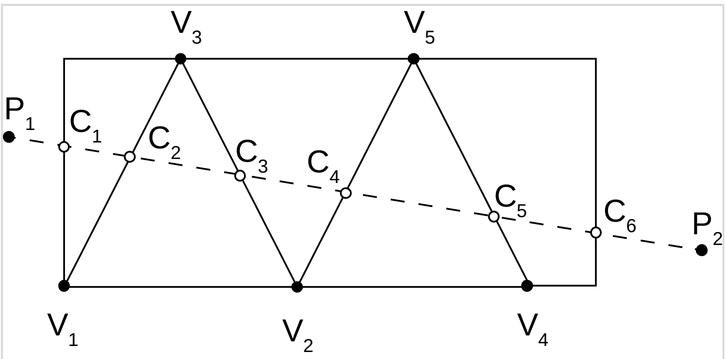


Figure 5-35: Fillet Plate Starts & Ends outside Shell Plate

Figure 5-33, Figure 5-34 and Figure 5-35 are schematic views of some triangles on a shell plate surface and one line on a curve of a fillet plate touching the shell plate, such as one of the fillet plates touching the shell plates Figure 5-30 and Figure 5-32. The small line of the arc is shown by the dotted line whose vertices are P_1, P_2. In Figure 5-34, P_1 is on the surface of the triangle described by V_1, V_2 and V_3. P_2 is on the surface of the triangle described by V_4, V_6 and V_7. In Figure 5-34, P_1 is on the surface of the triangle described by V_1, V_2 and V_3. P_2 is outside the shell plate under consideration. In Figure 5-35, both vertices of the fillet plate, P_1, P_2, start and finish outside the shell plate. Different methods of calculation are employed for each of these three cases.

For the example in Figure 5-33, the length is given by the general equation, (5.1) where the number of contact point pairs is 1 and $R_{ix} = P_{1x}$, $R_{i+1x} = P_{2x}$, $R_{iy} = P_{1y}$, $R_{i+1y} = P_{2y}$, $R_{iz} = P_{1z}$, $R_{i+1z} = P_{2z}$.

For the example in Figure 5-34, the length is given by the general equation, (5.1) where the number of contact point pairs is 2, i.e. P_1 and C_4, and $R_{ix} = P_{ix}$, $R_{i+1x} = C_{4x}$, $R_{iy} = P_{iy}$, $R_{i+1y} = C_{4y}$, $R_{iz} = P_{iz}$, $R_{i+1z} = C_{4z}$.

For the example given in Figure 5-35, the length is given by the general equation (5.1), where the number of contact point pairs are 5, and $R_{ix} = C_{ix}$, $R_{i+1x} = C_{i+1x}$, $R_{iy} = C_{iy}$, $R_{i+1y} = C_{i+1y}$, $R_{iz} = C_{iz}$, $R_{i+1z} = C_{i+1z}$ in general, $i = 1$ to $i = n - 1$, where n is the total number of points of intersection.

Referring to Figure 5-36, the points R,S,T are the vertices of a triangle. Two $v = \overrightarrow{RT}$ vectors $u = \overrightarrow{RS}$ are from vertex R, to vertices S and T respectively. These vertices are in an anticlockwise sequence. The normal to the surface of the triangle is given by $u \times v$. For vertex *P* to lie on the same plane as the shell plate triangle, $(w \times v) \times (v \times u) = 0$ i.e. the cross product of the normals of the two surfaces must be zero. However, in practice, this is not likely to be exactly zero, since computer calculations are carried out using floating-point arithmetic. Hence, if the cross product of the normal of the two surfaces is near to zero, it is taken to be that vertex *P* lies on the surface of the triangle. For near to zero, this research has taken 0.0001, which has proved to take into account floating point arithmetic vagaries.

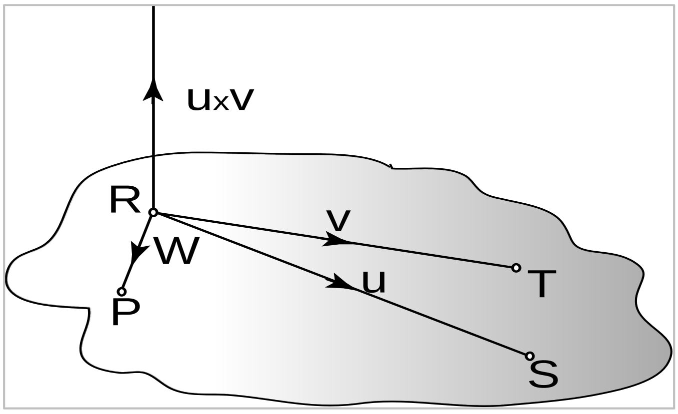


Figure 5-36: Point on a Plane

As mentioned in section 5.2, the extrusion of the surface has to be taken into account. In which case, the points *R, S* and *T* have to be adjusted for this.

$$R' = R + (thickness * extrusion\, vector) \tag{5.25}$$

$$S' = S + (thickness * extrusion\, vector) \tag{5.26}$$

$$T' = T + (thickness * extrusion\, vector) \tag{5.27}$$

And $u' = \overrightarrow{R'S'}$, $v' = \overrightarrow{R'T'}$ and $w' = \overrightarrow{R'P'}$, then for *P’* to be on the surface, $(w' \times v') \times (v' \times u') = 0$.

If it is determined that point *P* lies on the same surface as the triangle, the next step is to determine if the point is within the triangle. The method used for this has been discussed in section 5.7, where the vertex is projected onto a 2D plane (see Figure 5-20) and then ray casting (see Figure 5-21). For examples shown in Figure 5-34 and Figure 5-35 where either one or both vertices are outside the shell plate, the method used to calculate the contact length, is by checking for 3D line intersection of the line between the vertices and the sides of the triangles. This method has been discussed in section 5.7.

5.11.2. Shell Plate and Planar Panel

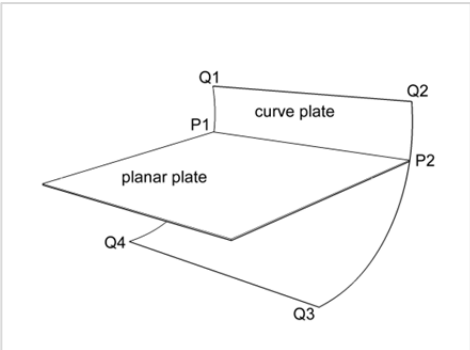


Figure 5-37: Ends of Planar Plate Coincide with Edge of Curve Plate.

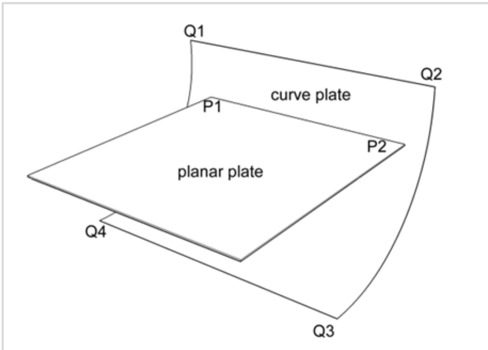


Figure 5-38: Ends of Planar Plate Wholly within Curve Plate

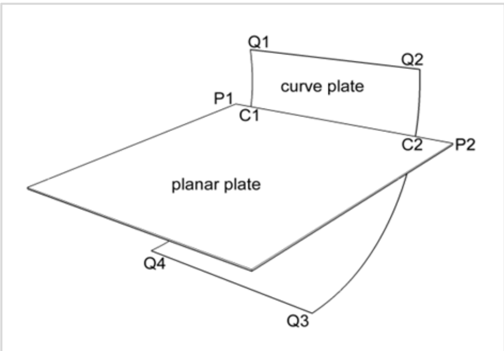


Figure 5-39: Vertices of Planar Plate Outside of Curve Plate

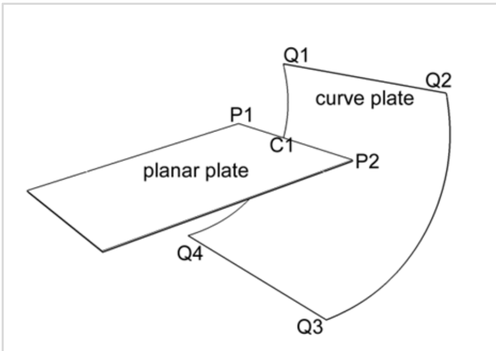


Figure 5-40: Part of Planar Plate Touches Curve Plate (Outside at P1)

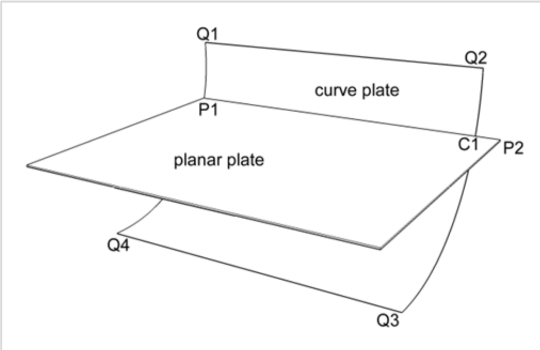


Figure 5-41: Planar Plate Longer than Curve Plate at P2

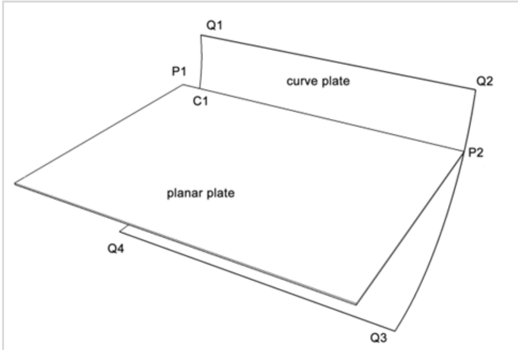


Figure 5-42: Planar Plate Longer than Curve Plate at P1

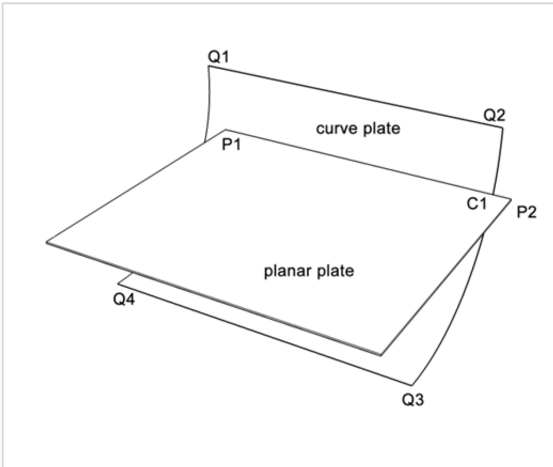


Figure 5-43:Part of Planar Plate Touches Curve Plate (Outside at P2)

There are a number of ways in which a planar plate may connect to a shell plate. In Figure 5-37 to Figure 5-43 Q_1, Q_2, Q3 and Q_4 are the extreme vertices of a shell plate. The curve paths of the shell plate are on path Q_1-Q_4 and Q_2-Q_3. The edge of the planar plate that touches the curve plate is described by the vertices P_1 and P_2. C_1 and C_2 are the points on the edge of the planar plate described by P_1 and P_2 that are cut by the curve path of the shell plate on Q_1-Q_4 and Q_2-Q_3. Note that C_1 and C_2 do not always exist, as may be see in Figure 5-37 to Figure 5-43.

The diagrams shown in Figure 5-37 to Figure 5-43 illustrate the possible ways a planar plate can make a fillet joint with a shell plate. The various methods of location identification and calculation that have been used hitherto may also be applied in these cases.

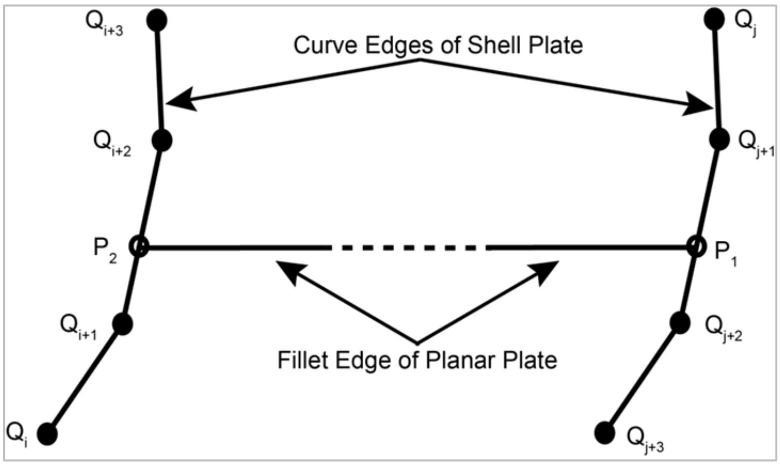


Figure 5-44: Fillet Plate Vertices on Shell Plate Edge

Figure 5-37 has the two vertices of the planar plate P_1 and P_2 coincident with the curve edge Q_1-Q_4 and Q_2-Q_3. Iteration is performed on the set of boundary edges identified in equation (5.24) to see if the vertices of the planar plate fall on the shell plate edge. Figure 5-44 gives a diagrammatic view of a planar plate cutting the boundary paths of a shell plate. The vertices P_1 and P_2 represent the two ends of the planar plate that touch the curve edge of the shell plate. As mentioned, the shell plate is triangulated; the curve edge is composed of a set of small tangents to a curve. These small tangents are straight lines. From Figure 5-44, it can be seen that vertex P_1 lies on the line described by the two vertices Q_{j+1}, Q_{j+2}, and vertex P_2 lies on the line described by the two vertices Q_{i+1}, Q_{i+2}. The method of determining if a vertex lies on a line described by two other vertices has been described above in section 5.6 and Figure 5-18 illustrates a vertex lying between two other vertices.

The case when the planar plate vertices lie with the boundary of the shell plate is illustrated in Figure 5-37. A detail of one vertex on the shell plate is shown in Figure 5-45. A typical triangle, T_i, is illustrated in Figure 5-45 with vertices $VT_{i,1}$, $VT_{i,2}$, $VT_{i,3}$, where $VT_{i,n}$ is the n^{th} vertex of triangle T_i where n=1,2,3. It needs to be noted that the triangles shown in this diagram are schematic only. (For a realistic view of a shell plate's triangles, refer to Figure 5-26, Figure 5-27, Figure 5-28 and Figure 5-29.)

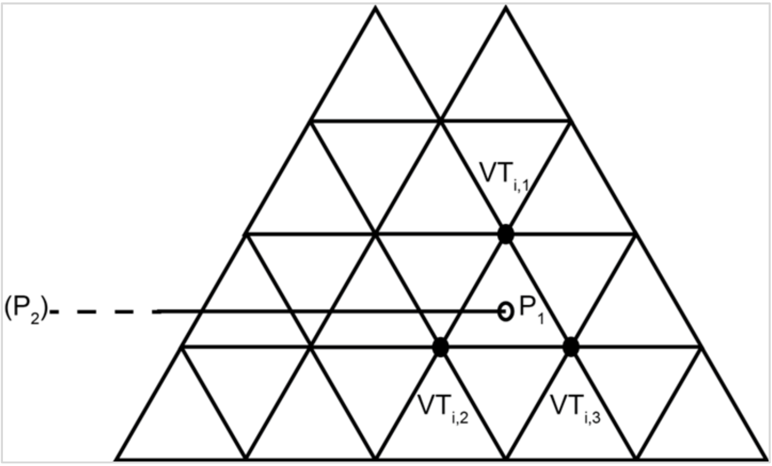


Figure 5-45: Fillet Plate Vertices within Shell Plate

The method used to determine if a point lies on a planar surface has been described in section 5.7 and illustrated in Figure 5-20 and Figure 5-21. To determine if vertex P1, in Figure 5-45, lies in a triangle of the shell plate, all triangles that comprise the set of triangles of the shell plate are iterated, each triangle being treated as a plate in its own right. Each vertex on the planar plate is validated to see if it lies on any part of the shell plate. In the example shown in Figure 5-38, only two vertices of the planar plate lay on the shell plate, in which case the contact length is given by general equation (5.1), where $R_{ix} = P_{1x},\ R_{i+1x} = P_{2x}$, $R_{iy} = P_{1y},\ R_{i+1y} = P_{2y}$, $R_{iz} = P_{1z},\ R_{i+1z} = P_{2z}$ for just on pair of contact points.

In the general case, there are multiple vertices of a planar plate that touch the surface of the shell plate. This is exactly analogous to the work described in section 5.8 where one planar plate has discontinuous contact with another planar plate. In section

5.8, the planar plate has a single surface for comparison; with the shell plate, there are multiple surfaces as it were, each surface being a triangle. The contact length of a planar plate with discontinuous contact to a shell plate is given by the general equation (5.1), where *i, i+1*, are consecutive vertices that touch the shell plate and where $R_{ix} = P_{ix}$, $R_{i+1x} = P_{i+1x}$, $R_{iy} = P_{iy}$, $R_{i+1y} = P_{i+1y}$, $R_{iz} = P_{iz}$, $R_{i+1z} = P_{i+1z}$.

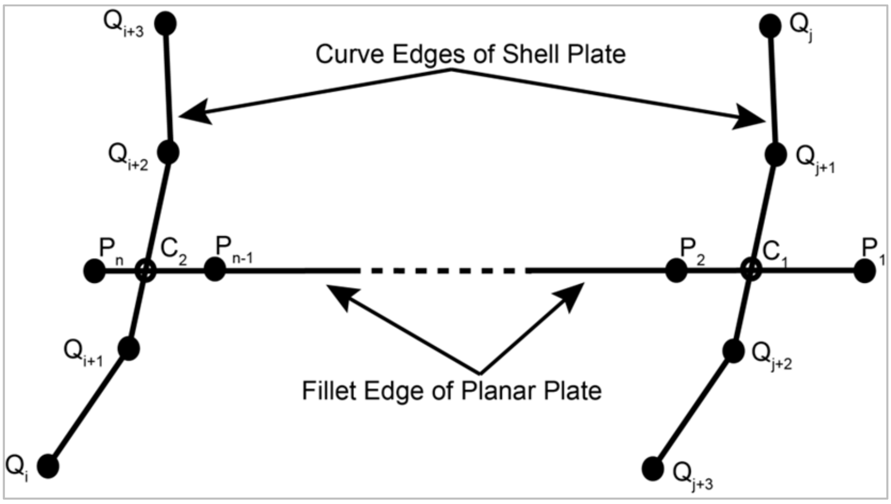


Figure 5-46: Fillet Plate's Vertices Outside the Shell Plate

Figure 5-46, is a schematic of a fillet plate whose vertices, P_1 and P_2 begin and ends outside the boundary of a shell plate. The fillet plate crosses the shell plate's boundary curve at C_1 and C_2. The contact length is given by the general equation (5.1), where, $R_{ix} = C_{2x}$, $R_{i+1x} = C_{1x}$, $R_{iy} = C_{2y}$, $R_{i+1y} = C_{1y}$, $R_{iz} = C_{2z}$, $R_{i+1z} = C_{1z}$ for a single set of point a point pairs.

The method of determining the intersection points, C_1 and C_2 requires the iteration of all vertex pairs in the edge set shown in matrix equation (5.24) to obtain the vector between the vertex points, $\vec{V}_2$, which is given in equation (5.29). The vector between the vertex points of the planar plate, $\vec{V}_1$, is given in equation (5.28). The cross-product is given in equation (5.30). For vectors not to be parallel, the cross product must be other than zero.

$$\vec{V}_1 = P_2 - P_1 \tag{5.28}$$

$$\vec{V}_2 = Q_{k+1} - Q_k \tag{5.29}$$

$$\vec{V}_3 = \vec{V}_1 \times \vec{V}_2 \tag{5.30}$$

Referring to Figure 5-22 and to equation (5.19), the vectors intersect if $(\mathrm{t}-\mathrm{s}).(\mathrm{a}\times b)=0$. The lines cross when $p=q \;\; or \;\; t+\lambda a = s+\varepsilon b$, which leads to equations (5.20), (5.21), (5.22) and (5.23). From equation (5.23) if $dv \neq 0$ the vectors do not intersect. The values for ε and λ are solved using equations (5.20), (5.21) and (5.22). Once solved, the points of intersection, C_1 and C_2, are readily calculated and from this, the contact and then the welding length may be readily calculated.

The example used here was one where the fillet plate had a continuous edge contact with the shell plate. This may not always be the case. The fillet plate may have discontinuous connect with the shell plate. Examples of such plates are illustrated in Figure 5-23 and Figure 5-24. The general equation (5.1) applies here but in distinct parts:

- Length from $C_1 \; to \; P_2$

  Where

  $R_{ix} = C_{1x}$, $R_{i+1x} = P_{2x}$, $R_{iy} = C_{1y}$, $R_{i+1y} = P_{2y}$, $R_{iz} = C_{1z}$, $R_{i+1z} = P_{2z}$

- Length from points $P_2 \; to \; P_{n-1}$

  Where

$R_{ix} = P_{ix}$, $R_{i+1x} = P_{i+1x}$, $R_{iy} = P_{iy}$, $R_{i+1y} = P_{i+1y}$, $R_{iz} = P_{iz}$, $R_{i+1z} = P_{i+1z}$

and i=2, to n-1, for all touching vertex pairs

- Length from P_{n-1} *to* C_2.

Where

$R_{ix} = P_{n-1x}$, $R_{i+1x} = C_{2x}$, $R_{iy} = P_{n-1y}$, $R_{i+1y} = C_{2y}$,

$R_{iz} = P_{n-1z}$, $R_{i+1z} = C_{2z}$

5.12. Other Part Connections

The preceding sections 5.7 to 5.11 have dealt with the various combinations of plate connections. The established methods of iteration of vertices, determining whether a vertex lies on a plane/surface or not, if a vertex lies between two other vertices and determining where edges of plates intersect can be just as easily applied to stiffeners, brackets and flanges.

5.12.1. Bracket Connections

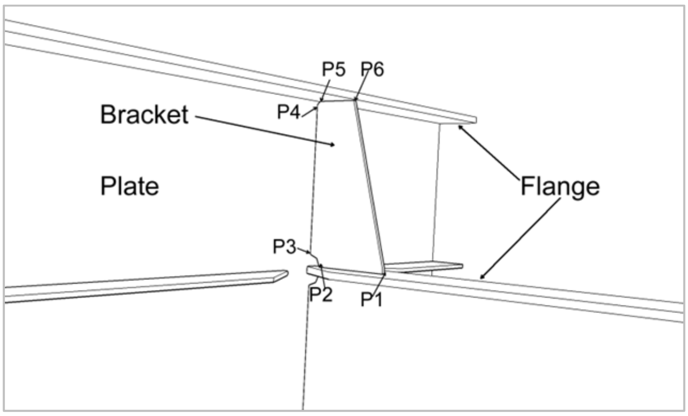


Figure 5-47: Example of a Bracket Touching a Plate and Flanges

Table 5-1 above shows that brackets have possible connections to plates, other brackets, stiffeners and flanges.

Figure 5-47 illustrates the case where a bracket has a fillet connection with a plate and two flanges. The bracket has vertices P_1 and P_2 making a fillet connection with a flange plate and vertices P_5, and P_6 making a fillet connection with another flange plate. Vertices P_3 and P_4 make a fillet connection with a plate.

The discussion in section 5.7 on the measuring of plate-to-plate fillet connections is directly applicable here. All vertex pairs (P_1 & P_2, P_3 & P_4, P_5 & P_6) shown in Figure 5-47 are treated as vertices on a surface, exactly as in Figure 5-19, example (1). The necessary calculations have been discussed in detail in section 5.7, and will not be repeated here. The example in Figure 5-47 shows just two vertices on three edges of the bracket making a fillet connection with another part. If the profile of the bracket is such that it has discontinuous contact with a surface, then the general equation (5.1) applies where $R_{ix} = P_{1x}$, $R_{i+1x} = P_{i+1x}$, $R_{iy} = P_{1y}$, $R_{i+1y} = P_{i+1y}$, $R_{iz} = P_{1z}$, $R_{i+1z} = P_{i+1z}$.

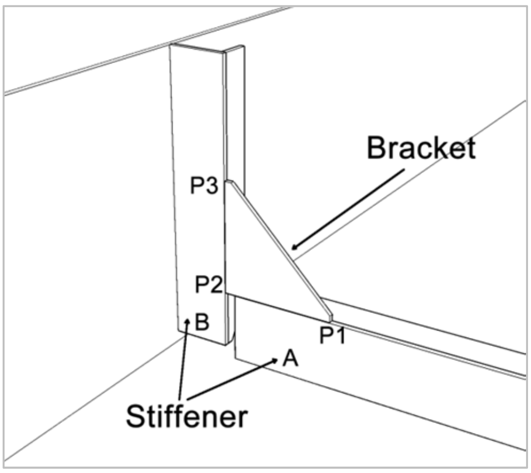


Figure 5-48: Bracket to Stiffener Edges

Figure 5-48 shows a bracket making a fillet edge connection with two stiffeners. The bracket has three vertices of interest, P_1, P_2 and P_3. Vertex pair P_2 & P_3

lay on one stiffener. However, only vertex P_1 lies on the other stiffener, with only part of one edge of the bracket making a fillet connection with the stiffener.

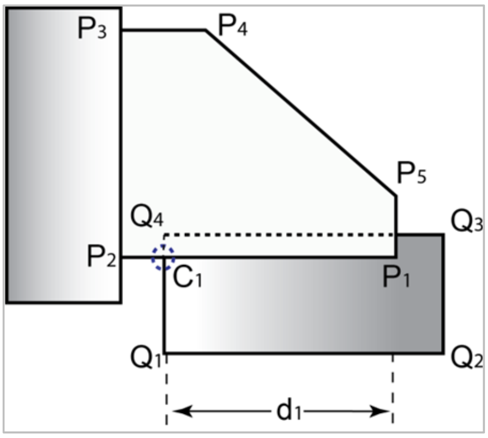


Figure 5-49: Schematic View of Figure 5-48

The vertex pair P_2 & P_3 are treated as vertices on a surface, exactly as in Figure 5-19, example (1). The challenge is the calculation of the actual contact length of the edge defined by vertex pair P_1 & P_2 make with the stiffener A. A schematic view of the bracket-stiffener contact is shown in Figure 5-49, where the location C_1 is at the intersection made by the edge of the bracket defined by vertex pair P_1 & $P_{2,}$ and the end of the stiffener. The contact distance is shown as d_1. The method for calculating the length d_1 which requires finding of the intersection of the edge defined by the vertex pair P_1 & P_2 and the edge defined by the vertex pair Q_4 & Q_1 which is at the point C_1, has been discussed in section 5.7 with example (4) and (5) showing intersection of edges. The discussion relating to Figure 5-22, the intersection of skew lines in 3D space, is directly applicable here.

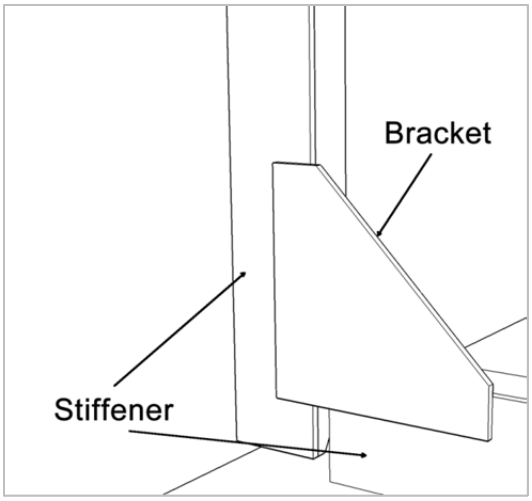


Figure 5-50: Bracket Overlapping Stiffeners

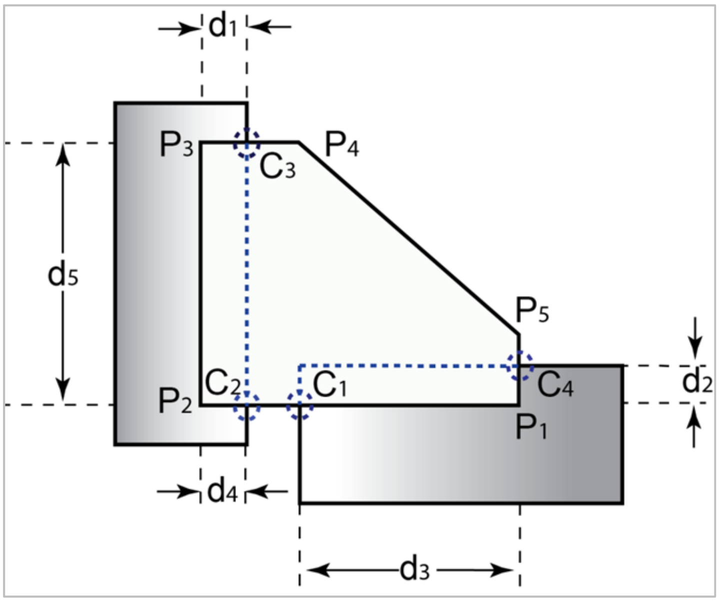


Figure 5-51: Schematic view of Figure 5-48

Brackets may also overlap two stiffeners, as illustrated in Figure 5-50, with a schematic view shown in Figure 5-51. The discussion on how to find the points of intersection of the parts edges has been covered immediately above. The points of intersection of the parts edges are annotated as C_1, C_2, C_3 and C_4 in Figure 5-51.

The distances shown as d_1, d_2, d_3, d_4 and d_5 are the calculated contact lengths. These lengths are considered as fillet lengths.

In the discussion on bracket connections, it has been shown that the methods used for their calculation are not peculiar to brackets, but have already been developed in plate-to-plate fillet connections.

5.12.2. Stiffener Connections

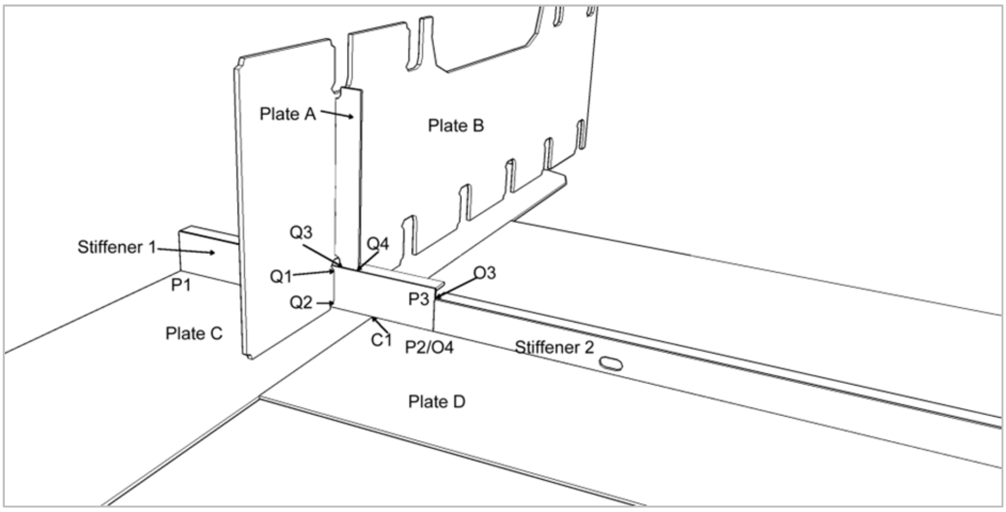


Figure 5-52: Illustration of Stiffener Connections

As may be seen in Table 5-1 stiffeners have connectivity to plates, flanges and other stiffeners. It will be seen that the methods for calculating the connection lengths are the same as have already been discussed in earlier parts of this chapter.

Figure 5-52 illustrates stiffeners connections. Stiffener 1 makes a fillet connection with Plate C and Plate D where vertex P_1 lies on Plate C and vertex P_2 lies on Plate D. The edge defined by vertices P_1 & P_2 intersect Plate C and Plate D at point C_1. It makes a fillet connection as it passes through Plate B, along the edge defined by vertices Q_1 & Q_2. It makes a fillet connection with Plate A along the edge defined by the vertices Q_3 & Q_4.

All contact lengths in the following table use the general equation (5.1). The substitutions for the values, $R_{ix}, R_{i+1x}, R_{iy}, R_{i+1y}, R_{iz}, R_{i+1z}$, are given as well as which vertex on what plate to use.

Table 5-16: Vertex Substitution for General Equation 5.1 for Figure 5-52

Connection	Part#1/ Vertices	Part#2/ Vertices	Vertex Substitutions
Butt	Stiffener 1	Stiffener 2 O_2, O_3	$R_{ix} = O_{3x}$, $R_{i+1x} = O_{4x}$, $R_{iy} = O_{3y}$, $R_{i+1y} = O_{4y}$, $R_{iz} = O_{3z}$, $R_{i+1z} = O_{4z}$
Fillet	Stiffener 1 Q_1, Q_2	Plate B	$R_{ix} = Q_{1x}$, $R_{i+1x} = Q_{2x}$, $R_{iy} = Q_{1y}$, $R_{i+1y} = Q_{2y}$, $R_{iz} = Q_{1z}$, $R_{i+1z} = Q_{2z}$.
Fillet	Stiffener 1	Plate A Q_3, Q_4	$R_{ix} = Q_{3x}$, $R_{i+1x} = Q_{4x}$, $R_{iy} = Q_{3y}$, $R_{i+1y} = Q_{4y}$, $R_{iz} = Q_{3z}$, $R_{i+1z} = Q_{4z}$.
Fillet	Stiffener 1 P_1	Plate C C_1	$R_{ix} = P_{1x}$, $R_{i+1x} = C_{1x}$, $R_{iy} = P_{1y}$, $R_{i+1y} = C_{1y}$, $R_{iz} = P_{1z}$, $R_{i+1z} = C_{1z}$

5.12.3. Flange Connections

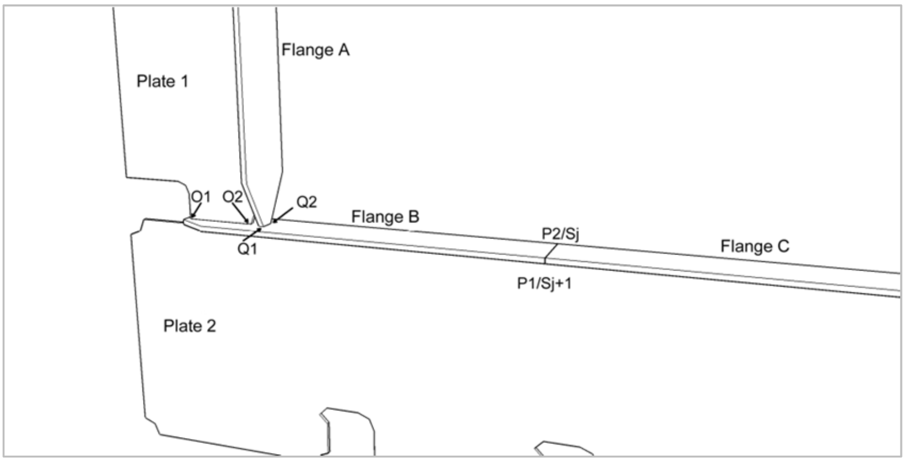


Figure 5-53: Examples of Flange Connections

Flanges connect to plates in a fillet connection and to other flanges in either a fillet or butt connection.

Figure 5-53 illustrates the three types of flange connections. Flange A makes a fillet connection with Flange B along the edge defined by vertices Q_1, Q_2. Flange A, also makes a fillet connection with Plate 1. Flange B makes a butt connection with Flange C along the edge defined by vertices P_1, P_2 on Flange B, or S_j, S_{j+1} on Flange C. Flange B makes a fillet connection with Plate 1 and Plate 2. Flange C makes a fillet connection with Plate 2.

The methods for calculating butt and fillet connection lengths discussed in sections 5.6 and 5.7 apply here.

Flange A's fillet connection with Plate 1 and Plate 2 is directly analogous to a plate-to-plate fillet connection as discussed in section 5.7, as also is Flange B and Flange C's fillet connection with Plate 2 and Flange A's fillet connection with Flange B. The various fillet connections for plates illustrated in Figure 5-19 with its associated discussion covers all the cases illustrated in Figure 5-53.

The butt connection between Flange B and Flange C is calculated in the same way as the butt connections discussed in section 5.6 with reference to the discussion related to Figure 5-12 to Figure 5-17.

5.12.4. Stiffener Connections

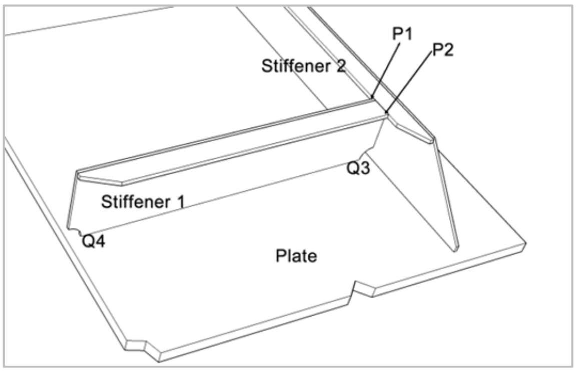


Figure 5-54: Stiffener Fillet Connection. Top View

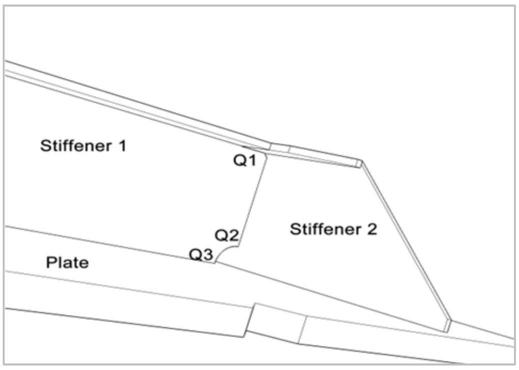


Figure 5-55: Stiffener Fillet Connection. Lower View

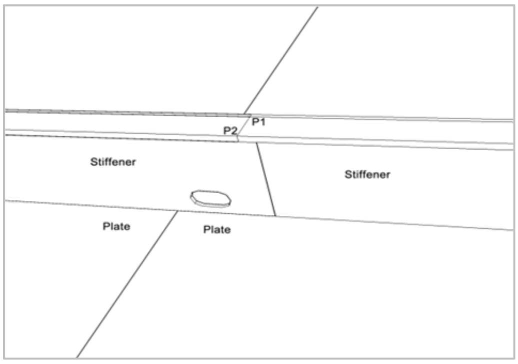


Figure 5-56: Stiffener Butt Connection. Top View

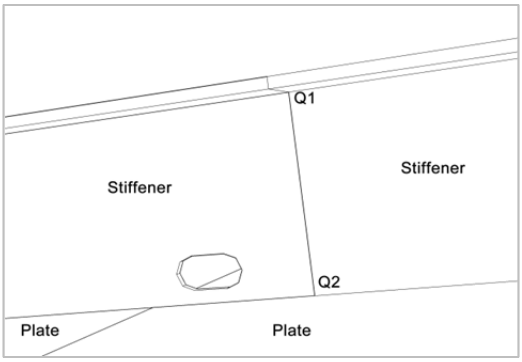


Figure 5-57: Stiffener Butt Connection. Lower View

Stiffeners make fillet connection to plates and other stiffeners. Figure 5-54 shows the top view of a stiffener fillet connection. The vertices P_1 & P_2 form the edge of contact between the stiffeners. Figure 5-55 shows the lower view of the stiffeners shown in Figure 5-54. The vertices Q_1 & Q_2 form the edge of contact between the stiffeners. The vertex Q_3 is on vertex on an edge that makes a fillet connection with the plate. The stiffener-to-stiffener fillet contact length illustrated in Figure 5-53nis given by the general equation (5.1), for a single pair of vertices, where $R_{ix} = P_{1x}$, $R_{i+1x} = P_{2x}$, $R_{iy} = P_{1y}$, $R_{i+1y} = P_{2y}$, $R_{iz} = P_{1z}$, $R_{i+1z} = P_{2z}$.

Similarly the stiffener-to-stiffener fillet contact length illustrated in Figure 5-55 is given by the general equation (5.1), for a single pair of vertices, where $R_{ix} = Q_{1x}$, $R_{i+1x} = Q_{2x}$, $R_{iy} = Q_{1y}$, $R_{i+1y} = Q_{2y}$, $R_{iz} = Q_{1z}$, $R_{i+1z} = Q_{2z}$ The methods discussed in section 5.7 apply here.

Stiffener fillet connections to a plate, as illustrated in Figure 5-54, is exactly analogous to example (1) in Figure 5-19. The methods used for the fillet length calculation of plate B1 with plate A is used here.

5.13. Summary

This chapter has discussed how equations were derived for the calculation of welding lengths for differing plate thickness and connection types. It then discussed the derivation of several methods necessary to calculate contact lengths between different part types and connection types. The calculation of the welding length follows from the calculation of the contact length, plate thickness and joint type.

The sub-blocks/subassemblies sequencing identified in chapter 4, together with their work content is the necessary precursor input for the calculations in the optimizing engine. This is the subject of the next chapter.

Chapter 6. Optimizing Engine: Development and Usage

The sequencing engine produced the on-block assembly sequence; the inference engine identified the plates that make up panels as well as subassemblies. It also calculated the welding lengths involved. The optimizing engine uses the sequences from the sequencing engine. The core of the optimizing engine is an algorithm developed in this research that automatically identifies sub-blocks from the set of subassemblies so that the on-block vertical welding is minimized and thus reducing unnecessary work, which is in line with Lean principles. The resultant work is a combination of sub-blocks and subassemblies that are to be assembled on the block. The order of construction for these sub-blocks/subassemblies to arrive at the assembly point Just-in-Time or as near as JIT as possible requires the use of an optimizing engine.

This chapter describes the development of the optimizing engine used in this research. The use of parallel machines is considered as a way of solving the issue of using a number of teams for the construction of the subassemblies. Scheduling of parallel machines in a machine shop environment has been identified as a *NP*-hard problem. Some (Karger et al, 2010) have suggested that a Greedy algorithm may be used for optimizing the work sequence, although it is often more natural to focus on a Genetic Algorithm as a search heuristic rather than a Greedy algorithm. The Greedy algorithm does has some production and execution advantages, namely, it is fast in execution, it can be optimal in some instances, but not in all, it is good as a first try to obtain a result and it is easy to implement. On the other hand, the Genetic algorithm is able to solve optimization problems that can be described in terms chromosome encoding. However, there is no certainty that the optimized result is a global optimum and the execution times required to produce a result will vary with the random nature of the data.

This research compares and contrasts a number of varieties of the Greedy algorithm with a Genetic algorithm to identify what type of algorithm is best suited.

6.1. Notation Used in This Chapter

The notation used in this chapter is the notation used by Graham et al. (1979) which is now the standard notation. The term 'job' or 'jobs' is used here as it is a standard term. In this research, a job will be the construction of a subassembly or sub-block.

S is the schedule for a set of jobs, *J*.

j is an individual job of the set *J*.

p_j is the amount of time it takes for a machine to process job *j*.

C_j^S is the completion time of job *j* in schedule *S*.

$makespan\, C_{\max}^S = \max_j C_j^S$ is the maximum completion time of any job in schedule *S*.

$\frac{1}{n}\sum_{j=1}^{n} C_j^S$ is the average completion of schedule *S*.

A scheduling problem is denoted by $\alpha | \beta | \gamma$ where $\alpha =$ the machine environment, $\beta =$ various constraints and characteristics, $\gamma =$ either $\sum C_j$ or $C_{\max}$.

$1 || \sum C_j$ = 1 machine | no pre-emption of independent jobs | required to minimize their average completion time.

R_j^S = The time job *j* is required at the final assembly point for on-block assembly[1]

$D_j^{M_i}$ = Construction time for job *j* on machine M_i.[2]

6.2. Introduction

The construction sequencing of the sub-blocks and subassemblies that compose an offshore rig's block is only part of the process of achieving an efficient construction process. The work content of each sub-block or subassembly differs from each other. The question arises is how to schedule the construction work on each of the sub-block/subassemblies so that an optimum Makespan is achieved while, at the same time, preserving the required assembly sequence so that the number of completed parts that go to the stack because they are ready before they are required is minimized?

The use of the Lean principle of continuous supply, in line with the proposal of Liker and Lamb (2002), is one of the aims here although with the varying construction times, it is not that easy to implement. However, some progress towards this has been made in the use of the developed algorithms with the reduction of the double handling of completed-before-required sub-blocks/subassemblies, thus reducing unnecessary work.

The work on constructing the sub-blocks/subassemblies is done by a few teams of works, usually two or three teams. The use of teams to work on the construction of the sub-blocks/subassemblies leads to considering the teams as machines working in parallel. With this in mind, this research has employed the concept of parallel

[1] This is not a part of the standard notation. It is used only in this research and follows the same form as Graham's notation for consistency
[2] This is not a part of the standard notation. It is used only in this research and follows the same form as Graham's notation for consistency

machines in the development of its algorithms. Several researchers (Blazewicz et al. (1983); (Chen et al. 1988; Karger et al. 2010), have pointed out that scheduling with parallel identical machines without pre-emption is an *NP*-hard problem. Mosheiov (2001) has suggested that flow-time minimization on just two parallel identical machines is polynomial solvable. The proposed solution is too problem specific for this research since it incorporates a learning effect. This research assumes that each team is of equal competency. Further, unlike Mosheiov (2001) who considers a learning effect on the production of work, this is not considered here. For there to be a significant learning effect, the work must have a degree of repetition. This is not the case here as each sub-block/subassembly may be considered as a unique construction. The only repetition here is that of welding, and the welders have to be of a sufficient skill level before they are employed.

Two classes of algorithms have been employed here by way of comparison; each algorithm assumes that there is no pre-emption. The first algorithm is a Greedy algorithm, as Karger et al. (2010) has shown that the Greedy *shortest processing time* (SPT) algorithm for the average-completion-time problem $\mathrm{P} \| \sum C_j$ gives an optimal schedule. This shows that a heuristic is capable of producing an optimal solution. Here the claim is not for an optimal solution but that the greedy algorithm is very useful in solving *NP*-hard problems of parallel machine processing as it easily leads to a very acceptable solution. However, the Greedy algorithm does not have the power of a Genetic algorithm. In this work both Greedy and Genetic algorithms have been developed by way of comparison.

6.3. Optimize the Building of Sub-Blocks

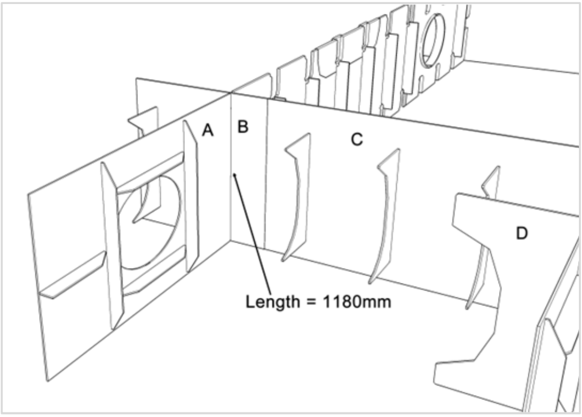


Figure 6-1: Example of a Potential Sub-Block

There are two main reasons for building sub-blocks for assembly rather than just assembling subassemblies directly on the base panel. The first reason is one that is difficult to measure, as it involves the working environment. Every time a subassembly or sub-block is moved from its construction point to the block assembly point a number of steps take place; each of which is time consuming. The part is transported from one place to another by an overhead crane. This requires a lifting supervisor to be present to check for the safe movement of the part. When the part is moved all work in its path has to stop and the workers vacate the path. This has an impact on other work. Anecdotal evidence and some small-scale observation by Abbott et al. (2011) indicated that some workers are interrupted just less than 9% of their working time by part movement. Three subassemblies combined into one sub-block removes two part movements to the assembly point, which is clear reduction in reducing wasted time.

The second reason of building a sub-block is to reduce the on-block vertical welding. Invariably, the on-block welding between subassemblies is vertical welding. This is a manual process. Vertical welding takes twice as long as horizontal welding for the same unit length. If the subassemblies are combined off-block to compose a sub-block, they can be horizontally welded. If the weld is long enough with a clear welding path, a welding machine can be employed which is nominally twice as fast as manual welding. Clearly, there is a reduction in construction time if sub-blocks can be automatically identified and built off-block. This is in line with Lean principles of reducing or eliminating unnecessary work. Figure 6-1 is an example in point, where subassemblies A, B have a common vertical weld. Subassemblies B and C also have a common vertical weld. The length of the vertical weld in both cases is shown as 1180 mm. If all or some of these subassemblies could be combined off-block, that would save on-block welding, and since the welding path is unencumbered, the horizontal welding could be done by automatic machine, which is four times faster than manual vertical welding.

The assembly sequence of the plates, which is equivalent to the subassemblies, was produced by the sequencing engine. This is the starting point for producing an optimized sub-block/subassembly sequence. The selection of the combination of subassemblies to form a sub-block depends upon local physical constraints. The clearance length, width and height of the working environment and the lifting crane capacity are taken into account in the development of the algorithm. The principle used is one of combining subassemblies until the addition of a subassembly violates the local physical constraints. All the subassemblies combined up to this point, but not including the subassembly that violated the limits, become a sub-block. The subassembly that violated the constraint begins a new sequence.

The algorithm for selection can be summarised in the following pseudo code:-

```
Read 1st Entry in Optimized Plate Sequence List
     Move Entry Data to Temporary Sub-Block List
      Do Until End (Read Next Entry in Optimized Plate Sequence List)
          If  (Current Read Entry + Entry in Temporary Sub-Block List)
                   Violate Local Physical Constraints:
              Add Temporary Sub-block List to Final Sub-Block Sequence List
              Move Current Read Entry to Temporary Sub-Block List
          Else
              Add Current Read Entry to Temporary Sub-Block List
          EndIf
     EndDo
```

6.4. The Greedy Algorithms

The focus in this section is on the development of a number of variants of a Greedy algorithm in an attempt to improve on the basic Greedy algorithm. One motivation for this was to find an algorithm that was not computationally heavy. Mosheiov (2001) work on two identical parallel machines that required computational resources of $O(n^4)$. The problem being dealt with in this research is constrained in that there is a predefined assembly sequence, with the view of minimizing the waiting time and number of parts waiting to be installed because they are ready before being required. This is not a shop-flow issue, since all sub-blocks/subassemblies are confined to a single process, in a sense, on any *machine*, prior to being ready for assembly. The Greedy algorithm with many variations is discussed by Karger et al. (2010).

The Greedy algorithm takes the sub-blocks more or less in their required assembly sequence with some variation. Other algorithms used in the parallel machine environment, such as the *Longest Processing Time First* (LPT) (Karger et al. 2010), do not do this; they focus only on the minimization of the *makespan*. With LPT, the completion sequence of the work is immaterial. With this research, the completion sequence of the work is important.

Makespan is defined as $C_{\max}^{S} = \max_j C_j^{S}$ of a schedule *S* to be the maximum completion time of any job in *S*, where C_j^{S} is the completion time of job *j* in schedule *S*. For *m* machines where the processing time for job j is p_j, then $C_{\max}^{*} \geq \sum_{j=1}^{n} \frac{p_j}{m}$ where *n* is the number of jobs, $C_{\max}^{*}$ is the makespan of the optimal schedule and $C_{\max}^{*} \geq p_j$ for all jobs *j*. That is, makespan for the whole assembly process cannot be less than the sum of the average processing time for all the individual jobs.

A Greedy algorithm, when used for scheduling work for identical parallel machines, assigns the next job to be processed to the next free machine. The work content of each sub-block/subassembly is calculation from its set-up time and its welding length. Knowing how long it takes to complete a sub-block/subassembly is necessary to derive the assembly sequence.

To use the Greedy algorithm with the desired output of the work to be in or near to the required assembly sequence, the work is ordered into the required assembly sequence and the work is then allocated to the next free machine.

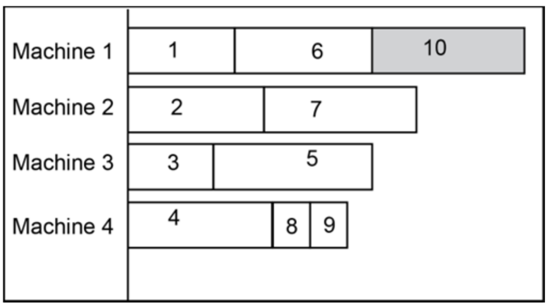


Figure 6-2: Test Assignment of Job 10

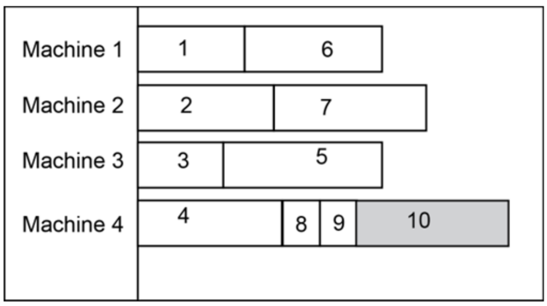


Figure 6-3: Final Assignment of Job 10

Figure 6-2 and Figure 6-3 illustrate the initial and final stage of assigning job 10 to one of four parallel machines. The assignment of any job is identical to the assignment of job 10. Job 10, when it comes to its assignment, is assigned to each machine, 1, 2, 3 and 4 in turn, and at each point of assignment, the completion time C_{10}^{S} of the schedule is calculated. The machine assignment that results in $\min C_{10}^{S}$ is the machine to which the job is assigned which, in the case of the example, is machine 4.

More formally, the j^{th} job is added to the machine with the minimum completion time, i.e. the j^{th} job is added to $\min C^{m}$, where C^{m} is the completion time for machine *m*. Since the j^{th} job is added in this way, the overall schedule is always going to be a minimum for sequential additions, although this is highly unlikely to be a minimum if the order of the assignment of jobs can be changed as with a Genetic algorithm as will be shown later.

6.4.1. Greedy Algorithm with Backtracking

It may be possible to improve the Greedy algorithm by applying a back-tracking technique (Karger et al. 2010). With backtracking, the job is added to the minimum machine schedule, provided that when it is added to the minimum machine schedule, the maximum schedule is not exceeded. If the maximum schedule is exceeded, the job is swapped with the previously added job, provided the overall schedule is now earlier than when the new job was added to the minimum machine schedule.

If *j* jobs have been added to the schedule using the Greedy algorithm, the j^{th} job was added to the machine with a completion time earlier than $C_j - p_j$. In the same way, the $j+1^{th}$ job is added to the machine that has a completion time earlier than $C_{j+1} - p_{j+1}$

When the j^{th} job is added to the schedule, it is added to the machine, m, with the shortest completion time, $\min C^m$. This machine is chosen irrespective of the work content of job j.

The $j+1^{th}$ job is added to the schedule. If $C_{j+1} \le C^*_{\max}$ then the addition of the job to the assigned machine is accepted. If, however, $C_{j+1} > C^*_{\max}$, then back tracking is tested with j^{th} job being replaced by $j+1^{th}$ job in the schedule. Following the replacement of the j^{th} job by the $j+1^{th}$ job, the j^{th} is added to the schedule, which will be a different machine, provided that $C'_j \le C_{j+1}$ where C'_j is the revised completion time for job j. Otherwise, job $j+1$ is added as per the greedy algorithm.

The waiting time is the time between the completion of successive jobs. This may be seen as the time that the downstream process has to wait for its next job. If the j^{th} and $j+1^{th}$ jobs are not swapped, the waiting time is given by

$$\begin{aligned} W_j &= C_{j+1} - C_j \text{ if } C_{j+1} > C_j \\ &= 0 \text{ if } C_{j+1} \le C_j \end{aligned} \tag{6.1}$$

The waiting time, if the j^{th} and $j+1^{th}$ jobs are swapped, is given by

$$\begin{aligned} W'_j &= C_{j+1} - C'_j \text{ if } C_{j+1} > C'_j \\ &= 0 \text{ if } C_{j+1} \le C'_j \end{aligned} \tag{6.2}$$

When swapping j^{th} and $j+1^{th}$ jobs, the condition for the swap is that $C'_j \le C_{j+1}$. The waiting time is zero time units. Without the swap, the waiting time is $C_{j+1} - C_j \ge 0$.

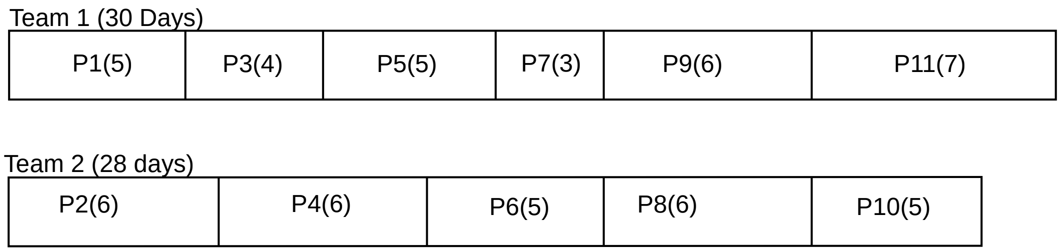


Figure 6-4: Result Using the Greedy Algorithm Only

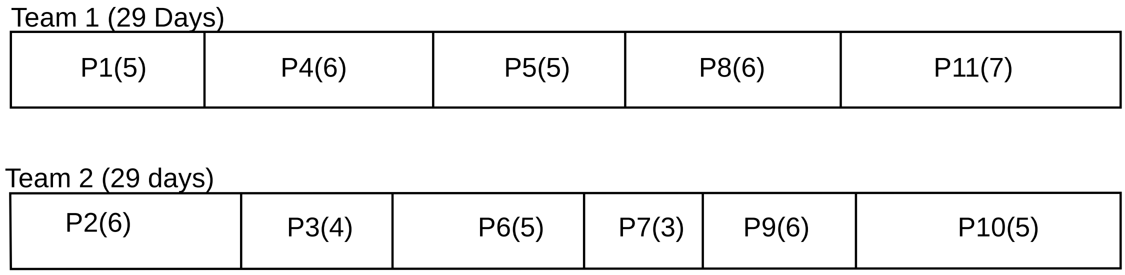


Figure 6-5: Result Using the Greedy Algorithm with Backtracking

Figure 6-4 illustrates the standard Greedy algorithm. Figure 6-5 illustrates the difference between the standard Greedy algorithmic allocation of the work and the Greedy algorithmic allocation of work with backtracking. In Figure 6-5, two teams are working in parallel on 11 sub-blocks labelled P1 to P11. The values in parenthesis, e.g. P1(5), represents the construction duration. In this example, these values can be considered as days. In Figure 6-4, the Greedy algorithm has a *makespan* of 30 days. The total work content of all 11 sub-blocks is 58 days. In Figure 6-5, the Greedy algorithm has a *makespan* of 29 days, a saving of 1 day over the normal Greedy algorithmic schedule.

6.4.2. Greedy Algorithm with Backtracking and Post Reordering

In the previous section, it was shown that by applying backtracking to a standard Greedy algorithm, there is a possibility of improving the *makespan* of the schedule. Improving the *makespan* is clearly a priority in any construction or even production environment. Focusing only on *makespan* only is a too narrow focus for Lean. Sub-blocks/subassemblies completed before they are required will be placed in a stack. It is assumed here that for the teams, there is a single stack. Placing a sub-block/subassembly in a stack can lead to multiple handling, when the sub-block/subassembly at the top of the stack is not the next one required. In such cases, it is necessary to 'dig' for the required sub-block/subassembly. In order to minimize the double handing of parts that have to be placed in a stack, a technique that this research has termed *Post reordering* may be employed.

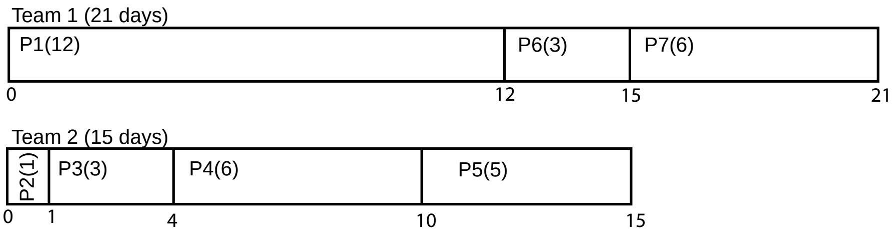


Figure 6-6: Greedy Algorithm with Backtracking

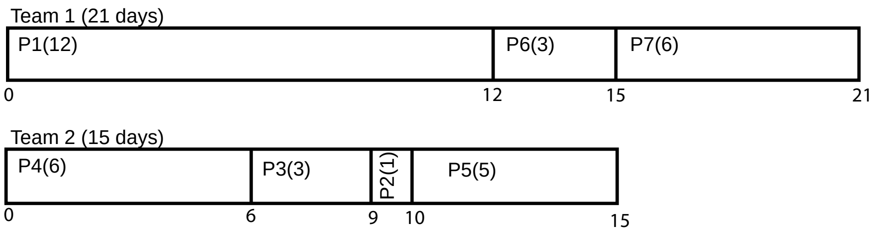


Figure 6-7: Greedy Algorithm with Backtracking and Post Reordering

Post reordering takes place after all the jobs have been allocated to their respective machines by an allocation algorithm. The purpose of post reordering is to reduce the double handling of work that has to be placed in a stack because it is ready before it is required.

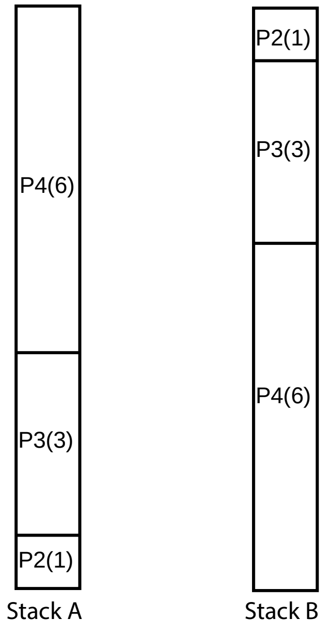


Figure 6-8: Stack Before and After Post Reordering

This process is illustrated with reference to Figure 6-6, Figure 6-7 and Figure 6-8. In Figure 6-6, job P1 finishes at time 12. With job P2 finishing at time 1, however, job P2 is not required until P1 is finished which is time 12. Similarly, for both jobs P3 and P4, they finish before they are required. Jobs that are ready earlier than their required time are stored in a holding area or stack until they are required. Due to the varying construction times of a job and allocation of work to more than one team, it is inevitable that some jobs will be ready before their required time and will have to be stored in a stack. In general the fitting into the final position on a block of a sub-block/subassembly is in terms of minutes (30 to 60 for example[3]), when compared with the construction time of the sub-block which can in terms of multiple hours or days. For scheduling purposes, the fitting time is ignored. Hence, job *j+1* is required at

[3] This is the usual case. Occasionally, there are some blocks that have very large plates that require scaffolding to be erected prior to their assembly.

the assembly point as soon as job *j,* has been fitted into its required position, which, as fitting time is being ignored, is as soon a job *j* has finished construction.

As illustrated, jobs P2, P3 and P4 in Figure 6-6 are completed before job P1. Jobs P2, P3 and P4 will be placed in a stack in the order in which they are completed as in Figure 6-8 Stack A. The bottom of the stack will be job P2 and the top of the stack will be job P4. When P1 is completed, P3 and P4 will have to be moved to gain access to P2. This requires multiple handing of the jobs, which is inefficient.

To eliminate this multiple handling in line with Lean principles, the sequence of construction for jobs P2, P3 and P4 can be reordered without any impact on the *makespan*, as may be seen in Figure 6-7 and Figure 6-8 Stack B. The reordering ensures that the stack has job P4 at the bottom and P2 at the top, so when job P1 is completed, jobs P2, P3 and P4 may be taken off the stack without any digging, as the required job will be on the top of the stack, thus completely eliminating multiple handing. Although, as mentioned, this does not impact upon the *makespan*, it does improve the general efficiency of the working of the shipyard by reducing double handling.

The reordering algorithm may be summarised as follows:-

For job *j,* if $R_j^S > C_j^S$, then the job is ready before it is required. i.e., the required time is after the completion time.

If $R_j^S = C_j^S$, the job is on time.

If $R_j^S < C_j^S$, the job is late. i.e., the required time is earlier than the completion time.

$$\begin{array}{l} \text{Begin:} \\ \quad \text{For } j = 1 \text{ to } n \\ \quad\quad \text{If } R_{j+1}^{S} \leq C_{j}^{S} \\ \quad\quad\quad \text{set } R_{j+1}^{S} = C_{j}^{S} \\ \quad \text{Endfor} \\ \text{End} \end{array} \tag{6.3}$$

A job's completion time is calculated from its position in the construction team's work sequence. This is illustrated in Figure 6-6 and Figure 6-7. Beginning at the start of a team's work sequence, the sum of all preceding job's construction duration plus the current job's construction duration is the current job's completion time. More formally, the completion of job j on machine M_i is given in equation (6.4).

$$C_j^{M_i} = \sum_{k=1}^{k=j} D_k^{M_i} \tag{6.4}$$

In order to ascertain the required time for job, the jobs are parsed in their required assembly sequence. The algorithm (6.3) is then applied to each job. Algorithm (6.3) calculates a new required time based on the preceding job's completion time. The required time for job $j+1$ is amended to the completion time of job j, if its required time is not greater than the completion of job j. In the algorithm (6.3), n is the number of jobs and j is an individual job in the range 1 to n.

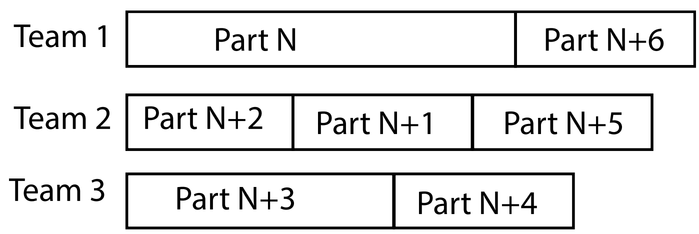


Figure 6-9: Backtracking Issue

Reordering has benefits in reducing the double handling of parts that are placed in a stack but it is not a panacea to the problem. The reordering approach adopted here is a good approximation and in a majority of cases does remove the need for double handling. However, under some circumstances the algorithm breaks down. Figure 6-9 illustrates the difficulty where algorithm has correctly reordered part N+1 and part N+2, since both parts are ready before part N. However, Part N+3, which is required after N+2, and completes after the reordered N+2, actually completes after N+2 and before N+1 and will be placed on the stack between N+2 and N+1. The solution to this issue, when using a Greedy algorithm only, may be found in simulation, which is beyond the scope of this current research, but is something to be considered for future work. It needs to be noted, however, that Genetic Algorithms have been successfully employed to resolve this problem.

6.4.3. Testing the Algorithms

To test a proof concept only for the algorithms, four computer models were developed using the Microsoft C# .NET platform. The 30 sub-blocks were randomly assigned construction times in the range 1 to 8 days, with the initial randomizing seed being set to 42. The total work content for the 30 blocks was 99 days. The 4 models were (a) a Greedy only algorithm, (b) a Greedy algorithm with post reordering, (c) a Greedy algorithm with backtracking only, and (d) a Greedy algorithm with backtracking and post reordering. It is more usual for the number of teams at a shipyard to be between two and four. The models developed are for two, three and four teams of construction workers.

The column headings in Table 6-1, Table 6-2, Table 6-3 and Table 6-4 have the following meaning:-

Column Heading	Meaning
Number of Teams	In the range 2 to 4
No. to Stack	Total number of sub-blocks that were ready before they were required and placed in the stack. Sub-block *i* is placed in the stack if $C_{i\text{-}1} > C_i$, where C_i is the completion time of sub-block *i*.
Max in Stack	The maximum number of sub-blocks in the stack at any one time
Total Stack Time	The time a sub-block is in the stack is given by $C_{i\text{-}1} - C_i$. The total stack time for all sub-blocks is given $\sum_{i=1}^{n}(C_{i\text{-}1} - C_i)$ for all $C_{i\text{-}1} > C_i$, where C_i is the completion time of sub-block i, n= the number of sub-blocks.
Avg. Stack Time	Average Stack Time = $\frac{1}{n}\sum_{i=1}^{n}(C_{i\text{-}1} - C_i)$ where n is the total number of sub-blocks placed in the stack.
JIT Items	Sub-block *i* is a JIT sub-block denoted as J_i Total of all JIT items is given by: - $J_s = \sum_{i=1}^{n} J_i$ where $J_i = 1$ if $C_i \leq C_{i-1}$, otherwise $J_i = 0$ where C_i is the completion time for sub-block/subassembly *i*.
% JIT Items	$\frac{J_s}{n} \times 100$ .Where J_s is the sum of all JIT sub-blocks.
Total Waiting Time	The waiting time for sub-block *i* is given by $W_i = C_i - C_{i-1}$ where $C_i > C_{i-1}$, otherwise $W_i = 0$.
Avg. Waiting Time	$\frac{1}{n}\sum_{i=1}^{n} W_i$ Where
Sequence Changes	Number of sub-blocks whose original sequence was changed
Optimum Makespan	$\frac{1}{T}\sum_{i=1}^{n} CT_i$ where CT_i is the construction time required for part *i*. *n*= number of part. *T* = number of construction teams.
Makespan	As defined in paragraph 6.4

Table 6-1: Greedy Algorithm Only

No. of Teams	No. in Stack	Max in Stack	Total Stack Time	Avg. Stack Time	JIT Items	% JIT Items	Total Waiting Time	Avg. Waiting Time	Sequence Changes	Optimum Makespan	Makespan
2	7	2	12	1.71	11	36.67	45	1.5	0	49.5	50
3	13	4	28	2.15	16	53.53	30	1.0	0	33	35
4	15	6	37	2.47	21	70.00	22	0.73	0	24.75	27

Table 6-2: Greedy Algorithm with Post Reordering

No. of Teams	No. in Stack	Max in Stack	Total Stack Time	Avg. Stack Time	JIT Items	% JIT Items	Total Waiting Time	Avg. Waiting Time	Sequence Changes	Optimum Makespan	Makespan
2	7	2	15	2.14	17	56.67	45	1.5	4	49.5	50
3	13	2	32	2.46	21	70.00	30	1.0	4	33	35
4	15	1	35	2.33	24	80.80	22	0.73	7	24.75	27

Table 6-3: Greedy Algorithm with Backtracking Only

No. of Teams	No. in Stack	Max in Stack	Total Stack Time	Avg. Stack Time	JIT Items	% JIT Items	Total Waiting Time	Avg. Waiting Time	Sequence Changes	Optimum Makespan	Makespan
2	8	2	17	2.13	13	43.33	45	1.5	0	49.5	50
3	13	5	34	2.62	19	63.33	30	1.0	0	33	35
4	11	5	22	2.2	18	60.00	21	0.7	0	24.75	26

Table 6-4: Greedy with Backtracking and Post Reordering

No. of Teams	No. in Stack	Max in Stack	Total Stack Time	Avg. Stack Time	JIT Items	% JIT Items	Total Waiting Time	Avg. Waiting Time	Sequence Changes	Optimum Makespan	Makespan
2	8	2	21	2.63	18	60.00	45	1.5	18	49.5	50
3	13	2	41	3.15	23	76.76	30	1.0	27	33	35
4	11	1	24	2.18	23	76.67	18	0.6	32	24.75	26

Table 6-5: Randomly Generated Construction Durations of Sub-Blocks/Subassemblies

Sub-block	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Construction Duration	5	6	4	6	6	5	6	2	2	15
Sub-block	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
Construction Duration	2	1	4	5	5	4	1	1	2	6
Sub-block	P21	P22	P23	P24	P25	P26	P27	P28	P29	P30
Construction Duration	3	2	1	1	2	2	1	5	3	1

6.4.4. Analysis of Results

Table 6-6: Summary of % JIT Items

	Table 6-1		Table 6-2		Table 6-3		Table 6-4	
No. of Teams	JIT Items	% JIT Items	JIT Items	% JIT Items	JIT Items	% JIT Items	JIT Items	% JIT Items
2	11	36.67	17	56.67	13	43.33	18	60.00
3	16	53.33	21	70.00	19	63.33	23	76.67
4	21	70.00	24	80.80	18	60.00	23	76.67

With this particular set of random data, it can be observed that a Greedy algorithm with backtrack &/or post reordering has an advantage over a Greedy only algorithm in terms of the number of JIT Items. Comparing Table 6-1 with Table 6-2 it can be seen that there is an increase in the number of JIT sub-blocks. Further, with post reordering, there is a clear reduction in the time wasted by a reduction in multiple handling of the sub-blocks. Similarly, when comparing Table 6-1 with Table 6-3 and Table 6-1 with Table 6-4 the same improvements in the JIT and reduction in multiple handling of sub-blocks can be observed. The greatest number of JIT items is when the Greedy algorithm is augmented with backtracking and post reordering.

6.5. A Genetic Algorithm

Genetic or evolutionary algorithms, first proposed by Holland (1992), are loosely based on population reproduction. All populations have a wide range of genetic diversity with no two individuals having exactly the same genetic makeup. With reproduction, the offspring will carry genes from both parents, although in the natural process what genes an offspring inherits is indeterminate. Some offspring inherit *good* genes from their parents, others *not-so-good.* Sometimes, there is a variation in the gene that is not accounted for by the inheritance. It is these basic principles that drive the genetic algorithm (GA).

The execution of the genetic algorithm has a number of stages. Firstly, an initial population of solutions is generated. These solutions are termed *chromosomes*. Each chromosome will have a level of *fitness* when evaluated against the objective function for the solution. The higher the fitness level, the better and more advantageous is the chromosome for the solution to the problem. The chromosomes with the highest fitness level are part of an *elite* group. The theory is that if these chromosomes are combined, by way of mixing their genes, termed *crossover*, then they will produce a better offspring, i.e. one with an even better fitness level. To maintain the diversity of a population, *mutations* or small random changes in the chromosomes are introduced. The initial population is used to generate a new population by the process of crossover and mutation. Each member of the population is tested for fitness, which over a period of time converges to an optimal solution; although this solution may not be a global optimal solution, it could be just a local optimal solution.

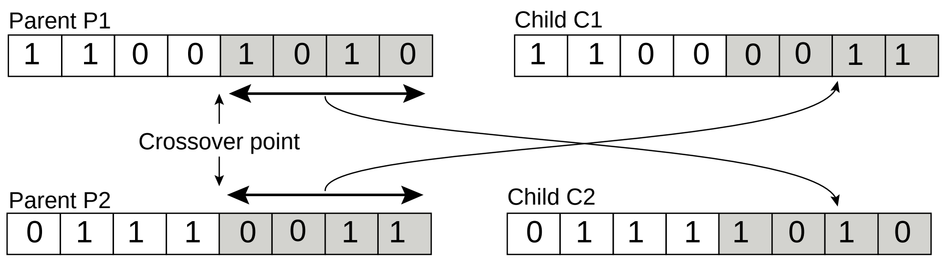


Figure 6-10: Single Point Crossover

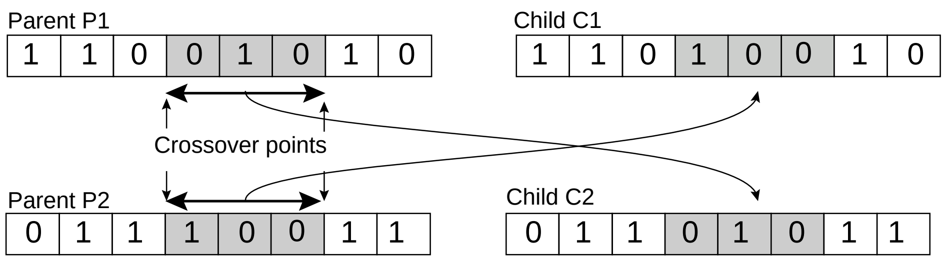


Figure 6-11: Two Point Crossover

The general principles of a GA are illustrated in Figure 6-10, Figure 6-11, and Figure 6-12. Figure 6-10 illustrates the parents selected from a population who have 8 chromosomes each. In this example, the crossover point is in the middle. The left hand 4 cells of parent P1 are combined with the right hand 4 cells to parent P2 to produce child C1. The left hand 4 cells of parent P2 are combined with the right hand 4 cells of parent P1 to produce child C2. These children become parents in the next population.

There are many ways of selecting a crossover point or crossover points. Figure 6-11 illustrates the case where 2 points in the gene are selected to make a crossover segment. The crossover segment in parent P1 is replaced by the crossover segment of parent P2 to produce child C1. Similarly, the crossover segment of parent P2 is replaced with the crossover segment of parent P1 to produce child C2. These children become parents in the next population.

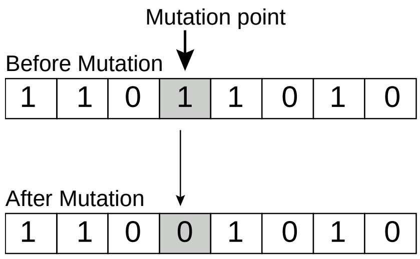


Figure 6-12: Single Point Mutation

In order to maintain genetic diversity, random mutations are introduced into the population. In the example shown in Figure 6-12, a single point mutation to the chromosome is made. Here the value 1 is made a 0. This mutation is added to the next population or evaluation.

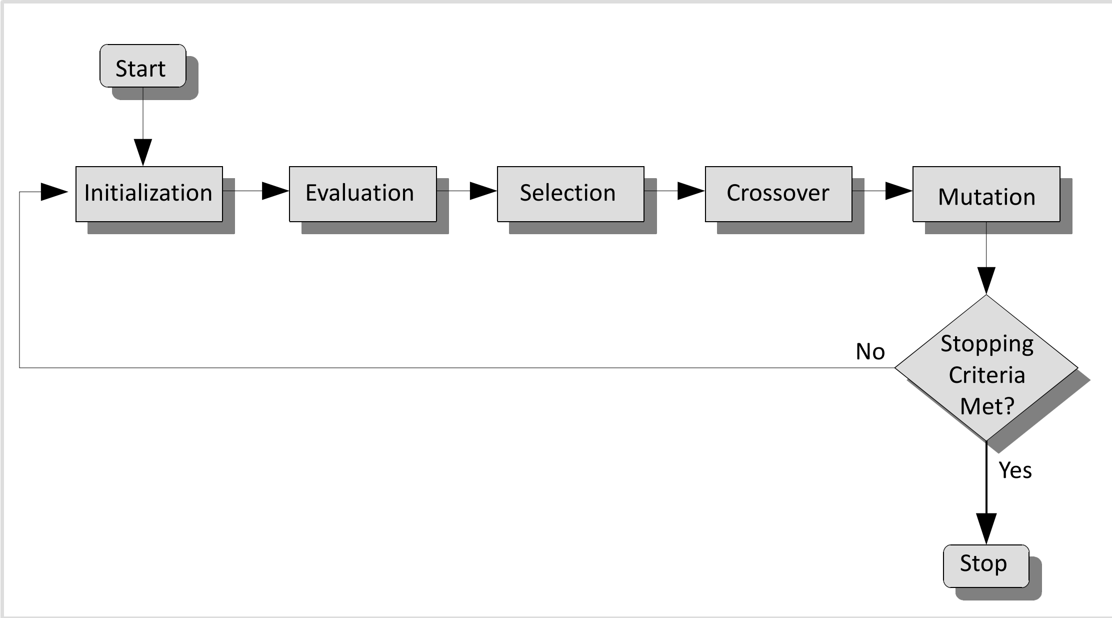


Figure 6-13: Process Flow of a GA

Figure 6-13 illustrates the process flow for using a GA. In this process, the initial population is replaced with a new population for ever iteration of the process.

6.5.1. Genetic Algorithm for Sequencing

It was seen in the earlier sections of this chapter that the various forms of the Greedy algorithm performed well in terms of makespan. However, being an algorithmic approach, they will not necessarily produce an optimum makespan. To achieve an optimum or near optimum makespan, it is necessary to use a Genetic or evolutionary Algorithm approach since, as has been mentioned, the scheduling of parallel machines is *NP*-hard, and although a heuristic approach may sometimes produce an optimum or near optimum, they are as efficient as a GA in obtaining optimums.

The initial purpose of the GA used in this research is to optimize the makespan for the construction of a set of sub-blocks/subassemblies. The GA used here keeps a percentage of the *fittest* members of the population to be incorporated in the new population. The fitness referred to here is that of the minimum makespan. A percentage of the population has its genes crossed over and is then added to the new population. A percentage of the population undergoes a mutation, to ensure diversity in the new population.

As with the Greedy algorithm, a number of teams are available for constructing the sub-blocks/subassemblies. The GA used in this research was programmed using the Microsoft .NET platform with C# as the programming language. C# has a built-in function for generating random numbers that are evenly distributed. A number of random numbers were generated in the program using the C# random function.

The steps for using the GA were as follow:-

1. Generate an initial set of 100 random solutions to the problem
2. Find the fitness of each member of the population
3. Select the fittest 5%
4. Select 70% for gene crossover
5. Select 25% for mutation selected from the crossover population
6. Repeat steps 1 to 5 for 100 iterations.

The number of parts (sub-blocks or subassemblies) required for construction was fixed. The construction time required for each part was randomly generated. The team that each part was assigned to was randomly generated. The number of teams varied from two to a maximum of four. This was to reflect the usual construction environment at the shipyard. A population of 100 parents was generated.

The makespan of a population was calculated for each of the 100 parents. The population was ordered into ascending makespan sequence, i.e. the parents with the shortest makespan are the first in the sequence, and that the parents with the longest makespan are the last in the sequence. The first 5 parents in the newly ordered sequence are selected to begin a new population. Parents 6 to 75, the next best 70 makespan, are chosen for gene crossover to add to the new population. Finally, from these 70 parents, 25 were randomly chosen for mutation.

Table 6-7: Parts Allocated to Teams for Parent *n*

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
1	1	2	2	1	3	2	1	2	1	1	3	3	3	1	1	3	3	2	1

Table 6-8: Parts Allocated to Teams for Parent *n+1*

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
1	3	3	1	1	3	2	3	2	3	2	1	3	2	2	1	3	3	2	1

To illustrate this particular GA, in this description, a string of just 20 parts will be used. Each collection of 20 parts represents a parent in a population of 100. Each part is identified by its number, P1, P2,…, P20. Each part has a randomly assigned construction time, which is constant for each generated population. For the purpose of this part of the discussion, 3 teams of workers will be used to construct all the parts. Each part is randomly allocated to team 1, 2 or 3. The randomly assigned team is shown immediately below the part number in Table 6-7 and Table 6-8.

The *elite* or *fittest* 5% of the population are retained. Since the total population is 100, the retention is 5. The process of crossover involves 70% of the population which starts immediately after the top 5% have been selected. The crossover, in this illustration, involved pairs of parents, the first pair being parents, 6 and 7, with the next pair being 8 and 9. (These crossover points are for illustrative purposes only, as in the GA, the crossover points are chosen at random.) The elite genes are those that have the minimum makespan. As there may be more than 5% of these elite genes, a further distinction is also introduced of having the maximum JIT% items and also the minimum number of items in the stack. In terms of order in each population, they are ordered in the sequence shown in (6.5), from which the first 5 genes are selected to form part of the next population.

$$\min(\sum_{j=1}^{n} \max_j C_j^S) > (\frac{J_s}{n} \times 100) > \min(\sum_{i=1}^{n} (C_{i-1} - C_i)) \quad (6.5)$$

Table 6-9: Parts Allocated to Teams for Parent n after the crossover

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
1	1	2	2	1	3	2	1	2	3	2	1	3	2	2	1	3	3	2	1

Table 6-10: Parts Allocated to Teams for Parent n+1 after the crossover

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
1	3	3	1	1	3	2	3	2	1	1	3	3	3	1	1	3	3	2	1

Table 6-11: Parts Allocated to Teams for Parent *m* before mutation

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
1	1	2	2	1	3	2	1	2	1	1	3	3	3	1	1	3	3	2	1

Table 6-12: Parts Allocated to Teams for Parent *m* after mutation

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
1	1	2	1	1	3	2	1	2	1	1	3	3	3	1	1	3	2	2	1

The crossover used in this GA is a 2-point crossover. The points for crossover are chosen randomly so that each pair of parents that participate in the crossover has different crossover points, subject to the randomizing functions limitations. In the examples illustrated in Table 6-7 and Table 6-8, the crossover points are between parts 9 and 15. The crossover result is shown in

Table 6-9 and Table 6-10. Table 6-11 and Table 6-12 show the chromosome before and after a mutation takes place. The mutation applies to the team the part that is allocated to for construction. The mutation is wholly within a single parent. The total number of genes mutated within any parent is 10% of the total number of genes in the parent. In the example being used to illustrate this GA, that is 2 genes or 2 parts. The parents chosen for mutation are randomly selected from parents 6 to 75. Just 25 parents are randomly selected from this population pool.

In validating the GA and later comparing it with the variants of the Greedy algorithm, random seeds were generated randomly. Initially, a random seed of 42 was chosen to be the seed to generate 10 random numbers that would be used as random seeds in the execution of the programs. The purpose of generating 10 random seeds was to eliminate any bias of manually choosing a set of 10 numbers for random seed, although the initial number of 42 was chosen manually and not randomly. The random numbers generated and used as seeds in random number generation were, in order of generation: - 6681, 1409, 1256, 5228, 1685, 2626, 7244, 5129, 1737 and 7612. However, in the display of data following, the seeds are displayed in ascending numerical sequence. The random seeds generated were also used in the generation of random construction durations for the parts and for randomly selecting the team the part is initially allocated to for construction. The number of parts used in the testing of the algorithms ranged from 30 to 100 in steps of 10 parts i.e. 30, 40, 50, 60, 70, 80, 90 and 100. This was considered a reasonable range as some blocks have fewer sub-blocks/subassemblies than others and this range covered all that was practically necessary.

6.5.2. Performance of the Genetic Algorithm

The GA was tested on an i-3770 CPU with 32GB of main memory, under the 64 bit Microsoft Windows 7 operating system. The line charts for 3 teams, which are taken as illustrative of the convergence of the GA and may be found in appendix A.1.1 to A.1.8 of this thesis, show the performance of the algorithm for each of the random seeds used (1256, 1409, 1685, 1737, 2626, 5129, 5228, 6881, 7244, 7612) and for the range of parts in the set, i.e. 30, 40, 50, 60, 70, 80, 90 and 100. It can be seen from the charts that the algorithm converges within less than 70 generations and, in some cases, within less than 10 generations. The execution time of the program for all 10 random seeds and all ranges of parts in the same execution of program was less than 17 seconds, which gives an average execution time for a single random seed of less than 1.7 seconds for 7 sets of parts, or 0.282 seconds per part set.

6.5.3. Details of Algorithm Simulation

The earlier simulation exercise on variations of the Greedy algorithm led to a more detailed simulation in conjunction with the GA. The four variations of the Greedy algorithm were executed under the same condition as the GA.

The GA initially had a single objective, that of Makespan. Subsequently, an additional objective was combined with this, that of the maximum number of Just-In-Time parts. Upon investigation, it was seen that there were a number of occasions where the maximum number in the stack at any one time was quite high. Several being greater than five sub-blocks/subassemblies. In practice, this would not be the case, simply for safety reasons among other things. An additional condition was added to the objective function to ensure that the maximum number in the stack was less than or equal to five. This was achieved by heavily penalizing populations with the maximum number in the stack greater than five. The appendices Table A-1 to Table A-160 show the details of each population by random seed and number of teams. Table 6-13 and Table 6-14 below, are summaries of the tables in the appendices, Table A-1 to Table A-160.

Table 6-13: Comparison of Makespan Variations for Different GAs

	GA – Makespan		GA – Makespan/JIT		GA- Makespan/Max in Stack/JIT	
	Min %	Max %	Min %	Max %	Min %	Max %
2 Teams	0.00	0.35	0.00	0.35	0.00	0.35
3 Teams	0.00	0.65	0.00	0.95	0.00	0.65
4 Teams	0.00	2.21	0.33	2.17	0.14	2.08

Table 6-13 is a summary of the variations between the theoretical makespan and the actual makespan for one of the variations on the GA and two, three and four teams. In general, the GAs performs very well in coming very close to the theoretical maximum.

Table 6-14: Comparison of Makespan Variations for Various Greedy Algorithms

	Greedy		Greedy + Backtracking		Greedy + Post Reordering		Greedy + Backtracking + Post Reordering	
	Min %	Max %	Min %	Max %	Min %	Max %	Min %	Max %
2 Teams	0.00	4.32	0.00	3.73	0.00	4.32	0.00	3.73
3 Teams	0.00	9.36	0.00	7.74	0.00	9.36	0.00	7.74
4 Teams	0.86	13.89	0.52	11.93	0.86	13.89	0.52	11.93

Table 6-14 is a summary of the minimum and maximum variations from the theoretical makespan for each of the four variations of the Greedy algorithm for two, three and four teams. It needs to be observed, that for the makespan, there is no difference between the Greedy algorithm and the Greedy with post reordering. Similarly, there is no difference between the Greedy algorithm with backtracking and the Greedy algorithm with backtracking and post reordering. This is because post reordering has no effect on the makespan, since the parts are just reordered within a team.

Comparison of Table 6-13 with Table 6-14 shows the efficiency of the GA over any variety of the Greedy algorithm. The following charts give a more detailed analysis of the data produced from the program simulations. The first set of charts Figure 6-14 to Figure 6-27 show the percentage difference between the theoretical makespan and the actual makespan for the different algorithms.

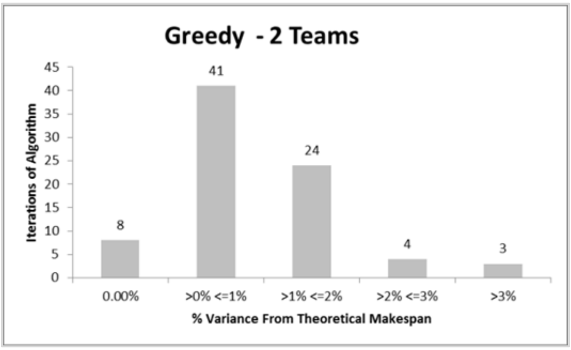


Figure 6-14: %Variance of Makespan (Greedy 2 Teams)

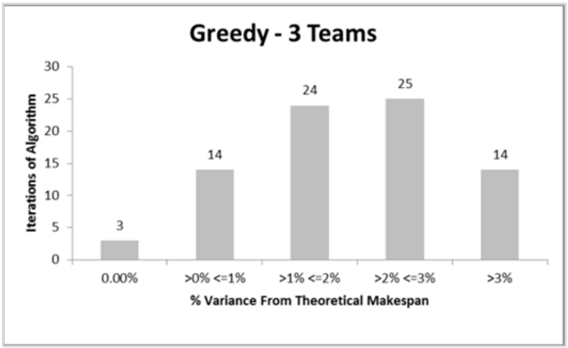


Figure 6-15: %Variance of Makespan (Greedy 3 Teams)

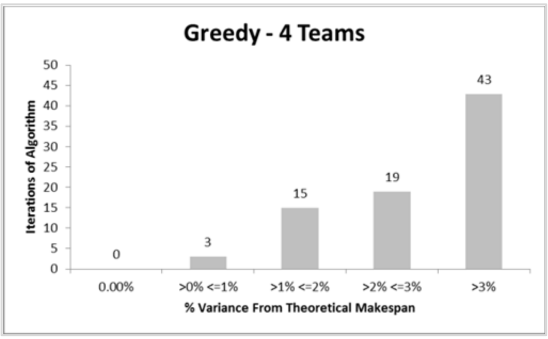


Figure 6-16%Variance of Makespan (Greedy 4 Teams)

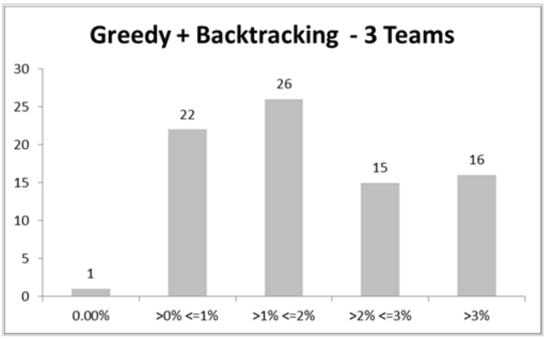


Figure 6-17: %Variance of Makespan Greedy+ Backtracking (3 Teams)

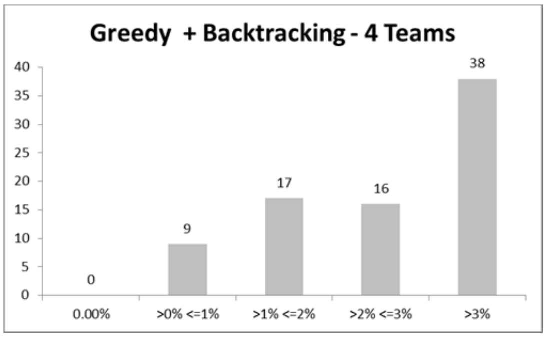


Figure 6-18: %Variance of Makespan Greedy + Backtracking (4 Teams)

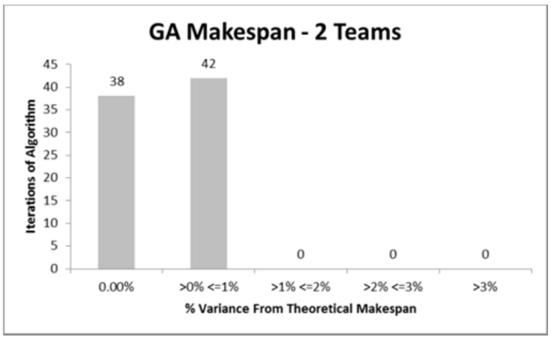


Figure 6-19: %Variance of GA Makespan (2 Teams)

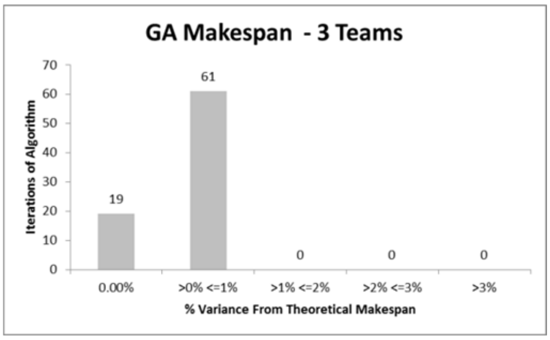


Figure 6-20: %Variance of GA Makespan (3 Teams)

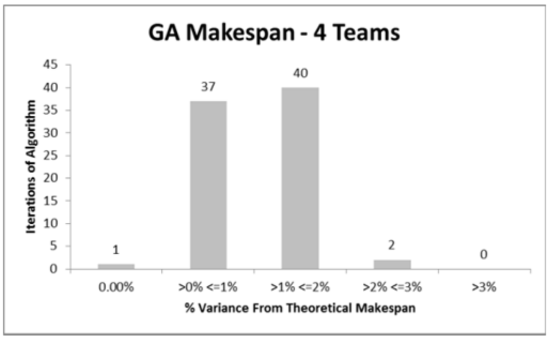


Figure 6-21: %Variance of GA Makespan (4 Teams)

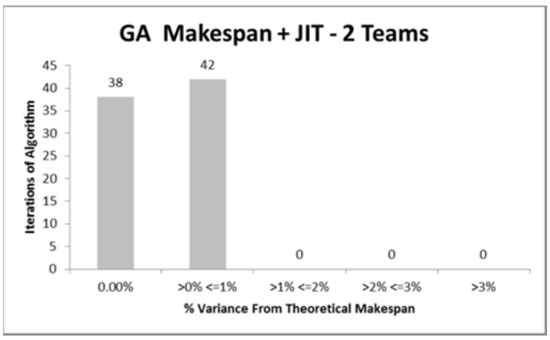


Figure 6-22: %Variance GA Makespan + JIT (2 Teams)

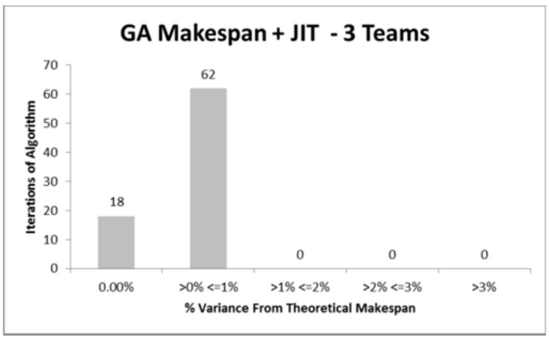


Figure 6-23: %Variance GA Makespan + JIT (3 Teams)

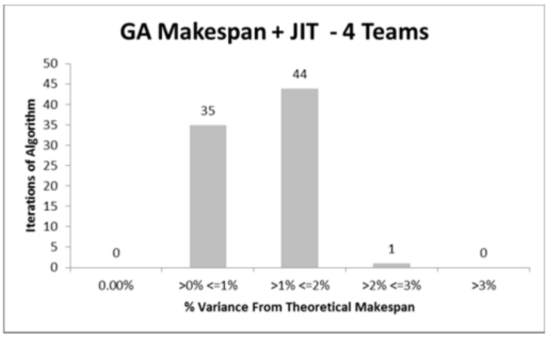


Figure 6-24: %Variance GA Makespan + JIT (4 Teams)

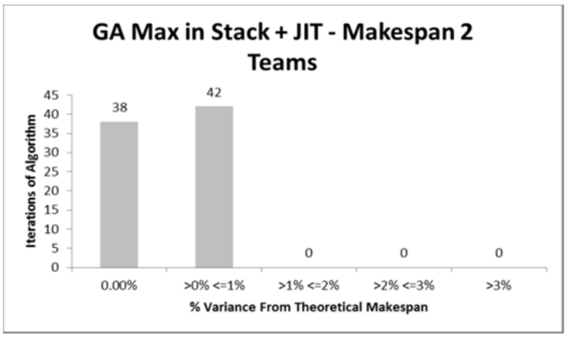


Figure 6-25: %Variance of GA Makespan + Max In Stack + JIT (2Teams)

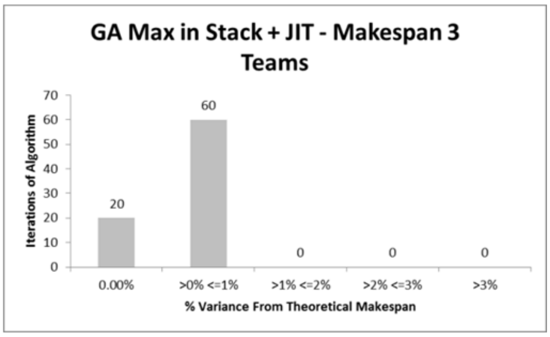


Figure 6-26:%Variance of GA Makespan + Max In Stack + JIT (3 Teams)

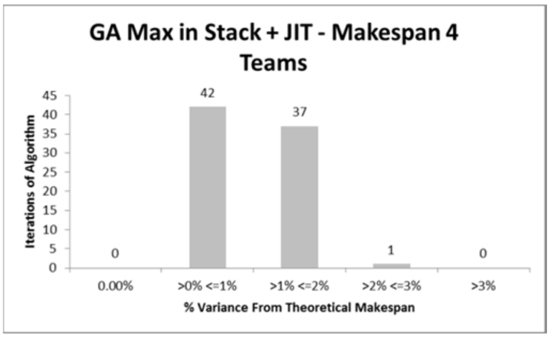


Figure 6-27:%Variance of GA Makespan + Max In Stack + JIT (4Teams)

The charts show that GAs have a smaller percentage variance with the theoretical Makespan than do the Greedy algorithms. Across all teams and for all GA variants, only for 3 of the 720 samples is the variance greater than 2%, i.e. 0.41% of the samples are greater than 2%. For the variants of the Greedy algorithm, 202 samples of the 480 samples have variances greater than 2%. i.e 42.1% of samples are greater than 2%. As minimizing the makespan was one of the objectives, it is clear that a GA should be adopted.

Another objective of this work is to have the parts arriving JIT, or as near JIT as possible. The following charts show number of items, of 80 in each sample. The bands of values used are shown in Table 6-15.

Table 6-15: Bands for JIT Charts

Band 1	Band 2	Band 3	Band 4	Band 5	Band 6	Band 7
<=20%	>20% & <=30%	>30 & <=40	>40% & <=50%	>50% & <=60%	>60% & <=70%	>70%

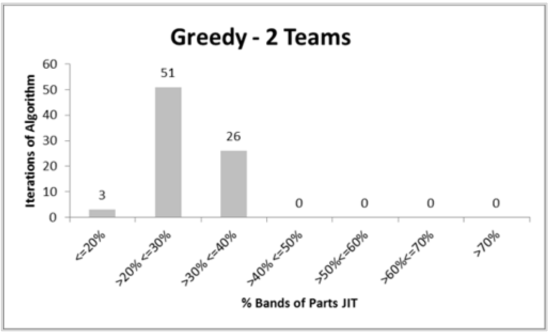


Figure 6-28: No. of JIT Items by % Bands (Greedy 2 Teams)

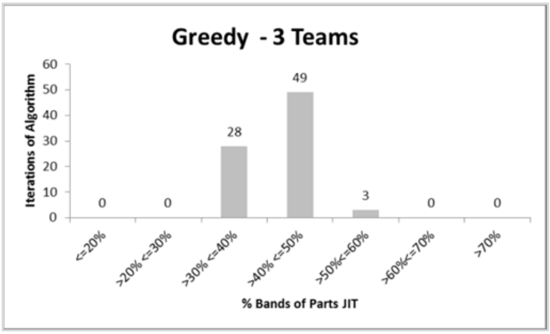


Figure 6-29: No. of JIT Items by % Bands (Greedy 3 Teams)

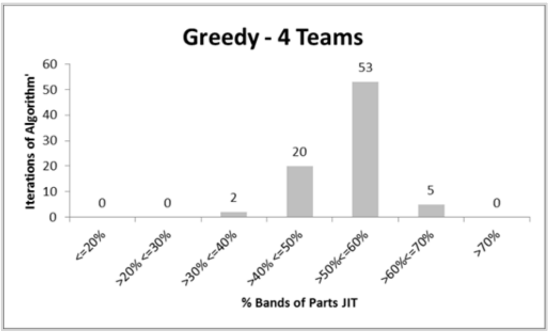


Figure 6-30: No. of JIT Items by % Bands (Greedy 4 Teams)

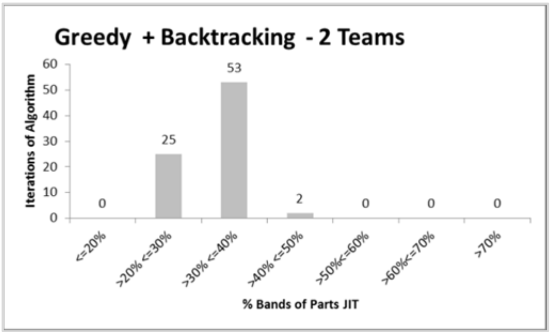


Figure 6-31: No. of JIT Items by % Bands (Greedy+ Backtracking 2 Teams)

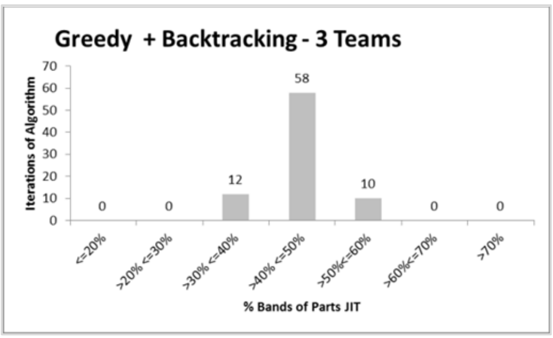


Figure 6-32: No. of JIT Items by % Bands (Greedy+ Backtracking 3 Teams)

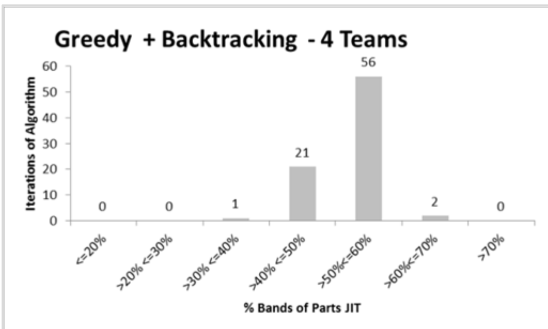


Figure 6-33: No. of JIT Items by % Bands (Greedy+ Backtracking 4 Teams)

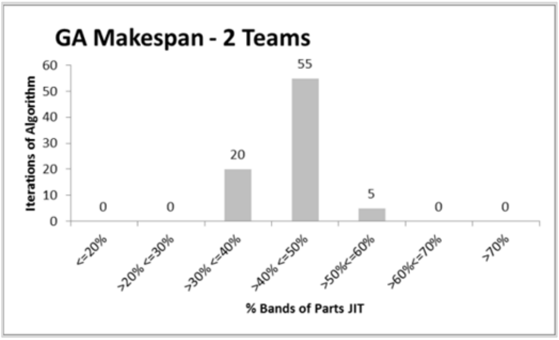


Figure 6-34: No. of JIT Items by % Bands (GA Makespan 2 Teams)

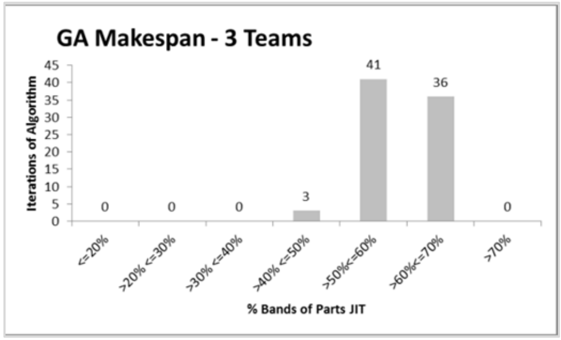


Figure 6-35: No. of JIT Items by % Bands (GA Makespan 3 Teams)

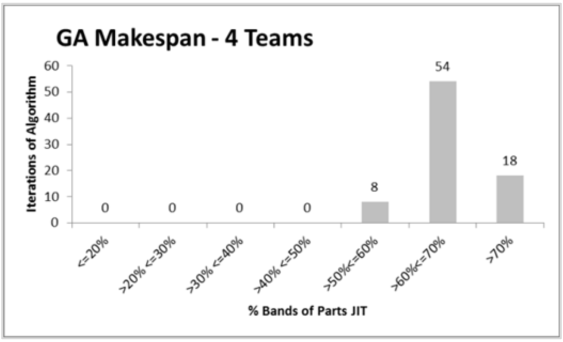


Figure 6-36: Number of JIT Items by % Bands (GA Makespan 4 Teams)

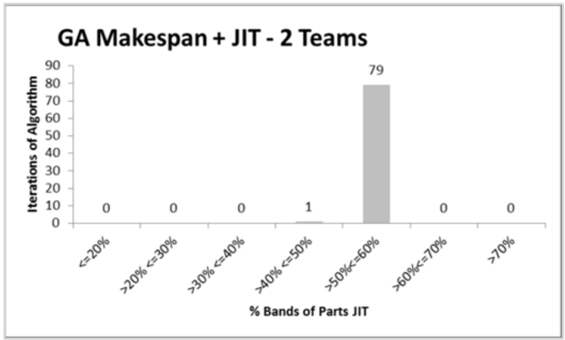


Figure 6-37: Number of JIT Items by % Bands (GA Makespan + JIT 2 Teams)

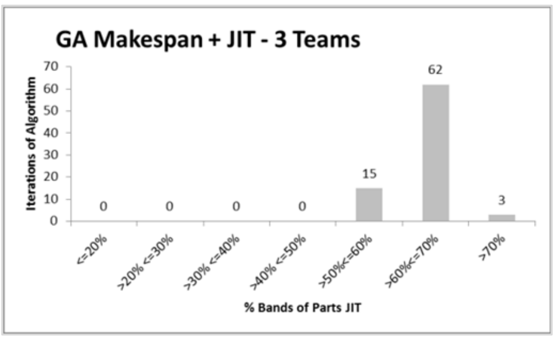


Figure 6-38: Number of JIT Items by % Bands (GA Makespan + JIT 3 Teams)

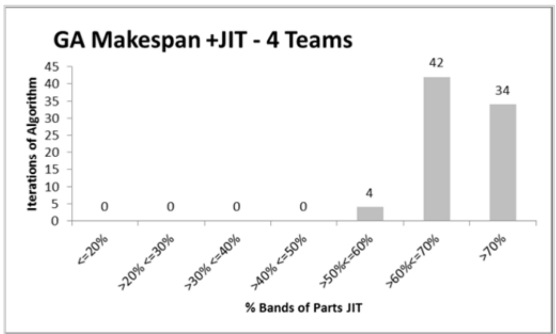


Figure 6-39: Number of JIT Items by % Bands (GA Makespan + JIT 4 Teams)

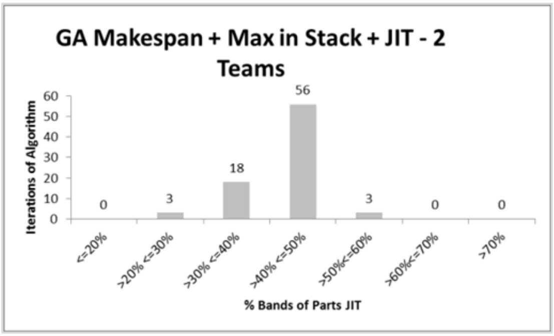


Figure 6-40: Number of JIT Items by % Bands (GA Makespan + Max in Stack + JIT 2 Teams)

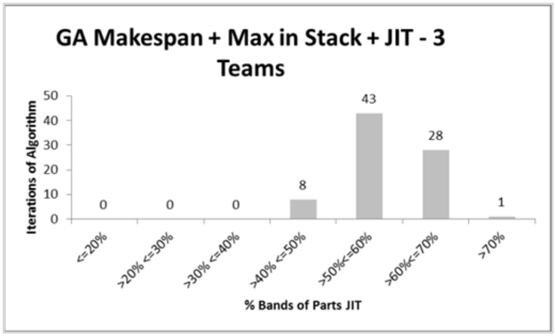


Figure 6-41: Number of JIT Items by % Bands (GA Makespan + Max in Stack + JIT 3 Teams)

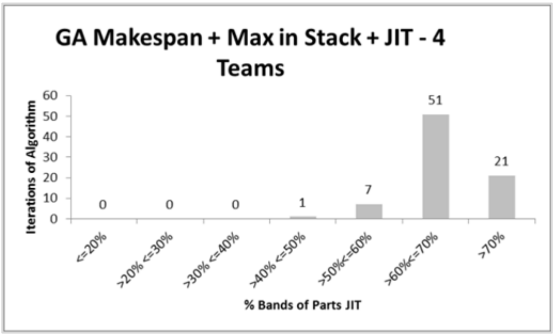


Figure 6-42: Number of JIT Items by % Bands (GA Makespan + Max in Stack + JIT 4 Teams)

In analysing the charts in Figure 6-28 to Figure 6-42, it is evident that when comparing Greedy with Greedy plus backtracking, has a higher percentage of items arriving JIT for 2 and 3 teams summed across all quantities (30, 40, 50, 60, 70, 80, 90, 100 parts) and for 4 teams the percentages are equal. This shows that if the work is speed across more than 3 teams that more items will be completed JIT, however, as was seen in Figure 6-18, the makespan for 4 teams under these conditions was the worse of the set of Greedy algorithms. This indicated that an increase in the number of teams does not necessary improve the efficiency of the work. Although the makespan is shorter in execution time, it is not more efficient.

When it comes to the GAs, all "Makespan" and "Makespan + JIT", (Figure 6-34 to Figure 6-39) except for Figure 6-28, have the majority of their parts in bands 5 and band 6. However, when limits on the number of items that can be held in the stack are added Figure 6-40 indicates that the majority of JIT items fall in band 3 and band 4. From Figure 6-41 and Figure 6-42 the majority of JIT items are in band 5 and band 6 with a better packing in the higher bands than in Figure 6-38 and Figure 6-39 for 3 teams and Figure 6-41 and Figure 6-42 for 4 teams.

6.6. Solution Quality

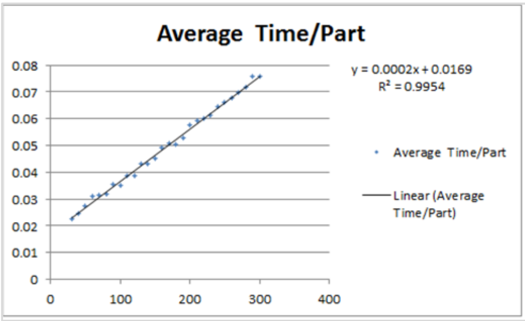


Figure 6-43: Greedy Times Per Part

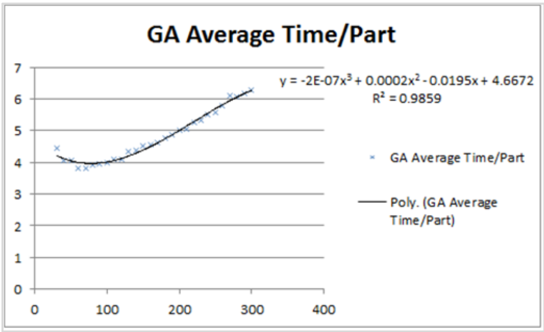


Figure 6-44: GA Times Per Part

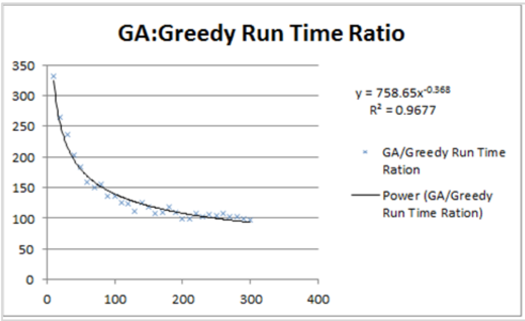


Figure 6-45: Ratio of GA to Greedy

To analyse the quality of the algorithms, and to reduce any interference from any background work associated with the dotNet framework, the Greedy with backtracking and the Genetic Algorithm with backtracking were modified to use the console out rather than the windows output. This removed some known variables, such as the refreshing of the window.

A block may have many parts, but the algorithm for sequencing was tested using just the plates, and typically, a block may have 30 major plates as a low number and 300 as a high number. The two algorithms were executed both using 3 teams and with the number of plates ranging from 30 to 300 on a i7-2640M CPU @ 2.80GHz and 4.00 GB using 64-bit windows 7 operating system.

The execution times in milliseconds to three decimal places were captured for the core processing part of each program. In pure processing times, the Greedy algorithm with backtracking outperformed the Genetic Algorithm. It should be emphasised that the termination condition for the GA was set to be 100 generations.

Figure 6-43 is the graph of the Greedy algorithm with backtracking for three teams. The X-axis values are the number of parts, the Y-axis values are for processing a single part in milliseconds. The time for a single part was calculated by dividing the total time for each set of parts by the number of parts. The average is that for all 10 random seeds for each number of parts. The graph shows that the algorithm has a linear increase in time. Using Excel curve fitting function, the graph has an equation of $y = 0.0002x + 0.0169$. This is a good fit with $R^2 = 0.9954$.

Figure 6-44 is the graph of the Genetic Algorithm with backtracking for three teams. The axes are the same as for Figure 6-43. Using Excel for curve fitting yields:- $y = -0.07x^3 + 0.0002x^2 - 0.0195x + 4.6672$, with $R^2 = 0.9859$. Although this fit is not as good as the Greedy algorithm, it is an acceptable fit.

The curve for the Greedy algorithm is linear and for the Genetic algorithm cubic.

The ratio of times is given by $\frac{Time\ per\ part\ for\ GA}{Time\ per\ part\ for\ Greedy}$. Figure 6-45 shows the results of this curve which has the characteristics of a power curve. Excel curve fitting function yields a power equation of $y = 758.65x^{-0.368}$ and $R^2 = 0.9677$. Although the fit is not as good as the fit for the Greedy or Genetic algorithm, it is still in the acceptable range.

6.7. Limitations of the Algorithms

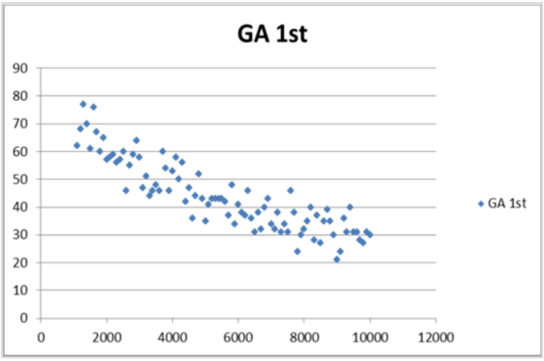


Figure 6-46: GA with Minimum Makespan

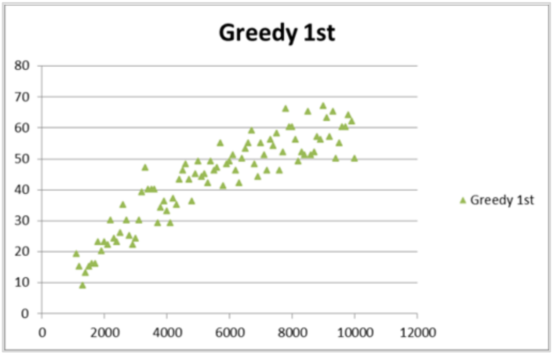


Figure 6-47: Greedy with Minimum Makespan

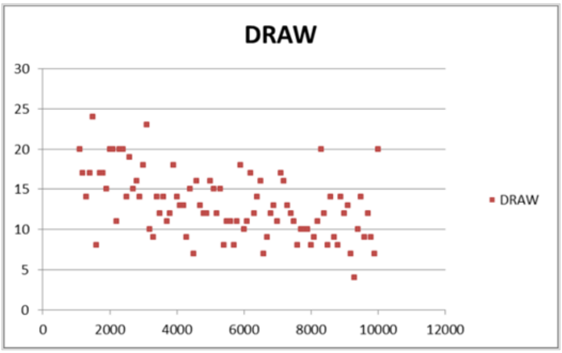


Figure 6-48: Occasions of Equal Makespan

While a comparison of the Genetic and Greedy algorithm has been covered, it was for 30 to 100 parts in increments of 10 parts. In every case, the GA outperformed the Greedy algorithm in minimizing the makespan. When analysing the algorithms for 30 to 300 plates in increments of 10, in the immediately preceding section, in 280 iterations of the algorithms, the GA outperformed the Greedy algorithm 273 times, 28 times, there was no difference in the makespan and on 6 occasions, the Greedy outperformed the GA. This raised the question as to whether the GA as used would produce the minimum makespan across a larger range of values. To test this, both the Greedy and GA programs were run with a number of parts ranging from 1000 to 10000 in increments of 10 parts. The seed for the random numbers were the same as had been used in all previous tests, and the initial and subsequent populations for each generation for the GA was fixed at 100.

The results were grouped at each hundred. For example, the 10 parts from 1110 to 1200 each had a result for each of the random seeds, thus giving 100 random values. The number of times the GA and the Greedy algorithm had the minimum makespan was calculated, as were the number of times the GA and Greedy algorithm had an identical makespan.

Figure 6-48 shows that for one grouping of 100, both algorithms had an identical makespan of 24 times and, at the other end of the scale, an identical makespan was recorded just 4 times. The average number of times both algorithms produced an identical makespan across all executions of the algorithms was 13. That is 13% of the time.

Figure 6-46 shows that the Genetic algorithm produces the minimum makespan in a declining pattern. The larger the number of parts to be processed by the GA, the least likely it is to produce the minimum makespan. Figure 6-47 shows the Greedy algorithm with the inverse results, the smaller the number of parts in the population, the least likely the Greedy algorithm will produce a minimum makespan. The point where the GA does not produce the minimum makespan is around the 5500 number of parts. For all the executions of the GA, the population size is 100. This clearly illustrates the limitation of the original parameters for the GA, which were selected for a realistic set of values for block construction.

Diaz-Gomez and Hougen (2007), examining the problem of an initial population size for a GA, show that as the population size increases and chromosome length increases, the diversity of the population becomes independent of both. In the work here, only the population size changes but the chromosome length is static, hence diversity is not so great. Tsoy (2003) examines the workings of a GA with a fixed population size and a limit to the number of generations. As Tsoy points out, it is the nature of the GA that predictive methods of ideal population size and generations are absent. A large population should give a better solution, but at the cost overall performance of the GA.

The GA used previously in this research gave a very good result with a small space and a fixed population. It did not give a good result with a large sample and a fixed population. Taking the immediately above mentioned research into account, the GA parameters were modified.

The elite selection of 5% remained, but the percentage of genes for crossover was changed from 70% to 80% with a decrease of the mutation percentage from 25% to 15%. Further, the convergence condition of 100 generations was enhanced by adding two more possible conditions which were, (1) the difference between the theoretical makespan and the calculated makespan was less than 2 and (2) when $\frac{CalculatedMakespan - TheoreticalMakespan}{TheoreticalMakespan} < 0.0002$. The reason for the selection of 0.0002 will be made clear later.

The population selection parameters were modified to be a percentage of the number of parts (or sample space). Three values, 60%, 40% and 20% were chosen to reflect a reasonable range of percentages, with the condition that no initial population size was less than 100. The population size for generation two onwards did one of two things, (1) the population size was set to 100 or (2) the initial population size was maintained the same for all generations of the GA.

For purposes of identity, simulations whose initial number of population was a percentage of the number of parts and whose number of populations in subsequent generations was fixed to 100 are referred to as 20% fixed, 40% fixed and 60% fixed. The simulations whose initial number of populations were a percentage of the number of parts participating in the simulation and whose population for all subsequent generations did not change, i.e. the population was percentage of the number of parts for all generations of the simulation, are referred to as 20% variable, 40% variable and 60% variable.

Table 6-16: Simulation Results to Derive Termination Ratio

	60% fixed	40% fixed	20% fixed	60% variable	40% variable	20% variable
Average of ratios	0.00026	0.000237	0.00025	0.000172	0.000176	0.000184
Maximum	0.001229	0.001002	0.001006	0.034312	0.001002	0.001006
Median	0.0002	0.000186	0.000204	0.000141	0.000146	0.00015

Running the simulations using the same 10 random numbers as used in earlier in this research, the average, maximum and median of the ratio of $\frac{CalculatedMakespan - TheoreticalMakespan}{TheoreticalMakespan}$ was calculated, the results of which can be seen in Table 6-16. The number of parts used for each simulation execution ranged from 500 to 10,000 in increments of 500.

The results from the simulations were compared in a number of ways. The simulations with ‘fixed’ populations were compared with each other. The simulations with ‘variable’ populations were compared with each other. Both ‘fixed’ and ‘variable’ population simulations were compared with the Greedy with Backtracking algorithm.

Table 6-17: Comparison of Best makespan with ‘fixed’ Populations

	Percentage of times that 60% fixed has best makespan	Percentage of times that 40% fixed has best makespan	Percentage of times that 20% fixed has best makespan	Percentage of times that there is no difference in makespan
60% fixed compared to 40% fixed	29%	38%	-	33%
60% fixed compared to 20% fixed	37.5%	-	36%	26.5%
40% fixed compared to 20% fixed	-	43%	33%	24%
Average	-	-	-	27.83

Table 6-17 has to be read by row, then column. For example, row 1 shows the comparison of two simulation runs: 60% ‘fixed’ and 40% ‘fixed’. Column 1 shows the percentage of times 60% ‘fixed’ simulation had a smaller makespan than the 40% ‘fixed’ simulation.

In analysing Table 6-17 above, it can be seen that having 40% of the sample space as the initial population size produces a better result than either 60% or 20%. The 40% selection against the 60% selection has the 40% selection with the best makespan of 38% of the number of simulation executions. Whereas the 60% selection only manages 29% of the number of simulations executed.

Table 6-18: Comparison of Best makespan with 'variable' Populations

	Percentage of times that 60% variable has best makespan	Percentage of times that 40% variable has best makespan	Percentage of times that 20% variable has best makespan	Percentage of times that there is no difference in makespan
60% variable compared 40% variable	26%	23.5%	-	50.5%
60% variable compared to 20% variable	30.5%	-	20.5%	49%
40% variable compared to 20% variable	-	30%	23%	47%
Average	-	-	-	48.95%

Table 6-18 is read in the same way as Table 6-17. Examining

Table 6-18, it can be seen that by having the number of populations equal to a percentage of the sample space give equitable results. The "no difference" column, the last column in the table, indicates that on average, 48.93% no percentage variation in the population 'variable' is dominant. The largest is when comparing 60% result with 20% result, with 60% having only 30% of its simulation results being dominant.

Table 6-19: Comparison of 'fixed' GA and Greedy Algorithm with Backtracking

	% of times that GA 'fixed' had best makespan	% of times that Greedy had best makespan	% of times no difference in makespan
GA 60% 'fixed' vs Greedy with Backtracking	48.5%	38%	13.5%
GA 40% 'fixed' vs Greedy with Backtracking	49.5%	36%	14.5%
GA 20% 'fixed' vs Greedy with Backtracking	46.5%	34.5%	19.5%
Average	48.17%	36%	15.83%

Table 6-19 shows clearly, that the GA produces a minimum makespan between 46.5% and 48.5% of the time. The Greedy algorithm with backtracking produces a minimum makespan of 36% of the time, on average.

Table 6-20: Comparison of ‘variable’ GA and Greedy Algorithm with Backtracking

	% of times GA % ‘variable’ had best makespan	% of times Greedy with Backtracking had best makespan	% of times there is no difference in makespan
GA 60% ‘variable’ vs Greedy with Backtracking	60%	23.5%	16.5%
GA 40% ‘variable’ vs Greedy with Backtracking	63.5%	20.5%	16%
GA 20% ‘variable’ vs Greedy with Backtracking	57.5%	24%	18.5%
Average	60.33%	22.67%	17%

Table 6-20 shows that the GA produces a minimum makespan on average of 60.33% of the time as opposed to the Greedy algorithm, which produces a minimum makespan of 22.67% of the time. Clearly, the GA is dominant with the ‘variable’ population.

Table 6-21: Comparison of ‘fixed’ and ‘variable’ GA Populations

	60% fixed vs 60% variable	60% fixed vs 40% variable	60% fixed vs 20% variable	40% fixed vs 40 variable	40% fixed vs 20% variable	20% fixed vs 20% variable
% times ‘fixed’ has better makespan	9.0%	11.0%	17.5%	14.5%	18.5%	0%
% times no difference	38.5%	34.0%	29.0%	36.5%	32.5%	100%
% times ‘variable’ has better makespan	52.5%	55.0%	53.5%	49.0%	49.0%	0%

Table 6-21 gives a comparison of makespan between the two versions of the GA, i.e. ‘fixed’ and ‘variable. The second row in the table shows the percentage of times the ‘fixed’ GA has a minimum makespan. The very best here is in the case of 40% ‘fixed’ when compared with 20% ‘variable’. However, it needs to be noted, as may be seen in the final column, that the 20% ‘variable’ has no advantage over the 20% ‘fixed’.

The conclusion of these simulations is that the GA is quite dominant in producing a minimum makespan and that the ‘variable’ population produces a minimum makespan more often than the ‘fixed’ population. A ‘variable’ population of 40% produces a minimum makespan more frequently than either 60% or 20%.

6.8. Summary

In this chapter, four variations on a Greedy algorithm and a Genetic algorithm have been shown. Programs written in C#.net have been used to test the effectiveness of the algorithms. To ensure a fair test, 10 random numbers were taken as seeds for the random number generator that were used to generate the construction duration. The important value for an organization is the makespan; a secondary important value is the number of parts that have to be stored in a stack. For safety reasons, these need to be limited. This has been included in the GA; it is not a factor that can be included in the Greedy algorithms. The GA outperformed the Greedy algorithms in the variance of makespan, the variance of JIT items. Further, as many factors can be taken into account when designing the GA, it is clear that using the GA is a good approach for scheduling the construction sequences for the parts.

The execution time for the GA was longer than the Greedy algorithms, but the difference is of little consequence as it was in the order of a few seconds.

In testing the limitations of the GA, it was seen that using a fixed population for the initial population and for all subsequent generations produced good results when bench marked against the Greedy algorithm with Backtracking, provided that the sample space was less than about 1000. (i.e. the population was 10% of the sample space or greater.)

Modifying the parameters of the GA, it was found that an initial population of 40% of the sample space, which was kept at 40% of the sample space for all generations, produced the minimum makespan more frequently.

Chapter 7. Case Study

7.1. Introduction

The use of Polychromatic Set Theory for identifying the connectivity of parts looks as if there is a requirement for sparse matrix algorithms; however, this is not the case as will be seen in the development of the model. The heuristic developed for sequencing in chapter 4 will be used here to identify the assembly sequence. Again it needs to be emphasised that this is an assembly sequence of the sub-blocks/subassemblies in their final position. The full welding takes place after the sub-blocks/subassemblies have been tack-welded into their position.

Another possible concern in using Polychromatic sets is that it necessitates are large number of iterations; this is not the case, as will be demonstrated in this chapter.

7.2. Creating the Model

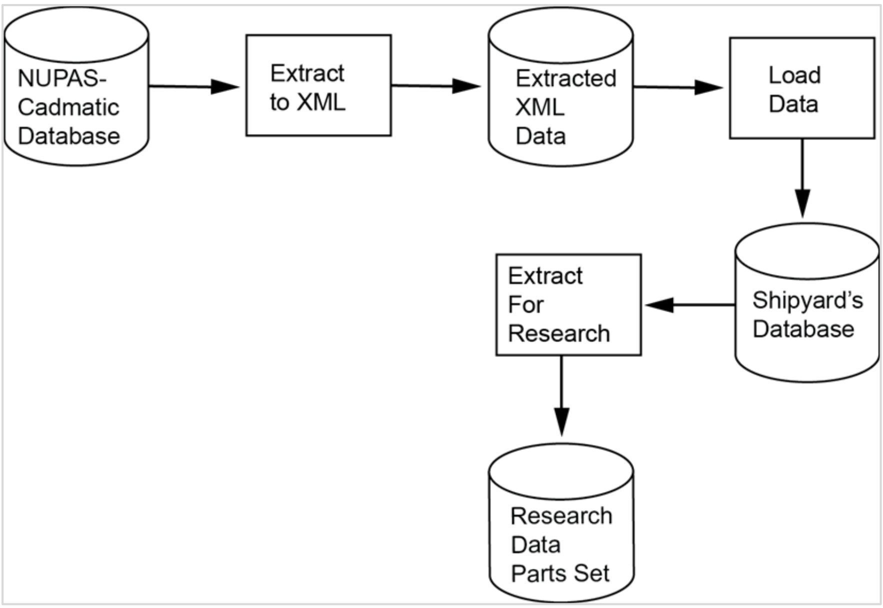


Figure 7-1: Creation of Research Data

Figure 7-1 illustrates the flow process for creating the research parts set data. The research database has been designed for compactness and for the efficient use of Polychromatic sets. The shipyard's database segmented the different part types, plate, stiffener, flange and bracket, into different database tables. This distinction has been preserved with some enhancements.


```
<part type="BRACKET">
    <keys logkey="23" pltkey="13" attkey="-1" b_attkey="-1" shpkey="-1"></keys>
    <wbs block="8" panel="F114C" part="905" assemblyorder="-99612" processcode="9048"></wbs>
    <geo weight="10.305450" thickness="10.000000" cogx="57005.003906" cogy="9812.486328"
    cogz="11320.597656" length="853.016113" width="404.613495" area="?"></geo>
    <dump_3d>
        <sweep nr_curves="1" length="10.000000" direction="1.000000 0.000000 0.000000">
            <curve nr_segments="11" x_start="57000.000000" y_start="9388.030273" z_start="11500.599609">
            <linearsegment end_x="57000.000000" end_y="9938.000000" end_z="11494.799805">
            </linearsegment>
            <circulararc arc_center_x="57000.000000" arc_center_y="9988.000000"
arc_center_z="11494.299805" arc_mp_x="57000.000000" arc_mp_y="9952.477539" arc_mp_z="11459.109375"
arc_end_x="57000.000000" arc_end_y="9988.030273" arc_end_z="11444.296875">
            </circulararc>
            <linearsegment end_x="57000.000000" end_y="9988.000000" end_z="10894.299805">
            </linearsegment>
            <linearsegment end_x="57000.000000" end_y="9963.000000" end_z="10894.299805">
            </linearsegment>
            <linearsegment end_x="57000.000000" end_y="9963.000000" end_z="10916.799805">
            </linearsegment>
            <circulararc arc_center_x="57000.000000" arc_center_y="9713.000000" arc_center_z="10916.799805"
arc_mp_x="57000.000000" arc_mp_y="9944.221680" arc_mp_z="11011.859375" arc_end_x="57000.000000"
arc_end_y="9890.708984" arc_end_z="11092.639648">
            </circulararc>
            <linearsegment end_x="57000.000000" end_y="9585.320313" end_z="11401.200195">
            </linearsegment>
            <circulararc arc_center_x="57000.000000" arc_center_y="9407.610352" arc_center_z="11225.400391"
arc_mp_x="57000.000000" arc_mp_y="9505.125000" arc_mp_z="11455.568359" arc_end_x="57000.000000"
arc_end_y="9410.281250" arc_end_z="11475.359375">
            </circulararc>
            <linearsegment end_x="57000.000000" end_y="9387.769531" end_z="11475.599609">
            </linearsegment>
            <linearsegment end_x="57000.000000" end_y="9388.030273" end_z="11500.599609">
            </linearsegment>
           </curve>
    </sweep>
  </dump_3d>
</part>
```


Figure 7-2: Example of XML Data

From Figure 7-1, it can be seen that the data used in this research was extracted from the candidate company's database. The candidate shipyard uses NUPAS as its ship's drawing software. Since the offshore rig data is not directly accessible, NUPAS provides an exporter program that extracts from its own database into an XML format.

Figure 7-2 is part of an XML file extracted from the NUPAS-Cadmatic data. This part is a bracket. The XML statements describe the profile of the bracket, including any enclosed holes. The coordinates are in absolute terms for the rig. At the shipyard, this data is read and loaded into the shipyard's own database for their internal use. The data used in this research was extracted from the shipyard's own database, also in an XML format, then uploaded to the research database. The data extracted from the shipyard's database and used in this research already has its own *part identification* as each part has a unique identification: *Part weight, part thickness, part extrusion vector, part vertices* and *curved profile*.

7.2.1. Design of the Research Database

The keys for the various parts, plate, stiffener, flange and bracket, on the shipyard's database are compound keys, that is, the keys are composed of several parts of the data contained within the record. The shipyard's parts database key is composed of *Project Number, Block Name, Plate Name, Plate Key,*and *Plate Number.* The *Plate Name, Plate Key* and *Plate Number* are data from the NUPAS XML file. This compound key uniquely identifies a single part.

When the XML data are loaded to the research database, a single table holding all the parts is populated. This table has a straightforward key, comprising of the *Project Identity*, which is a unique number for each new project. The *Block Identity,* is a unique number within each project and the *Part Identity*, is a number assigned to the part, unique for each block and within a range that identifies the part type. Plates are numbered in the range 1 to 500,000. Flanges use the range 500,001 to 1,000,000.

Stiffeners use the range 1,000,001 to 1,500,000. Brackets use the range 1,500,001 to 2,000,000. The unique sequencing of parts into specific ranges gives a simple method to identify the part type, reduces the amount of data to be checked and allows for rapid access to the entry of the part in its set. This will be explained later. This database design is used to facilitate the use of Polychromatic set usage, and is discussed in the section 7.3. To be able to relate to the original part identification, a *key table* for each part type is loaded with the unique key and the parts related information. For example, a plate on the research database may have the unique identifier of 1. This is of no use to a user, who would refer to the plate BB1-19-114, i.e. the plate name is BB1, the plate key is 19 and the plate number is 114. (This information originates from the NUPAS database.)

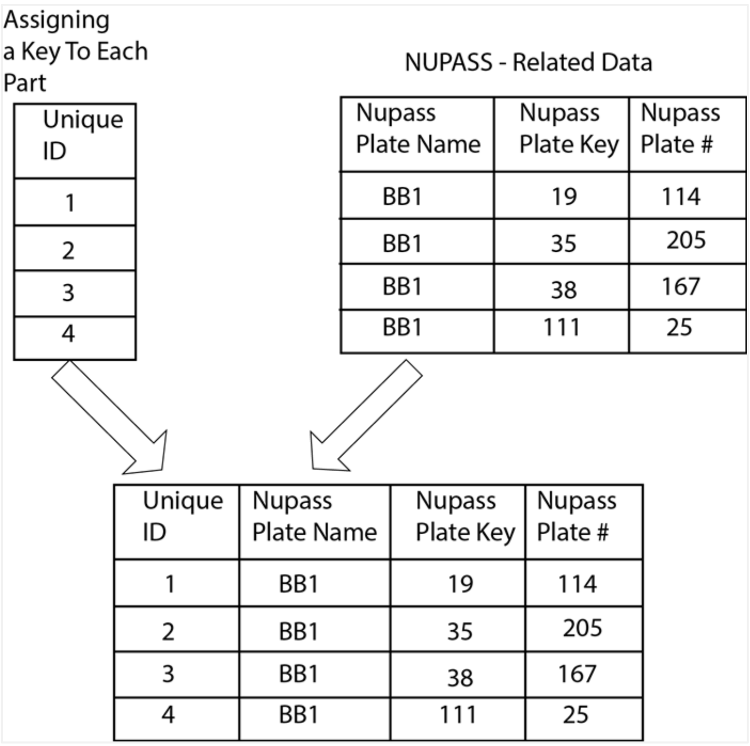


Figure 7-3: Linking Unique Keys to NUPAS Data

The *parts set* data table holds minimal information that is necessary: - *Project Unique Identity, Block Unique Identity, Part Unique Identity, Bound Box, Thickness,* *Length, Profile Data* and *Weight*. The first 3 pieces of data have been referred to already. The *Bounding Box* is simply the 3D coordinates necessary to define a box within which the part fits.

The *Profile Data* holds an extrusion vector. The extrusion vector, as its name suggests, gives the direction of extrusion of a part. Parts with a butt connection may not necessarily have the same extrusion direction. The extrusion direction depends on how the engineer draws a particular part in the NUPAS drawing package. For planar objects, the extrusion direction may be the same as the surface normal, but not necessarily so. The engineer could draw the end profile of a part and extrude it to the length of the part. Although this is very unlikely, it is possible. A planar plate is illustrated in Figure 7-5. For curved parts, the extrusion would be synonymous with the length of the part, as may be seen in Figure 7-4.

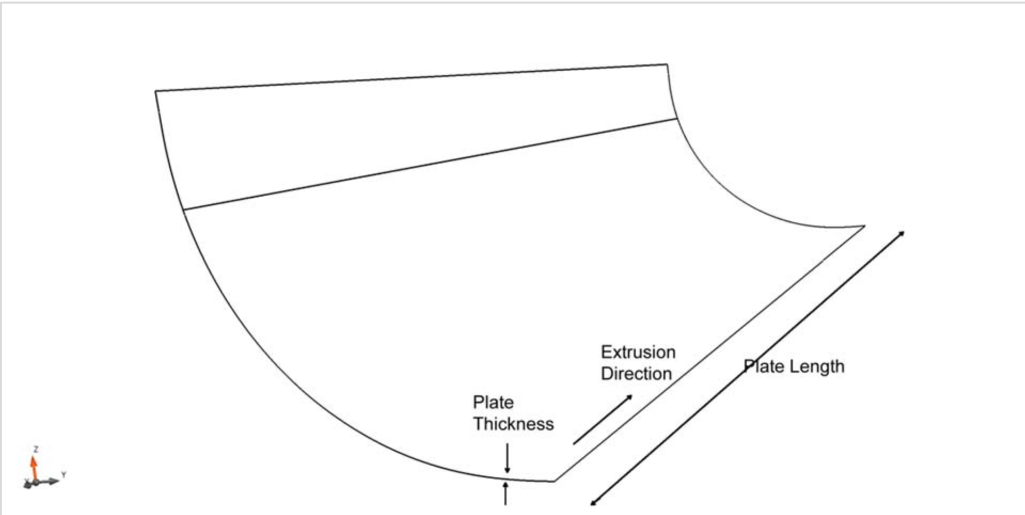


Figure 7-4: Curve Panel Showing Fundamental Data

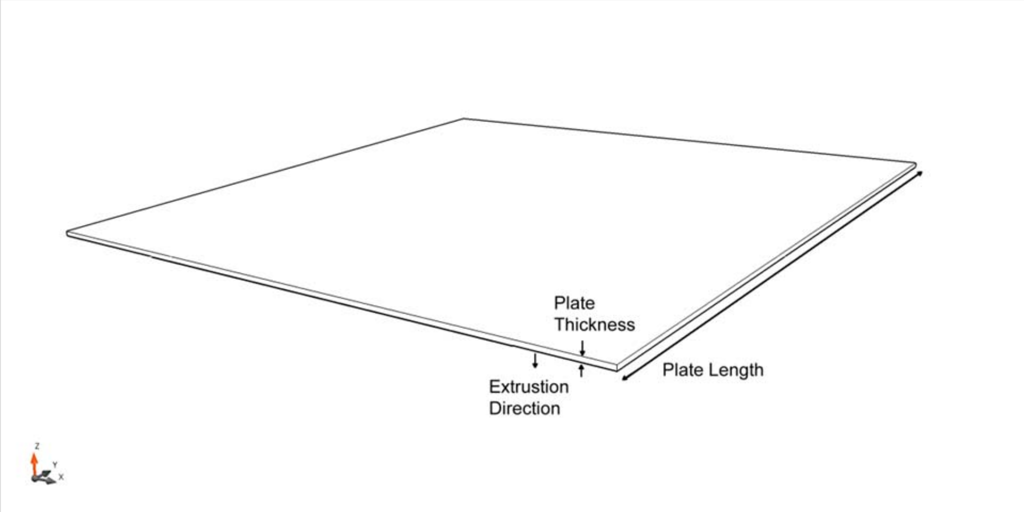


Figure 7-5: Planar Panel Showing Fundamental Data

7.3. Creating Set Indices

The Polychromatic sets are derived by taking the Cartesian cross product of various part types used in a block's construction. When searching for part combinations and details about the matching parts' colours (or properties), it would appear that it is always necessary to iterate through all elements of a set until the required element is located and its property or properties retrieved. However, by using the unique range number for each part type, this proves not to be the case. An index to a parts position within a set is generated when the sets are initially created.

As each part type is allocated a different range of numbers, the uniquely assigned number minus the start of the range gives the relative position of the part within its set. For example, the range for flanges is 500,001 to 1,000,000. The third flange read from the database would have the sequence number of 500,003. Its relative number within the set is $500003 - 500001 = 2$. Using the relative position within a set enables direct access to the information; the need to iterate through the set is thus eliminated, as will be illustrated in the next paragraph.

When building sets of connectivity, the information, although shown in sparse matrix form elsewhere in this research and in the literature, is not held in this form by the computer program. For each plate, for example, the computer program keeps a record of connectivity to all part types only where there is a connection; it does not hold a full Polychromatic Set matrix, as this is unnecessary. The reference held is that of the unique key of the part within its own set. This allows for rapid direct access to the information required, rather than repeated iteration through the set; thus computational time is reduced to a minimum.

Table 7-1 gives a practical illustration of how related information is held. The part illustrated, (not specified), is linked to 4 other plates, 5 stiffeners, 2 flanges and 3 brackets. To obtain more information about the stiffeners, the stiffeners at relative positions, 33, 4, 18, 48 and 74 in the stiffener set would be interrogated.

Table 7-1: Example of Linkage Using Relative Indexes

Part Links To:	Relative Set Entries				
Plate	14	36	95	194	
Stiffener	33	4	18	48	74
Flange	3	6			
Bracket	10	15	29		

7.4. Welding Lengths from Case Study

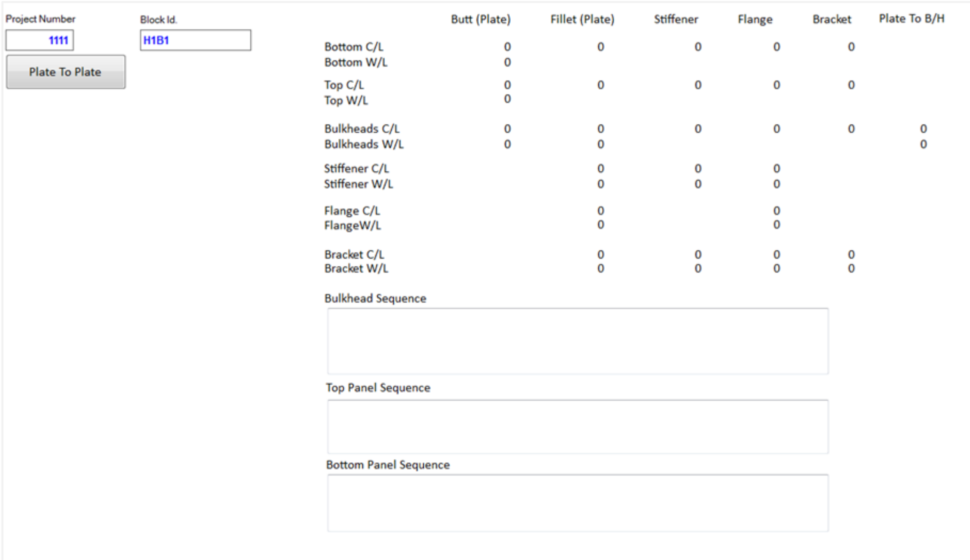


Figure 7-6: ‘Screen Shot’ of Program to Test the Model

The program developed to test the model reads data from a database. The data was extracted from the candidate shipyard’s database. The schematic of this is in Figure 7-1.

Figure 7-6 is a screen shot of the program after the project and block used to test the model have been selected. The project number is 1111 and the block identity is H1B1. Both these numbers are from the original NUPAS imported data.

The connectivity between parts was validated a step at a time. This can be seen in the first *button* displayed in Figure 7-6, *Plate To Plate*. This builds the plate-to-plate set connectivity. Figure 7-7 shows the program screen after all connectivity has been set up. The connectivity is set up one stage at a time. This was done to facilitate checking during the development of the model. That is, the connectivity follows the sequences of button shown on the left in Figure 7-7, beginning with plate-to-plate, followed by stiffener-to-plate and finishing with bracket-to-bracket. In a working environment, all connectivity relationships would be created at the instance the block is selected.

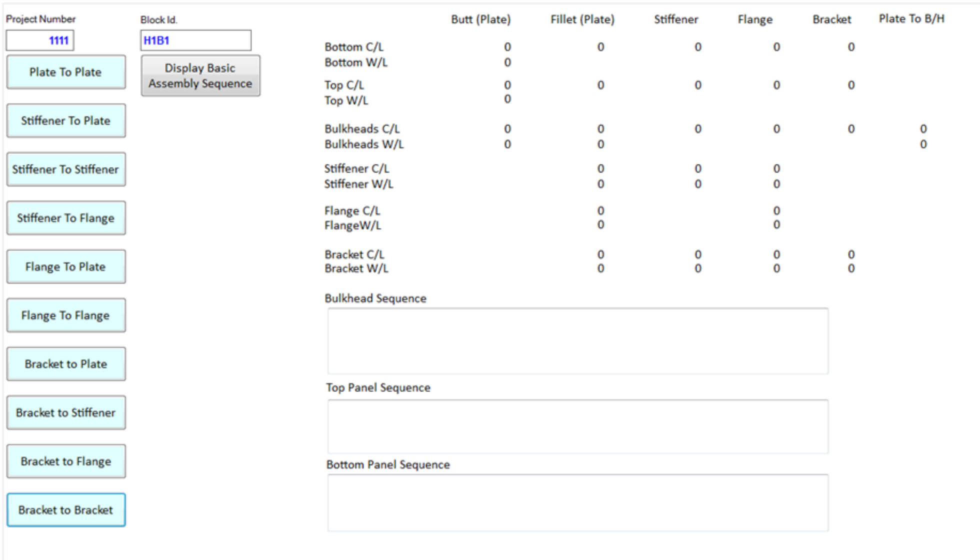


Figure 7-7: 'Screen Shot' Following Establishment of All Connectivity

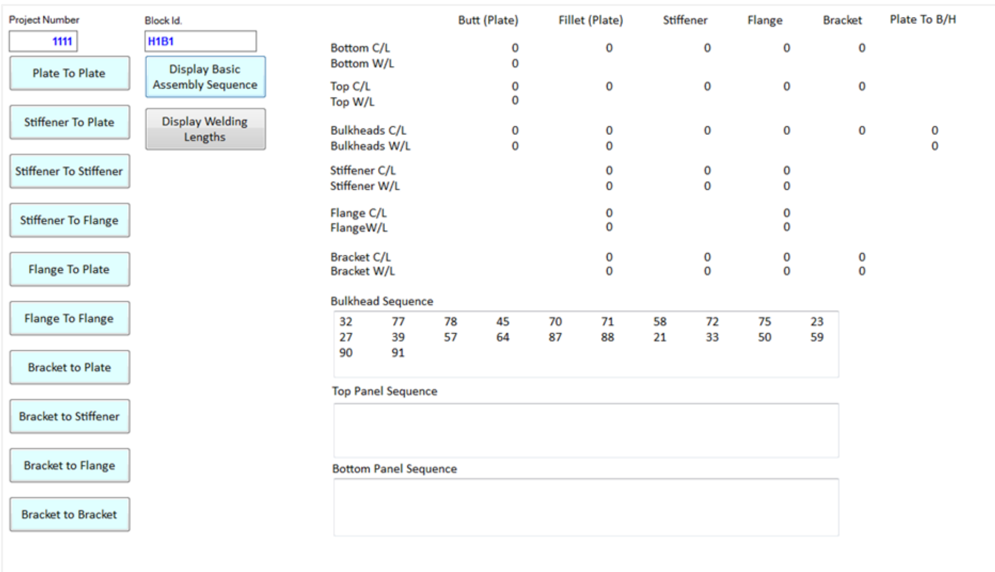


Figure 7-8: Model with Bulkhead Assembly Sequence Displayed

Following the establishment of the connectivity of all parts, the model has all the information that it requires to display the basic assembly sequence, which is shown in Figure 7-8. Note the assembly sequence is that of the sub-blocks/subassemblies, (referred to Bulkhead Sequences in the model.) The numbers shown refer to the unique reference given to the parts. These plates and their associated numbers are shown in Figure 7-9.

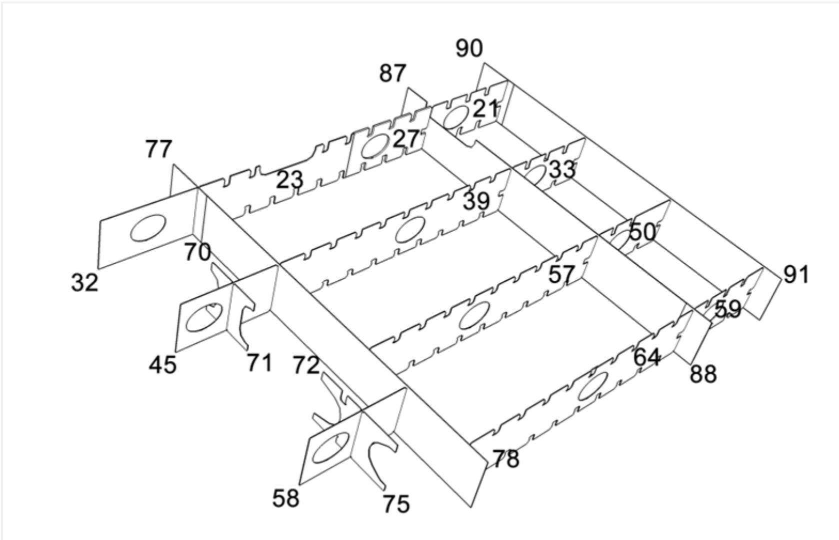


Figure 7-9: Vertical Plate with Relative Set Index Values Shown

Similarly, the top and bottom plate sequence is display using the unique sequence numbers.

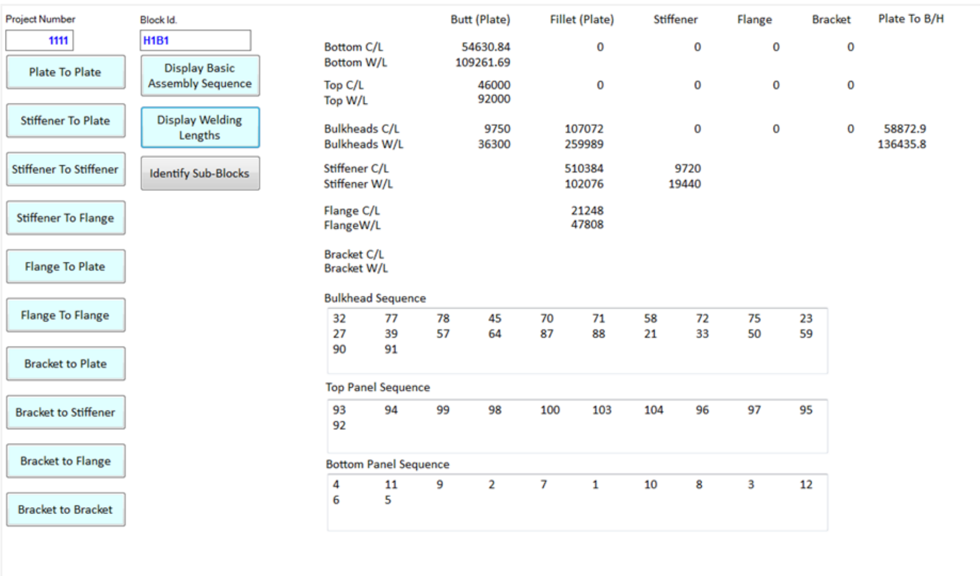


Figure 7-10: Program Output with Top and Bottom Panel Sequences

The bottom plate sequence list shown in Figure 7-10, beginning with 4, 11 and ending with 6, 5 compares exactly with the sequence that would be expected from examining the plates in Figure 7-11. Similarly, the top panel sequence list shown in Figure 7-10, beginning with 93, 94 and ending with 95, 92 compares exactly with the sequence that would be expected from examining the plates in Figure 7-12.

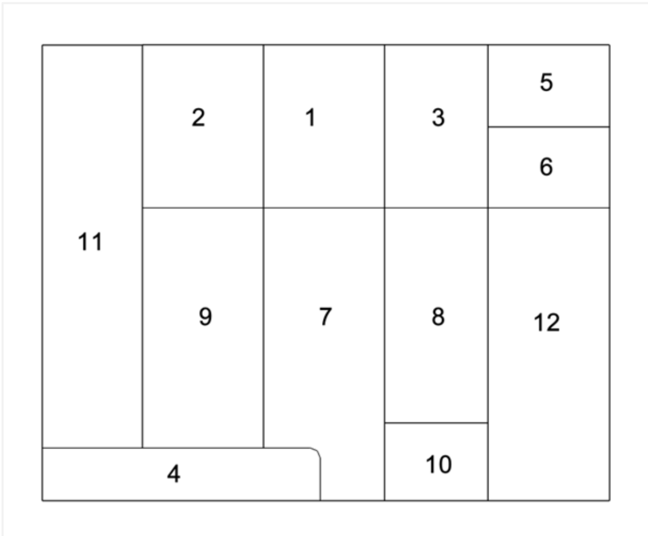


Figure 7-11: Bottom Plates Showing Unique Keys

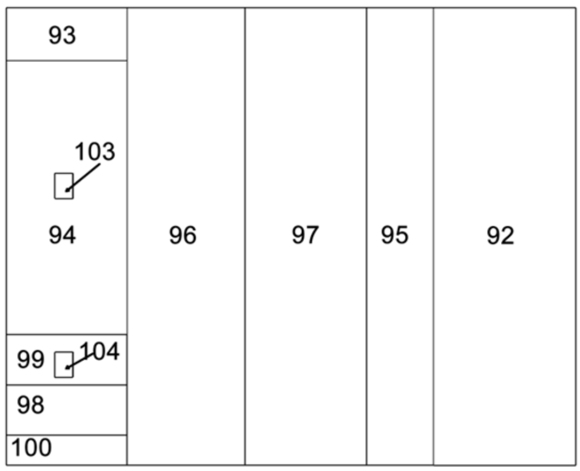


Figure 7-12: Top Plate Showing Unique Keys

Figure 7-13, shows the plates that are to be combined to form a sub-block prior to their assembly on the base panel. Bulks 32 and 77 are combined to form a sub-block. Also, plates 45, 70 and 71 form another sub-block, with plates 58, 72 and 75 forming the final sub-block. All other bulkheads are subassemblies, as they are not welded to any other plate prior to being assembled in their final location on the base panel.

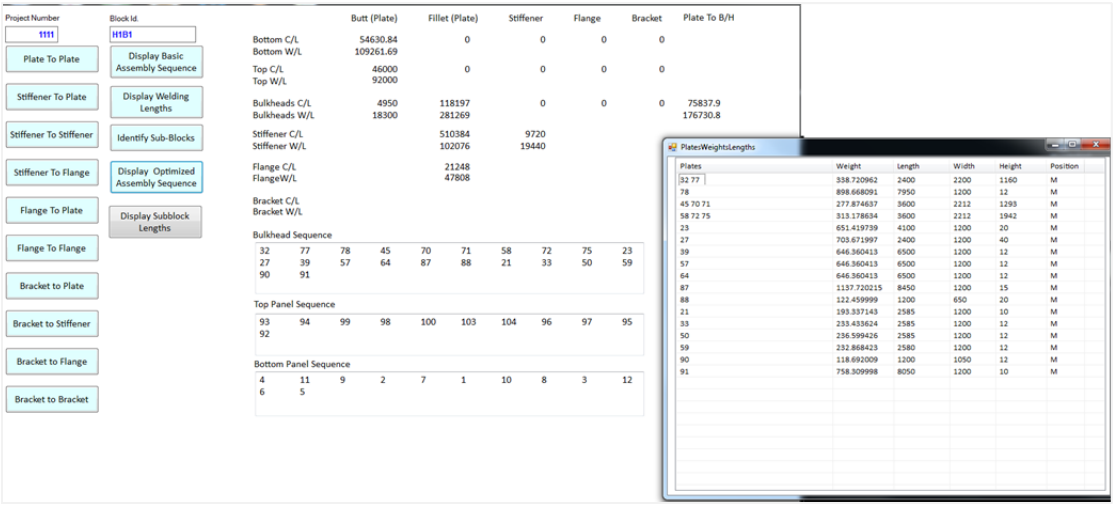


Figure 7-13: Plates, Sub-Blocks, Weights and Dimensions

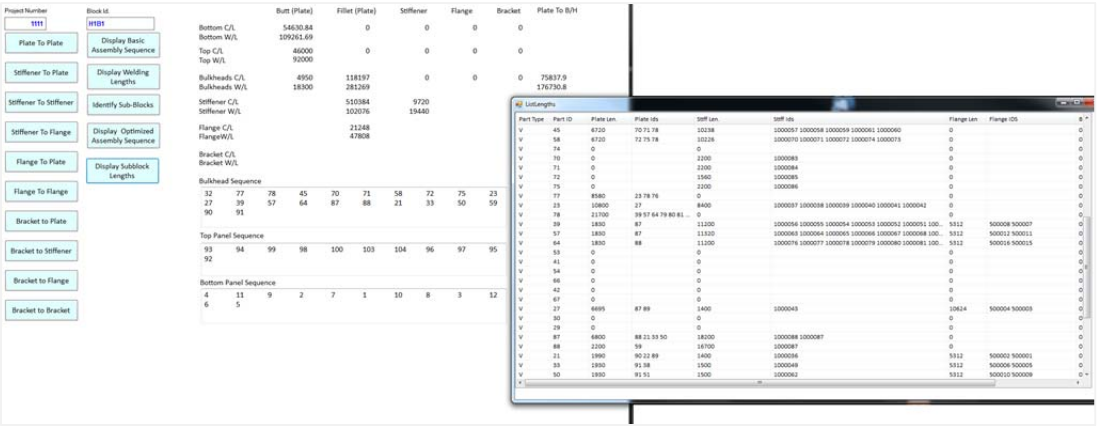


Figure 7-14: Details of Connectivity and Welding Lengths

Figure 7-14, shows the plates in the block and to what they are connected. The top line shows that the plate with the unique identity of 45 is connected to plates whose unique identities are 70, 71, 78 and stiffeners whose unique identities are 1000057, 1000058, 1000059, 1000061 and 1000060. There is no double accounting in the sense that if a plate with a unique identity of 99 is connected to another plate with a unique identity 405, the lengths are shown for only one of the plates. The other plate will show a length of zero.

7.5. Sub-Block Construction Schedule

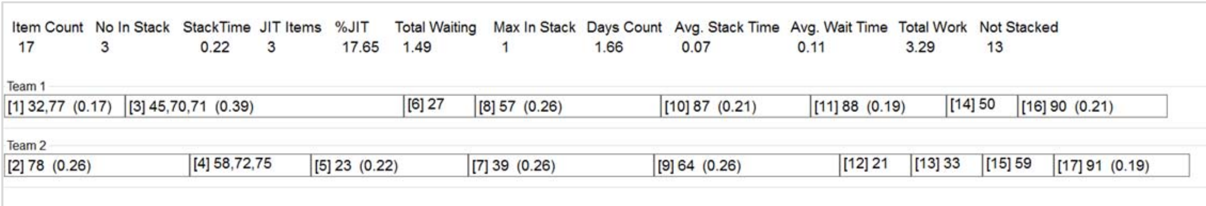


Figure 7-15: Greedy Only (2 Teams)

Figure 7-15 is a screen shot of the “Greedy Only” sequencing program described in chapter 6. All other screen shots of the variations of the Greedy algorithm with 2, 3 and 4 teams and the GA with 2, 3 and 4 teams have been placed in the appendices as Figure A-1 to Figure A-12. The data is that of the bulkheads as shown in Figure 7-9. All time values are in days. The results for each algorithm with both 2 and 3 teams being used are shown. The figures below show the order in which sub-blocks are constructed and the team to which the work of construction is allocated. The numbers in each box show the unique plate identities. For example, in Figure 7-15 for team 1, the first piece of work is the construction of the sub-block that consists of plates 32 and 77. The amount of time calculated to complete this work is 0.26 days.

The calculation of work is based on the welding length and the method of welding. For welding lengths greater than 1000 mm, it is assumed that an automatic welding machine is to be used with a welding speed of 300 mm/minute. For welding lengths less than or equal to 1000 mm, it is assumed that the welding will be carried out manually with a welding speed of 200 mm/minute. Setup time is taken to be 1 minute/300mm length plus 1 minute. After welding, some grinding is often required. 5 minutes is added for each piece that is welded to allow for grinding.

In these calculations, the delivery time of parts to the point of construction is not considered, neither is any re-work that is necessary for poorly welded joints.

Table 7-2: Summary of Makespan for Figure A-1 to Figure A-12 in Days

Algorithm	2 Teams	3 Teams	4 Teams
Greedy Only	1.66	1.15	0.91
Greedy + Post Reordering	1.66	1.15	0.91
Greedy + Backtracking	1.73	1.21	0.91
Greedy + Backtracking + Post Reordering	1.73	1.21	0.91
Genetic Algorithm	1.65	1.10	0.83

Table 7-2 summarizes the makespan for the model data using the four versions of the Greedy algorithm and the GA whose screen shots may be seen in the appendices, Figure A-1 to Figure A-12. The GA, even on this small sample set, proves to be the algorithm that has the optimum makespan. The backtracking algorithm suffers when the last few items have long construction durations. For 4 teams, it does not matter what version of the Greedy algorithm is used; they all yield the same makespan. This is no doubt due to the small data set.

Table 7-3:Sub-Block Welding

Combined Parts	Welding Length	Off-Block Timing	On-Block Timing
32, 77	2400 mm	8 minutes	24 minutes
45,70,71	2212 mm	7.4 minutes	22.1 minutes
58,72, 75	2212 mm	7.4 minutes	22.1 minutes

The combining of subassemblies into sub-blocks reduces vertical on-block welding. Table 7-3 gives a comparison of sub-block welding off-block using automatic welding machines and manual vertical welding on-block. The savings in time, in man-days, are clear. The welding lengths are taken from Figure 7-13.

The base panel and the top panel of the block have construction times of 1.55 days and 1.31 days respectively. These are constructed in the union-melt and panel line before being moved to the block assembly position. The construction teams shown in appendices Figure A-1 to Figure A-9 did not carry out this work; hence it does not appear there. The union-melt setup time for each panel pair is taken as 15 minutes. The welding time for each panel pair is taken as 280 mm/minute.

The stiffeners on the top and bottom panels are welded up to 14 at a time. In calculating the welding time, the stiffener setup time is 1 minute for each 300mm contact length. The welding speed of the welding machine is 300mm/minutes. Even though the stiffeners are welded in parallel time, not all welding machines start at the same time. A very conservative one-minute time lag between starting each welding machine is added for the 2^{nd} to the last stiffener. If, for example, there are 10 stiffeners to be welded in parallel, all of equal length and it takes 7 minutes to weld one stiffener, the total time-lapse time to weld 10 stiffeners is 7+9 = 16 minutes. Grinding time is taken as 1 minute for each 1000mm of contact length.

7.6. Summary

This chapter has shown that given only the topological data together with the part type for each part in a block, a database can be constructed that is highly efficient for using Polychromatic set theory to build connectivity relationship and calculate the contact lengths of parts.

Using an algorithm, the sequence of assembly has been identified. Using the sequence of assembly, the optimal combination of sub-blocks/subassemblies was identified for construction prior to being assembled on the base panel. The benefits of this are, (1) reduction of unnecessary work by reducing on-block welding through the construction of sub-blocks, (2) reduction of part moving within the construction yard. Part movements, from the construction point to the assembly point, disrupt work along the movement path. Combining 3 subassemblies into 1 sub-block automatically reduces such disruptions by two-thirds. Combining subassemblies into sub-blocks improves the productivity in non-related areas, thus eliminating unnecessary waste.

The total welding length was calculated from the contact length, the joint type and plate thickness. The amount of work required to construct a sub-block/subassembly was calculated from the welding length. A schedule for the working among or between the teams was calculated using a Genetic or Greedy algorithm.

Chapter 8. Conclusion and Recommendations

This chapter summarises the work covered in the earlier chapters, giving the current conclusions of the research. This work is by no means the end of the research as it can clearly be extended to all upstream scheduling and can form part of a plan to move the construction of offshore rigs towards Lean construction.

8.1. Summary of the Research

The research, as described in the preceding chapters, introduced the necessity of correct part sequencing for an efficient and effective assembly process. Most research for assembly sequencing has focused on the production process rather than any construction process. The production process is a repetitive process, whereas the construction, certainly of offshore rigs, is anything but repetitive. Production processes have repetitive parts in repetitive sequence at fixed time intervals. The nearest to a repetitive process in the construction of offshore rigs is the welding process. This is not to imply that all welding is the same. There are different types of welds: butt and fillet. There are different orientations of welding: horizontal, vertical and overhead. There are different methods of welding: manual and automatic. The combination of sub-blocks/subassemblies and the configuration of them that is required to construct a block is often unique to an individual offshore rig.

The theory of Polychromatic set has gained increasing attention since its introduction by V.V.Pavlov in 1994. Polychromatic set theory has been applied to a diverse set of engineering and industrial problems. The theory was introduced in chapter 3 with an illustration on how it can be used, using an example of notebook computers and their configuration.

Chapter 4 described how Polychromatic set theory was used in the building of a knowledge base and how this was used in the reasoning engine to identify part connectivity and part relationships and in the sequencing engine to build the plate assembly sequence.

The knowledge base is fundamental to this work. The reasoning engine uses the knowledge base to reason the topological relationships between the different part types that compose an offshore rig's block. Using these reasons, results of the topological relationships and connection modes, butt or fillet, a heuristic was developed to identify the assembly sequence of the plates, which are the base for subassemblies. Knowing the assembly sequence of the plates automatically is the same as knowing the assembly sequence of the subassemblies. Sub-blocks are a combination of subassemblies. The sub-blocks/subassemblies are constructed prior to them being assembled in their final position on the block.

Chapter 5 discussed the calculation of the welding length, which is more than just the part-to-part contact length. The calculation of the part-to-part contact length was discussed at length, covering the multiplicity of part-to-part connection combinations. The proper calculation of part-to-part welding length is important for the calculation of work content. Knowing the work content is necessary for calculating a sub-block/subassembly construction schedule.

Chapter 6 covered the inference engine and optimizing engine. There was discussion on several algorithmic possibilities for creating an optimized construction schedule for the sub-blocks/subassemblies in order to achieve a minimum makespan for the construction, as well as attempting to ensure that the sub-blocks/subassemblies arrive JIT or as near to JIT as possible at their required assembly point.

The inference engine identified the plates that compose a panel with its associated stiffeners. The panels and stiffeners are assembled and welded in the panel line area of the shipyard.

The plate assembly identification, which is produced by the sequencing engine, together with the subassembly identification, which is a product of the inference engine, are used by the optimizing engine to identify which subassemblies, subject to the local constraints, may be combined to make a sub-block. One of the objectives of this research has been to reduce the on-block welding of sub-blocks/subassemblies, since this will usually be a vertical weld and vertical welding takes twice as long as horizontal welding. By suitably combining sub-blocks/subassemblies prior to them being located in their final assembly position, the construction time of the whole block can be reduced. The combination of the subassemblies into sub-blocks uses the connectivity of parts from the reasoning engine. Beginning with a single subassembly, other subassemblies, in their construction sequence, which is from the sequencing engine, are added until local constraints are exceeded. All additions up to, but not including the subassembly that forced a violation of the local constraints, form a sub-block. The process is repeated with the subassembly that violated the local constraint. This is in line with the Lean approach of the research of reducing unnecessary work.

Welding is a lengthy process. It is the process that consumes the largest amount of manpower in the construction process. To know the welding length required to construct any sub-block/subassembly allows for the calculation of the amount of work required to construct the component, and hence facilitates for the effective scheduling to optimize the work. The welding length is not the same as the contact length. The welding length is a multiple of the contact length and is a function of the thickness of the steel being welded as well as the length being welded. The combinations of part connections is large. Chapter 5 discusses these combinations in detail and how, initially, the contact length and, finally, the welding length are derived.

Chapter 6 compared and contrasted a number of algorithms that have the possibility of being used to schedule the work of assembling the sub-blocks/subassemblies. The concept of parallel machines has been adopted for the work allocated to the teams of workers who would construct the sub-blocks/subassemblies. As has been highlighted, optimizing scheduling for parallel machines is an *NP*-hard problem and some researchers suggest the use of a greedy algorithm. The problem at hand is more constrained than some shop scheduling problems in that one of the driving factors here is JIT. The assembly sequence of the sub-blocks/subassemblies on the block has been defined by the heuristic developed in chapter 4. The greedy algorithm, which was used to minimize the makespan, was modified by using backtracking and, after the final sequence was derived, the use of reordering to minimize double handling was introduced. By way of comparison, a genetic algorithm whose objective was to minimize the makespan was developed. The GA produced a shorter makespan than the greedy algorithms. As post reordering was implemented in the GA, the issue of double handling, which would have been high, was dealt with.

The case study used a real block taken from a project used by the candidate shipyard. The block was considered complex enough to test the heuristic and scheduling algorithms. The results showed that it was possible to take a block from the design engineer's database and, without any further manual additions or changes to the data, produce a construction and schedule that minimizes the makespan for the block assembly and reduces waste in the process.

8.2. Main Contribution

The contribution of this research is the whole framework that was developed to solve the problem of sub-block/subassembly sequencing. The framework starts with only digital data that describes the parts in 3D. The digital data, an example of which may be seen in Figure 7-2, is in initially in an XML format. The XML data is read and loaded to an SQL database in a format that is more suitable for process. The information required from the XML data is minimal and is adumbrated below.

- Part Unique Identity
- Part Type (Plate, Stiffener, Flange, Bracket)
- Part Weight
- Part Thickness
- Part Extrusion Vector
- Part Length
- Part Profile (series of 3D vertices)

Using just this minimal data, the developed framework uses Polychromatic sets to construct a knowledge base of parts (plates, stiffeners, flanges and brackets). The knowledge base is interrogated by a reasoning engine using only the basic part vertices, thickness and extrusion direction vector. The reasoning engine builds and augments the knowledge base with the parts connectivity and topological relationship with other parts in the set.

The part connectivity and relationship between parts that have been derived from the minimal data is used by the inference engine. The plate is the basic unit of connectivity. By using the plate as the basic unit of connectivity, this research has reduced the complexity of the problem, firstly, by identifying base and, if they exist, top panels as being composed of a series of connected plates and secondly, for other plates, by identifying the subassemblies. Once connectivity and relationship has been identified, the welding length can be calculated.

The research was able to derive a formula, based on observed data, for the calculation of the welding length, which depends on the joint type and the connection type (butt or fillet). The expected duration of the welding process is also calculated. This is based on either the average manual welding speed or the speed of the automatic welding machine. The choice of manual or automatic is dependent on the length of weld, as the automatic welding machine requires a minimum length for operation.

The sequencing engine has, at its heart, sequencing heuristic. The heuristic uses the connectivity, part relationship to identity the plate assembly sequence. The brackets, flanges and stiffeners are also associated with a plate; hence, the necessary welding length can be calculated.

The optimizing engine adopts algorithmic process when it reads the data from the sequencing engine to identify sub-blocks from the set of subassemblies such that the sub-blocks do not violate the physical constraints of the construction area in terms of weight and size and concomitantly minimizes on-block vertical welding.

The construction sequence of the now optimized assembly sequence is optimized such that there is a minimum of waiting time at the assembly point for the ‘next’ piece to be assembled. This optimization is achieved by use of a Genetic Algorithm (GA). The result of the GA is further enhanced by a process referred to as Post Ordering.

This reduces and probably eliminates the double handling of the parts. This does not affect the makespan, but it does eliminate unnecessary work, which is in line with Lean principles.

8.3. Future Work

This work can be seen as an important step to creating a more efficient and lean production process for offshore rigs. Starting with the basic building block of automatically calculating the welding length from a design engineer's electronic drawings, and hence the calculation of the work required to produce the block. From the same engineer's electronic drawing, the construction sequence of a block can be formulated. With both the work content and the sequence of sub-block construction, a good schedule based on real values can be generated for the whole block. What works for one block, works for all blocks on the offshore rig. A realistic construction schedule based on real values that are intrinsic to each block may be derived. As was seen in chapter 6, with the comparison of the Greedy and Genetic Algorithm, there will always be periods of time when those fitters working on the block assembly have to wait for the sub-blocks to be constructed. This is not a problem but an opportunity to redeploy the teams to other blocks. A good schedule for the whole rig construction will redeploy workers when and where necessary to optimize the use of labour and thus improve productivity.

What the part is to the block, the block is to the rig. The work sequencing algorithms for sub-blocks/subassemblies can be used for whole blocks, where teams are replaced by work assembly areas. In this way, the whole rig can be scheduled for construction.

This research only considered areas shown in the dotted ellipse, as shown in Figure 8-1. To have a schedule that takes advantage of Lean principles, the ellipse can be extended to include cutting and bending. It was mentioned that cutting software such as *nestix* is not tied to a schedule and will just produce cut parts that maximizes usage of the steel plate. It is possible to give a block construction schedule to the engineer based on the derived schedules for the rig construction and the engineer can then use this so that the cutting of parts is more closely related to the schedule. In *nestix*, for example, an engineer would list the blocks for which *nestix* is to produce a cutting plan for the parts.

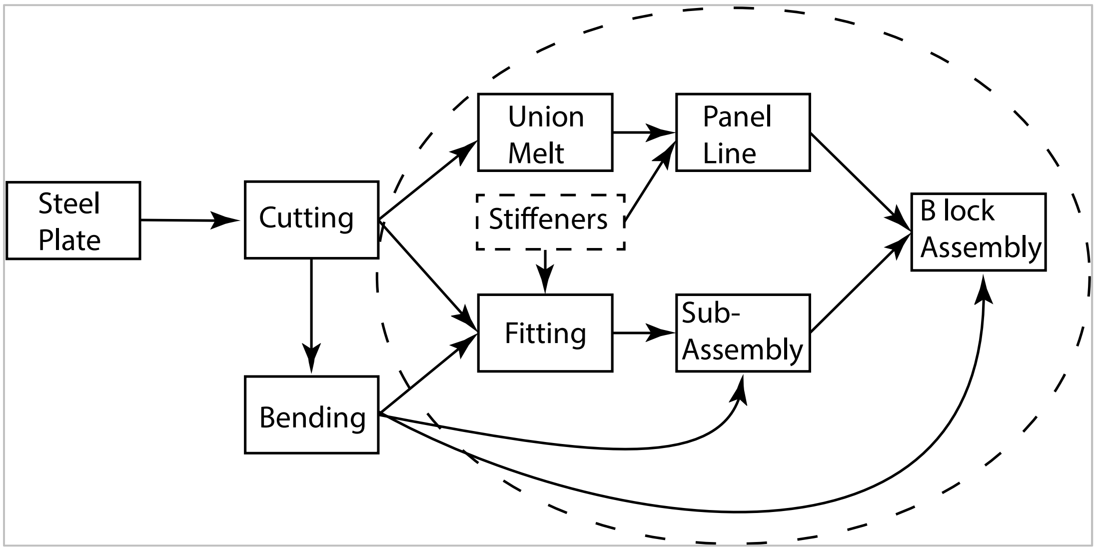


Figure 8-1: Area of Research

Bibliography

Abbott, E., and Chua, D., K. H.; "The Application of Polychromatic Set Theory in the Assembly Sequencing of Blocks for Offshore Rigs." *Proc., Proceedings of the 4th International Conference on Engineering, Project, and Production Management (EPPM 2013)*, 15.

Åhlström, P. (1998). "Sequences in the Implementation of Lean Production." *European Management Journal*, 16(3), 8.

Bai, Y. W., Chen, Z. N., Bin, H. Z., and Hun, J. (2005). "An effective integration approach toward assembly sequence planning and evaluation." *International Journal of Advanced Manufacturing Technology*, 27(1-2), 96-105.

Biskup, D., Herrmann, J., and Gupta, J. N. D. (2008). "Scheduling identical parallel machines to minimize total tardiness." *International Journal of Production Economics*, 115(1), 134-142.

Blazewicz, J., Lenstra, J. K., and Rinnooy Kan, A. H. G. "Scheduling subject to resource constraints: classification and complexity." *Proc., Combinatorial Optimization Conference, 24-28 Aug. 1981*, 11-24.

Bonneville, F., Perrard, C., and Henrioud, J. M. "A genetic algorithm to generate and evaluate assembly plans." *Proc., Proceedings 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation. ETFA'95, 10-13 Oct. 1995*, IEEE Comput. Soc. Press, 231-239.

Bourjault, A. (1984). "Contribution a une approche mathodolgique de l'asemblage automatise: elaboation automatique des sequences operatoires (Contribution to a systematic approach of automatic assembly: automatic determination of operation sequences)." PhD, Universite de Franche-Comte, Besancon, France.

Cagliano, R., Caniato, F., and Spina, G. (2004). "Lean, Agile and traditional supply: how do they impact manufacturing performance?", 10(4-5), 151-164.

Chakrabarty, S., and Wolter, J. (1997). "A structure-oriented approach to assembly sequence planning." *IEEE Transactions on Robotics and Automation*, 13(1), 14-29.

Chen, B., Potts, C. N., and Woeginger, G. J. (1988). *A Review of Machine Schedling: Complexity, Algorithms and Approximations*, Kluwer Academic Press.

Chen, S.-F., and Liu, Y.-J. (2001). "An adaptive genetic assembly-sequence planner." *International Journal of Computer Integrated Manufacturing*, 14(5), 489-500.

Chen, Y., Liu, Y., Liu, J., and He, F. (2011). "Design of PDM multi-version management model based on polychromatic sets theory." *Advanced Materials Research*, 271-273, 28-33.

De Fazio, T. L., and Whitney, D. E. (1987). "SIMPLIFIED GENERATION OF ALL MECHANICAL ASSEMBLY SEQUENCES." *IEEE journal of robotics and automation*, RA-3(6), 640-658.

Diaz-Gomez, P. A and Hougen D. F. (2007) "Initial population for genetic algorithms: a metric approach." *International Conference on Genetic and Evolutionary Methods (GEM '07),* p 43-9.

Dini, G., and Santochi, M. (1992). "Automated sequencing and subassembly detection in assembly planning." *CIRP Annals - Manufacturing Technology*, 41(1), 1-4.

Dong, T., Tong, R., Zhang, L., and Dong, J. (2007). "A knowledge-based approach to assembly sequence planning." *International Journal of Advanced Manufacturing Technology*, 32(11-12), 1232-1244.

Gao, X., and Li, Z. "Computer aided conceptual design of mechanical product using polychromatic sets." *Proc., 2006 IEEE International Conference on Mechatronics and Automation, ICMA 2006, June 25, 2006 - June 28, 2006*, Inst. of Elec. and Elec. Eng. Computer Society, 1169-1174.

Gao, X., and Wang, X. (2013). "Special net structure and its application in workflow modeling." *International Journal of u- and e- Service, Science and Technology*, 6(4), 127-138.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979). "Optimization and approximation in deterministic sequencing and scheduling: a survey." *Discrete Optimisation, Aug. 1977*, II, 287-326.

Guo, S., and Lan, Y. "A formal evaluation approach to UML state model based on polychromatic sets." *Proc., 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), 12-14 Oct. 2008*, IEEE, 4 pp.

Guo, X., Ma, L., and Shi, Y. "Construct assembly model for assembly sequence planning." *Proc., 2010 IEEE International Conference on Information Theory and Information Security, 17-19 Dec. 2010*, IEEE, 25-29.

Heckbert, P. S. (1994). *Graphics gems IV*, AP Professional, Boston.

Henrioud, J. M., Relange, L., and Perrard, C. (2002). "Generation of precedence hypergraphs for assembly system design." *IFAC 15th World Congress*, 15(1), 6.

Herron, C., and Hicks, C. (2008). "The transfer of selected lean manufacturing techniques from Japanese automotive manufacturing into general manufacturing (UK) through change agents." 24(4), 524-531.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT Press, Cambridge, Mass.

Homem de Mello, L. S., and Sanderson, A. C. (1991). "A correct and complete algorithm for the generation of mechanical assembly sequences." *IEEE Transactions on Robotics and Automation*, 7(2), 228-240.

Horowitz, E., and Sahni, S. (1976). "Exact and approximate algorithms for scheduling nonidentical processors." *Journal of the Association for Computing Machinery*, 23(2), 317-327.

Jin, C., Wang, Y., Zhang, W., and Lin, Y. "Study on semi-finished ship structural components assembly sequence optimization." *Proc., 2010 6th International Conference on Natural Computation, ICNC'10, August 10, 2010 - August 12, 2010*, IEEE Computer Society, 2706-2709.

Karger, D., Stein, C., and Wein, J. (2010). *Algorithms and theory of computation handbook*, Chapman & Hall.

Kim, H., Lee, J.-G., Lee, S.-S., and Park, J. H. "A simulation-based shipbuilding system for evaluation of validity in design and manufacturing." *Proc., System Security and Assurance, October 5, 2003 - October 8, 2003*, Institute of Electrical and Electronics Engineers Inc., 522-529.

Kolic, D., Fafandjel, N., and Zamarin, A. "Lean Manufacturing Methodology for Shipyards." *Proc., The 20th Symposium on Theory and Practice of Shipbuilding*, 12.

Kolich, D., Storch, R. L., and Fafandjel, N. "Lean manufacturing in shipbuilding with Monte Carlo simulation." *Proc., International Conference on Computer Applications* *in Shipbuilding 2011, September 20, 2011 - September 22, 2011*, Royal Institution of Naval Architects, 159-167.

Krafcik, J., K. (1998). "Triumph of the Lean Production System." *Sloan Management Review*, 30(1), 12.

Lai, H.-Y., and Huang, C.-T. (2004). "A systematic approach for automatic assembly sequence plan generation." *International Journal of Advanced Manufacturing Technology*, 24(9-10), 752-763.

Lang, S., Dutta, N., Hellesoy, A., Daniels, T., Liess, D., Chew, S., and Canhetti, A. (2001). "Shipbuilding and Lean Manufacturing - A Case Study." *THE SOCIETY OF NAVAL ARCHITECTS AND MARINE ENGINEERS*.

Lee, K., and Gossard, D. C. (1985). "A hierarchical data structure for representing assemblies. I." *Computer Aided Design*, 17(1), 15-19.

Li, S., and Wang, X. (2012). "Polychromatic set theory-based spectrum access in cognitive radios." *IET Communications*, 6(8), 909-916.

Li, Y., Li, Z., and Liu, E. "Genetic algorithm method and model constrains based on polychromatic sets theory in sequencing machining step on a machining center." *Proc., 2009 IEEE International Conference on Automation and Logistics, ICAL 2009, August 5, 2009 - August 7, 2009*, IEEE Computer Society, 1812-1816.

Li, Z., and Xu, L. (2003). "Polychromatic sets and its application in simulating complex objects and systems." *Computers & Operations Research*, 30(6), 851-860.

Li, Z., and Xu, L. (2003). "Polychromatic sets and its application in simulating complex objects and systems." *Computers & Operations Research*, 30(6), 851-860.

Li, Z., Xu, L., and Zhao, S. (2006). "Polychromatic Sets Theory and its Applications in Enterprise Information Systems." *International Federtion for Information Processing*, 205, 8.

Liker, J. K., and Lamb, T. (2002). "What is lean ship construction and repair?" *Journal of Ship Production*, 18(3), 121-142.

Liu, H., Hong, J., Li, Y., and Li, Z. "Process model of vehicle body-in-white welding assembly based on polychromatic sets." *Proc., International Conference on Advanced Technology of Design and Manufacture (ATDM 2010), 23-25 Nov. 2010*, IET, 316-321.

Liu, Z., Chua, D. K. H., Abbott, E. L. S., and Yeoh, K.-W. (2010). "Optimization of Resource Leveling in Shipbuilding with Variation-Float TradeOffs." *6th International Conference of Innovation in Architecture, Engineering & Construction (AEC)*.

Liu, Z., Chua, D. K. H., and Yeoh, K. W. (2011). "Aggregate production planning for shipbuilding with variation-inventory trade-offs." *International Journal of Production Research*, 49(20), 6249-6272.

Lu, T., Zhang, B., and JIa, P. "Assembly sequence planning based on graph reduction." *Proc., Proceedings of TENCON '93. IEEE Region 10 International Conference on Computers, Communications and Automation, 19-21 Oct. 1993*, IEEE, 119-122.

Minzu, V., Bratcu, A., and Henrioud, J. M. "Construction of the precedence graphs equivalent to a given set of assembly sequences." *Proc., Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning (ISATP'99), 21-24 July 1999*, IEEE, 14-19.

Mosheiov, G. (2001). "Parallel machine scheduling with a learning effect." 52(10), 1165-1169.

Nawaz, M., Enscore Jr, E. E., and Ham, I. (1983). "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem." *Omega*, 11(1), 91-95.

Paez, O., Salem, S., Solomon, J., and Genaidy, A. (2005). "Moving from lean manufacturing to lean construction: Toward a common sociotechnological framework." *Human Factors and Ergonomics In Manufacturing*, 15(2), 233-245.

Parnaby, J. "COMBINING JAPANESE METHODOLOGY WITH TECHNOLOGY TO ACHIEVE MORE COMPETITIVE MANUFACTURING SYSTEMS." *Proc., Automation '86: The Plastics and Rubber Institute's Second International Conference on Competitive Manufacturing Systems - Polymer Processing.*, Plastics & Rubber Inst, 2. 1-2. 8.

Parnaby, J. (1988). "SYSTEMS APPROACH TO THE IMPLEMENTATION OF JIT METHODOLOGIES IN LUCAS INDUSTRIES." *International Journal of Production Research*, 26(3), 483-492.

Pavlov, V. V., Li, Z., and Gao, F. (2001). "Application of the polychromatic sets for modeling and analysis of manufacturing system." *Hsi-An Chiao Tung Ta Hsueh/Journal of Xi'an Jiaotong University*, 35(9), 926-929+986.

Salem, O., Solomon, J., Genaidy, A., and Luegring, M. (2005). "Site Implementation and Assessment of Lean Construction Techniques." *Lean Construction Journal*, 2(2).

Salem, O., Solomon, J., Genaidy, A., and Minkarah, I. (2006). "Lean construction: from theory to implementation." *Journal of Management in Engineering*, 22(4), 168-175.

Sinanoglu, C., and Borklu, H. R. (2005). "An assembly sequence-planning system for mechanical parts using neural network." *Assembly Automation*, 25(1), 38-52.

Staats, B. R., Brunner, D. J., and Upton, D. M. (2011). "Lean principles, learning, and knowledge work: Evidence from a software services provider." *Journal of Operations Management*, 29(5), 376-390.

Storch, R. L., and Lim, S. (1999). "Improving flow to achieve lean manufacturing in shipbuilding." *Production Planning and Control*, 10(2), 127-137.

Tapping, D., Luyster, T., and Shuker, T. (2002). "Value Stream Management." *Productivity Press, New York.*

Tianyang, D., Ruofeng, T., Ling, Z., and Jinxiang, D. (2007). "A knowledge-based approach to assembly sequence planning." *International Journal of Advanced Manufacturing Technology*, 32(11-12), 1232-1244.

Tsoy, Y. R. (2003). "*7th Korea-Russia International Symposium on Science and Technology*", vol 3 181-7.

Vikram;, M. V. S. R. K. T. S. N. D. (2013). "A Deterministic Approach For Routing Through Polychromatic Sets in Wireless Adhoc Networks." *International Journal of Innovative Research in Science, Engineering and Technology*, 2(11), 6.

Wang, S., Hong, J., Li, Y. L., Zhang, Y., and Li, Z. B. "Assembly planning of aircraft based on polychromatic sets." *Proc., IEEE International Conference on Industrial Engineering and Engineering Management, IEEM2010, December 7, 2010 - December 10, 2010*, IEEE Computer Society, 600-605.

Wang, X., and Li, S. (2013). "Scalable routing modeling for wireless Ad Hoc networks by using polychromatic sets." *IEEE Systems Journal*, 7(1), 50-58.

Warnecke, H. J., and Huser, M. (1995). "Lean production." *International Journal of Production Economics*, 41(1-3), 37-37.

Wilson, R. H. "Minimizing user queries in interactive assembly planning." *Proc., Proceedings of 1993 IEEE International Conference on Robotics and Automation, 2-6 May 1993*, IEEE Comput. Soc. Press, 322-327.

Womack, J. P., Jones, D., and Roos, D. (1991). *The Machine That Changed the World: The Story of Lean Produciton*.

Womack, J. P., and Jones, D. T. (1994). "From lean production to lean enterprise " *Harvard Business Review 74*(Mar-Apr), 21.

Xu, Z., Li, Y., Zhang, J., Cheng, H., Jiang, S., and Tang, W. (2012). "A dynamic assembly model for assembly sequence planning of complex product based on polychromatic sets theory." *Assembly Automation*, 32(2), 152-162.

Yalaoui, F., and Chengbin, C. (2002). "Parallel machine scheduling to minimize total tardiness." *International Journal of Production Economics*, 76(3), 265-279.

Yan, L., Jiang, L., and Li, Z. (2006). "A Disassembly Model based on Polychromatic Sets Theory for Manufacturing Systems." *International Federtion for Information Processing*, 205(Research and Practical Issues of Enterprise Systems.), 11.

Zhang, W. (1989). "Representation of assembly and automatic robot planning by Petri net." *IEEE Transactions on Systems, Man and Cybernetics*, 19(2), 418-422.

Zhang, Y., Li, Z., Xu, L., and Wang, J. (2011). "A new method for automatic synthesis of tolerances for complex assemblies based on polychromatic sets." *Enterprise Information Systems*, 5(3), 337-358.

Zhao, S., and Li, Z. (2008). "A new assembly sequences generation of three dimensional product based on polychromatic sets." *Information Technology Journal*, 7(1), 112-118.

Zhao, S., Li, Z., and Chang, N. "A new modelling method for design information based polychromatic sets theory." *Proc., IEEE International Conference on Robotics and Biomimetics - ROBIO2006, 17-20 Dec. 2006*, IEEE, 208-213.

Zhao, S., Li, Z., and Wu, F. "Assembly sequences algorithm of car body for car enterprise information system." *Proc., 2006 IEEE International Conference on Systems, Man and Cybernetics, October 8, 2006 - October 11, 2006*, Institute of Electrical and Electronics Engineers Inc., 401-406.

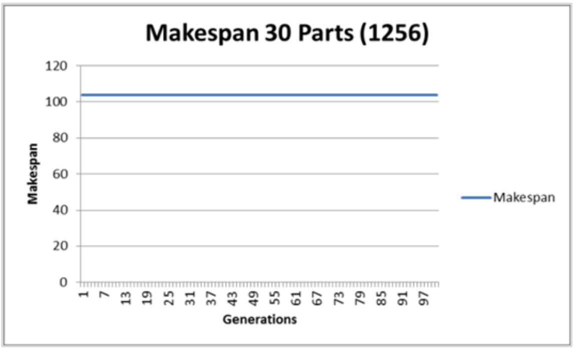
Zimmer, E., Salem, O., Genaidy, A., and Shell, R. (2008). "Case Study: Lean Supply Chain Management in Construction Projects." *16th Annual Conferencd of the International Group of Lean Construction Supply Chain Management*Manchester UK.

Zong, Z., Xi, X., Jiang, L., and Yang, B. "Manufacturability evaluation of auto panels using polychromatic sets." *Proc., 2009 International Asia Conference on Informatics in Control, Automation, and Robotics, CAR 2009, February 1, 2009 - September 2, 2009*, Inst. of Elec. and Elec. Eng. Computer Society, 372-375.

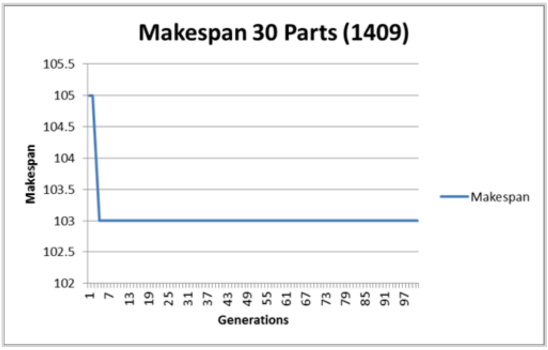
A. Appendices

A1 Genetic Algorithm Results for 3 Teams

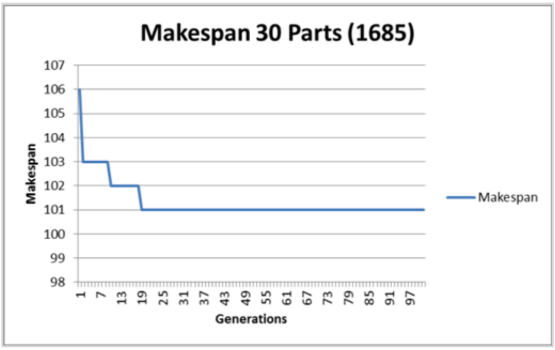
A1.1. Charts for 30 Parts



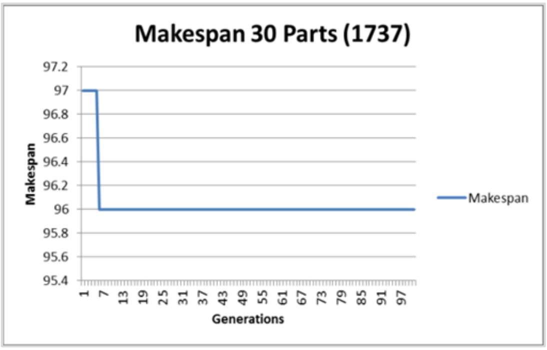
Makespan = 104, Optimum Makespan = 103.33



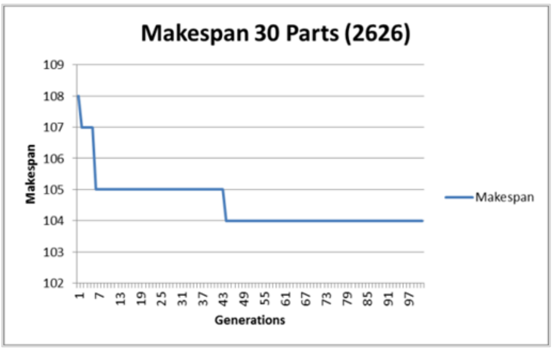
Makespan = 103, Optimum Makespan = 102.33



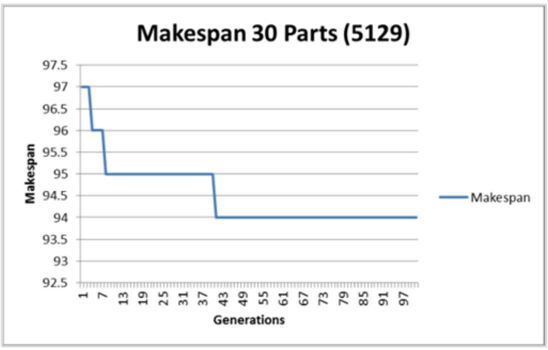
Makespan = 101,Optimum Makespan =100.67



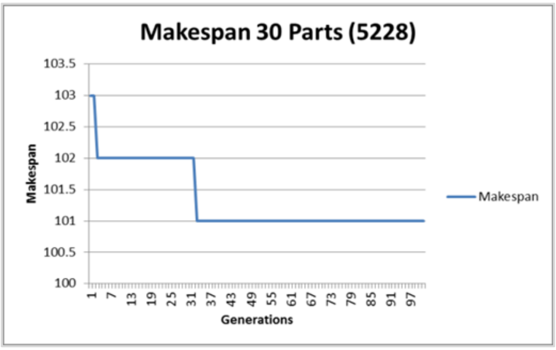
Makespan = 96, Optimum Makespan =95.67



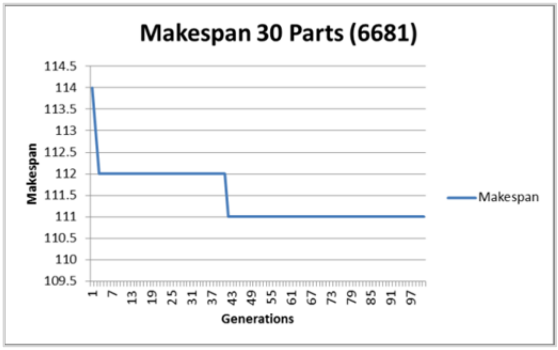
Makespan = 104, Optimum Makespan = 104



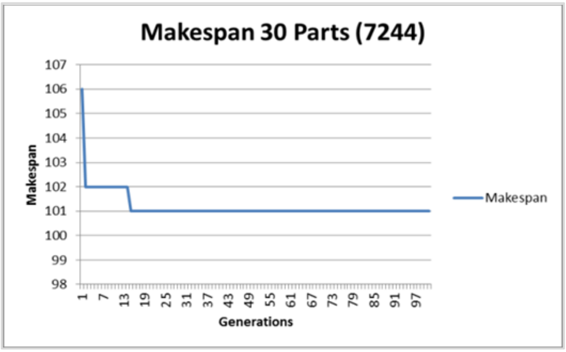
Makespan = 94, Optimum Makespan =94



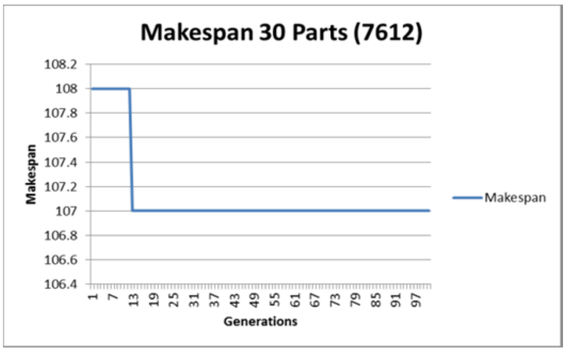
Makespan = 101, Optimum Makespan =101



Makespan = 111, Optimum Makespan =110.67

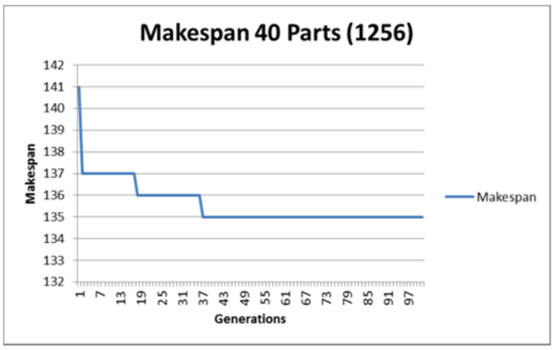


Makespan = 101, Optimum Makespan = 101

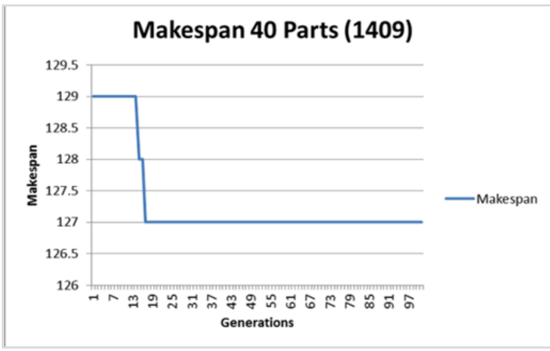


Makespan = 107, Optimum Makespan = 106.33

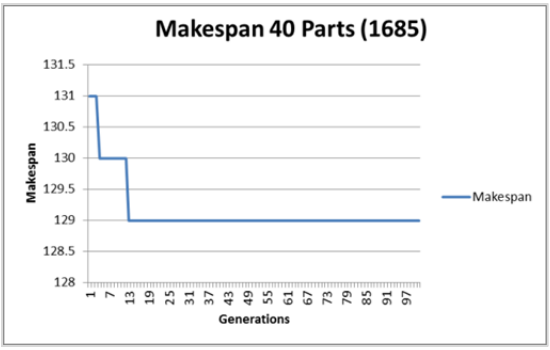
A1.2. Charts for 40 Parts



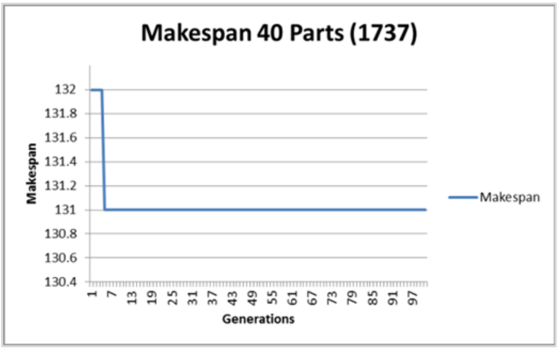
Makespan = 135, Optimum Makespan = 134.67



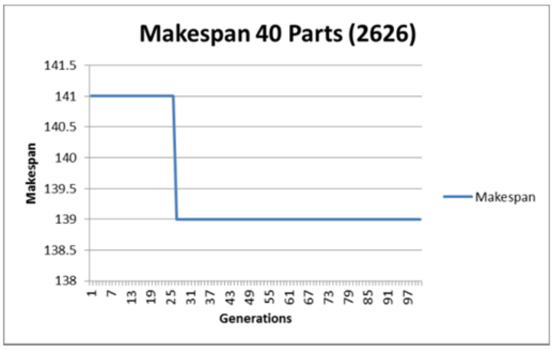
Makespan = 127, Optimum Makespan = 127



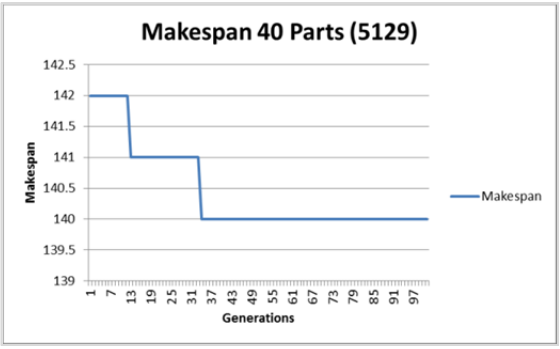
Makespan = 129, Optimum Makespan = 129



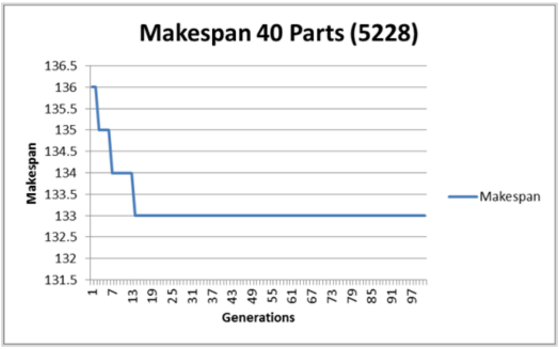
Makespan = 131, Optimum Makespan = 130.67



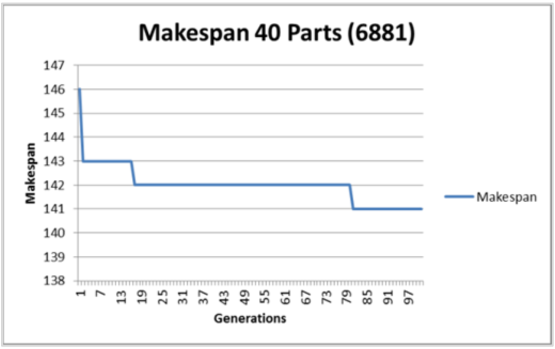
Makespan = 139, Optimum Makespan = 139



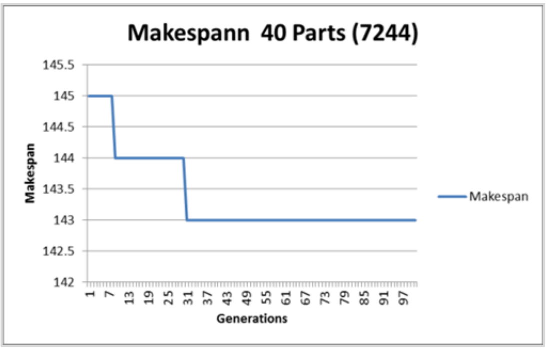
Makespan = 140, Optimum Makespan = 140.67



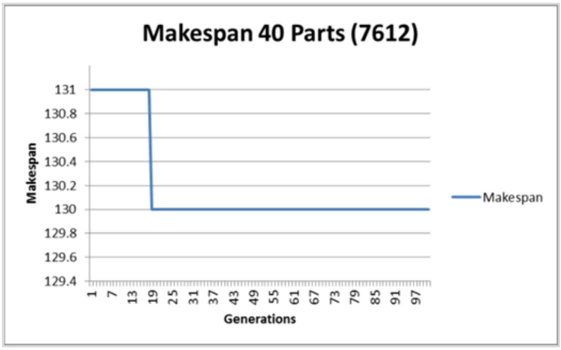
Makespan = 133, Optimum Makespan = 132.67



Makespan = 141, Optimum Makespan = 141

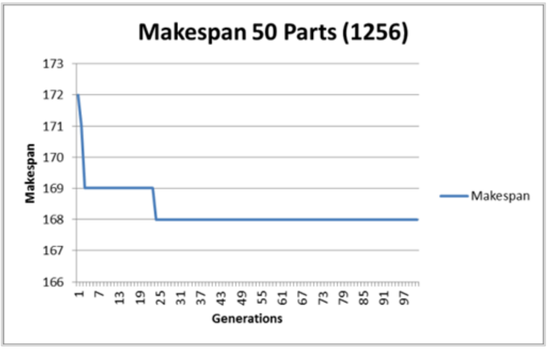


Makespan = 143, Optimum Makespan = 142.67

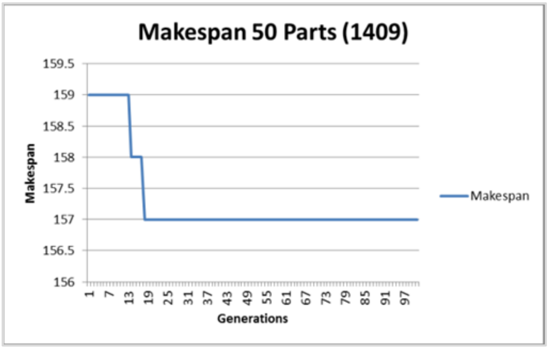


Makespan = 130, Optimum Makespan = 129.67

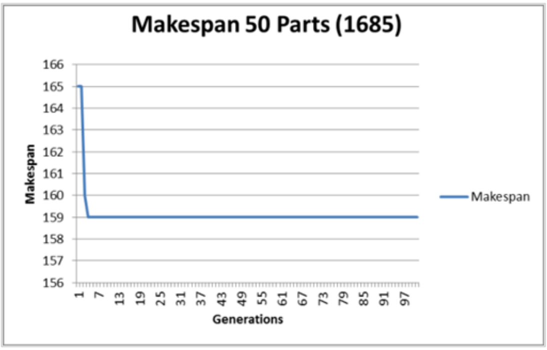
A1.3. Charts for 50 Parts



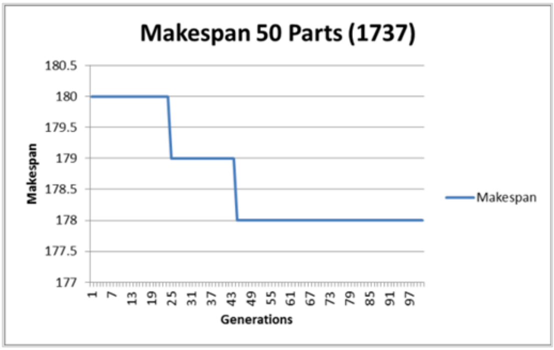
Makespan = 168, Optimum Makespan = 167.67



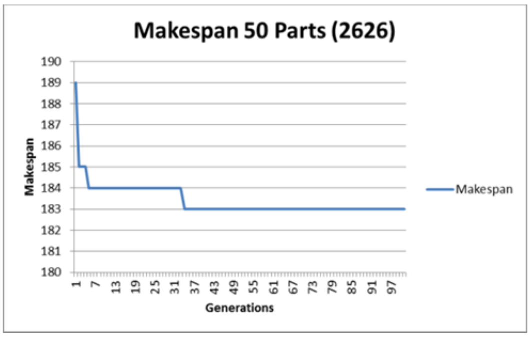
Makespan = 157, Optimum Makespan = 157.67



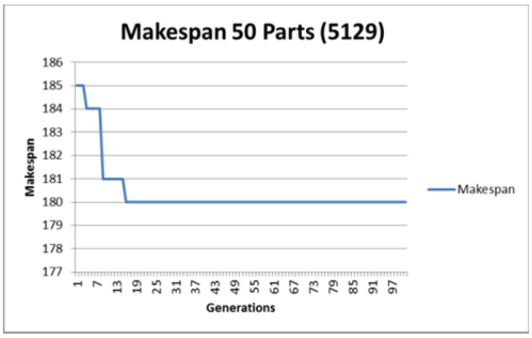
Makespan = 159, Optimum Makespan = 158.67



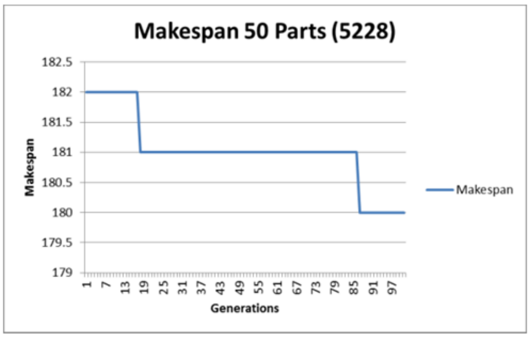
Makespan = 178, Optimum Makespan = 177.33



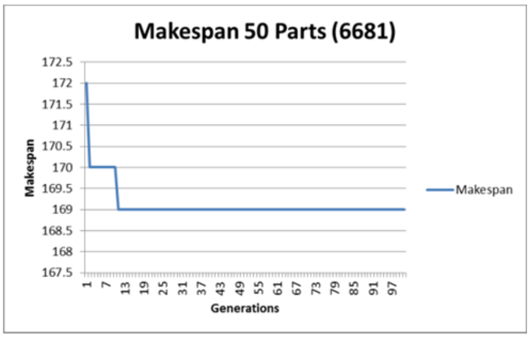
Makespan = 183, Optimum Makespan = 183



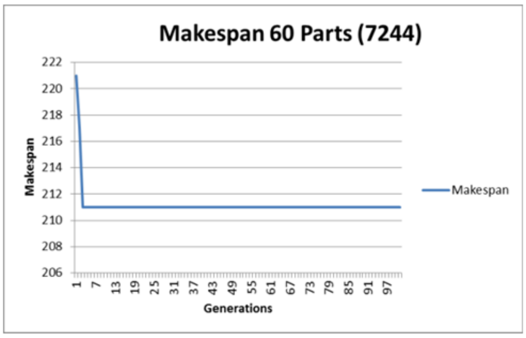
Makespan = 180, Optimum Makespan = 180



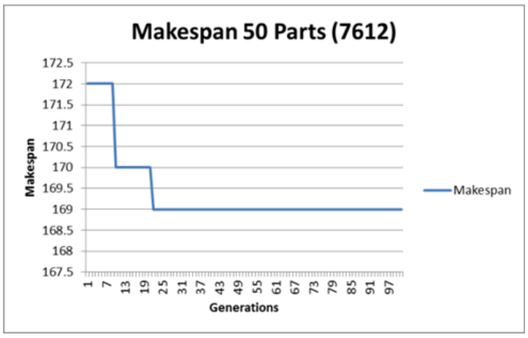
Makespan = 180, Optimum Makespan = 180



Makespan = 169, Optimum Makespan = 168.33

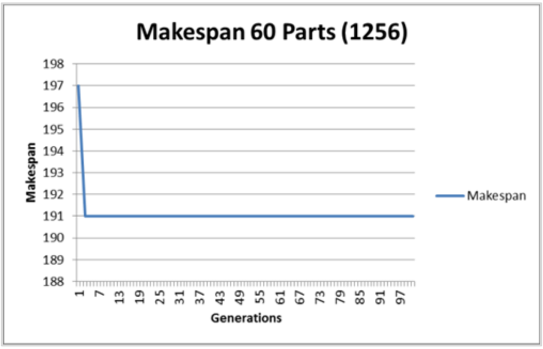


Makespan = 182, Optimum Makespan = 129.67

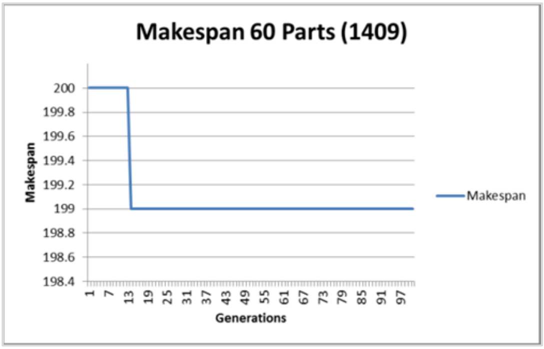


Makespan = 130, Optimum Makespan = 129.67

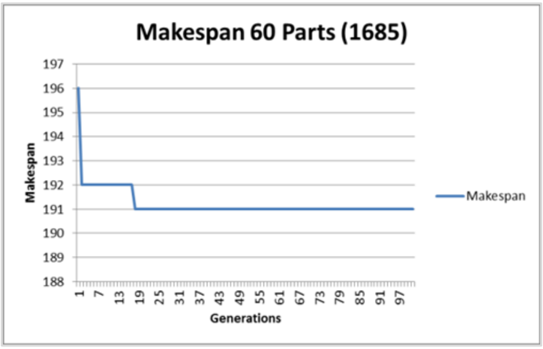
A1.4. Charts for 60 Parts



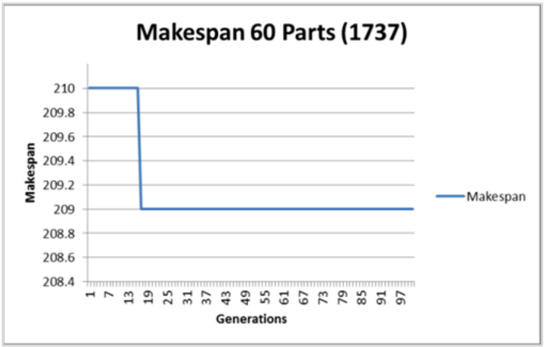
Makespan = 191, Optimum Makespan = 190.67



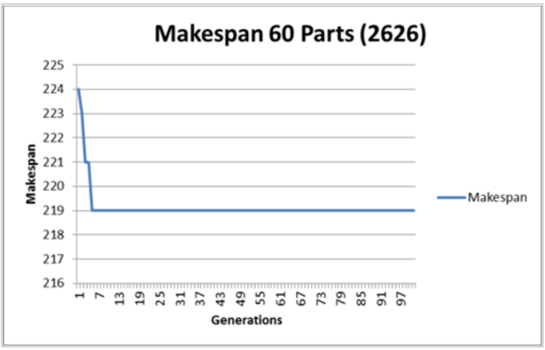
Makespan = 199, Optimum Makespan = 198.67



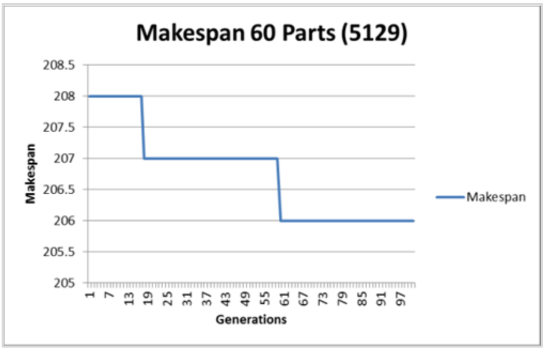
Makespan = 190, Optimum Makespan = 190



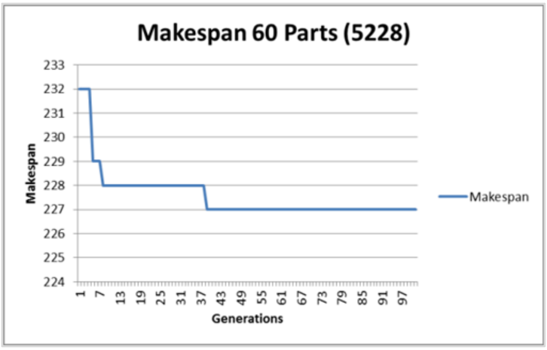
Makespan = 209, Optimum Makespan = 208.33



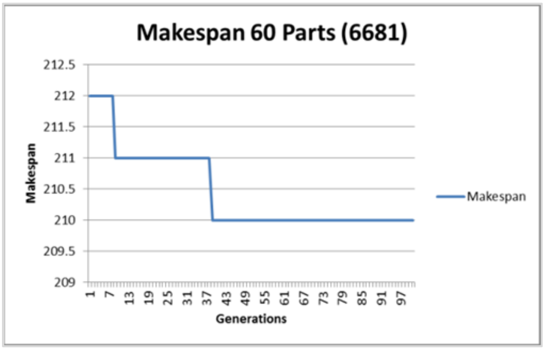
Makespan = 219, Optimum Makespan = 218.33



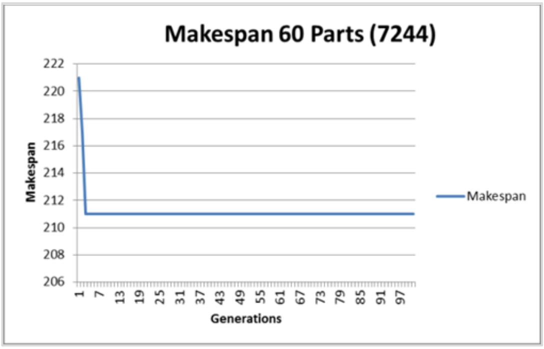
Makespan = 205, Optimum Makespan = 205



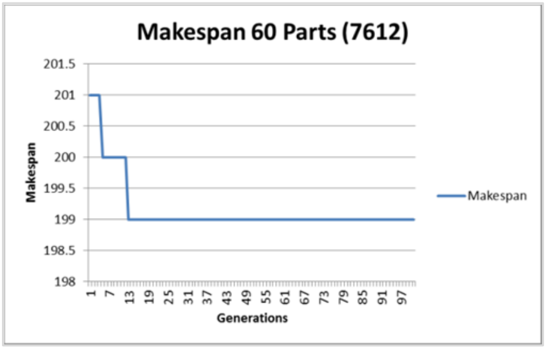
Makespan = 227, Optimum Makespan = 226.33



Makespan = 210, Optimum Makespan = 210

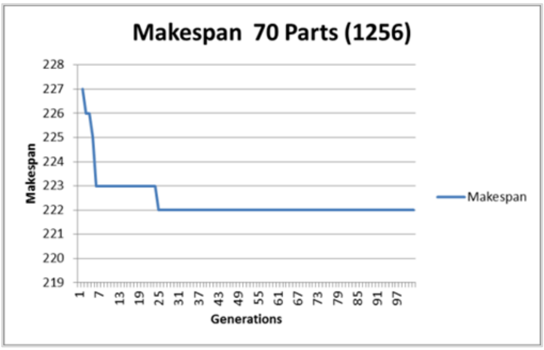


Makespan = 210, Optimum Makespan = 210

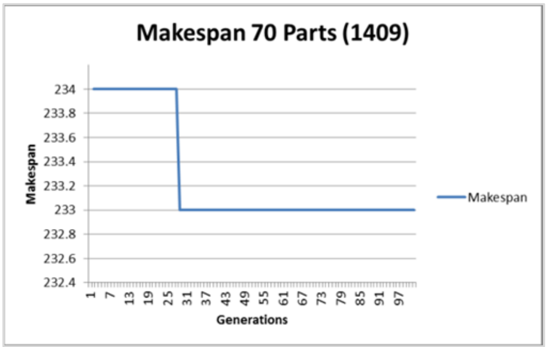


Makespan = 199, Optimum Makespan = 198.33

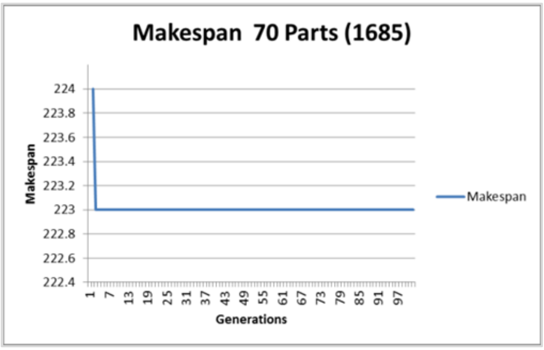
A1.5. Charts for 70 Parts



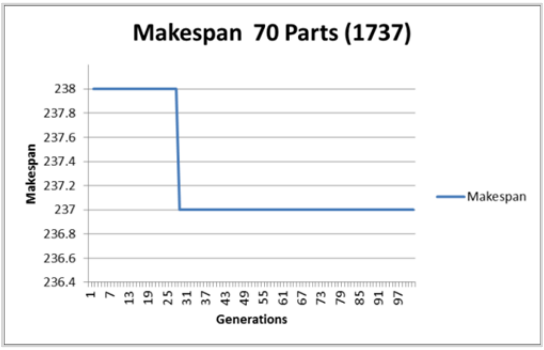
Makespan = 222, Optimum Makespan = 221.67



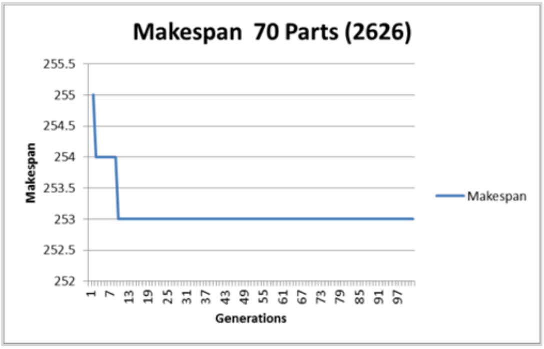
Makespan = 233, Optimum Makespan = 233



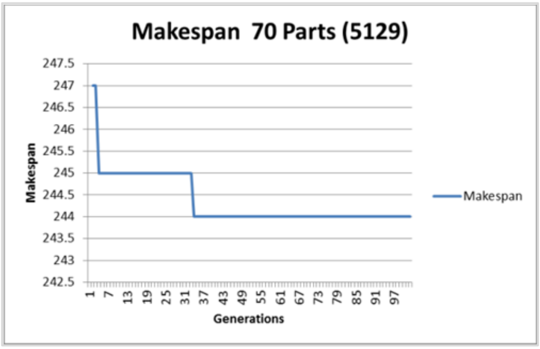
Makespan = 223, Optimum Makespan = 222



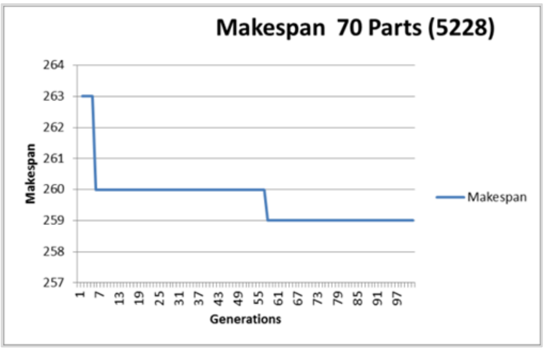
Makespan = 237, Optimum Makespan = 236.33



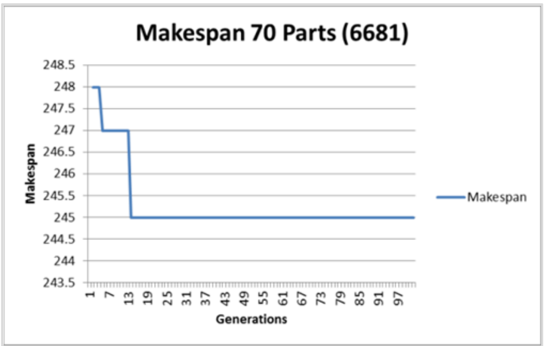
Makespan = 252, Optimum Makespan = 251.67



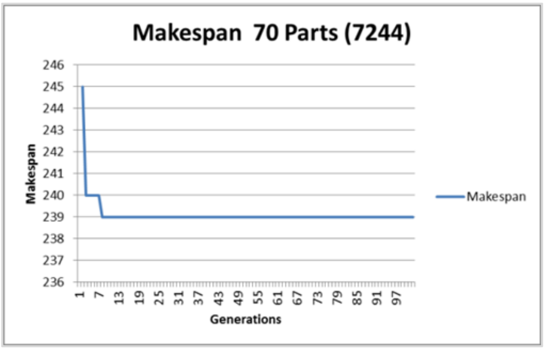
Makespan = 244, Optimum Makespan = 243.33



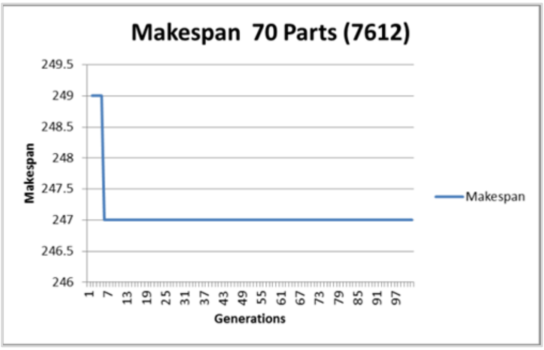
Makespan = 259, Optimum Makespan = 258.67



Makespan = 245, Optimum Makespan = 245

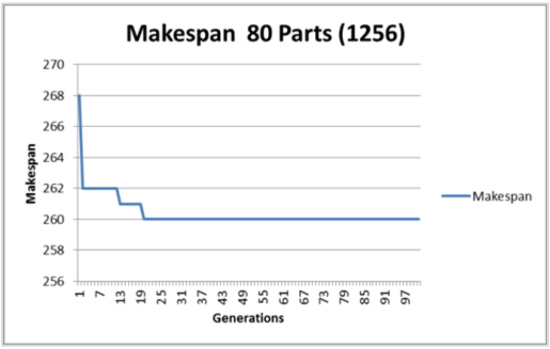


Makespan = 238, Optimum Makespan = 237.67

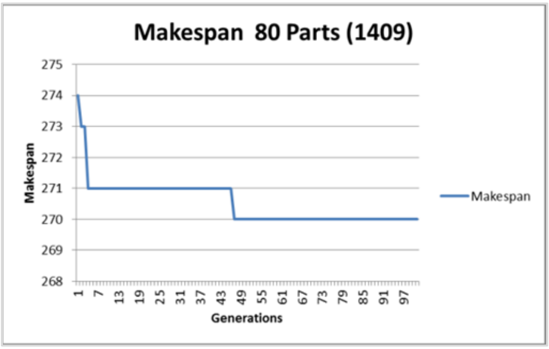


Makespan = 247, Optimum Makespan = 246

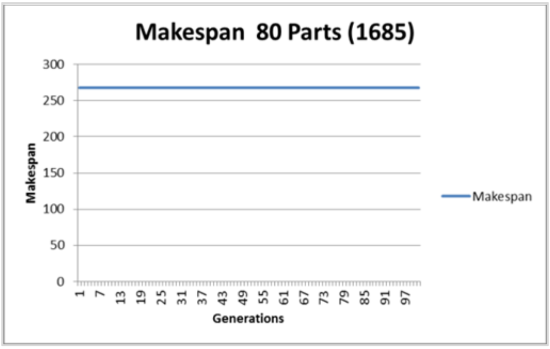
A1.6. Charts for 80 Parts



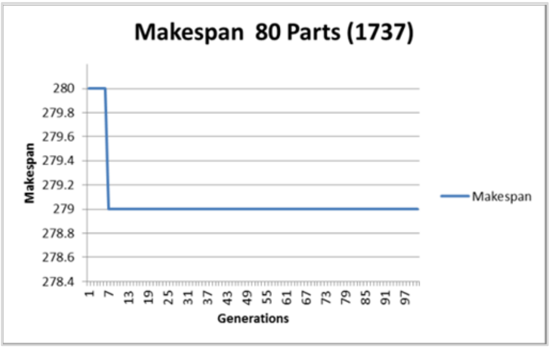
Makespan = 260, Optimum Makespan = 259.33



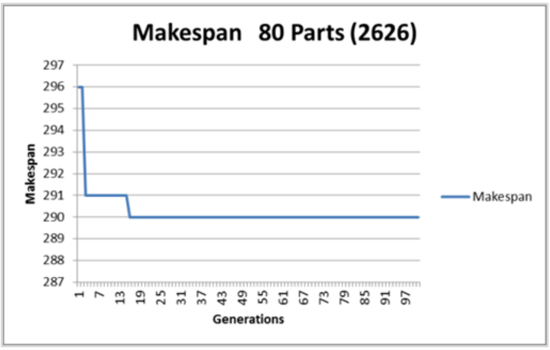
Makespan =270, Optimum Makespan = 269.67



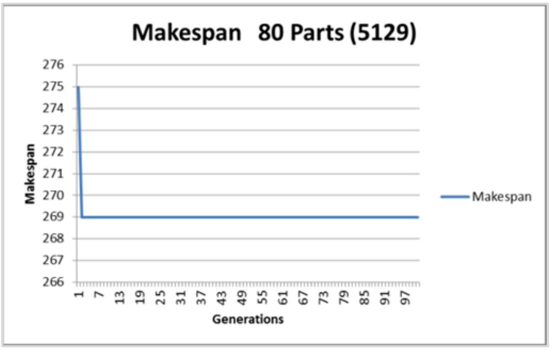
Makespan = 268, Optimum Makespan = 267.33



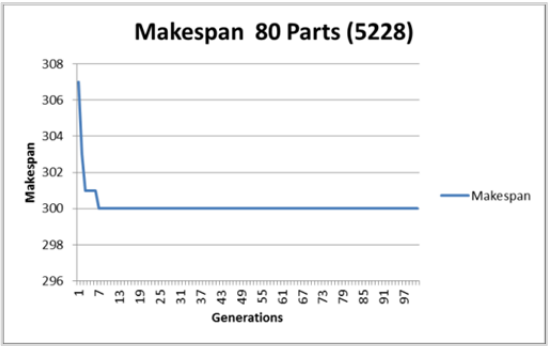
Makespan = 279, Optimum Makespan = 278.33



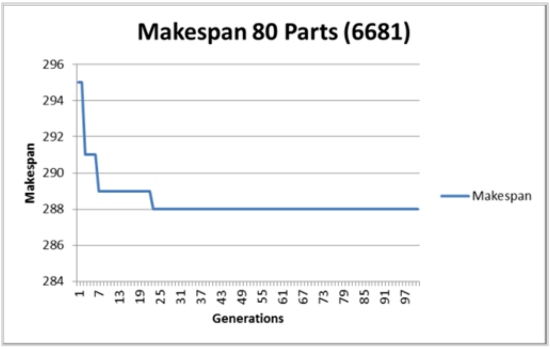
Makespan = 290, Optimum Makespan = 289



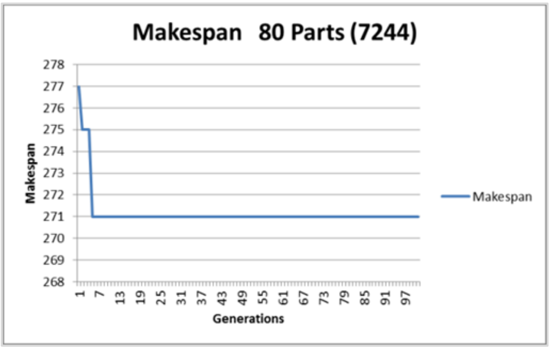
Makespan = 269, Optimum Makespan = 268.33



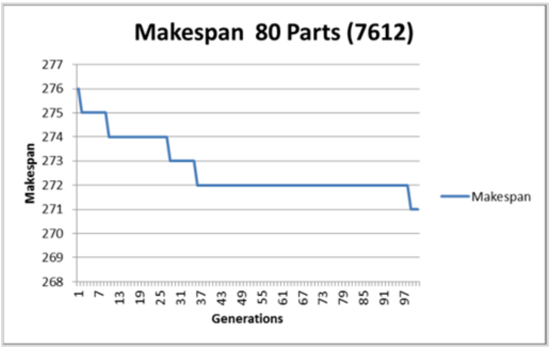
Makespan = 300, Optimum Makespan = 299.67



Makespan = 288, Optimum Makespan = 287.33

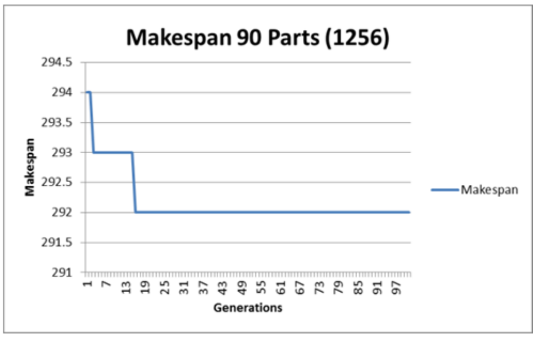


Makespan = 271, Optimum Makespan = 270.33

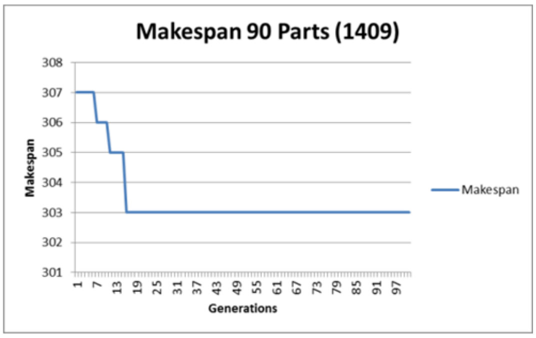


Makespan = 271, Optimum Makespan = 270.67

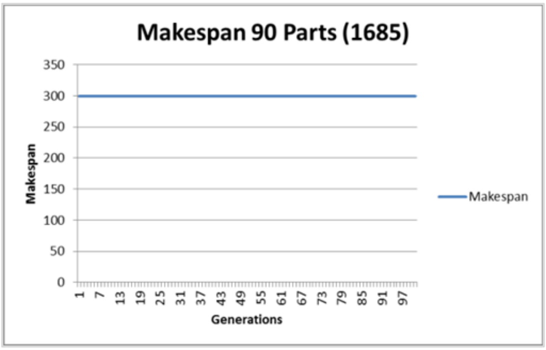
A1.7. Charts for 90 Parts



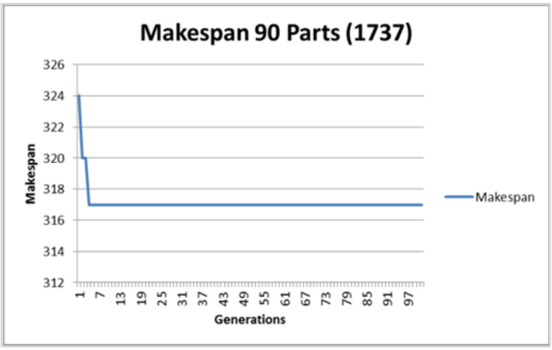
Makespan = 292, Optimum Makespan =291.33



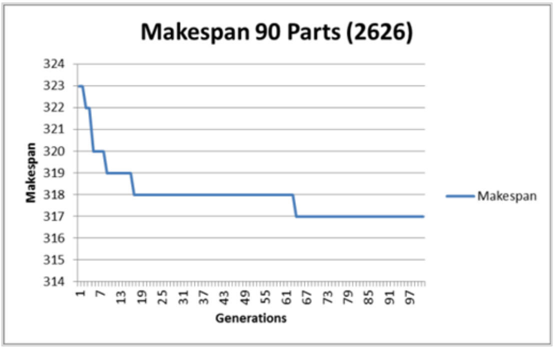
Makespan = 303, Optimum Makespan = 302.33



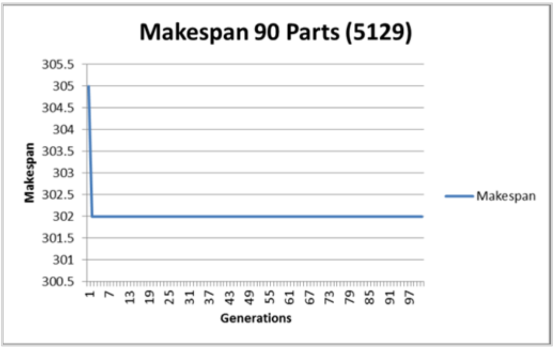
Makespan = 299, Optimum Makespan = 298.33



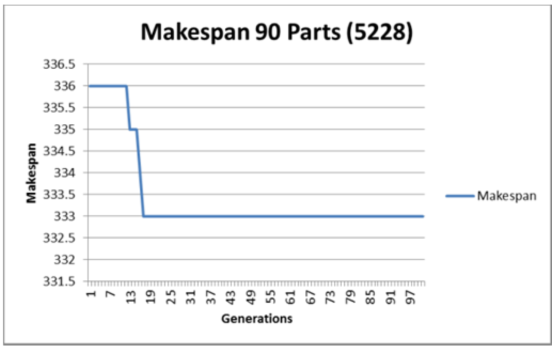
Makespan = 317, Optimum Makespan = 316.33



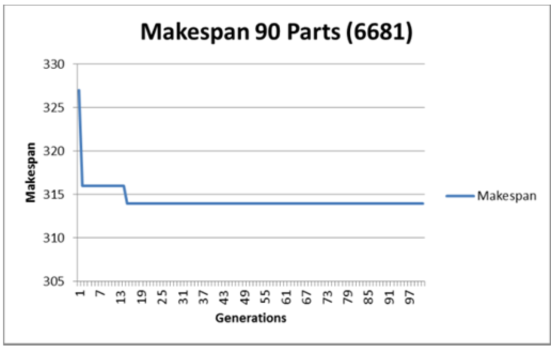
Makespan = 317, Optimum Makespan = 316.33



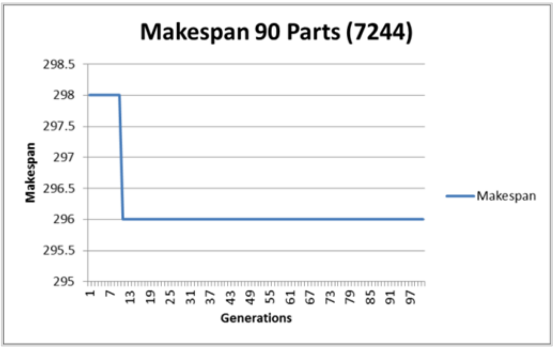
Makespan = 302, Optimum Makespan = 301



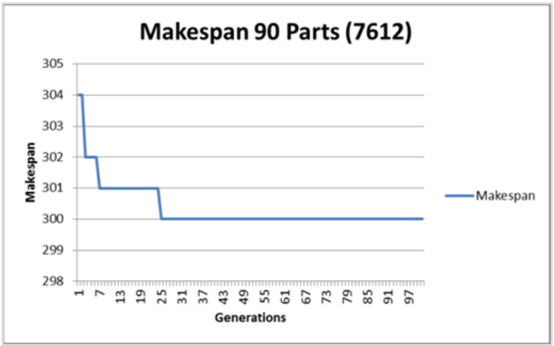
Makespan = 333, Optimum Makespan = 333



Makespan = 315, Optimum Makespan = 314

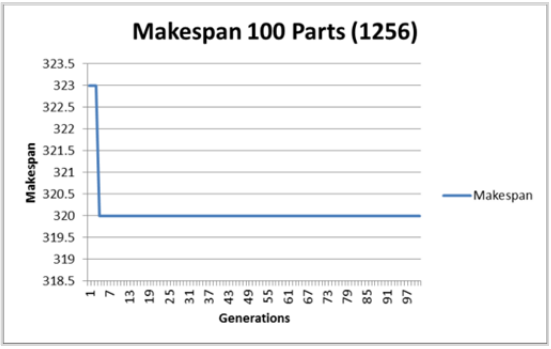


Makespan = 296, Optimum Makespan = 295.33

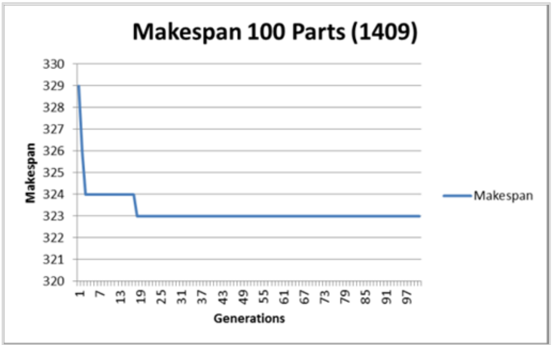


Makespan = 300, Optimum Makespan = 299.33

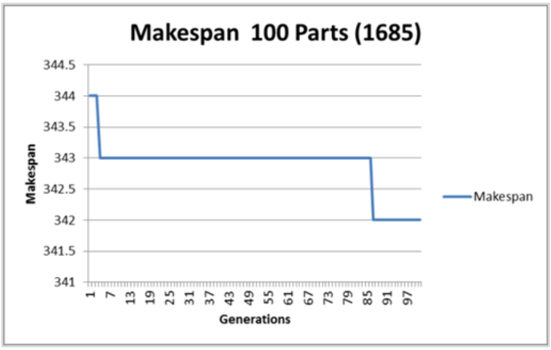
A1.8. Charts for 100 Parts



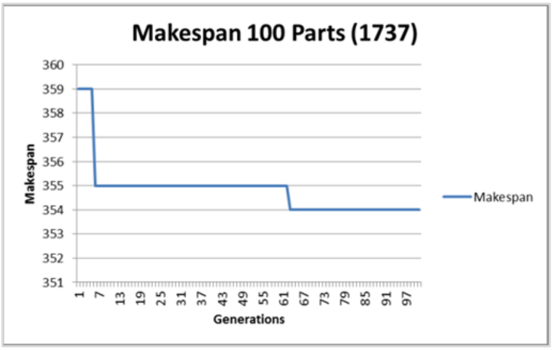
Makespan = 320, Optimum Makespan =319.67



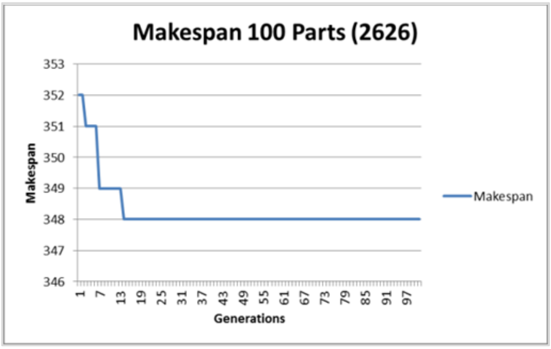
Makespan = 324, Optimum Makespan = 323



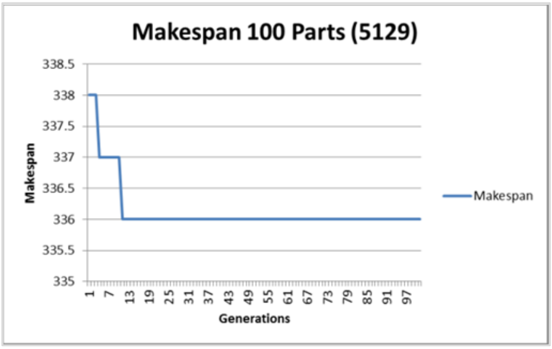
Makespan = 342, Optimum Makespan = 341.67



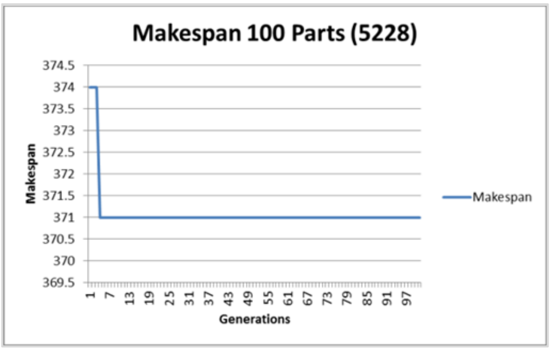
Makespan = 254, Optimum Makespan = 253.67



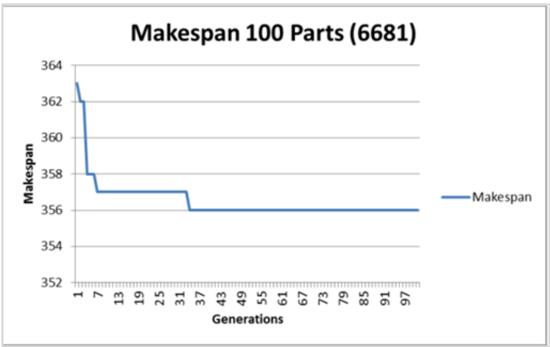
Makespan = 347, Optimum Makespan = 346.67



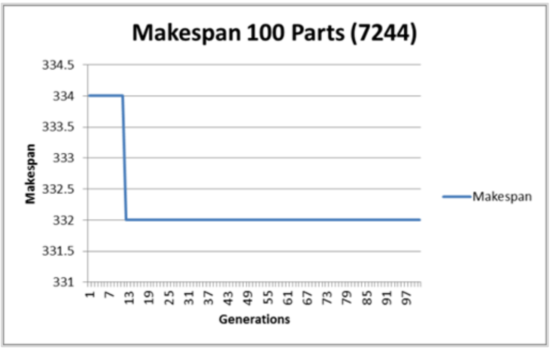
Makespan = 336, Optimum Makespan = 335.33



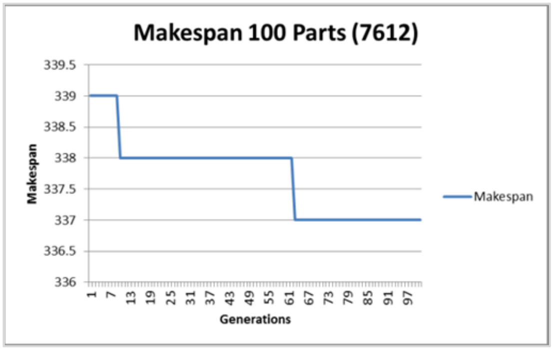
Makespan = 371, Optimum Makespan = 370



Makespan = 356, Optimum Makespan = 355.67



Makespan = 333, Optimum Makespan = 332



Makespan = 337, Optimum Makespan = 336.67

A2. Details of Output from Algorithms

In Table A-1 to Table A-136 the meaning of the column headings is given below.

Column Heading	Meaning
Seed	The seed used for the random number generation in the program.
No. To Stack	This is the number of parts that were placed in the stack because they were ready prior to the previous part being completed. Part *i* is placed in the stack if $C_{i\text{-}1} > C_i$, where C_i is the completion time of part *i*.
Max. in Stack	The maximum number of parts in the stack at any one time.
Avg. Stack Time	The time an item is in the stack is given by $C_{i\text{-}1} - C_i$. Average Stack Time = $\frac{1}{n}\sum_{i=1}^{n}(C_{i\text{-}1} - C_i)$ where *n* is the total number of parts placed in the stack.
% JIT	$J_s = \sum_{i=1}^{n} J_i$ where $J_i = 1$ if $C_i \le C_{i-1}$, otherwise $J_i = 0$ where C_i is the completion time for part *i*.
Total Work	$\sum_{i=1}^{n} C_i$ where C_i is the construction time required for part *i*. *n*= number of parts
Makespan	$C_{\max}^S = \max_j C_j^S$ of a schedule *S* to be the maximum completion time of any job in *S*, where C_j^S is the completion time of job *j* in schedule *S*.
Optimum Makespan	$\frac{1}{T}\sum_{i=1}^{n} C_i$ where C_i is the construction time required for part *i*. *n*= number of part. *T* = number of construction teams.
% Difference	$\frac{\sum_{i=1}^{n} C_i - \frac{1}{T}\sum_{i=1}^{n} C_i}{\frac{1}{T}\sum_{i=1}^{n} C_i} \times 100$ where C_i is the construction time required for part *i*, *n*= number of part, *T* = number of construction teams.
Double Handled	Number of parts that were placed in the stack but were not in the required usage sequence so had other parts had to be double handled to access the part
Reduced/ Double Handled	The effects of Post Ordering reduces the number of double handled parts.
Teams = Makespan	How many of the teams makespan was the minimum makespan

A2.1. Greedy Algorithm – 2 Teams

Table A-1: 2 Teams, 30 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	10	2	3	145	7.63	33	310	155	162	4.32%	1
1409	10	2	4.4	138	7.26	33	307	153.5	154	0.32%	3
1685	11	2	5.27	150	8.82	40	302	151	151	0.00%	2
1737	9	2	4.56	140	7.78	36	287	143.5	144	0.35%	1
2626	10	3	6.6	146	8.11	36	312	156	158	1.27%	2
5129	7	3	5.86	139	6.95	30	282	141	144	2.08%	1
5228	11	2	4.64	137	8.06	40	303	151.5	156	2.88%	1
6681	6	1	4.33	167	7.59	23	332	166	168	1.19%	0
7244	7	1	4.14	148	7.05	26	303	151.5	152	0.33%	0
7612	12	3	4.25	153	9	40	319	159.5	162	1.54%	2
Average	9.3	2.1	4.71	146.3	7.83	33.7	305.7	152.85	155.1	1.43%	1.3
Min	6	1	3	137	6.95	23	282	141	144	0.00%	0
Max	12	3	6.6	167	9	40	332	166	168	4.32%	3

Table A-2: 2 Teams, 40 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	13	2	3.85	187	7.48	35	404	202	204	0.98%	1
1409	14	2	4.07	176	7.04	35	381	190.5	192	0.78%	4
1685	15	2	5.13	194	8.43	40	387	193.5	195	0.77%	3
1737	12	2	4.75	199	7.96	35	392	196	203	3.45%	1
2626	13	3	5.54	197	8.21	37	417	208.5	209	0.24%	2
5129	9	3	6	210	7.5	27	419	209.5	215	2.56%	1
5228	13	2	4.46	181	7.54	37	398	199	200	0.50%	1
6681	8	2	4.88	213	7.34	25	423	211.5	214	1.17%	1
7244	9	1	4.33	214	7.93	30	428	214	218	1.83%	0
7612	15	3	4.53	186	8.09	40	389	194.5	195	0.26%	2
Average	12.1	2.2	4.75	195.7	7.75	34.1	403.8	201.9	204.5	1.25%	1.6
Min	8	1	3.85	176	7.04	25	381	190.5	192	0.24%	0
Max	15	3	6	214	8.43	40	428	214	218	3.45%	4

Table A-3:2 Teams, 50 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	17	2	3.53	236	7.61	36	503	251.5	253	0.59%	1
1409	18	2	3.83	228	7.35	36	470	235	244	3.69%	5
1685	20	3	5.85	237	8.46	42	476	238	238	0.00%	4
1737	13	2	4.85	266	7.82	30	532	266	270	1.48%	1
2626	16	3	5.5	265	8.55	36	549	274.5	277	0.90%	2
5129	10	3	5.6	270	7.3	24	540	270	275	1.82%	1
5228	15	2	4.8	259	8.35	36	540	270	278	2.88%	1
6681	12	2	4.58	252	7.2	28	505	252.5	253	0.20%	1
7244	11	2	4.18	271	7.97	30	544	272	275	1.09%	1
7612	17	3	4.59	244	8.13	38	504	252	253	0.40%	2
Average	14.9	2.4	4.73	252.8	7.87	33.6	516.3	258.15	261.6	1.30%	1.9
Min	10	2	3.53	228	7.2	24	470	235	238	0.00%	1
Max	20	3	5.85	271	8.55	42	549	274.5	278	3.69%	5

Table A-4: 2 Teams, 60 parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	21	3	3.62	272	7.35	36	572	286	289	1.04%	2
1409	22	2	4.32	286	7.94	38	596	298	302	1.32%	6
1685	22	3	5.91	285	8.14	40	570	285	286	0.35%	4
1737	15	2	4.87	309	7.36	28	625	312.5	313	0.16%	1
2626	19	3	5.47	317	8.57	36	655	327.5	329	0.46%	3
5129	12	3	5.67	304	6.91	25	615	307.5	309	0.49%	1
5228	16	2	4.56	321	8.03	31	679	339.5	340	0.15%	1
6681	14	2	4.36	314	7.48	28	630	315	315	0.00%	1
7244	15	2	4.27	311	7.97	33	630	315	315	0.00%	2
7612	21	3	5.24	293	8.14	38	595	297.5	302	1.49%	3
Average	17.7	2.5	4.83	301.2	7.79	33.3	616.7	308.35	310	0.55%	2.4
Min	12	2	3.62	272	6.91	25	570	285	286	0.00%	1
Max	22	3	5.91	321	8.57	40	679	339.5	340	1.49%	6

Table A-5: 2 Teams, 70 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	24	3	4.04	320	7.44	37	665	332.5	337	1.34%	2
1409	25	2	4.52	338	7.86	37	699	349.5	354	1.27%	6
1685	24	3	5.79	336	8	38	666	333	337	1.19%	4
1737	18	2	5.06	353	7.35	30	709	354.5	357	0.70%	1
2626	21	3	5.1	366	8.13	34	755	377.5	378	0.13%	3
5129	13	3	5.46	367	6.92	22	730	365	372	1.88%	1
5228	20	2	4.75	370	8.04	32	776	388	389	0.26%	2
6681	17	2	3.94	372	7.59	28	735	367.5	373	1.47%	1
7244	19	2	4.21	355	8.07	35	713	356.5	359	0.70%	3
7612	22	3	5.09	360	8.18	35	738	369	369	0.00%	3
Average	20.3	2.5	4.80	353.7	7.76	32.8	718.6	359.3	362.5	0.89%	2.6
Min	13	2	3.94	320	6.92	22	665	332.5	337	0.00%	1
Max	25	3	5.79	372	8.18	38	776	388	389	1.88%	6

Table A-6: 2 Teams, 80 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	26	3	4.12	377	7.85	38	778	389	394	1.27%	2
1409	29	2	5.03	396	8.25	38	809	404.5	412	1.82%	8
1685	26	3	5.62	406	8.12	36	802	401	407	1.47%	4
1737	21	2	5.19	418	7.6	30	835	417.5	422	1.07%	1
2626	24	3	5.08	427	8.37	35	867	433.5	439	1.25%	3
5129	17	3	5.35	399	6.76	25	805	402.5	404	0.37%	2
5228	23	2	4.52	434	8.19	32	899	449.5	453	0.77%	2
6681	18	2	3.83	430	7.54	27	862	431	431	0.00%	1
7244	21	2	3.9	402	7.88	35	811	405.5	406	0.12%	3
7612	24	3	5.04	397	7.78	35	812	406	406	0.00%	4
Average	22.9	2.5	4.77	408.6	7.83	33.1	828	414	417.4	0.81%	3
Min	17	2	3.83	377	6.76	25	778	389	394	0.00%	1
Max	29	3	5.62	434	8.37	38	899	449.5	453	1.82%	8

Table A-7: 2 Teams, 90 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	30	3	4.4	420	7.92	40	874	437	437	0.00%	3
1409	32	2	4.88	440	8	37	907	453.5	456	0.55%	8
1685	28	3	5.57	448	7.72	34	895	447.5	449	0.33%	5
1737	24	2	4.92	472	7.61	30	949	474.5	476	0.32%	1
2626	26	3	5.04	466	8.03	34	949	474.5	478	0.73%	4
5129	20	3	5.1	450	6.82	25	903	451.5	455	0.77%	3
5228	26	2	4.5	481	8.02	32	999	499.5	500	0.10%	2
6681	21	2	4	474	7.52	28	942	471	475	0.84%	2
7244	25	2	3.72	442	7.75	35	886	443	446	0.67%	3
7612	27	3	4.78	441	7.6	34	898	449	450	0.22%	4
Average	25.9	2.5	4.69	453.4	7.70	32.9	920.2	460.1	462.2	0.45%	3.5
Min	20	2	3.72	420	6.82	25	874	437	437	0.00%	1
Max	32	3	5.57	481	8.03	40	999	499.5	500	0.84%	8

Table A-8: 2 Teams, 100 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	35	3	4.23	465	8.02	41	959	479.5	482	0.52%	3
1409	36	2	4.86	471	7.72	38	969	484.5	487	0.51%	8
1685	29	3	5.72	515	7.69	32	1025	512.5	516	0.68%	5
1737	26	2	5.08	534	7.63	29	1061	530.5	538	1.39%	1
2626	28	3	5.11	510	7.85	34	1040	520	522	0.38%	4
5129	24	3	4.92	503	6.99	27	1006	503	508	0.98%	3
5228	28	2	4.68	542	7.97	31	1110	555	561	1.07%	2
6681	23	2	3.91	539	7.59	28	1067	533.5	540	1.20%	2
7244	27	2	4.07	498	7.78	35	996	498	502	0.80%	4
7612	28	3	4.71	499	7.45	32	1010	505	508	0.59%	4
Average	28.4	2.5	4.73	507.6	7.67	32.7	1024.3	512.15	516.4	0.81%	3.6
Min	23	2	3.91	465	6.99	27	959	479.5	482	0.38%	1
Max	36	3	5.72	542	8.02	41	1110	555	561	1.39%	8

A2.2. Greedy Algorithm with Post Ordering – 2 Teams

Table A-9: 2 Teams, 30 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	10	2	3	145	7.63	33	310	155	162	4.32%	1
1409	10	2	4.4	138	7.26	33	307	153.5	154	0.32%	3
1685	12	2	4.83	150	8.82	40	302	151	151	0.00%	2
1737	9	2	4.56	140	7.78	36	287	143.5	144	0.35%	1
2626	11	3	6	146	8.11	36	312	156	158	1.27%	2
5129	9	3	4.56	139	6.95	30	282	141	144	2.08%	1
5228	12	2	4.25	137	8.06	40	303	151.5	156	2.88%	1
6681	7	1	3.71	167	7.59	23	332	166	168	1.19%	0
7244	8	1	3.63	148	7.05	26	303	151.5	152	0.33%	0
7612	12	3	4.25	153	9	40	319	159.5	162	1.54%	2
Average	10	2.1	4.32	146.3	7.83	33.7	305.7	152.85	155.1	1.43%	1.3
Min	7	1	3	137	6.95	23	282	141	144	0.00%	0
Max	12	3	6	167	9	40	332	166	168	4.32%	3

Table A-10: 2 Teams, 40 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	14	2	3.57	187	7.48	35	404	202	204	0.98%	1
1409	14	2	4.07	176	7.04	35	381	190.5	192	0.78%	4
1685	16	2	4.81	194	8.43	40	387	193.5	195	0.77%	3
1737	12	2	4.75	199	7.96	35	392	196	203	3.45%	1
2626	15	3	4.8	197	8.21	37	417	208.5	209	0.24%	2
5129	11	3	4.91	210	7.5	27	419	209.5	215	2.56%	1
5228	15	2	3.87	181	7.54	37	398	199	200	0.50%	1
6681	10	2	3.9	213	7.34	25	423	211.5	214	1.17%	1
7244	12	1	3.25	214	7.93	30	428	214	218	1.83%	0
7612	16	3	4.25	186	8.09	40	389	194.5	195	0.26%	2
Average	13.5	2.2	4.22	195.7	7.75	34.1	403.8	201.9	204.5	1.25%	1.6
Min	10	1	3.25	176	7.04	25	381	190.5	192	0.24%	0
Max	16	3	4.91	214	8.43	40	428	214	218	3.45%	4

Table A-11: 2 Teams, 50 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	18	2	3.33	236	7.61	36	503	251.5	253	0.59%	1
1409	18	2	3.83	228	7.35	36	470	235	244	3.69%	5
1685	21	3	5.57	237	8.46	42	476	238	238	0.00%	4
1737	13	2	4.85	266	7.82	30	532	266	270	1.48%	1
2626	18	3	4.89	265	8.55	36	549	274.5	277	0.90%	2
5129	12	3	4.67	270	7.3	24	540	270	275	1.82%	1
5228	18	2	4	259	8.35	36	540	270	278	2.88%	1
6681	14	2	3.93	252	7.2	28	505	252.5	253	0.20%	1
7244	15	2	3.07	271	7.97	30	544	272	275	1.09%	1
7612	19	3	4.11	244	8.13	38	504	252	253	0.40%	2
Average	16.6	2.4	4.23	252.8	7.87	33.6	516.3	258.15	261.6	1.30%	1.9
Min	12	2	3.07	228	7.2	24	470	235	238	0.00%	1
Max	21	3	5.57	271	8.55	42	549	274.5	278	3.69%	5

Table A-12: 2 Teams, 60 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	22	3	3.45	272	7.35	36	572	286	289	1.04%	2
1409	23	2	4.13	286	7.94	38	596	298	302	1.32%	6
1685	24	3	5.42	285	8.14	40	570	285	286	0.35%	4
1737	15	2	4.87	309	7.36	28	625	312.5	313	0.16%	1
2626	22	3	4.73	317	8.57	36	655	327.5	329	0.46%	3
5129	15	3	4.53	304	6.91	25	615	307.5	309	0.49%	1
5228	19	2	3.84	321	8.03	31	679	339.5	340	0.15%	1
6681	17	2	3.59	314	7.48	28	630	315	315	0.00%	1
7244	20	2	3.2	311	7.97	33	630	315	315	0.00%	2
7612	23	3	4.78	293	8.14	38	595	297.5	302	1.49%	3
Average	20	2.5	4.25	301.2	7.79	33.3	616.7	308.35	310	0.55%	2.4
Min	15	2	3.2	272	6.91	25	570	285	286	0.00%	1
Max	24	3	5.42	321	8.57	40	679	339.5	340	1.49%	6

Table A-13: 2 Teams, 70 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	26	3	3.73	320	7.44	37	665	332.5	337	1.34%	2
1409	26	2	4.35	338	7.86	37	699	349.5	354	1.27%	6
1685	27	3	5.15	336	8	38	666	333	337	1.19%	4
1737	19	2	4.79	353	7.35	30	709	354.5	357	0.70%	1
2626	24	3	4.46	366	8.13	34	755	377.5	378	0.13%	3
5129	16	3	4.44	367	6.92	22	730	365	372	1.88%	1
5228	23	2	4.13	370	8.04	32	776	388	389	0.26%	2
6681	20	2	3.35	372	7.59	28	735	367.5	373	1.47%	1
7244	25	2	3.2	355	8.07	35	713	356.5	359	0.70%	3
7612	25	3	4.48	360	8.18	35	738	369	369	0.00%	3
Average	23.1	2.5	4.21	353.7	7.76	32.8	718.6	359.3	362.5	0.89%	2.6
Min	16	2	3.2	320	6.92	22	665	332.5	337	0.00%	1
Max	27	3	5.15	372	8.18	38	776	388	389	1.88%	6

Table A-14: 2 Teams, 80 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	31	3	3.45	377	7.85	38	778	389	394	1.27%	2
1409	31	2	4.71	396	8.25	38	809	404.5	412	1.82%	8
1685	29	3	5.03	406	8.12	36	802	401	407	1.47%	4
1737	22	2	4.95	418	7.6	30	835	417.5	422	1.07%	1
2626	28	3	4.36	427	8.37	35	867	433.5	439	1.25%	3
5129	20	3	4.55	399	6.76	25	805	402.5	404	0.37%	2
5228	26	2	4	434	8.19	32	899	449.5	453	0.77%	2
6681	22	2	3.14	430	7.54	27	862	431	431	0.00%	1
7244	28	2	2.93	402	7.88	35	811	405.5	406	0.12%	3
7612	28	3	4.32	397	7.78	35	812	406	406	0.00%	4
Average	26.5	2.5	4.14	408.6	7.83	33.1	828	414	417.4	0.81%	3
Min	20	2	2.93	377	6.76	25	778	389	394	0.00%	1
Max	31	3	5.03	434	8.37	38	899	449.5	453	1.82%	8

Table A-15: 2 Teams, 90 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	36	3	3.67	420	7.92	40	874	437	437	0.00%	3
1409	34	2	4.59	440	8	37	907	453.5	456	0.55%	8
1685	31	3	5.03	448	7.72	34	895	447.5	449	0.33%	5
1737	25	2	4.72	472	7.61	30	949	474.5	476	0.32%	1
2626	31	3	4.23	466	8.03	34	949	474.5	478	0.73%	4
5129	23	3	4.43	450	6.82	25	903	451.5	455	0.77%	3
5228	29	2	4.03	481	8.02	32	999	499.5	500	0.10%	2
6681	26	2	3.23	474	7.52	28	942	471	475	0.84%	2
7244	32	2	2.91	442	7.75	35	886	443	446	0.67%	3
7612	31	3	4.16	441	7.6	34	898	449	450	0.22%	4
Average	29.8	2.5	4.10	453.4	7.70	32.9	920.2	460.1	462.2	0.45%	3.5
Min	23	2	2.91	420	6.82	25	874	437	437	0.00%	1
Max	36	3	5.03	481	8.03	40	999	499.5	500	0.84%	8

Table A-16: 2 Teams, 100 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	41	3	3.61	465	8.02	41	959	479.5	482	0.52%	3
1409	38	2	4.61	471	7.72	38	969	484.5	487	0.51%	8
1685	32	3	5.19	515	7.69	32	1025	512.5	516	0.68%	5
1737	27	2	4.89	534	7.63	29	1061	530.5	538	1.39%	1
2626	34	3	4.21	510	7.85	34	1040	520	522	0.38%	4
5129	27	3	4.37	503	6.99	27	1006	503	508	0.98%	3
5228	31	2	4.23	542	7.97	31	1110	555	561	1.07%	2
6681	28	2	3.21	539	7.59	28	1067	533.5	540	1.20%	2
7244	35	2	3.14	498	7.78	35	996	498	502	0.80%	4
7612	32	3	4.13	499	7.45	32	1010	505	508	0.59%	4
Average	32.5	2.5	4.16	507.6	7.67	32.7	1024.3	512.15	516.4	0.81%	3.6
Min	27	2	3.14	465	6.99	27	959	479.5	482	0.38%	1
Max	41	3	5.19	542	8.02	41	1110	555	561	1.39%	8

A2.3 Greedy Algorithm with Backtracking – 2 Teams

Table A-17: 2 Teams, 30 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	9	1	6.67	144	7.2	30	310	155	161	3.73%	1
1409	11	2	8.36	138	7.67	36	307	153.5	154	0.32%	1
1685	6	1	6.17	153	6.65	20	302	151	154	1.95%	0
1737	11	2	8.73	141	7.83	36	287	143.5	145	1.03%	2
2626	9	3	5.33	146	7.3	30	312	156	158	1.27%	0
5129	11	2	5.91	137	7.61	36	282	141	142	0.70%	3
5228	8	2	6.5	135	6.43	26	303	151.5	154	1.62%	1
6681	11	2	7.45	168	9.33	36	332	166	169	1.78%	0
7244	8	2	8.63	151	7.19	26	303	151.5	155	2.26%	2
7612	9	1	9.56	154	7.7	30	319	159.5	163	2.15%	2
Average	9.3	1.8	7.33	146.7	7.49	30.6	305.7	152.85	155.5	1.68%	1.2
Min	6	1	5.33	135	6.43	20	282	141	142	0.32%	0
Max	11	3	9.56	168	9.33	36	332	166	169	3.73%	3

Table A-18: 2 Teams, 40 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	12	1	6.5	189	7	30	404	202	206	1.94%	1
1409	12	2	8.25	177	6.56	30	381	190.5	193	1.30%	1
1685	8	1	5.38	196	6.32	20	387	193.5	197	1.78%	0
1737	13	3	7.62	195	7.5	32	392	196	199	1.51%	2
2626	13	3	5.62	198	7.62	32	417	208.5	210	0.71%	1
5129	14	2	6.71	209	8.36	35	419	209.5	214	2.10%	2
5228	12	2	4.83	182	6.74	30	398	199	201	1.00%	1
6681	15	2	5.87	211	8.79	37	423	211.5	212	0.24%	0
7244	9	2	7.89	213	7.1	22	428	214	217	1.38%	2
7612	10	1	9.7	188	6.48	25	389	194.5	197	1.27%	2
Average	11.8	1.9	6.84	195.8	7.25	29.3	403.8	201.9	204.6	1.32%	1.2
Min	8	1	4.83	177	6.32	20	381	190.5	193	0.24%	0
Max	15	3	9.7	213	8.79	37	428	214	217	2.10%	2

Table A-19: 2 Teams, 50 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	15	1	7.33	242	7.12	30	503	251.5	259	2.90%	1
1409	15	2	7.27	226	6.65	30	470	235	242	2.89%	1
1685	10	1	5.2	240	6.15	20	476	238	241	1.24%	0
1737	16	3	9.63	263	7.97	32	532	266	267	0.37%	3
2626	14	3	7.5	265	7.57	28	549	274.5	277	0.90%	1
5129	18	2	8	270	8.71	36	540	270	275	1.82%	2
5228	13	2	6.08	253	7.03	26	540	270	272	0.74%	1
6681	18	2	5.06	256	8.26	36	505	252.5	257	1.75%	0
7244	15	2	6.73	268	7.88	30	544	272	272	0.00%	3
7612	13	1	10.15	249	6.92	26	504	252	258	2.33%	2
Average	14.7	1.9	7.30	253.2	7.43	29.4	516.3	258.15	262	1.49%	1.4
Min	10	1	5.06	226	6.15	20	470	235	241	0.00%	0
Max	18	3	10.15	270	8.71	36	549	274.5	277	2.90%	3

Table A-20: 2 Teams, 60 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	18	1	6.11	271	6.61	30	572	286	288	0.69%	1
1409	19	2	6.89	285	7.13	31	596	298	301	1.00%	1
1685	14	2	5.5	284	6.31	23	570	285	285	0.00%	1
1737	18	3	9	309	7.54	30	625	312.5	313	0.16%	3
2626	17	3	7.65	316	7.52	28	655	327.5	328	0.15%	1
5129	22	2	7.73	309	8.35	36	615	307.5	314	2.07%	2
5228	17	2	6.71	327	7.79	28	679	339.5	346	1.88%	1
6681	21	2	6	319	8.39	35	630	315	320	1.56%	0
7244	16	2	6.81	313	7.28	26	630	315	317	0.63%	3
7612	17	1	9	289	6.88	28	595	297.5	298	0.17%	3
Average	17.9	2	7.14	302.2	7.38	29.5	616.7	308.35	311	0.83%	1.6
Min	14	1	5.5	271	6.31	23	570	285	285	0.00%	0
Max	22	3	9	327	8.39	36	679	339.5	346	2.07%	3

Table A-21: 2 Teams, 70 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	21	1	6.43	317	6.6	30	665	332.5	334	0.45%	1
1409	22	2	6.91	335	7.13	31	699	349.5	351	0.43%	1
1685	18	2	5.61	332	6.51	25	666	333	333	0.00%	1
1737	21	3	8.76	353	7.35	30	709	354.5	357	0.70%	4
2626	19	3	8.26	370	7.4	27	755	377.5	382	1.18%	1
5129	26	2	7.73	365	8.49	37	730	365	370	1.35%	3
5228	19	2	6.53	370	7.4	27	776	388	389	0.26%	1
6681	24	2	5.75	369	8.2	34	735	367.5	370	0.68%	0
7244	19	2	6.21	353	7.06	27	713	356.5	357	0.14%	3
7612	20	2	8.85	368	7.51	28	738	369	377	2.12%	2
Average	20.9	2.1	7.10	353.2	7.37	29.6	718.6	359.3	362	0.73%	1.7
Min	18	1	5.61	317	6.51	25	665	332.5	333	0.00%	0
Max	26	3	8.85	370	8.49	37	776	388	389	2.12%	4

Table A-22: 2 Teams, 80 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	24	1	6.96	372	6.76	30	778	389	389	0.00%	1
1409	24	2	7.04	389	7.07	30	809	404.5	405	0.12%	1
1685	22	2	7.55	400	7.02	27	802	401	401	0.00%	2
1737	25	3	8.64	417	7.72	31	835	417.5	421	0.83%	3
2626	21	3	8.19	422	7.28	26	867	433.5	434	0.12%	1
5129	29	2	7.69	400	8	36	805	402.5	405	0.62%	3
5228	21	2	7.52	435	7.5	26	899	449.5	454	0.99%	1
6681	28	2	6.71	434	8.51	35	862	431	435	0.92%	1
7244	23	2	6.3	405	7.23	28	811	405.5	409	0.86%	4
7612	24	2	8.13	399	7.25	30	812	406	408	0.49%	3
Average	24.1	2.1	7.47	407.3	7.43	29.9	828	414	416.1	0.49%	2
Min	21	1	6.3	372	6.76	26	778	389	389	0.00%	1
Max	29	3	8.64	435	8.51	36	899	449.5	454	0.99%	4

Table A-23: 2 Teams, 90 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	27	1	6.81	422	6.81	30	874	437	439	0.46%	1
1409	27	2	6.63	439	7.08	30	907	453.5	455	0.33%	1
1685	25	2	7.44	448	7	27	895	447.5	449	0.33%	3
1737	29	3	9.14	474	7.9	32	949	474.5	478	0.73%	4
2626	25	3	7.64	464	7.25	27	949	474.5	476	0.32%	1
5129	33	3	7.64	447	7.98	36	903	451.5	452	0.11%	3
5228	24	2	7.83	485	7.46	26	999	499.5	504	0.89%	1
6681	31	3	6.16	473	8.16	34	942	471	474	0.63%	1
7244	26	2	6.08	441	7	28	886	443	445	0.45%	4
7612	25	2	8.36	447	6.98	27	898	449	456	1.54%	2
Average	27.2	2.3	7.37	454	7.36	29.7	920.2	460.1	462.8	0.58%	2.1
Min	24	1	6.08	422	6.81	26	874	437	439	0.11%	1
Max	33	3	9.14	485	8.16	36	999	499.5	504	1.54%	4

Table A-24: 2 Teams, 100 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	30	2	6.93	464	6.72	30	959	479.5	481	0.31%	2
1409	28	2	6.43	469	6.61	28	969	484.5	485	0.10%	1
1685	29	2	7.48	513	7.33	29	1025	512.5	514	0.29%	3
1737	31	3	8.9	529	7.78	31	1061	530.5	533	0.47%	4
2626	29	3	8.41	515	7.36	29	1040	520	527	1.33%	2
5129	36	3	7.83	504	8	36	1006	503	509	1.18%	4
5228	27	2	7.11	537	7.46	27	1110	555	556	0.18%	1
6681	34	3	6.59	535	8.23	34	1067	533.5	536	0.47%	1
7244	28	2	6.75	496	6.99	28	996	498	500	0.40%	4
7612	28	2	8.29	497	7	28	1010	505	506	0.20%	2
Average	30	2.4	7.47	505.9	7.35	30	1024.3	512.15	514.7	0.49%	2.4
Min	27	2	6.43	464	6.61	27	959	479.5	481	0.10%	1
Max	36	3	8.9	537	8.23	36	1110	555	556	1.33%	4

A2.4. Greedy Algorithm with Post Ordering and Backtracking – 2 Teams

Table A-25: 2 Teams, 30 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	10	2	5.4	144	7.58	30	310	155	161	3.73%	1
1409	8	3	5.88	138	6.57	23	307	153.5	154	0.32%	1
1685	7	1	5.43	153	6.95	20	302	151	154	1.95%	0
1737	11	2	6.82	141	7.83	33	287	143.5	145	1.03%	2
2626	8	1	3.88	146	6.95	23	312	156	158	1.27%	0
5129	12	2	5.58	137	8.06	36	282	141	142	0.70%	3
5228	7	2	5.71	135	6.14	20	303	151.5	154	1.62%	1
6681	11	1	5.55	168	9.33	33	332	166	169	1.78%	0
7244	10	2	6.4	151	7.95	30	303	151.5	155	2.26%	2
7612	10	2	7.1	154	8.11	30	319	159.5	163	2.15%	2
Average	9.4	1.8	5.78	146.7	7.55	27.8	305.7	152.85	155.5	1.68%	1.2
Min	7	1	3.88	135	6.14	20	282	141	142	0.32%	0
Max	12	3	7.1	168	9.33	36	332	166	169	3.73%	3

Table A-26: 2 Teams, 40 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	13	2	5.54	189	7.27	30	404	202	206	1.94%	1
1409	8	3	5.88	177	5.71	17	381	190.5	193	1.30%	1
1685	9	1	4.89	196	6.53	20	387	193.5	197	1.78%	0
1737	13	3	6	195	7.5	30	392	196	199	1.51%	2
2626	12	1	4.67	198	7.33	27	417	208.5	210	0.71%	1
5129	15	2	6.4	209	8.71	35	419	209.5	214	2.10%	2
5228	11	2	4.18	182	6.5	25	398	199	201	1.00%	1
6681	15	1	4.47	211	8.79	35	423	211.5	212	0.24%	0
7244	11	2	6	213	7.61	25	428	214	217	1.38%	2
7612	12	2	6.33	188	6.96	27	389	194.5	197	1.27%	2
Average	11.9	1.9	5.44	195.8	7.29	27.1	403.8	201.9	204.6	1.32%	1.2
Min	8	1	4.18	177	5.71	17	381	190.5	193	0.24%	0
Max	15	3	6.4	213	8.79	35	428	214	217	2.10%	2

Table A-27: 2 Teams, 50 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	16	2	5.56	242	7.33	30	503	251.5	259	2.90%	1
1409	11	3	5.18	226	5.95	20	470	235	242	2.89%	1
1685	11	1	4.82	240	6.32	20	476	238	241	1.24%	0
1737	17	3	6.82	263	8.22	32	532	266	267	0.37%	3
2626	14	3	6	265	7.57	26	549	274.5	277	0.90%	1
5129	18	2	6.56	270	8.71	34	540	270	275	1.82%	2
5228	12	2	4.75	253	6.84	22	540	270	272	0.74%	1
6681	18	1	4.11	256	8.26	34	505	252.5	257	1.75%	0
7244	17	2	5.65	268	8.38	32	544	272	272	0.00%	3
7612	15	2	7	249	7.32	28	504	252	258	2.33%	2
Average	14.9	2.1	5.65	253.2	7.49	27.8	516.3	258.15	262	1.49%	1.4
Min	11	1	4.11	226	5.95	20	470	235	241	0.00%	0
Max	18	3	7	270	8.71	34	549	274.5	277	2.90%	3

Table A-28: 2 Teams, 60 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	19	2	4.95	271	6.78	30	572	286	288	0.69%	1
1409	15	3	4.47	285	6.48	23	596	298	301	1.00%	1
1685	15	2	5.2	284	6.45	23	570	285	285	0.00%	1
1737	19	3	6.53	309	7.73	30	625	312.5	313	0.16%	3
2626	17	3	6.18	316	7.52	26	655	327.5	328	0.15%	1
5129	21	2	5.67	309	8.13	33	615	307.5	314	2.07%	2
5228	16	2	5.75	327	7.6	25	679	339.5	346	1.88%	1
6681	21	1	4.76	319	8.39	33	630	315	320	1.56%	0
7244	18	2	5.78	313	7.63	28	630	315	317	0.63%	3
7612	19	2	6.74	289	7.23	30	595	297.5	298	0.17%	3
Average	18	2.2	5.60	302.2	7.39	28.1	616.7	308.35	311	0.83%	1.6
Min	15	1	4.47	271	6.45	23	570	285	285	0.00%	0
Max	21	3	6.74	327	8.39	33	679	339.5	346	2.07%	3

Table A-29: 2 Teams, 70 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	23	2	5.13	317	6.89	31	665	332.5	334	0.45%	1
1409	18	3	4.89	335	6.57	24	699	349.5	351	0.43%	1
1685	18	2	4.89	332	6.51	24	666	333	333	0.00%	1
1737	22	3	6.64	353	7.51	30	709	354.5	357	0.70%	4
2626	20	3	6.55	370	7.55	27	755	377.5	382	1.18%	1
5129	25	2	6	365	8.3	34	730	365	370	1.35%	3
5228	18	2	5.67	370	7.25	24	776	388	389	0.26%	1
6681	24	1	4.67	369	8.2	32	735	367.5	370	0.68%	0
7244	20	2	5.35	353	7.2	27	713	356.5	357	0.14%	3
7612	22	2	6.91	368	7.83	30	738	369	377	2.12%	2
Average	21	2.2	5.67	353.2	7.38	28.3	718.6	359.3	362	0.73%	1.7
Min	18	1	4.67	317	6.51	24	665	332.5	333	0.00%	0
Max	25	3	6.91	370	8.3	34	776	388	389	2.12%	4

Table A-30: 2 Teams, 80 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	27	2	5.48	372	7.15	32	778	389	389	0.00%	1
1409	20	3	5.5	389	6.59	23	809	404.5	405	0.12%	1
1685	23	2	5.83	400	7.14	27	802	401	401	0.00%	2
1737	24	3	6.46	417	7.58	28	835	417.5	421	0.83%	3
2626	22	3	6.59	422	7.4	26	867	433.5	434	0.12%	1
5129	28	2	5.79	400	7.84	33	805	402.5	405	0.62%	3
5228	21	2	6.14	435	7.5	25	899	449.5	454	0.99%	1
6681	28	1	5.79	434	8.51	33	862	431	435	0.92%	1
7244	24	2	5	405	7.36	28	811	405.5	409	0.86%	4
7612	26	2	6.62	399	7.53	31	812	406	408	0.49%	3
Average	24.3	2.2	5.92	407.3	7.46	28.6	828	414	416.1	0.49%	2
Min	20	1	5	372	6.59	23	778	389	389	0.00%	1
Max	28	3	6.62	435	8.51	33	899	449.5	454	0.99%	4

Table A-31: 2 Teams, 90 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	29	2	5.55	422	7.03	31	874	437	439	0.46%	1
1409	23	3	5.22	439	6.65	24	907	453.5	455	0.33%	1
1685	27	2	5.59	448	7.23	28	895	447.5	449	0.33%	3
1737	28	3	6.96	474	7.77	30	949	474.5	478	0.73%	4
2626	25	3	6.2	464	7.25	26	949	474.5	476	0.32%	1
5129	32	2	5.38	447	7.84	34	903	451.5	452	0.11%	3
5228	23	2	6.43	485	7.35	24	999	499.5	504	0.89%	1
6681	31	3	5.32	473	8.16	33	942	471	474	0.63%	1
7244	27	2	4.93	441	7.11	28	886	443	445	0.45%	4
7612	27	2	6.41	447	7.21	28	898	449	456	1.54%	2
Average	27.2	2.4	5.80	454	7.36	28.6	920.2	460.1	462.8	0.58%	2.1
Min	23	2	4.93	422	6.65	24	874	437	439	0.11%	1
Max	32	3	6.96	485	8.16	34	999	499.5	504	1.54%	4

Table A-32: 2 Teams, 100 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	32	2	5.78	464	6.93	31	959	479.5	481	0.31%	2
1409	24	3	5.04	469	6.25	23	969	484.5	485	0.10%	1
1685	29	2	5.9	513	7.33	28	1025	512.5	514	0.29%	3
1737	30	3	6.87	529	7.67	29	1061	530.5	533	0.47%	4
2626	29	3	6.69	515	7.36	28	1040	520	527	1.33%	2
5129	35	2	5.77	504	7.88	34	1006	503	509	1.18%	4
5228	26	2	5.85	537	7.36	25	1110	555	556	0.18%	1
6681	33	3	5.45	535	8.11	32	1067	533.5	536	0.47%	1
7244	30	2	5.33	496	7.19	29	996	498	500	0.40%	4
7612	31	2	6.23	497	7.31	30	1010	505	506	0.20%	2
Average	29.9	2.4	5.89	505.9	7.34	28.9	1024.3	512.15	514.7	0.49%	2.4
Min	24	2	5.04	464	6.25	23	959	479.5	481	0.10%	1
Max	35	3	6.87	537	8.11	34	1110	555	556	1.33%	4

A2.5. Greedy Algorithm – 3 Teams

Table A-33: 3 Teams, 30 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	10	3	5.4	97	5.11	33	310	103.33	114	9.36%	2
1409	12	2	5.5	89	5.56	43	307	102.33	105	2.54%	1
1685	13	3	6.31	101	6.73	46	302	100.67	102	1.31%	2
1737	11	2	4.36	92	5.75	43	287	95.67	96	0.35%	3
2626	11	3	7.18	92	5.75	43	312	104	104	0.00%	2
5129	12	4	4.83	94	6.27	46	282	94	99	5.05%	3
5228	17	5	5.71	87	7.25	56	303	101	106	4.72%	3
6681	9	2	5.44	112	5.89	33	332	110.67	113	2.06%	1
7244	11	2	6	100	5.88	40	303	101	104	2.88%	3
7612	10	3	7.2	102	6	40	319	106.33	111	4.20%	1
Average	11.6	2.9	5.79	96.6	6.02	42.3	305.7	101.9	105.4	3.25%	2.1
Min	9	2	4.36	87	5.11	33	282	94	96	0.00%	1
Max	17	5	7.2	112	7.25	56	332	110.67	114	9.36%	3

Table A-34: 3 Teams, 40 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	16	3	5.31	118	5.13	40	404	134.67	135	0.25%	3
1409	18	4	5.5	113	5.65	47	381	127	129	1.55%	3
1685	17	3	6.65	130	6.5	47	387	129	131	1.53%	4
1737	15	2	4.8	135	6.14	42	392	130.67	139	6.00%	3
2626	17	3	7.59	131	6.55	47	417	139	143	2.80%	3
5129	14	4	4.86	143	6.22	40	419	139.67	148	5.63%	3
5228	21	5	5.71	116	6.44	52	398	132.67	135	1.73%	4
6681	13	2	5	142	5.92	37	423	141	143	1.40%	1
7244	14	2	5.29	141	5.88	37	428	142.67	145	1.61%	3
7612	14	3	6.93	124	5.39	40	389	129.67	133	2.51%	1
Average	15.9	3.1	5.76	129.3	5.98	42.9	403.8	134.6	138.1	2.50%	2.8
Min	13	2	4.8	113	5.13	37	381	127	129	0.25%	1
Max	21	5	7.59	143	6.55	52	428	142.67	148	6.00%	4

Table A-35: 3 Teams, 50 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	20	3	5.2	155	5.54	42	503	167.67	172	2.52%	4
1409	21	4	5.05	152	6.08	48	470	156.67	168	6.75%	3
1685	22	3	6.45	161	6.44	48	476	158.67	162	2.06%	5
1737	17	2	4.94	177	5.9	38	532	177.33	181	2.03%	3
2626	21	3	6.81	173	6.65	46	549	183	185	1.08%	4
5129	17	4	4.53	179	5.97	38	540	180	184	2.17%	3
5228	24	5	5.5	166	6.64	48	540	180	185	2.70%	4
6681	17	2	5	174	5.8	38	505	168.33	175	3.81%	2
7244	17	2	5.59	182	6.07	38	544	181.33	186	2.51%	4
7612	18	3	6	164	5.66	40	504	168	173	2.89%	2
Average	19.4	3.1	5.51	168.3	6.08	42.4	516.3	172.1	177.1	2.85%	3.4
Min	17	2	4.53	152	5.54	38	470	156.67	162	1.08%	2
Max	24	5	6.81	182	6.65	48	549	183	186	6.75%	5

Table A-36: 3 Teams, 60 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	26	5	5.04	178	5.56	45	572	190.67	195	2.22%	6
1409	25	4	5	192	6.4	48	596	198.67	208	4.49%	3
1685	26	3	6.54	191	6.16	46	570	190	192	1.04%	7
1737	22	2	5	206	5.89	40	625	208.33	210	0.79%	4
2626	25	3	7.08	211	6.59	45	655	218.33	223	2.09%	5
5129	21	4	4.33	205	5.69	38	615	205	210	2.38%	4
5228	28	5	5.25	216	7.2	48	679	226.33	235	3.69%	4
6681	20	2	5.1	215	5.97	38	630	210	216	2.78%	2
7244	23	3	5.74	207	6.09	41	630	210	211	0.47%	6
7612	22	3	5.95	195	5.57	40	595	198.33	204	2.78%	4
Average	23.8	3.4	5.50	201.6	6.11	42.9	616.7	205.57	210.4	2.27%	4.5
Min	20	2	4.33	178	5.56	38	570	190	192	0.47%	2
Max	28	5	7.08	216	7.2	48	679	226.33	235	4.49%	7

Table A-37: 3 Teams, 70 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	32	5	5.19	209	5.81	47	665	221.67	226	1.92%	7
1409	30	4	5.33	221	6.31	48	699	233	237	1.69%	4
1685	31	3	6.23	228	6.33	47	666	222	229	3.06%	8
1737	26	3	4.88	234	5.85	41	709	236.33	238	0.70%	4
2626	29	4	6.62	244	6.59	45	755	251.67	256	1.69%	7
5129	23	4	4.43	247	5.74	37	730	243.33	252	3.44%	4
5228	33	5	5.64	243	6.94	48	776	258.67	262	1.27%	5
6681	23	2	4.65	254	6.05	38	735	245	255	3.92%	3
7244	27	3	5.56	240	6	41	713	237.67	244	2.60%	7
7612	23	3	5.74	241	5.6	37	738	246	250	1.60%	4
Average	27.7	3.6	5.43	236.1	6.12	42.9	718.6	239.53	244.9	2.19%	5.3
Min	23	2	4.43	209	5.6	37	665	221.67	226	0.70%	3
Max	33	5	6.62	254	6.94	48	776	258.67	262	3.92%	8

Table A-38: 3 Teams, 80 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	35	5	5.17	245	5.83	46	778	259.33	262	1.02%	7
1409	36	4	5.56	260	6.67	50	809	269.67	276	2.29%	5
1685	34	3	5.88	273	6.5	46	802	267.33	274	2.43%	8
1737	30	3	4.57	280	6.09	41	835	278.33	284	2.00%	4
2626	32	4	6.53	278	6.47	45	867	289	290	0.34%	8
5129	26	4	4.23	265	5.41	37	805	268.33	270	0.62%	4
5228	36	5	5.36	286	6.81	46	899	299.67	305	1.75%	5
6681	27	2	4.67	289	6.02	38	862	287.33	290	0.92%	4
7244	28	3	5.43	270	5.74	40	811	270.33	274	1.34%	7
7612	28	3	5.54	262	5.57	40	812	270.67	271	0.12%	4
Average	31.2	3.6	5.29	270.8	6.11	42.9	828	276	279.6	1.28%	5.6
Min	26	2	4.23	245	5.41	37	778	259.33	262	0.12%	4
Max	36	5	6.53	289	6.81	50	899	299.67	305	2.43%	8

Table A-39: 3 Teams, 90 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	40	5	5.55	276	5.87	46	874	291.33	293	0.57%	9
1409	41	4	5.29	290	6.59	50	907	302.33	306	1.20%	5
1685	39	3	5.69	300	6.38	46	895	298.33	301	0.89%	9
1737	35	3	4.54	315	6.18	42	949	316.33	319	0.84%	6
2626	34	4	6.5	312	6.12	42	949	316.33	324	2.37%	9
5129	31	4	4.61	300	5.56	38	903	301	305	1.31%	6
5228	39	5	5.33	314	6.68	46	999	333	333	0.00%	5
6681	34	3	4.59	315	6.18	42	942	314	316	0.63%	5
7244	33	3	5.33	295	5.67	41	886	295.33	299	1.23%	7
7612	32	3	5.28	299	5.75	41	898	299.33	308	2.81%	5
Average	35.8	3.7	5.27	301.6	6.10	43.4	920.2	306.73	310.4	1.18%	6.6
Min	31	3	4.54	276	5.56	38	874	291.33	293	0.00%	5
Max	41	5	6.5	315	6.68	50	999	333	333	2.81%	9

Table A-40: 3 Teams, 100 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	44	5	5.43	309	5.83	46	959	319.67	326	1.94%	9
1409	47	4	5.11	309	6.44	51	969	323	325	0.62%	6
1685	41	3	5.85	345	6.27	44	1025	341.67	346	1.25%	9
1737	40	3	4.8	361	6.45	43	1061	353.67	365	3.11%	7
2626	41	4	5.95	340	6.3	45	1040	346.67	352	1.52%	10
5129	35	4	4.71	338	5.63	39	1006	335.33	343	2.24%	7
5228	42	5	5.29	358	6.63	45	1110	370	377	1.86%	5
6681	37	3	4.59	365	6.29	41	1067	355.67	366	2.82%	6
7244	37	3	5.46	336	5.89	42	996	332	340	2.35%	7
7612	35	3	5.31	332	5.63	40	1010	336.67	341	1.27%	5
Average	39.9	3.7	5.25	339.3	6.14	43.6	1024.3	341.43	348.1	1.90%	7.1
Min	35	3	4.59	309	5.63	39	959	319.67	325	0.62%	5
Max	47	5	5.95	365	6.63	51	1110	370	377	3.11%	10

A2.6. Greedy Algorithm with Post Ordering – 3 Teams

Table A-41: 3 Teams, 30 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	10	3	5.4	97	5.11	33	310	103.33	114	9.36%	2
1409	13	2	5.08	89	5.56	43	307	102.33	105	2.54%	1
1685	14	3	5.86	101	6.73	46	302	100.67	102	1.31%	2
1737	12	2	4	92	5.75	43	287	95.67	96	0.35%	3
2626	13	3	6.08	92	5.75	43	312	104	104	0.00%	2
5129	14	4	4.14	94	6.27	46	282	94	99	5.05%	3
5228	17	5	5.71	87	7.25	56	303	101	106	4.72%	3
6681	10	2	4.9	112	5.89	33	332	110.67	113	2.06%	1
7244	12	2	5.5	100	5.88	40	303	101	104	2.88%	3
7612	10	3	7.2	102	6	40	319	106.33	111	4.20%	1
Average	12.5	2.9	5.39	96.6	6.02	42.3	305.7	101.90	105.4	3.25%	2.1
Min	10	2	4	87	5.11	33	282	94	96	0.00%	1
Max	17	5	7.2	112	7.25	56	332	110.67	114	9.36%	3

Table A-42: 3 Teams, 40 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	16	3	5.31	118	5.13	40	404	134.67	135	0.25%	3
1409	19	4	5.21	113	5.65	47	381	127.00	129	1.55%	3
1685	19	3	5.95	130	6.5	47	387	129	131	1.53%	4
1737	16	2	4.5	135	6.14	42	392	130.67	139	6.00%	3
2626	19	3	6.79	131	6.55	47	417	139	143	2.80%	3
5129	16	4	4.25	143	6.22	40	419	139.67	148	5.63%	3
5228	21	5	5.71	116	6.44	52	398	132.67	135	1.73%	4
6681	15	2	4.33	142	5.92	37	423	141	143	1.40%	1
7244	15	2	4.93	141	5.88	37	428	142.67	145	1.61%	3
7612	14	3	6.93	124	5.39	40	389	129.67	133	2.51%	1
Average	17	3.1	5.39	129.3	5.98	42.9	403.8	134.60	138.1	2.50%	2.8
Min	14	2	4.25	113	5.13	37	381	127	129	0.25%	1
Max	21	5	6.93	143	6.55	52	428	142.67	148	6.00%	4

Table A-43: 3 Teams, 50 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack		Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handling
1256	21	3	4.95	155	5.54	42	503	167.67	172	2.52%	4
1409	24	4	4.42	152	6.08	48	470	156.67	168	6.75%	3
1685	24	3	5.92	161	6.44	48	476	158.67	162	2.06%	5
1737	18	2	4.67	177	5.9	38	532	177.33	181	2.03%	3
2626	23	3	6.22	173	6.65	46	549	183	185	1.08%	4
5129	19	4	4.05	179	5.97	38	540	180	184	2.17%	3
5228	24	5	5.5	166	6.64	48	540	180	185	2.70%	4
6681	19	2	4.47	174	5.8	38	505	168.33	175	3.81%	2
7244	19	2	5	182	6.07	38	544	181.33	186	2.51%	4
7612	18	3	6	164	5.66	40	504	168	173	2.89%	2
Average	20.9	3.1	5.12	168.3	6.08	42.4	516.3	172.10	177.1	2.85%	3.4
Min	18	2	4.05	152	5.54	38	470	156.67	162	1.08%	2
Max	24	5	6.22	182	6.65	48	549	183	186	6.75%	5

Table A-44: 3 Teams, 60 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	27	5	4.85	178	5.56	45	572	190.67	195	2.22%	6
1409	29	4	4.31	192	6.4	48	596	198.67	208	4.49%	3
1685	28	3	6.07	191	6.16	46	570	190	192	1.04%	7
1737	23	2	4.78	206	5.89	40	625	208.33	210	0.79%	4
2626	27	3	6.56	211	6.59	45	655	218.33	223	2.09%	5
5129	23	4	3.96	205	5.69	38	615	205	210	2.38%	4
5228	29	5	5.07	216	7.2	48	679	226.33	235	3.69%	4
6681	23	2	4.43	215	5.97	38	630	210	216	2.78%	2
7244	25	3	5.28	207	6.09	41	630	210	211	0.47%	6
7612	22	3	5.95	195	5.57	40	595	198.33	204	2.78%	4
Average	25.6	3.4	5.13	201.6	6.11	42.9	616.7	205.57	210.4	2.27%	4.5
Min	22	2	3.96	178	5.56	38	570	190	192	0.47%	2

Table A-45: 3 Teams, 70 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	33	5	5.03	209	5.81	47	665	221.67	226	1.92%	7
1409	34	4	4.71	221	6.31	48	699	233	237	1.69%	4
1685	33	3	5.85	228	6.33	47	666	222	229	3.06%	8
1737	28	3	4.54	234	5.85	41	709	236.33	238	0.70%	4
2626	32	4	6	244	6.59	45	755	251.67	256	1.69%	7
5129	26	4	3.92	247	5.74	37	730	243.33	252	3.44%	4
5228	34	5	5.47	243	6.94	48	776	258.67	262	1.27%	5
6681	27	2	3.96	254	6.05	38	735	245	255	3.92%	3
7244	29	3	5.17	240	6	41	713	237.67	244	2.60%	7
7612	24	3	5.5	241	5.6	37	738	246	250	1.60%	4
Average	30	3.6	5.02	236.1	6.12	42.9	718.6	239.53	244.9	2.19%	5.3
Min	24	2	3.92	209	5.6	37	665	221.67	226	0.70%	3
Max	34	5	6	254	6.94	48	776	258.67	262	3.92%	8

Table A-46: 3 Teams, 80 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	37	5	4.89	245	5.83	46	778	259.33	262	1.02%	7
1409	40	4	5	260	6.67	50	809	269.67	276	2.29%	5
1685	37	3	5.41	273	6.5	46	802	267.33	274	2.43%	8
1737	32	3	4.28	280	6.09	41	835	278.33	284	2.00%	4
2626	36	4	5.81	278	6.47	45	867	289	290	0.34%	8
5129	30	4	3.67	265	5.41	37	805	268.33	270	0.62%	4
5228	37	5	5.22	286	6.81	46	899	299.67	305	1.75%	5
6681	31	2	4.06	289	6.02	38	862	287.33	290	0.92%	4
7244	32	3	4.75	270	5.74	40	811	270.33	274	1.34%	7
7612	30	3	5.17	262	5.57	40	812	270.67	271	0.12%	4
Average	34.2	3.6	4.83	270.8	6.11	42.9	828	276	279.6	1.28%	5.6
Min	30	2	3.67	245	5.41	37	778	259.33	262	0.12%	4
Max	40	5	5.81	289	6.81	50	899	299.67	305	2.43%	8

Table A-47: 3 Teams, 90 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	42	5	5.29	276	5.87	46	874	291.33	293	0.57%	9
1409	45	4	4.82	290	6.59	50	907	302.33	306	1.20%	5
1685	42	3	5.29	300	6.38	46	895	298.33	301	0.89%	9
1737	37	3	4.3	315	6.18	42	949	316.33	319	0.84%	6
2626	38	4	5.82	312	6.12	42	949	316.33	324	2.37%	9
5129	35	4	4.09	300	5.56	38	903	301	305	1.31%	6
5228	42	5	4.95	314	6.68	46	999	333	333	0.00%	5
6681	38	3	4.11	315	6.18	42	942	314	316	0.63%	5
7244	37	3	4.76	295	5.67	41	886	295.33	299	1.23%	7
7612	35	3	4.83	299	5.75	41	898	299.33	308	2.81%	5
Average	39.1	3.7	4.83	301.6	6.10	43.4	920.2	306.73	310.4	1.18%	6.6
Min	35	3	4.09	276	5.56	38	874	291.33	293	0.00%	5
Max	45	5	5.82	315	6.68	50	999	333	333	2.81%	9

Table A-48: 3 Teams, 100 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Avg. Waiting Time	Total Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	46	5	5.2	309	5.83	46	959	319.67	326	1.94%	9
1409	51	4	4.71	309	6.44	51	969	323	325	0.62%	6
1685	44	3	5.45	345	6.27	44	1025	341.67	346	1.25%	9
1737	42	3	4.57	361	6.45	43	1061	353.67	365	3.11%	7
2626	45	4	5.42	340	6.3	45	1040	346.67	352	1.52%	10
5129	39	4	4.23	338	5.63	39	1006	335.33	343	2.24%	7
5228	45	5	4.93	358	6.63	45	1110	370	377	1.86%	5
6681	41	3	4.15	365	6.29	41	1067	355.67	366	2.82%	6
7244	42	3	4.81	336	5.89	42	996	332	340	2.35%	7
7612	38	3	4.89	332	5.63	40	1010	336.67	341	1.27%	5
Average	43.3	3.7	4.84	339.3	6.14	43.6	1024.3	341.43	348.1	1.90%	7.1
Min	38	3	4.15	309	5.63	39	959	319.67	325	0.62%	5
Max	51	5	5.45	365	6.63	51	1110	370	377	3.11%	10

A2.7. Greedy Algorithm with Backtracking – 3 Teams

Table A-49: 3 Teams, 30 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	14	4	6.79	95	6.33	46	310	103.33	112	7.74%	1
1409	15	3	4.27	88	6.29	50	307	102.33	104	1.60%	4
1685	14	3	5.36	105	7	46	302	100.67	106	5.03%	2
1737	12	2	2.75	94	6.71	50	287	95.67	98	2.38%	2
2626	16	4	6.06	94	7.23	53	312	104	106	1.89%	3
5129	14	3	3.79	89	5.93	46	282	94	94	0.00%	1
5228	16	5	4.38	86	6.62	53	303	101	105	3.81%	1
6681	9	2	6.78	116	6.44	36	332	110.67	117	5.41%	2
7244	10	3	5.2	100	5.88	40	303	101	104	2.88%	1
7612	13	3	5.31	104	6.5	43	319	106.33	113	5.90%	2
Average	13.3	3.2	5.07	97.1	6.49	46.3	305.7	101.90	105.9	3.66%	1.9
Min	9	2	2.75	86	5.88	36	282	94	94	0.00%	1
Max	16	5	6.79	116	7.23	53	332	110.67	117	7.74%	4

Table A-50: 3 Teams, 40 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	18	4	5.94	120	5.71	45	404	134.67	137	1.70%	1
1409	19	3	4	114	5.7	47	381	127	130	2.31%	5
1685	20	3	5.2	133	7	50	387	129	134	3.73%	3
1737	16	3	2.25	132	6.6	47	392	130.67	136	3.92%	2
2626	21	4	7.19	131	7.28	52	417	139	143	2.80%	4
5129	17	3	4.12	141	6.41	42	419	139.67	146	4.34%	1
5228	22	5	3.5	114	6.71	55	398	132.67	133	0.25%	2
6681	17	3	5.53	141	7.05	47	423	141	142	0.70%	4
7244	14	3	5.29	141	6.13	40	428	142.67	145	1.61%	2
7612	18	3	5.33	125	5.95	45	389	129.67	134	3.23%	2
Average	18.2	3.4	4.84	129.2	6.45	47	403.8	134.60	138	2.46%	2.6
Min	14	3	2.25	114	5.7	40	381	127	130	0.25%	1
Max	22	5	7.19	141	7.28	55	428	142.67	146	4.34%	5

Table A-51: 3 Teams, 50 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	23	4	5.04	156	6	46	503	167.67	173	3.08%	2
1409	22	3	3.5	152	5.63	44	470	156.67	168	6.75%	5
1685	24	3	5.5	161	6.44	48	476	158.67	162	2.06%	4
1737	20	3	3.1	179	6.88	46	532	177.33	183	3.10%	2
2626	26	4	7	174	7.57	52	549	183.00	186	1.61%	6
5129	20	3	3.8	177	6.1	40	540	180.00	182	1.10%	1
5228	24	5	4.58	165	6.6	48	540	180	184	2.17%	2
6681	19	3	5.21	172	6.14	42	505	168.33	173	2.70%	4
7244	19	3	5.79	182	6.5	42	544	181.33	186	2.51%	3
7612	22	3	4.95	162	6	44	504	168	171	1.75%	2
Average	21.9	3.4	4.85	168	6.39	45.2	516.3	172.10	176.8	2.68%	3.1
Min	19	3	3.1	152	5.63	40	470	156.67	162	1.10%	1
Max	26	5	7	182	7.57	52	549	183	186	6.75%	6

Table A-52: 3 Teams, 60 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	26	4	4.69	176	5.33	43	572	190.67	193	1.21%	3
1409	27	3	3.44	186	5.81	45	596	198.67	202	1.65%	5
1685	31	3	5.23	192	6.86	51	570	190	193	1.55%	5
1737	23	3	2.91	205	6.21	43	625	208	209	0.32%	2
2626	29	4	6.59	208	6.93	48	655	218.33	220	0.76%	6
5129	25	3	3.64	204	6	41	615	205	209	1.91%	2
5228	27	5	4.85	213	6.66	45	679	226.33	232	2.44%	2
6681	24	3	5.08	211	6.39	43	630	210	212	0.94%	5
7244	24	4	5.5	207	6.27	43	630	210	211	0.47%	4
7612	27	3	4.74	190	5.94	45	595	198.33	199	0.34%	3
Average	26.3	3.5	4.67	199.2	6.24	44.7	616.7	205.57	208	1.16%	3.7
Min	23	3	2.91	176	5.33	41	570	190	193	0.32%	2
Max	31	5	6.59	213	6.93	51	679	226.33	232	2.44%	6

Table A-53: 3 Teams, 70 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	32	4	4.5	208	5.62	45	665	221.67	225	1.48%	3
1409	34	3	4	223	6.37	48	699	233	239	2.51%	8
1685	36	3	5.08	226	6.85	51	666	222	227	2.20%	6
1737	28	3	3.25	237	6.24	44	709	236.33	241	1.94%	3
2626	36	5	7.11	249	7.55	51	755	251.67	261	3.58%	7
5129	28	3	3.82	243	5.93	40	730	243.33	248	1.88%	2
5228	34	5	4.68	241	6.89	48	776	258.67	260	0.51%	5
6681	28	3	4.68	250	6.41	42	735	245	251	2.39%	5
7244	25	4	5.44	238	5.67	38	713	237.67	242	1.79%	5
7612	30	3	4.67	242	6.21	42	738	246	251	1.99%	3
Average	31.1	3.6	4.72	235.7	6.37	44.9	718.6	239.53	244.5	2.03%	4.7
Min	25	3	3.25	208	5.62	38	665	221.67	225	0.51%	2
Max	36	5	7.11	250	7.55	51	776	258.67	261	3.58%	8

Table A-54: 3 Teams, 80 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	37	4	4.92	248	5.9	46	778	259.33	265	2.14%	4
1409	38	3	4.05	259	6.32	47	809	269.67	275	1.94%	8
1685	41	3	5.56	276	7.26	51	802	267.33	277	3.49%	6
1737	31	3	3.39	283	6.29	42	835	278.33	287	3.02%	3
2626	42	5	6.83	280	7.57	52	867	289	292	1.03%	8
5129	32	3	3.53	265	5.64	40	805	268.33	270	0.62%	2
5228	37	5	4.7	288	6.86	46	899	299.67	307	2.39%	5
6681	34	3	5.41	291	6.77	45	862	287.33	292	1.60%	6
7244	28	4	5.54	268	5.47	37	811	270.33	272	0.61%	5
7612	35	3	4.4	266	6.05	43	812	270.67	275	1.58%	3
Average	35.5	3.6	4.83	272.4	6.41	44.9	828	276	281.2	1.84%	5
Min	28	3	3.39	248	5.47	37	778	259.33	265	0.61%	2
Max	42	5	6.83	291	7.57	52	899	299.67	307	3.49%	8

Table A-55: 3 Teams, 90 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	43	4	5.02	277	6.02	47	874	291.33	294	0.91%	5
1409	42	3	3.95	289	6.15	46	907	302.33	305	0.87%	9
1685	44	3	5.27	299	6.64	48	895	298.33	300	0.56%	6
1737	37	3	3.76	316	6.45	44	949	316.33	320	1.15%	5
2626	45	5	6.51	307	6.98	50	949	316.33	319	0.84%	8
5129	35	3	3.89	298	5.52	38	903	301	303	0.66%	3
5228	41	5	4.66	317	6.6	45	999	333	336	0.89%	6
6681	39	4	5.15	314	6.54	45	942	314	315	0.32%	7
7244	34	4	4.97	294	5.55	40	886	295.33	298	0.89%	6
7612	40	3	4.3	300	6.12	44	898	299.33	309	3.13%	4
Average	40	3.7	4.75	301.1	6.26	44.7	920.2	306.73	309.9	1.02%	5.9
Min	34	3	3.76	277	5.52	38	874	291.33	294	0.32%	3
Max	45	5	6.51	317	6.98	50	999	333.00	336	3.13%	9

Table A-56: 3 Teams, 100 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	48	5	4.75	308	6.04	48	959	319.67	325	1.64%	6
1409	45	3	3.89	309	5.72	45	969	323	325	0.62%	9
1685	48	3	5.13	345	6.76	48	1025	341.67	346	1.25%	6
1737	42	3	4.1	356	6.59	45	1061	353.67	360	1.76%	7
2626	50	5	6.26	337	6.88	50	1040	346.67	349	0.67%	9
5129	37	3	3.76	339	5.47	37	1006	335.33	344	2.52%	3
5228	45	5	4.58	357	6.61	45	1110	370	376	1.60%	6
6681	41	4	5	358	6.39	43	1067	355.67	359	0.93%	6
7244	37	4	5.11	330	5.5	39	996	332	334	0.60%	6
7612	44	3	4.5	334	6.07	44	1010	336.67	343	1.85%	5
Average	43.7	3.8	4.71	337.3	6.20	44.4	1024.3	341.43	346.1	1.34%	6.3
Min	37	3	3.76	308	5.47	37	959	319.67	325	0.60%	3
Max	50	5	6.26	358	6.88	50	1110	370	376	2.52%	9

A2.8. Greedy Algorithm with Post Ordering and Backtracking – 3 Teams

Table A-57: 3 Teams, 30 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	14	4	6.79	95	6.33	43	310	103.33	112	7.74%	1
1409	15	3	4.27	88	6.29	46	307	102.33	104	1.60%	4
1685	14	3	5.14	105	7	43	302	100.67	106	5.03%	2
1737	12	2	2.75	94	6.71	46	287	95.67	98	2.38%	2
2626	16	4	5.94	94	7.23	50	312	104	106	1.89%	3
5129	14	3	3.79	89	5.93	43	282	94	94	0.00%	1
5228	16	5	4.38	86	6.62	50	303	101	105	3.81%	1
6681	9	2	6.78	116	6.44	33	332	110.67	117	5.41%	2
7244	10	3	5.2	100	5.88	36	303	101	104	2.88%	1
7612	13	3	5.31	104	6.5	40	319	106.33	113	5.90%	2
Average	13.3	3.2	5.04	97.1	6.49	43	305.7	101.90	105.9	3.66%	1.9
Min	9	2	2.75	86	5.88	33	282	94	94	0.00%	1
Max	16	5	6.79	116	7.23	50	332	110.67	117	7.74%	4

Table A-58: 3 Teams, 40 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	18	4	5.94	120	5.71	42	404	134.67	137	1.70%	1
1409	19	3	4	114	5.7	45	381	127	130	2.31%	5
1685	20	3	5.05	133	7	47	387	129	134	3.73%	3
1737	16	3	2.25	132	6.6	45	392	130.67	136	3.92%	2
2626	21	4	7.1	131	7.28	50	417	139.00	143	2.80%	4
5129	17	3	4.12	141	6.41	40	419	140	146	4.34%	1
5228	22	5	3.5	114	6.71	52	398	132.67	133	0.25%	2
6681	16	4	5.13	141	6.71	42	423	141	142	0.70%	4
7244	14	3	5.29	141	6.13	37	428	142.67	145	1.61%	2
7612	18	3	5.33	125	5.95	42	389	129.67	134	3.23%	2
Average	18.1	3.5	4.77	129.2	6.42	44.2	403.8	134.60	138	2.46%	2.6
Min	14	3	2.25	114	5.7	37	381	127	130	0.25%	1
Max	22	5	7.1	141	7.28	52	428	142.67	146	4.34%	5

Table A-59: 3 Teams, 50 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack		Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	23	4	5.04	156	6	44	503	167.67	173	3.08%	2
1409	22	3	3.5	152	5.63	42	470	156.67	168	6.75%	5
1685	24	3	5.38	161	6.44	46	476	158.67	162	2.06%	4
1737	20	3	2.7	179	6.88	44	532	177.33	183	3.10%	2
2626	26	4	6.92	174	7.57	50	549	183	186	1.61%	6
5129	20	3	3.8	177	6.1	38	540	180	182	1.10%	1
5228	24	5	4.58	165	6.6	46	540	180	184	2.17%	2
6681	18	4	4.83	172	5.93	38	505	168	173	2.70%	4
7244	20	3	5.45	182	6.74	42	544	181.33	186	2.51%	3
7612	22	3	4.95	162	6	42	504	168	171	1.75%	2
Average	21.9	3.5	4.72	168	6.39	43.2	516.3	172.10	176.8	2.68%	3.1
Min	18	3	2.7	152	5.63	38	470	156.67	162	1.10%	1
Max	26	5	6.92	182	7.57	50	549	183	186	6.75%	6

Table A-60: 3 Teams, 60 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	26	4	4.69	176	5.33	41	572	190.67	193	1.21%	3
1409	27	3	3.44	186	5.81	43	596	198.67	202	1.65%	5
1685	31	3	5.13	192	6.86	50	570	190	193	1.55%	5
1737	23	3	2.57	205	6.21	41	625	208.33	209	0.32%	2
2626	29	4	6.52	208	6.93	46	655	218.33	220	0.76%	6
5129	25	3	3.64	204	6	40	615	205	209	1.91%	2
5228	27	5	4.85	213	6.66	43	679	226.33	232	2.44%	2
6681	23	4	4.78	211	6.21	40	630	210	212	0.94%	5
7244	25	4	5.24	207	6.47	43	630	210	211	0.47%	4
7612	27	3	4.74	190	5.94	43	595	198.33	199	0.34%	3
Average	26.3	3.6	4.56	199.2	6.24	43	616.7	205.57	208	1.16%	3.7
Min	23	3	2.57	176	5.33	40	570	190	193	0.32%	2
Max	31	5	6.52	213	6.93	50	679	226.33	232	2.44%	6

Table A-61: 3 Teams, 70 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	32	4	4.5	208	5.62	44	665	221.67	225	1.48%	3
1409	34	3	4	223	6.37	47	699	233	239	2.51%	8
1685	36	3	5	226	6.85	50	666	222	227	2.20%	6
1737	28	3	2.96	237	6.24	42	709	236.33	241	1.94%	3
2626	36	5	7.06	249	7.55	50	755	251.67	261	3.58%	7
5129	28	3	3.82	243	5.93	38	730	243.33	248	1.88%	2
5228	34	5	4.68	241	6.89	47	776	258.67	260	0.51%	5
6681	27	4	4.41	250	6.25	40	735	245	251	2.39%	5
7244	26	4	5.19	238	5.8	38	713	237.67	242	1.79%	5
7612	30	3	4.67	242	6.21	41	738	246	251	1.99%	3
Average	31.1	3.7	4.63	235.7	6.37	43.7	718.6	239.53	244.5	2.03%	4.7
Min	26	3	2.96	208	5.62	38	665	221.67	225	0.51%	2
Max	36	5	7.06	250	7.55	50	776	258.67	261	3.58%	8

Table A-62: 3 Teams, 80 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	37	4	4.92	248	5.9	45	778	259.33	265	2.14%	4
1409	38	3	4.05	259	6.32	46	809	269.67	275	1.94%	8
1685	42	3	5.14	276	7.46	51	802	267.33	277	3.49%	6
1737	31	3	3.13	283	6.29	41	835	278.33	287	3.02%	3
2626	42	5	6.79	280	7.57	51	867	289	292	1.03%	8
5129	32	3	3.53	265	5.64	38	805	268.33	270	0.62%	2
5228	37	5	4.7	288	6.86	45	899	299.67	307	2.39%	5
6681	34	4	4.71	291	6.77	43	862	287.33	292	1.60%	6
7244	29	4	5.31	268	5.58	37	811	270.33	272	0.61%	5
7612	35	3	4.4	266	6.05	42	812	270.67	275	1.58%	3
Average	35.7	3.7	4.67	272.4	6.44	43.9	828	276	281.2	1.84%	5
Min	29	3	3.13	248	5.58	37	778	259.33	265	0.61%	2
Max	42	5	6.79	291	7.57	51	899	299.67	307	3.49%	8

Table A-63: 3 Teams, 90 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	43	4	5.02	277	6.02	46	874	291.33	294	0.91%	5
1409	42	3	3.95	289	6.15	45	907	302.33	305	0.87%	9
1685	45	3	4.89	299	6.8	48	895	298.33	300	0.56%	6
1737	37	3	3.54	316	6.45	43	949	316.33	320	1.15%	5
2626	45	5	6.47	307	6.98	48	949	316.33	319	0.84%	8
5129	35	3	3.71	298	5.52	37	903	301	303	0.66%	3
5228	41	5	4.66	317	6.6	44	999	333	336	0.89%	6
6681	39	4	4.54	314	6.54	44	942	314	315	0.32%	7
7244	35	4	4.8	294	5.65	40	886	295.33	298	0.89%	6
7612	40	3	4.3	300	6.12	43	898	299.33	309	3.13%	4
Average	40.2	3.7	4.59	301.1	6.28	43.8	920.2	306.73	309.9	1.02%	5.9
Min	35	3	3.54	277	5.52	37	874	291.33	294	0.32%	3
Max	45	5	6.47	317	6.98	48	999	333	336	3.13%	9

Table A-64: 3 Teams, 100 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Avg. Waiting Time	Total Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	48	5	4.75	308	6.04	47	959	319.67	325	1.64%	6
1409	45	3	3.87	309	5.72	44	969	323	325	0.62%	9
1685	49	3	4.78	345	6.9	48	1025	341.67	346	1.25%	6
1737	42	3	3.9	356	6.59	44	1061	353.67	360	1.76%	7
2626	50	5	6.22	337	6.88	49	1040	346.67	349	0.67%	9
5129	37	3	3.59	339	5.47	36	1006	335.33	344	2.52%	3
5228	45	5	4.58	357	6.61	44	1110	370	376	1.60%	6
6681	41	4	4.41	358	6.39	42	1067	355.67	359	0.93%	6
7244	38	4	4.95	330	5.59	39	996	332	334	0.60%	6
7612	44	3	4.36	334	6.07	43	1010	336.67	343	1.85%	5
Average	43.9	3.8	4.54	337.3	6.23	43.6	1024.3	341.43	346.1	1.34%	6.3
Min	37	3	3.59	308	5.47	36	959	319.67	325	0.60%	3
Max	50	5	6.22	358	6.9	49	1110	370	376	2.52%	9

A2.9. Greedy Algorithm – 4 Teams

Table A-65: 4 Teams, 30 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	14	3	5.43	73	5.62	53	310	77.5	90	13.89%	1
1409	15	2	6.6	66	4.71	50	307	76.75	82	6.40%	3
1685	16	3	6.38	79	6.08	53	302	75.5	80	5.63%	4
1737	14	2	4.43	69	5.31	53	287	71.75	73	1.71%	3
2626	16	4	6.38	69	5.31	53	312	78	81	3.70%	2
5129	17	5	4.29	68	6.18	60	282	70.5	73	3.42%	2
5228	18	5	7.44	61	5.55	60	303	75.75	80	5.31%	2
6681	10	2	4.8	88	5.18	40	332	83	89	6.74%	0
7244	11	2	6.18	74	4.35	40	303	75.75	78	2.88%	2
7612	13	3	6.69	79	5.27	46	319	79.75	88	9.38%	2
Average	14.4	3.1	5.86	72.6	5.36	50.8	305.7	76.43	81.4	5.91%	2.1
Min	10	2	4.29	61	4.35	40	282	70.5	73	1.71%	0
Max	18	5	7.44	88	6.18	60	332	83	90	13.89%	4

Table A-66: 4 Teams, 40 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	20	3	5.75	86	5.38	57	404	101	103	1.94%	2
1409	22	4	6.23	81	4.76	55	381	95.25	97	1.80%	4
1685	23	3	6.39	99	6.19	57	387	96.75	100	3.25%	5
1737	18	5	5.17	102	5.37	50	392	98	106	7.55%	4
2626	22	4	6.45	97	5.71	55	417	104.25	109	4.36%	3
5129	20	5	4.65	109	6.06	52	419	104.75	114	8.11%	2
5228	24	5	7.33	87	5.8	60	398	99.5	106	6.13%	4
6681	15	2	4.87	109	4.95	42	423	105.75	110	3.86%	1
7244	16	3	5.44	106	5.05	45	428	107	110	2.73%	2
7612	20	3	6.45	90	5	52	389	97.25	99	1.77%	4
Average	20	3.7	5.87	96.6	5.43	52.5	403.8	100.95	105.4	4.15%	3.1
Min	15	2	4.65	81	4.76	42	381	95.25	97	1.77%	1
Max	24	5	7.33	109	6.19	60	428	107	114	8.11%	5

Table A-67: 4 Teams, 50 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	R	3	5.62	116	5.8	58	503	125.75	133	5.45%	4
1409	28	4	5.46	115	5.48	56	470	117.5	131	10.31%	5
1685	29	3	6.72	121	6.05	58	476	119	122	2.46%	7
1737	21	5	4.95	135	5.4	48	532	133	139	4.32%	4
2626	26	4	6.58	131	5.7	52	549	137.25	143	4.02%	3
5129	26	5	3.96	136	6.18	54	540	135	141	4.26%	2
5228	28	5	6.96	121	5.76	56	540	135	140	3.57%	4
6681	22	2	4.59	130	5.2	48	505	126.25	131	3.63%	3
7244	19	3	5.53	135	4.82	42	544	136	139	2.16%	2
7612	24	3	6	123	5.59	54	504	126	132	4.55%	5
Average	24.9	3.7	5.64	126.3	5.60	52.6	516.3	129.08	135.1	4.47%	3.9
Min	19	2	3.96	115	4.82	42	470	117.5	122	2.16%	2
Max	29	5	6.96	136	6.18	58	549	137.25	143	10.31%	7

Table A-68: 4 Teams, 60 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	34	3	5.56	129	5.86	61	572	143	146	2.05%	6
1409	35	4	5.63	141	5.88	58	596	149	157	5.10%	6
1685	34	3	6.82	146	5.84	56	570	142.5	147	3.06%	8
1737	28	5	5.36	157	5.61	51	625	156.25	161	2.95%	6
2626	30	4	6.83	159	5.48	50	655	163.75	171	4.24%	4
5129	31	5	4.23	157	5.81	53	615	153.75	162	5.09%	4
5228	31	5	6.81	157	5.61	51	679	169.75	176	3.55%	4
6681	26	2	4.69	159	5.3	48	630	157.5	160	1.56%	3
7244	26	4	5.46	156	5.03	46	630	157.5	160	1.56%	3
7612	30	3	5.9	146	5.62	55	595	148.75	155	4.03%	7
Average	30.5	3.8	5.73	150.7	5.60	52.9	616.7	154.18	159.5	3.32%	5.1
Min	26	2	4.23	129	5.03	46	570	142.5	146	1.56%	3
Max	35	5	6.83	159	5.88	61	679	169.75	176	5.10%	8

Table A-69: 4 Teams, 70 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	40	7	5.45	156	6	61	665	166.25	173	3.90%	6
1409	40	4	5.9	162	5.59	57	699	174.75	178	1.83%	6
1685	42	4	6.43	172	6.37	60	666	166.5	173	3.76%	10
1737	34	5	5.12	178	5.56	52	709	177.25	182	2.61%	6
2626	37	5	6.59	183	5.72	52	755	188.75	195	3.21%	6
5129	37	5	4.22	187	6.03	54	730	182.5	192	4.95%	5
5228	37	5	7	182	5.69	52	776	194	201	3.48%	6
6681	29	2	4.86	191	5.31	47	735	183.75	192	4.30%	3
7244	31	4	5.1	182	5.06	47	713	178.25	186	4.17%	4
7612	33	3	5.55	184	5.58	51	738	184.5	193	4.40%	7
Average	36	4.4	5.62	177.7	5.69	53.3	718.6	179.65	186.5	3.66%	5.9
Min	29	2	4.22	156	5.06	47	665	166.25	173	1.83%	3
Max	42	7	7	191	6.37	61	776	194	201	4.95%	10

Table A-70: 4 Teams, 80 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	45	7	5.24	186	6.2	61	778	194.5	203	4.19%	7
1409	46	4	6.11	194	5.88	57	809	202.25	210	3.69%	7
1685	46	4	6.02	207	6.27	57	802	200.5	208	3.61%	10
1737	40	5	4.95	211	5.86	53	835	208.75	215	2.91%	7
2626	43	5	6.42	209	5.81	53	867	216.75	221	1.92%	6
5129	40	5	4.38	198	5.5	53	805	201.25	203	0.86%	6
5228	39	5	7	216	5.54	50	899	224.75	235	4.36%	6
6681	34	2	5	220	5.37	47	862	215.5	221	2.49%	3
7244	33	4	5.12	201	4.57	43	811	202.75	205	1.10%	4
7612	40	4	5.53	196	5.44	53	812	203	205	0.98%	9
Average	40.6	4.5	5.58	203.8	5.64	52.7	828	207	212.6	2.61%	6.5
Min	33	2	4.38	186	4.57	43	778	194.5	203	0.86%	3
Max	46	7	7	220	6.27	61	899	224.75	235	4.36%	10

Table A-71: 4 Teams, 90 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	52	7	5.54	205	6.41	63	874	218.5	222	1.58%	9
1409	51	4	5.96	216	5.68	56	907	226.75	232	2.26%	8
1685	53	4	6.15	229	6.36	58	895	223.75	230	2.72%	11
1737	45	5	5.11	236	5.9	54	949	237.25	240	1.15%	7
2626	46	5	6.41	234	5.44	51	949	237.25	246	3.56%	7
5129	47	5	4.55	224	5.74	55	903	225.75	229	1.42%	6
5228	47	5	6.49	233	5.68	53	999	249.75	252	0.89%	7
6681	41	3	5.41	238	5.41	50	942	235.5	239	1.46%	6
7244	37	4	5.16	224	4.57	44	886	221.5	228	2.85%	5
7612	46	4	4.98	221	5.53	54	898	224.5	230	2.39%	9
Average	46.5	4.6	5.58	226	5.67	53.8	920.2	230.05	234.8	2.03%	7.5
Min	37	3	4.55	205	4.57	44	874	218.5	222	0.89%	5
Max	53	7	6.49	238	6.41	63	999	249.75	252	3.56%	11

Table A-72: 4 Teams, 100 Parts (Greedy Only)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	59	7	5.51	229	6.54	64	959	239.75	246	2.54%	12
1409	58	4	5.86	229	5.59	58	969	242.25	245	1.12%	10
1685	56	4	6.21	262	6.09	56	1025	256.25	263	2.57%	11
1737	50	5	5.44	271	6.02	54	1061	265.25	275	3.55%	8
2626	52	5	6.23	252	5.48	53	1040	260.00	264	1.52%	8
5129	54	5	4.56	255	6.07	57	1006	251.50	260	3.27%	7
5228	51	5	6.39	265	5.76	53	1110	277.50	284	2.29%	7
6681	45	3	5.22	273	5.46	49	1067	266.75	274	2.65%	6
7244	41	4	5.27	251	4.74	46	996	249.00	255	2.35%	5
7612	51	4	4.82	251	5.58	54	1010	252.50	260	2.88%	9
Average	51.7	4.6	5.55	253.8	5.73	54.4	1024.3	256.08	262.6	2.47%	8.3
Min	41	3	4.56	229	4.74	46	959	239.75	245	1.12%	5
Max	59	7	6.39	273	6.54	64	1110	277.5	284	3.55%	12

A2.10. Greedy Algorithm with Post Ordering – 4 Teams

Table A-73: 4 Teams, 30 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	16	3	4.75	73	5.62	53	310	77.5	90	13.89%	1
1409	15	2	6.6	66	4.71	50	307	76.75	82	6.40%	3
1685	16	3	6.38	79	6.08	53	302	75.5	80	5.63%	4
1737	14	2	4.43	69	5.31	53	287	71.75	73	1.71%	3
2626	16	4	6.38	69	5.31	53	312	78	81	3.70%	2
5129	18	5	4.06	68	6.18	60	282	70.5	73	3.42%	2
5228	18	5	7.44	61	5.55	60	303	75.75	80	5.31%	2
6681	11	2	4.36	88	5.18	40	332	83	89	6.74%	0
7244	12	2	5.67	74	4.35	40	303	75.75	78	2.88%	2
7612	13	3	6.69	79	5.27	46	319	79.75	88	9.38%	2
Average	14.9	3.1	5.68	72.6	5.36	50.8	305.7	76.43	81.4	5.91%	2.1
Minimum	11	2	4.06	61	4.35	40	282	70.5	73	1.71%	0
Maximum	18	5	7.44	88	6.18	60	332	83	90	13.89%	4

Table A-74: 4 Teams, 40 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	23	3	5	86	5.38	57	404	101	103	1.94%	2
1409	22	4	6.23	81	4.76	55	381	95.25	97	1.80%	4
1685	23	3	6.39	99	6.19	57	387	96.75	100	3.25%	5
1737	18	5	5.17	102	5.37	50	392	98	106	7.55%	4
2626	22	4	6.45	97	5.71	55	417	104.25	109	4.36%	3
5129	21	5	4.43	109	6.06	52	419	104.75	114	8.11%	2
5228	24	5	7.33	87	5.8	60	398	99.5	106	6.13%	4
6681	16	2	4.56	109	4.95	42	423	105.75	110	3.86%	1
7244	18	3	4.83	106	5.05	45	428	107	110	2.73%	2
7612	20	3	6.45	90	5	52	389	97.25	99	1.77%	4
Average	20.7	3.7	5.68	96.6	5.43	52.5	403.8	100.95	105.4	4.15%	3.1
Minimum	16	2	4.43	81	4.76	42	381	95.25	97	1.77%	1
Maximum	24	5	7.33	109	6.19	60	428	107	114	8.11%	5

Table A-75: 4 Teams, 50 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	29	3	5.03	116	5.8	58	503	125.75	133	5.45%	4
1409	28	4	5.46	115	5.48	56	470	117.5	131	10.31%	5
1685	29	3	6.72	121	6.05	58	476	119	122	2.46%	7
1737	22	5	4.73	135	5.4	48	532	133	139	4.32%	4
2626	26	4	6.58	131	5.7	52	549	137.25	143	4.02%	3
5129	27	5	3.81	136	6.18	54	540	135	141	4.26%	2
5228	28	5	6.96	121	5.76	56	540	135	140	3.57%	4
6681	23	2	4.39	130	5.2	48	505	126.25	131	3.63%	3
7244	21	3	5	135	4.82	42	544	136	139	2.16%	2
7612	26	3	5.54	123	5.59	54	504	126	132	4.55%	5
Average	25.9	3.7	5.42	126.3	5.60	52.6	516.3	129.08	135.1	4.47%	3.9
Minimum	21	2	3.81	115	4.82	42	470	117.5	122	2.16%	2
Maximum	29	5	6.96	136	6.18	58	549	137.25	143	10.31%	7

Table A-76: 4 Teams, 60 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	37	3	5.11	129	5.86	61	572	143	146	2.05%	6
1409	35	4	5.63	141	5.88	58	596	149	157	5.10%	6
1685	34	3	6.82	146	5.84	56	570	142.5	147	3.06%	8
1737	29	5	5.17	157	5.61	51	625	156.25	161	2.95%	6
2626	30	4	6.83	159	5.48	50	655	163.75	171	4.24%	4
5129	32	5	4.09	157	5.81	53	615	153.75	162	5.09%	4
5228	31	5	6.81	157	5.61	51	679	169.75	176	3.55%	4
6681	28	2	4.36	159	5.3	48	630	157.5	160	1.56%	3
7244	28	4	5.07	156	5.03	46	630	157.5	160	1.56%	3
7612	32	3	5.53	146	5.62	55	595	148.75	155	4.03%	7
Average	31.6	3.8	5.54	150.7	5.60	52.9	616.7	154.18	159.5	3.32%	5.1
Minimum	28	2	4.09	129	5.03	46	570	142.5	146	1.56%	3
Maximum	37	5	6.83	159	5.88	61	679	169.75	176	5.10%	8

Table A-77: 4 Teams, 70 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	43	7	5.07	156	6	61	665	166.25	173	3.90%	6
1409	40	4	5.9	162	5.59	57	699	174.75	178	1.83%	6
1685	42	4	6.43	172	6.37	60	666	166.5	173	3.76%	10
1737	35	5	4.97	178	5.56	52	709	177.25	182	2.61%	6
2626	37	5	6.59	183	5.72	52	755	188.75	195	3.21%	6
5129	38	5	4.11	187	6.03	54	730	182.5	192	4.95%	5
5228	37	5	7	182	5.69	52	776	194	201	3.48%	6
6681	32	2	4.41	191	5.31	47	735	183.75	192	4.30%	3
7244	33	4	4.79	182	5.06	47	713	178.25	186	4.17%	4
7612	35	3	5.23	184	5.58	51	738	184.5	193	4.40%	7
Average	37.2	4.4	5.45	177.7	5.69	53.3	718.6	179.65	186.5	3.66%	5.9
Minimum	32	2	4.11	156	5.06	47	665	166.25	173	1.83%	3
Maximum	43	7	7	191	6.37	61	776	194	201	4.95%	10

Table A-78: 4 Teams, 80 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	49	7	4.82	186	6.2	61	778	194.5	203	4.19%	7
1409	46	4	6.11	194	5.88	57	809	202.25	210	3.69%	7
1685	46	4	6.02	207	6.27	57	802	200.5	208	3.61%	10
1737	41	5	4.83	211	5.86	53	835	208.75	215	2.91%	7
2626	43	5	6.42	209	5.81	53	867	216.75	221	1.92%	6
5129	43	5	4.07	198	5.5	53	805	201.25	203	0.86%	6
5228	40	5	6.83	216	5.54	50	899	224.75	235	4.36%	6
6681	37	2	4.59	220	5.37	47	862	215.5	221	2.49%	3
7244	35	4	4.83	201	4.57	43	811	202.75	205	1.10%	4
7612	42	4	5.26	196	5.44	53	812	203	205	0.98%	9
Average	42.2	4.5	5.38	203.8	5.64	52.7	828	207	212.6	2.61%	6.5
Minimum	35	2	4.07	186	4.57	43	778	194.5	203	0.86%	3
Maximum	49	7	6.83	220	6.27	61	899	224.75	235	4.36%	10

Table A-79: 4 Teams, 90 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	57	7	5.05	205	6.41	63	874	218.5	222	1.58%	9
1409	51	4	5.96	216	5.68	56	907	226.75	232	2.26%	8
1685	53	4	6.15	229	6.36	58	895	223.75	230	2.72%	11
1737	47	5	4.89	236	5.9	54	949	237.25	240	1.15%	7
2626	46	5	6.41	234	5.44	51	949	237.25	246	3.56%	7
5129	50	5	4.28	224	5.74	55	903	225.75	229	1.42%	6
5228	48	5	6.35	233	5.68	53	999	249.75	252	0.89%	7
6681	44	3	5.05	238	5.41	50	942	235.5	239	1.46%	6
7244	40	4	4.78	224	4.57	44	886	221.5	228	2.85%	5
7612	48	4	4.77	221	5.53	54	898	224.5	230	2.39%	9
Average	48.4	4.6	5.37	226	5.67	53.8	920.2	230.05	234.8	2.03%	7.5
Minimum	40	3	4.28	205	4.57	44	874	218.5	222	0.89%	5
Maximum	57	7	6.41	238	6.41	63	999	249.75	252	3.56%	11

Table A-80: 4 Teams, 100 Parts (Greedy with Post Ordering)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	64	7	5.08	229	6.54	64	959	239.75	246	2.54%	12
1409	58	4	5.86	229	5.59	58	969	242.25	245	1.12%	10
1685	56	4	6.21	262	6.09	56	1025	256.25	263	2.57%	11
1737	52	5	5.23	271	6.02	54	1061	265.25	275	3.55%	8
2626	53	5	6.11	252	5.48	53	1040	260	264	1.52%	8
5129	57	5	4.32	255	6.07	57	1006	251.5	260	3.27%	7
5228	53	5	6.15	265	5.76	53	1110	277.5	284	2.29%	7
6681	48	3	4.9	273	5.46	49	1067	266.75	274	2.65%	6
7244	46	4	4.7	251	4.74	46	996	249	255	2.35%	5
7612	53	4	4.64	251	5.58	54	1010	252.5	260	2.88%	9
Average	54	4.6	5.32	253.8	5.73	54.4	1024.3	256.08	262.6	2.47%	8.3
Minimum	46	3	4.32	229	4.74	46	959	239.75	245	1.12%	5
Maximum	64	7	6.21	273	6.54	64	1110	277.5	284	3.55%	12

A2.11. Greedy Algorithm with Backtracking – 4 Teams

Table A-81: 4 Teams, 30 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	16	6	5.06	71	5.46	53	310	77.5	88	11.93%	4
1409	14	3	5.5	67	4.47	46	307	76.75	83	7.53%	1
1685	16	3	5.19	80	6.15	53	302	75.5	81	6.79%	3
1737	13	2	2.77	69	4.93	50	287	71.75	73	1.71%	1
2626	17	4	3.94	68	5.67	56	312	78	80	2.50%	3
5129	19	4	5.37	68	6.8	63	282	70.5	73	3.42%	4
5228	19	6	5.37	61	6.1	63	303	75.75	80	5.31%	5
6681	11	2	4.45	88	4.89	36	332	83	89	6.74%	1
7244	16	3	5.5	75	5.77	53	303	75.75	79	4.11%	5
7612	17	4	5.41	78	7.09	60	319	79.75	87	8.33%	4
Average	15.8	3.7	4.86	72.5	5.73	53.3	305.7	76.425	81.3	5.84%	3.1
Minimum	11	2	2.77	61	4.47	36	282	70.5	73	1.71%	1
Maximum	19	6	5.5	88	7.09	63	332	83	89	11.93%	5

Table A-82: 4 Teams, 40 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	23	6	5.43	91	5.69	57	404	101	108	6.48%	4
1409	20	4	5.2	82	4.32	50	381	95.25	98	2.81%	3
1685	22	3	5.23	98	5.76	55	387	96.75	99	2.27%	6
1737	17	2	2.82	101	5.05	47	392	98	105	6.67%	1
2626	21	4	4.1	96	5.33	52	417	104.25	108	3.47%	4
5129	24	4	4.58	105	7	60	419	104.75	110	4.77%	4
5228	24	6	5.67	86	5.73	60	398	99.5	105	5.24%	6
6681	19	3	3.58	110	5.5	47	423	105.75	111	4.73%	2
7244	22	4	4.77	104	6.12	55	428	107	108	0.93%	5
7612	23	5	5.52	90	6	60	389	97.25	99	1.77%	5
Average	21.5	4.1	4.69	96.3	5.65	54.3	403.8	100.95	105.1	3.91%	4
Minimum	17	2	2.82	82	4.32	47	381	95.25	98	0.93%	1
Maximum	24	6	5.67	110	7	60	428	107	111	6.67%	6

Table A-83: 4 Teams, 50 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	27	6	4.7	114	5.18	54	503	125.75	131	4.01%	5
1409	27	4	4.44	113	5.14	54	470	117.5	129	8.91%	4
1685	28	3	5	120	5.71	56	476	119	121	1.65%	6
1737	21	2	3	135	5.19	46	532	133	139	4.32%	1
2626	27	4	4.41	129	5.86	54	549	137.25	141	2.66%	5
5129	28	4	4.29	135	6.43	56	540	135	140	3.57%	4
5228	26	6	5.31	122	5.3	52	540	135	141	4.26%	6
6681	25	3	3.28	128	5.33	50	505	126.25	129	2.13%	3
7244	27	4	5.26	139	6.32	54	544	136	143	4.90%	7
7612	27	5	5.26	123	5.86	56	504	126	132	4.55%	5
Average	26.3	4.1	4.50	125.8	5.63	53.2	516.3	129.08	134.6	4.10%	4.6
Minimum	21	2	3	113	5.14	46	470	117.5	121	1.65%	1
Maximum	28	6	5.31	139	6.43	56	549	137.25	143	8.91%	7

Table A-84: 4 Teams, 60 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	34	6	4.35	128	5.12	56	572	143	145	1.38%	7
1409	33	4	3.94	139	5.35	55	596	149	155	3.87%	4
1685	35	3	5.03	147	6.13	58	570	142.5	148	3.72%	8
1737	25	2	2.88	154	4.81	45	625	156.25	158	1.11%	1
2626	31	4	4.35	154	5.5	51	655	163.75	166	1.36%	5
5129	34	4	3.76	155	6.2	56	615	153.75	160	3.91%	5
5228	31	6	4.84	158	5.64	51	679	169.75	177	4.10%	6
6681	29	3	3.76	164	5.47	48	630	157.5	165	4.55%	4
7244	34	4	4.79	155	6.2	56	630	157.5	159	0.94%	7
7612	34	5	4.41	142	5.92	58	595	148.75	151	1.49%	5
Average	32	4.1	4.21	149.6	5.63	53.4	616.7	154.18	158.4	2.64%	5.2
Minimum	25	2	2.88	128	4.81	45	570	142.5	145	0.94%	1
Maximum	35	6	5.03	164	6.2	58	679	169.75	177	4.55%	8

Table A-85: 4 Teams, 70 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	40	6	4.3	154	5.31	57	665	166.25	171	2.78%	7
1409	39	4	3.87	165	5.5	55	699	174.75	181	3.45%	4
1685	41	4	4.8	170	6.07	58	666	166.5	171	2.63%	10
1737	30	2	3.07	180	4.86	45	709	177.25	184	3.67%	1
2626	38	4	4.39	187	6.03	54	755	188.75	199	5.15%	7
5129	38	4	3.79	185	5.97	54	730	182.5	190	3.95%	5
5228	37	6	5	180	5.63	52	776	194	199	2.51%	7
6681	32	3	3.66	186	5.03	45	735	183.75	187	1.74%	4
7244	35	4	4.63	181	5.32	50	713	178.25	185	3.65%	7
7612	39	5	4.59	188	6.48	57	738	184.5	197	6.35%	6
Average	36.9	4.2	4.21	177.6	5.62	52.7	718.6	179.65	186.4	3.59%	5.8
Minimum	30	2	3.07	154	4.86	45	665	166.25	171	1.74%	1
Maximum	41	6	5	188	6.48	58	776	194	199	6.35%	10

Table A-86: 4 Teams, 80 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	46	6	4.02	179	5.42	57	778	194.5	196	0.77%	7
1409	45	4	3.78	195	5.74	56	809	202.25	211	4.15%	4
1685	46	4	4.8	203	6.15	57	802	200.5	204	1.72%	10
1737	36	4	3.28	214	5.22	47	835	208.75	218	4.24%	4
2626	44	4	4.64	207	5.91	55	867	216.75	219	1.03%	8
5129	43	4	3.6	198	5.5	53	805	201.25	203	0.86%	5
5228	42	6	4.86	216	5.84	52	899	224.75	235	4.36%	9
6681	37	3	4.32	222	5.29	46	862	215.5	223	3.36%	4
7244	40	4	4.65	203	5.21	50	811	202.75	207	2.05%	8
7612	47	5	4.49	198	6.39	60	812	203	207	1.93%	8
Average	42.6	4.4	4.24	203.5	5.67	53.3	828	207	212.3	2.45%	6.7
Minimum	36	3	3.28	179	5.21	46	778	194.5	196	0.77%	4
Maximum	47	6	4.86	222	6.39	60	899	224.75	235	4.36%	10

Table A-87: 4 Teams, 90 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	53	6	4.49	207	5.75	58	874	218.5	224	2.46%	9
1409	51	4	3.71	215	5.66	56	907	226.75	231	1.84%	5
1685	54	4	4.67	225	6.43	60	895	223.75	226	1.00%	12
1737	43	4	3.67	239	5.43	50	949	237.25	243	2.37%	5
2626	50	4	4.38	231	5.92	55	949	237.25	243	2.37%	9
5129	48	4	3.85	223	5.44	53	903	225.75	228	0.99%	6
5228	50	6	4.94	233	5.97	55	999	249.75	252	0.89%	10
6681	44	5	4.2	238	5.29	48	942	235.5	239	1.46%	7
7244	45	4	4.36	221	5.02	50	886	221.5	225	1.56%	8
7612	52	5	4.27	222	6.17	58	898	224.5	231	2.81%	8
Average	49	4.6	4.25	225.4	5.71	54.3	920.2	230.05	234.2	1.77%	7.9
Minimum	43	4	3.67	207	5.02	48	874	218.5	224	0.89%	5
Maximum	54	6	4.94	239	6.43	60	999	249.75	252	2.81%	12

Table A-88: 4 Teams, 100 Parts (Greedy with Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Double Handled
1256	59	6	4.24	224	5.6	59	959	239.75	241	0.52%	10
1409	57	4	3.6	228	5.43	57	969	242.25	244	0.72%	6
1685	58	4	4.6	259	6.32	58	1025	256.25	260	1.44%	13
1737	48	4	3.52	271	5.53	50	1061	265.25	275	3.55%	7
2626	54	5	4.54	253	5.62	54	1040	260	265	1.89%	10
5129	52	4	3.79	255	5.43	52	1006	251.5	260	3.27%	6
5228	55	6	4.82	266	6.05	55	1110	277.5	285	2.63%	11
6681	49	5	4.02	272	5.44	49	1067	266.75	273	2.29%	7
7244	50	4	4.42	249	5.08	50	996	249	253	1.58%	9
7612	56	5	4.25	249	5.93	57	1010	252.5	258	2.13%	8
Average	53.8	4.7	4.18	252.6	5.64	54.1	1024.3	256.08	261.4	2.00%	8.7
Minimum	48	4	3.52	224	5.08	49	959	239.75	241	0.52%	6
Maximum	59	6	4.82	272	6.32	59	1110	277.5	285	3.55%	13

A2.12. Greedy Algorithm with Post Ordering and Backtracking – 4 Teams

Table A-89: 4 Teams, 30 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	16	6	5.06	71	5.46	50	310	77.5	88	11.93%	4
1409	14	3	5.5	67	4.47	43	307	76.75	83	7.53%	1
1685	16	3	5.19	80	6.15	50	302	75.5	81	6.79%	3
1737	13	2	2.77	69	4.93	46	287	71.75	73	1.71%	1
2626	17	4	3.94	68	5.67	53	312	78	80	2.50%	3
5129	19	4	5.37	68	6.8	60	282	70.5	73	3.42%	4
5228	19	6	5.37	61	6.1	60	303	75.75	80	5.31%	5
6681	11	2	4.45	88	4.89	33	332	83	89	6.74%	1
7244	16	3	5.5	75	5.77	50	303	75.75	79	4.11%	5
7612	15	4	5.07	78	6	50	319	79.75	87	8.33%	4
Average	15.6	3.7	4.82	72.5	5.62	49.5	305.7	76.43	81.3	5.84%	3.1
Min	11	2	2.77	61	4.47	33	282	70.5	73	1.71%	1
Max	19	6	5.5	88	6.8	60	332	83	89	11.93%	5

Table A-90: 4 Teams, 40 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	23	6	5.43	91	5.69	55	404	101	108	6.48%	4
1409	20	4	5.2	82	4.32	47	381	95.25	98	2.81%	3
1685	22	3	5.23	98	5.76	52	387	96.75	99	2.27%	6
1737	17	2	2.82	101	5.05	45	392	98	105	6.67%	1
2626	21	4	4.1	96	5.33	50	417	104.25	108	3.47%	4
5129	24	4	4.58	105	7	57	419	104.75	110	4.77%	4
5228	24	6	5.67	86	5.73	57	398	99.5	105	5.24%	6
6681	19	3	3.58	110	5.5	45	423	105.75	111	4.73%	2
7244	21	4	4.48	104	5.78	50	428	107	108	0.93%	5
7612	21	4	5.29	90	5.29	52	389	97.25	99	1.77%	5
Average	21.2	4	4.64	96.3	5.55	51	403.8	100.95	105.1	3.91%	4
Min	17	2	2.82	82	4.32	45	381	95.25	98	0.93%	1
Max	24	6	5.67	110	7	57	428	107	111	6.67%	6

Table A-91: 4 Teams, 50 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	27	6	4.7	114	5.18	52	503	125.75	131	4.01%	5
1409	27	4	4.44	113	5.14	52	470	117.5	129	8.91%	4
1685	28	3	5	120	5.71	54	476	119	121	1.65%	6
1737	21	2	3	135	5.19	44	532	133	139	4.32%	1
2626	27	4	4.41	129	5.86	52	549	137.25	141	2.66%	5
5129	28	4	4.29	135	6.43	54	540	135	140	3.57%	4
5228	26	6	5.31	122	5.3	50	540	135	141	4.26%	6
6681	25	3	3.28	128	5.33	48	505	126.25	129	2.13%	3
7244	26	4	5.04	139	6.04	50	544	136	143	4.90%	7
7612	25	4	5.04	123	5.35	50	504	126	132	4.55%	5
Average	26	4	4.45	125.8	5.55	50.6	516.3	129.08	134.6	4.10%	4.6
Min	21	2	3	113	5.14	44	470	117.5	121	1.65%	1
Max	28	6	5.31	139	6.43	54	549	137.25	143	8.91%	7

Table A-92: 4 Teams, 60 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	34	6	4.35	128	5.12	55	572	143	145	1.38%	7
1409	33	4	3.94	139	5.35	53	596	149	155	3.87%	4
1685	35	3	5.03	147	6.13	56	570	142.5	148	3.72%	8
1737	25	2	2.88	154	4.81	43	625	156.25	158	1.11%	1
2626	31	4	4.35	154	5.5	50	655	163.75	166	1.36%	5
5129	34	4	3.76	155	6.2	55	615	153.75	160	3.91%	5
5228	31	6	4.84	158	5.64	50	679	169.75	177	4.10%	6
6681	29	3	3.76	164	5.47	46	630	157.5	165	4.55%	4
7244	33	4	4.61	155	5.96	53	630	157.5	159	0.94%	7
7612	32	4	4.19	142	5.46	53	595	148.75	151	1.49%	5
Average	31.7	4	4.17	149.6	5.56	51.4	616.7	154.18	158.4	2.64%	5.2
Min	25	2	2.88	128	4.81	43	570	142.5	145	0.94%	1
Max	35	6	5.03	164	6.2	56	679	169.75	177	4.55%	8

Table A-93: 4 Teams, 70 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	40	6	4.3	154	5.31	55	665	166.25	171	2.78%	7
1409	39	4	3.87	165	5.5	54	699	174.75	181	3.45%	4
1685	41	4	4.8	170	6.07	57	666	166.5	171	2.63%	10
1737	30	2	3.07	180	4.86	44	709	177.25	184	3.67%	1
2626	38	4	4.39	187	6.03	52	755	188.75	199	5.15%	7
5129	38	4	3.79	185	5.97	52	730	182.5	190	3.95%	5
5228	37	6	5	180	5.63	51	776	194	199	2.51%	7
6681	32	3	3.66	186	5.03	44	735	183.75	187	1.74%	4
7244	34	4	4.44	181	5.17	47	713	178.25	185	3.65%	7
7612	37	4	4.41	188	6.06	52	738	184.5	197	6.35%	6
Average	36.6	4.1	4.17	177.6	5.56	50.8	718.6	179.65	186.4	3.59%	5.8
Min	30	2	3.07	154	4.86	44	665	166.25	171	1.74%	1
Max	41	6	5	188	6.07	57	776	194	199	6.35%	10

`Table A-94: 4 Teams, 80 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	46	6	4.02	179	5.42	56	778	194.5	196	0.77%	7
1409	45	4	3.78	195	5.74	55	809	202.25	211	4.15%	4
1685	46	4	4.8	203	6.15	56	802	200.5	204	1.72%	10
1737	36	4	3.28	214	5.22	46	835	208.75	218	4.24%	4
2626	44	4	4.64	207	5.91	53	867	216.75	219	1.03%	8
5129	43	4	3.6	198	5.5	52	805	201.25	203	0.86%	5
5228	42	6	4.86	216	5.84	51	899	224.75	235	4.36%	9
6681	37	3	4.32	222	5.29	45	862	215.5	223	3.36%	4
7244	39	4	4.49	203	5.08	47	811	202.75	207	2.05%	8
7612	45	5	4.33	198	6	56	812	203	207	1.93%	8
Average	42.3	4.4	4.21	203.5	5.62	51.7	828	207	212.3	2.45%	6.7
Min	36	3	3.28	179	5.08	45	778	194.5	196	0.77%	4
Max	46	6	4.86	222	6.15	56	899	224.75	235	4.36%	10

Table A-95: 4 Teams, 90 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	53	6	4.49	207	5.75	57	874	218.5	224	2.46%	9
1409	51	4	3.71	215	5.66	55	907	226.75	231	1.84%	5
1685	54	4	4.67	225	6.43	58	895	223.75	226	1.00%	12
1737	43	4	3.67	239	5.43	48	949	237.25	243	2.37%	5
2626	50	4	4.3	231	5.92	54	949	237.25	243	2.37%	9
5129	48	4	3.85	223	5.44	52	903	225.75	228	0.99%	6
5228	50	6	4.94	233	5.97	54	999	249.75	252	0.89%	10
6681	44	5	4.2	238	5.29	47	942	235.5	239	1.46%	7
7244	44	4	4.2	221	4.91	47	886	221.5	225	1.56%	8
7612	50	5	4.12	222	5.84	55	898	224.5	231	2.81%	8
Average	48.7	4.6	4.22	225.4	5.66	52.7	920.2	230.05	234.2	1.77%	7.9
Min	43	4	3.67	207	4.91	47	874	218.5	224	0.89%	5
Max	54	6	4.94	239	6.43	58	999	249.75	252	2.81%	12

Table A-96: 4 Teams, 100 Parts (Greedy with Post Ordering & Backtracking)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Makespan	% Difference	Reduced Double Handled
1256	59	6	4.24	224	5.6	58	959	239.75	241	0.52%	10
1409	57	4	3.6	228	5.43	56	969	242.25	244	0.72%	6
1685	58	4	4.6	259	6.32	57	1025	256.25	260	1.44%	13
1737	48	4	3.52	271	5.53	49	1061	265.25	275	3.55%	7
2626	54	5	4.46	253	5.62	53	1040	260	265	1.89%	10
5129	52	4	3.79	255	5.43	51	1006	251.5	260	3.27%	6
5228	55	6	4.82	266	6.05	54	1110	277.5	285	2.63%	11
6681	49	5	4.02	272	5.44	48	1067	266.75	273	2.29%	7
7244	49	4	4.29	249	4.98	48	996	249	253	1.58%	9
7612	54	5	4.11	249	5.66	54	1010	252.5	258	2.13%	8
Average	53.5	4.7	4.15	252.6	5.61	52.8	1024.3	256.08	261.4	2.00%	8.7
Min	48	4	3.52	224	4.98	48	959	239.75	241	0.52%	6
Max	59	6	4.82	272	6.32	58	1110	277.5	285	3.55%	13

A2.13. Genetic Algorithm (Makespan/Max in Stack/JIT Priority) – 2 Teams

Table A-97: 2 Teams, 30 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	13	2	13.54	138	4.6	43.33	310	155	155	0.00%	5	2
1409	14	4	27.57	138	4.6	46.67	307	153.5	154	0.32%	5	1
1685	12	2	13.17	150	5	40	302	151	151	0.00%	3	2
1737	12	3	23.5	140	4.67	43.33	287	143.5	144	0.35%	4	1
2626	11	4	22.82	144	4.8	40	312	156	156	0.00%	3	2
5129	12	3	26.83	136	4.53	43.33	282	141	141	0.00%	3	2
5228	14	2	9.86	133	4.43	46.67	303	151.5	152	0.33%	3	1
6681	9	3	15.67	165	5.5	33.33	332	166	166	0.00%	2	2
7244	15	5	40.53	148	4.93	50	303	151.5	152	0.33%	3	1
7612	11	3	26.36	151	5.03	36.67	319	159.5	160	0.31%	3	1
Average	12.3	3.1	21.99	144.3	4.81	42.33	305.7	152.85	153.1	0.16%	3.4	1.5
Min	9	2	9.86	133	4.43	33.33	282	141	141	0.00%	2	1
Max	15	5	40.53	165	5.5	50	332	166	166	0.35%	5	2

Table A-98: 2 Teams, 40 (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	16	2	11.13	185	4.63	42.5	404	202	202	0.00%	5	2
1409	18	2	9.72	175	4.38	45	381	190.5	191	0.26%	6	1
1685	18	3	41.22	193	4.83	45	387	193.5	194	0.26%	3	1
1737	18	5	34.11	192	4.8	47.5	392	196	196	0.00%	3	2
2626	20	3	16.15	197	4.93	50	417	208.5	209	0.24%	5	1
5129	18	3	11.28	205	5.13	45	419	209.5	210	0.24%	6	1
5228	16	4	34.06	180	4.5	42.5	398	199	199	0.00%	4	2
6681	15	3	27.13	211	5.28	37.5	423	211.5	212	0.24%	8	1
7244	17	3	32.18	210	5.25	42.5	428	214	214	0.00%	4	2
7612	12	5	28.33	186	4.65	32.5	389	194.5	195	0.26%	5	1
Average	16.8	3.3	24.53	193.4	4.84	43.00	403.8	201.9	202.2	0.15%	4.9	1.4
Min	12	2	9.72	175	4.38	32.5	381	190.5	191	0.00%	3	1
Max	20	5	41.22	211	5.28	50	428	214	214	0.26%	8	2

Table A-99: 2 Teams, 50 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	23	3	13.43	235	4.7	46	503	251.5	252	0.20%	8	1
1409	18	3	24.44	219	4.38	38	470	235	235	0.00%	4	2
1685	19	4	27.53	237	4.74	44	476	238	238	0.00%	5	2
1737	20	3	20.6	262	5.24	40	532	266	266	0.00%	6	2
2626	24	3	20.08	263	5.26	48	549	274.5	275	0.18%	8	1
5129	20	3	16.85	265	5.3	42	540	270	270	0.00%	5	2
5228	25	3	30.32	251	5.02	50	540	270	270	0.00%	7	2
6681	20	3	11.95	252	5.04	40	505	252.5	253	0.20%	6	1
7244	21	5	60.14	268	5.36	44	544	272	272	0.00%	3	2
7612	22	3	14	243	4.86	48	504	252	252	0.00%	5	2
Average	21.2	3.3	23.93	249.5	4.99	44.00	516.3	258.15	258.3	0.06%	5.7	1.7
Min	18	3	11.95	219	4.38	38	470	235	235	0.00%	3	1
Max	25	5	60.14	268	5.36	50	549	274.5	275	0.20%	8	2

Table A-100: 2 Teams, 60 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	27	6	38.3	269	4.48	45	572	286	286	0.00%	6	2
1409	28	4	32.75	282	4.7	46.67	596	298	298	0.00%	5	2
1685	28	4	23.14	284	4.73	46.67	570	285	285	0.00%	8	2
1737	22	3	11.91	309	5.15	38.33	625	312.5	313	0.16%	9	1
2626	25	3	23.2	316	5.27	43.33	655	327.5	328	0.15%	7	1
5129	21	4	33.38	303	5.05	36.67	615	307.5	308	0.16%	4	1
5228	26	5	31.77	321	5.35	43.33	679	339.5	340	0.15%	8	1
6681	26	5	21.92	314	5.23	43.33	630	315	315	0.00%	8	2
7244	30	3	29	311	5.18	51.67	630	315	315	0.00%	8	2
7612	28	4	22.93	289	4.82	46.67	595	297.5	298	0.17%	7	1
Average	26.1	4.1	26.83	299.8	5.00	44.17	616.7	308.35	308.6	0.08%	7	1.5
Min	21	3	11.91	269	4.48	36.67	570	285	285	0.00%	4	1
Max	30	6	38.3	321	5.35	51.67	679	339.5	340	0.17%	9	2

Table A-101: 2 Teams, 70 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	30	6	62.07	316	4.51	42.86	665	332.5	333	0.15%	6	1
1409	32	4	32.59	334	4.77	45.71	699	349.5	350	0.14%	9	1
1685	30	4	18.83	332	4.74	42.86	666	333	333	0.00%	9	2
1737	38	7	49.68	351	5.01	54.29	709	354.5	355	0.14%	7	1
2626	28	4	19.89	366	5.23	40	755	377.5	378	0.13%	7	1
5129	34	5	35.41	360	5.14	50	730	365	365	0.00%	7	2
5228	33	5	29.61	369	5.27	48.57	776	388	388	0.00%	6	2
6681	30	4	27.17	367	5.24	42.86	735	367.5	368	0.14%	7	1
7244	30	3	29.7	353	5.04	42.86	713	356.5	357	0.14%	6	1
7612	28	6	36.96	360	5.14	40	738	369	369	0.00%	5	2
Average	31.3	4.8	34.19	350.8	5.01	45.00	718.6	359.3	359.6	0.08%	6.9	1.4
Min	28	3	18.83	316	4.51	40	665	332.5	333	0.00%	5	1
Max	38	7	62.07	369	5.27	54.29	776	388	388	0.15%	9	2

Table A-102: 2 Teams, 80 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	41	5	47.8	372	4.65	52.5	778	389	389	0.00%	7	2
1409	34	3	13.06	389	4.86	42.5	809	404.5	405	0.12%	11	1
1685	44	7	56.39	400	5	56.25	802	401	401	0.00%	6	2
1737	32	5	23.53	414	5.18	43.75	835	417.5	418	0.12%	10	1
2626	30	4	21.37	422	5.28	37.5	867	433.5	434	0.12%	8	1
5129	39	5	31	398	4.98	48.75	805	402.5	403	0.12%	15	1
5228	28	3	10.25	431	5.39	37.5	899	449.5	450	0.11%	9	1
6681	37	4	31.65	430	5.38	46.25	862	431	431	0.00%	8	2
7244	28	4	19.11	402	5.03	35	811	405.5	406	0.12%	8	1
7612	31	4	11.23	397	4.96	42.5	812	406	406	0.00%	8	2
Average	34.4	4.4	26.54	405.5	5.07	44.25	828	414	414.3	0.07%	9	1.4
Min	28	3	10.25	372	4.65	35	778	389	389	0.00%	6	1
Max	44	7	56.39	431	5.39	56.25	899	449.5	450	0.12%	15	2

Table A-103: 2 Teams, 90 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	42	3	34.21	420	4.67	46.67	874	437	437	0.00%	9	2
1409	37	4	16.24	438	4.87	44.44	907	453.5	454	0.11%	10	1
1685	40	5	31.05	447	4.97	44.44	895	447.5	448	0.11%	10	1
1737	41	6	47	471	5.23	47.78	949	474.5	475	0.11%	8	1
2626	37	5	22.54	463	5.14	41.11	949	474.5	475	0.11%	10	1
5129	42	7	29.26	447	4.97	47.78	903	451.5	452	0.11%	9	1
5228	33	6	17.91	481	5.34	36.67	999	499.5	500	0.10%	11	1
6681	33	4	18.58	470	5.22	38.89	942	471	471	0.00%	8	2
7244	44	4	54.57	439	4.88	48.89	886	443	443	0.00%	7	2
7612	38	5	14.61	440	4.89	44.44	898	449	449	0.00%	17	2
Average	38.7	4.9	28.60	451.6	5.02	44.11	920.2	460.1	460.4	0.06%	9.9	1.4
Min	33	3	14.61	420	4.67	36.67	874	437	437	0.00%	7	1
Max	44	7	54.57	481	5.34	48.89	999	499.5	500	0.11%	17	2

Table A-104: 2 Teams, 100 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	46	6	22.39	463	4.63	46	959	479.5	480	0.10%	13	1
1409	45	6	31.8	469	4.69	46	969	484.5	485	0.10%	8	1
1685	43	6	30.47	512	5.12	43	1025	512.5	513	0.10%	12	1
1737	46	5	28.65	527	5.27	47	1061	530.5	531	0.09%	12	1
2626	39	6	14.36	508	5.08	40	1040	520	520	0.00%	11	2
5129	41	4	25.02	498	4.98	44	1006	503	503	0.00%	13	2
5228	51	5	81.04	536	5.36	51	1110	555	555	0.00%	7	2
6681	43	6	25.28	533	5.33	43	1067	533.5	534	0.09%	12	1
7244	47	6	25.96	494	4.94	48	996	498	498	0.00%	12	2
7612	36	4	18	496	4.96	37	1010	505	505	0.00%	13	2
Average	43.7	5.4	30.30	503.6	5.04	44.50	1024.3	512.15	512.4	0.05%	11.3	1.5
Min	36	4	14.36	463	4.63	37	959	479.5	480	0.00%	7	1
Max	51	6	81.04	536	5.36	51	1110	555	555	0.10%	13	2

A2.14. Genetic Algorithm (Makespan) – 3 Teams

Table A-104: 3 Teams, 30 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	17	4	11.65	87	2.9	56.67	310	103.33	104	0.64%	5	1
1409	21	7	19.86	87	2.9	70	307	102.33	103	0.65%	6	2
1685	16	7	18.63	100	3.33	56.67	302	100.67	101	0.33%	7	2
1737	16	2	8.75	92	3.07	56.67	287	95.67	96	0.35%	5	2
2626	18	3	32.17	92	3.07	60	312	104	104	0.00%	4	3
5129	18	5	15	89	2.97	63.33	282	94.00	94	0.00%	5	3
5228	20	5	17.05	82	2.73	66.67	303	101	101	0.00%	7	3
6681	14	4	17.07	110	3.67	50	332	110.67	111	0.30%	4	2
7244	13	3	12.92	97	3.23	50	303	101	101	0.00%	3	3
7612	16	3	15.88	98	3.27	53.33	319	106.33	107	0.62%	5	1
Average	16.9	4.3	16.90	93.4	3.11	58.334	305.7	101.90	102.2	0.29%	5.1	2.2
Min	13	2	8.75	82	2.73	50	282	94.00	94	0.00%	3	1
Max	21	7	32.17	110	3.67	70	332	111	111	0.65%	7	3

Table A-105: 3 Teams, 40 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	26	6	10.92	118	2.95	65	404	134.67	135	0.25%	7	2
1409	19	4	13.79	111	2.78	52.5	381	127	127	0.00%	8	3
1685	22	5	10.77	128	3.2	57.5	387	129	129	0.00%	8	3
1737	23	5	28.87	127	3.18	62.5	392	130.67	131	0.25%	5	2
2626	27	7	25.81	127	3.18	67.5	417	139	139	0.00%	8	3
5129	22	7	11.05	135	3.38	60	419	139.67	140	0.24%	9	2
5228	23	12	31.3	114	2.85	57.5	398	132.67	133	0.25%	5	2
6681	26	6	19.88	140	3.5	65	423	141	141	0.00%	9	3
7244	21	5	15.67	139	3.48	52.5	428	142.67	143	0.23%	6	2
7612	23	4	13.7	121	3.03	57.5	389	129.67	130	0.26%	8	2
Average	23.2	6.1	18.18	126	3.15	59.75	403.8	134.60	134.8	0.15%	7.3	2.4
Min	19	4	10.77	111	2.78	52.5	381	127.00	127	0.00%	5	2
Max	27	12	31.3	140	3.5	67.5	428	143	143	0.26%	9	3

Table A-106: 3 Teams, 50 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	28	4	13.29	151	3.02	56	503	167.67	168	0.20%	14	2
1409	32	5	11.72	141	2.82	66	470	156.67	157	0.21%	10	2
1685	31	6	21.06	158	3.16	62	476	158.67	159	0.21%	16	2
1737	33	9	23.24	174	3.48	68	532	177.33	178	0.37%	8	2
2626	28	3	11.64	171	3.42	58	549	183	183	0.00%	9	3
5129	34	5	16.71	175	3.5	68	540	180	180	0.00%	14	3
5228	29	5	15.59	161	3.22	60	540	180.00	180	0.00%	8	3
6681	27	6	29.11	168	3.36	54	505	168.33	169	0.39%	6	1
7244	28	7	24.93	178	3.56	56	544	181.33	182	0.37%	11	1
7612	27	4	12.33	159	3.18	58	504	168	168	0.00%	7	3
Average	29.7	5.4	17.96	163.6	3.27	60.6	516.3	172.10	172.4	0.18%	10.3	2.2
Min	27	3	11.64	141	2.82	54	470	156.67	157	0.00%	6	1
Max	34	9	29.11	178	3.56	68	549	183	183	0.39%	16	3

Table A-107: 3 Teams, 60 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	36	6	20.78	174	2.9	61.67	572	190.67	191	0.17%	13	2
1409	37	6	17.19	183	3.05	63.33	596	198.67	199	0.17%	9	2
1685	34	5	22	189	3.15	58.33	570	190	190	0.00%	8	3
1737	36	9	18.44	205	3.42	65	625	208.33	209	0.32%	8	1
2626	39	9	37.79	207	3.45	65	655	218.33	219	0.30%	8	1
5129	36	5	29.64	201	3.35	60	615	205	206	0.49%	11	1
5228	38	6	33.39	208	3.47	63.33	679	226.33	227	0.29%	13	2
6681	30	5	16.6	209	3.48	51.67	630	210	210	0.00%	9	3
7244	39	8	22.26	206	3.43	65	630	210	210	0.00%	17	3
7612	36	7	33.75	190	3.17	61.67	595	198.33	199	0.34%	13	1
Average	36.1	6.6	25.18	197.2	3.29	61.5	616.7	205.57	206	0.21%	10.9	1.9
Min	30	5	16.6	174	2.9	51.67	570	190.00	190	0.00%	8	1
Max	39	9	37.79	209	3.48	65	679	226	227	0.49%	17	3

Table A-108: 3 Teams, 70 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	48	8	39.04	205	2.93	68.57	665	221.67	222	0.15%	13	2
1409	47	6	25.45	217	3.1	67.14	699	233	233	0.00%	13	3
1685	37	6	15.89	222	3.17	54.29	666	222	223	0.45%	9	2
1737	44	11	27.27	233	3.33	65.71	709	236.33	237	0.28%	9	2
2626	46	8	29.35	240	3.43	65.71	755	251.67	252	0.13%	18	2
5129	42	5	18.9	239	3.41	61.43	730	243.33	244	0.27%	18	2
5228	40	8	18.58	240	3.43	60	776	258.67	259	0.13%	12	2
6681	40	10	28.08	244	3.49	60	735	245	245	0.00%	10	3
7244	43	6	28.05	234	3.34	61.43	713	237.67	238	0.14%	11	2
7612	46	9	23.41	238	3.4	65.71	738	246	247	0.40%	14	1
Average	43.3	7.7	25.40	231.2	3.30	62.999	718.6	239.53	240	0.20%	12.7	2.1
Min	37	5	15.89	205	2.93	54.29	665	221.67	222	0.00%	9	1
Max	48	11	39.04	244	3.49	68.57	776	259	259	0.45%	18	3

Table A-109: 3 Teams, 80 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	39	4	15.44	243	3.04	50	778	259.33	260	0.26%	22	1
1409	49	8	36.9	254	3.18	62.5	809	269.67	270	0.12%	23	2
1685	45	7	21.02	267	3.34	56.25	802	267.33	268	0.25%	14	1
1737	43	5	40.67	275	3.44	56.25	835	278.33	279	0.24%	9	2
2626	48	4	16.77	278	3.48	62.5	867	289	290	0.34%	16	1
5129	48	7	29.85	264	3.3	61.25	805	268.33	269	0.25%	12	2
5228	50	5	15.04	281	3.51	62.5	899	299.67	300	0.11%	11	2
6681	41	5	16.78	287	3.59	51.25	862	287.33	288	0.23%	12	2
7244	44	10	30.11	267	3.34	56.25	811	270.33	271	0.25%	8	2
7612	49	8	19.92	262	3.28	61.25	812	270.67	271	0.12%	14	2
Average	45.6	6.3	24.25	267.8	3.35	58	828	276.00	276.6	0.22%	14.1	1.7
Min	39	4	15.04	243	3.04	50	778	259.33	260	0.11%	8	1
Max	50	10	40.67	287	3.59	62.5	899	300	300	0.34%	23	2

Table A-110: 4 Teams, 90 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	52	16	45.21	275	3.06	61.11	874	291.33	292	0.23%	9	1
1409	58	6	21.83	287	3.19	64.44	907	302.33	303	0.22%	18	1
1685	55	13	39.25	298	3.31	62.22	895	298.33	299	0.22%	16	2
1737	52	6	25.04	313	3.48	60	949	316.33	317	0.21%	14	2
2626	54	10	33.41	305	3.39	60	949	316.33	317	0.21%	12	2
5129	51	5	31.94	296	3.29	60	903	301	301	0.00%	15	3
5228	54	6	21.41	314	3.49	60	999	333	333	0.00%	14	3
6681	54	7	37.72	314	3.49	60	942	314	315	0.32%	21	1
7244	47	5	12.72	292	3.24	54.44	886	295.33	296	0.23%	15	1
7612	54	7	15.52	291	3.23	60	898	299.33	300	0.22%	25	1
Average	53.1	8.1	28.41	298.5	3.32	60.221	920.2	306.73	307.3	0.19%	15.9	1.7
Min	47	5	12.72	275	3.06	54.44	874	291.33	292	0.00%	9	1
Max	58	16	45.21	314	3.49	64.44	999	333	333	0.32%	25	3

Table A-111: 3 Teams, 100 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	56	6	21.96	303	3.03	56	959	319.67	320	0.10%	24	2
1409	63	13	43.32	308	3.08	64	969	323	324	0.31%	10	1
1685	55	5	15.55	341	3.41	56	1025	341.67	342	0.10%	15	2
1737	57	4	13.05	350	3.5	59	1061	353.67	354	0.09%	17	2
2626	58	7	17.5	335	3.35	58	1040	346.67	347	0.10%	14	2
5129	60	8	32.45	331	3.31	60	1006	335.33	336	0.20%	21	1
5228	66	9	39.15	352	3.52	66	1110	370	371	0.27%	27	1
6681	58	8	25.72	355	3.55	58	1067	355.67	356	0.09%	15	2
7244	58	7	36.05	329	3.29	58	996	332	333	0.30%	13	2
7612	62	7	47.02	328	3.28	62	1010	336.67	337	0.10%	21	2
Average	59.3	7.4	29.18	333.2	3.33	59.7	1024.3	341.43	342	0.17%	17.7	1.7
Min	55	4	13.05	303	3.03	56	959	319.67	320	0.09%	10	1
Max	66	13	47.02	355	3.55	66	1110	370	371	0.31%	27	2

A2.15. Genetic Algorithm (Makespan) – 4 Teams

Table A-112: 4 Teams, 30 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	20	5	15.05	61	2.03	66.67	310	77.5	78	0.64%	4	2
1409	20	7	11.25	62	2.07	66.67	307	76.75	78	1.60%	4	1
1685	18	8	14.89	76	2.53	60	302	75.5	77	1.95%	4	1
1737	19	7	12.74	69	2.3	66.67	287	71.75	73	1.71%	5	2
2626	22	6	14.91	67	2.23	73.33	312	78	79	1.27%	7	2
5129	20	6	17.1	67	2.23	66.67	282	70.5	72	2.08%	6	1
5228	23	4	20.3	58	1.93	76.67	303	75.75	77	1.62%	8	1
6681	18	9	15.39	83	2.77	60	332	83	84	1.19%	7	2
7244	20	8	17.25	73	2.43	70	303	75.75	77	1.62%	7	2
7612	17	4	17.53	72	2.4	56.67	319	79.75	81	1.54%	8	1
Average	19.7	6.4	15.64	68.8	2.29	66.34	305.7	76.43	77.6	1.52%	6	1.5
Min	17	4	11.25	58	1.93	56.67	282	70.5	72	0.64%	4	1
Max	23	9	20.3	83	2.77	76.67	332	83	84	2.08%	8	2

Table A-113: 4 Teams, 40 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	29	6	23.1	85	2.13	72.5	404	101	102	0.98%	7	2
1409	19	3	6.42	80	2	57.5	381	95.25	96	0.78%	7	3
1685	28	11	12.86	97	2.43	70	387	96.75	98	1.28%	10	1
1737	24	7	13.96	96	2.4	60	392	98	100	2.00%	7	1
2626	24	6	9.71	94	2.35	65	417	104.25	106	1.65%	7	1
5129	26	6	22.65	101	2.53	65	419	104.75	106	1.18%	10	2
5228	28	8	19.46	82	2.05	70	398	99.5	101	1.49%	8	1
6681	22	5	12.68	106	2.65	55	423	105.75	107	1.17%	5	2
7244	25	8	15.68	103	2.58	67.5	428	107	107	0.00%	5	4
7612	27	6	11	89	2.23	67.5	389	97.25	98	0.77%	9	1
Average	25.2	6.6	14.75	93.3	2.34	65.00	403.8	100.95	102.1	1.13%	7.5	1.8
Min	19	3	6.42	80	2	55	381	95.25	96	0.00%	5	1
Max	29	11	23.1	106	2.65	72.5	428	107	107	2.00%	10	4

Table A-114: 4 Teams, 50 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	35	7	14.49	110	2.2	72	503	125.75	127	0.98%	16	2
1409	38	5	21.82	102	2.04	78	470	117.5	118	0.42%	11	3
1685	35	8	24.43	119	2.38	70	476	119	120	0.83%	11	3
1737	34	14	23.03	132	2.64	70	532	133	136	2.21%	7	1
2626	38	7	29.18	127	2.54	76	549	137.25	139	1.26%	14	2
5129	33	5	23.21	132	2.64	66	540	135	137	1.46%	6	1
5228	33	4	15.42	118	2.36	66	540	135	137	1.46%	8	2
6681	28	13	20.14	126	2.52	60	505	126.25	127	0.59%	9	3
7244	28	6	10.54	133	2.66	62	544	136	137	0.73%	9	1
7612	30	8	14	119	2.38	60	504	126	128	1.56%	12	1
Average	33.2	7.7	19.63	121.8	2.44	68.00	516.3	129.08	130.6	1.15%	10.3	1.9
Min	28	4	10.54	102	2.04	60	470	117.5	118	0.42%	6	1
Max	38	14	29.18	133	2.66	78	549	137.25	139	2.21%	16	3

Table A-115: 4 Teams, 60 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	38	7	18	127	2.12	65	572	143	144	0.69%	18	2
1409	38	4	14.71	135	2.25	63.33	596	149	151	1.32%	15	2
1685	39	7	13.13	144	2.4	65	570	142.5	145	1.72%	17	2
1737	39	10	19.59	153	2.55	66.67	625	156.25	157	0.48%	17	2
2626	38	7	14.29	153	2.55	66.67	655	163.75	165	0.76%	15	2
5129	38	8	15.71	150	2.5	65	615	153.75	155	0.81%	15	2
5228	43	7	29.7	152	2.53	71.67	679	169.75	171	0.73%	13	1
6681	43	11	28.67	158	2.63	71.67	630	157.5	159	0.94%	13	1
7244	43	8	17.72	156	2.6	71.67	630	157.5	160	1.56%	13	1
7612	36	6	17.67	141	2.35	65	595	148.75	150	0.83%	15	3
Average	39.5	7.5	18.92	146.9	2.45	67.17	616.7	154.18	155.7	0.99%	15.1	1.8
Min	36	4	13.13	127	2.12	63.33	570	142.5	144	0.48%	13	1
Max	43	11	29.7	158	2.63	71.67	679	169.75	171	1.72%	18	3

Table A-116: 4 Teams, 70 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	54	9	22.74	151	2.16	77.14	665	166.25	168	1.04%	19	2
1409	47	6	17.53	160	2.29	67.14	699	174.75	176	0.71%	19	2
1685	47	7	25.55	168	2.4	67.14	666	166.5	169	1.48%	18	1
1737	43	6	18.67	176	2.51	64.29	709	177.25	180	1.53%	9	1
2626	44	9	20.07	178	2.54	64.29	755	188.75	190	0.66%	21	1
5129	48	9	26.94	181	2.59	68.57	730	182.5	186	1.88%	18	1
5228	51	14	37.24	177	2.53	72.86	776	194	196	1.02%	14	1
6681	41	7	21.8	183	2.61	61.43	735	183.75	184	0.14%	12	3
7244	46	11	20.28	176	2.51	67.14	713	178.25	180	0.97%	12	2
7612	47	8	17.74	178	2.54	68.57	738	184.5	187	1.34%	21	2
Average	46.8	8.6	22.86	172.8	2.47	67.86	718.6	179.65	181.6	1.08%	16.3	1.6
Min	41	6	17.53	151	2.16	61.43	665	166.25	168	0.14%	9	1
Max	54	14	37.24	183	2.61	77.14	776	194	196	1.88%	21	3

Table A-117: 4 Teams, 80 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	54	8	20.5	179	2.24	67.5	778	194.5	196	0.77%	16	1
1409	56	11	26.75	188	2.35	70	809	202.25	204	0.86%	14	1
1685	56	8	28.38	201	2.51	70	802	200.5	202	0.74%	14	2
1737	55	8	24.75	207	2.59	71.25	835	208.75	211	1.07%	24	1
2626	52	5	13.98	208	2.6	66.25	867	216.75	220	1.48%	14	1
5129	56	12	38.3	200	2.5	70	805	201.25	205	1.83%	10	1
5228	58	14	32.86	208	2.6	72.5	899	224.75	227	0.99%	21	1
6681	52	8	19.81	217	2.71	65	862	215.5	218	1.15%	20	1
7244	52	6	19.62	200	2.5	65	811	202.75	204	0.61%	14	1
7612	53	9	18.83	197	2.46	67.5	812	203	206	1.46%	17	2
Average	54.4	8.9	24.38	200.5	2.51	68.50	828	207.00	209.3	1.09%	16.4	1.2
Min	52	5	13.98	179	2.24	65	778	194.5	196	0.61%	10	1
Max	58	14	38.3	217	2.71	72.5	899	224.75	227	1.83%	24	2

Table A-118: 4 Teams, 90 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	61	10	20.15	203	2.26	67.78	874	218.5	220	0.68%	31	1
1409	62	6	34.03	214	2.38	68.89	907	226.75	230	1.41%	14	2
1685	59	7	16.56	225	2.5	65.56	895	223.75	226	1.00%	21	2
1737	59	7	20.44	235	2.61	70	949	237.25	239	0.73%	25	3
2626	61	10	29.1	229	2.54	67.78	949	237.25	241	1.56%	17	1
5129	67	11	27.9	222	2.47	74.44	903	225.75	227	0.55%	16	2
5228	56	7	23.59	234	2.6	63.33	999	249.75	253	1.28%	24	1
6681	58	11	30.22	236	2.62	65.56	942	235.5	237	0.63%	24	2
7244	58	9	16.67	220	2.44	64.44	886	221.5	224	1.12%	20	2
7612	60	11	22.87	218	2.42	67.78	898	224.5	227	1.10%	25	1
Average	60.1	8.9	24.15	223.6	2.48	67.56	920.2	230.05	232.4	1.01%	21.7	1.7
Min	56	6	16.56	203	2.26	63.33	874	218.5	220	0.55%	14	1
Max	67	11	34.03	236	2.62	74.44	999	249.75	253	1.56%	31	3

Table A-119: 4 Teams, 100 Parts (GA – Makespan)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	61	10	20.15	203	2.26	67.78	874	218.5	220	0.68%	31	1
1409	62	6	34.03	214	2.38	68.89	907	226.75	230	1.41%	14	2
1685	59	7	16.56	225	2.5	65.56	895	223.75	226	1.00%	21	2
1737	59	7	20.44	235	2.61	70	949	237.25	239	0.73%	25	3
2626	61	10	29.1	229	2.54	67.78	949	237.25	241	1.56%	17	1
5129	67	11	27.9	222	2.47	74.44	903	225.75	227	0.55%	16	2
5228	56	7	23.59	234	2.6	63.33	999	249.75	253	1.28%	24	1
6681	58	11	30.22	236	2.62	65.56	942	235.5	237	0.63%	24	2
7244	58	9	16.67	220	2.44	64.44	886	221.5	224	1.12%	20	2
7612	60	11	22.87	218	2.42	67.78	898	224.5	227	1.10%	25	1
Average	60.1	8.9	24.15	223.6	2.48	67.56	920.2	230.05	232.4	1.01%	21.7	1.7
Min	56	6	16.56	203	2.26	63.33	874	218.5	220	0.55%	14	1
Max	67	11	34.03	236	2.62	74.44	999	249.75	253	1.56%	31	3

A2.16. Genetic Algorithm (Makespan/JIT) – 2 Teams

Table A-120: 2 Teams, 30 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	17	2	20.41	138	4.6	56.67	310	155	155	0.00%	6	2
1409	17	3	24.12	138	4.6	56.67	307	153.5	154	0.32%	5	1
1685	16	3	13.94	150	5	56.67	302	151	151	0.00%	4	2
1737	16	4	8.19	140	4.67	60	287	143.5	144	0.35%	4	1
2626	16	6	28.5	144	4.8	56.67	312	156	156	0.00%	3	2
5129	17	3	12.41	136	4.53	56.67	282	141	141	0.00%	6	2
5228	18	3	18.78	133	4.43	60	303	151.5	152	0.33%	6	1
6681	14	4	37.07	165	5.5	50	332	166	166	0.00%	6	2
7244	16	4	12.31	148	4.93	53.33	303	151.5	152	0.33%	5	1
7612	16	3	21.5	151	5.03	53.33	319	159.5	160	0.31%	5	1
Average	16.3	3.5	19.72	144.3	4.81	56.00	305.7	152.85	153.1	0.16%	5	1.5
Min	14	2	8.19	133	4.43	50	282	141	141	0.00%	3	1
Max	18	6	37.07	165	5.5	60	332	166	166	0.35%	6	2

Table A-122: 2 Teams, 40 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	22	5	31.36	185	4.63	55	404	202	202	0.00%	12	2
1409	23	5	27.91	175	4.38	57.5	381	190.5	191	0.26%	6	1
1685	22	3	15.95	193	4.83	55	387	193.5	194	0.26%	8	1
1737	20	4	37.75	192	4.8	55	392	196	196	0.00%	4	2
2626	21	6	49.62	197	4.93	55	417	208.5	209	0.24%	5	1
5129	21	5	19.48	205	5.13	52.5	419	209.5	210	0.24%	6	1
5228	22	4	26.18	180	4.5	57.5	398	199	199	0.00%	6	2
6681	22	5	27.14	211	5.28	55	423	211.5	212	0.24%	7	1
7244	22	6	37.45	210	5.25	55	428	214	214	0.00%	7	2
7612	22	6	20.09	186	4.65	55	389	194.5	195	0.26%	7	1
Average	21.7	4.9	29.29	193.4	4.838	55.25	403.8	201.9	202.2	0.15%	6.8	1.4
Min	20	3	15.95	175	4.38	52.5	381	190.5	191	0.00%	4	1
Max	23	6	49.62	211	5.28	57.5	428	214	214	0.26%	12	2

Table A-123: 2 Teams, 50 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	29	4	24.72	235	4.7	58	503	251.5	252	0.20%	9	1
1409	29	4	32.31	219	4.38	58	470	235	235	0.00%	7	2
1685	30	4	26.07	237	4.74	60	476	238	238	0.00%	10	2
1737	25	5	57.08	262	5.24	56	532	266	266	0.00%	4	2
2626	29	5	52.31	263	5.26	58	549	274.5	275	0.18%	13	1
5129	26	6	12.42	265	5.3	54	540	270	270	0.00%	9	2
5228	27	5	33.15	251	5.02	56	540	270	270	0.00%	6	2
6681	27	5	55.85	252	5.04	54	505	252.5	253	0.20%	4	1
7244	26	4	31.31	268	5.36	52	544	272	272	0.00%	5	2
7612	27	5	23.52	243	4.86	54	504	252	252	0.00%	7	2
Average	27.5	4.7	34.87	249.5	4.99	56	516.3	258.15	258.3	0.06%	7.4	1.7
Min	25	4	12.42	219	4.38	52	470	235	235	0.00%	4	1
Max	30	6	57.08	268	5.36	60	549	274.5	275	0.20%	13	2

Table A-124: 2 Teams, 60 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	33	5	31.73	269	4.48	55	572	286	286	0.00%	7	2
1409	32	5	28.81	282	4.7	53.33	596	298	298	0.00%	11	2
1685	31	5	16.84	284	4.73	53.33	570	285	285	0.00%	7	2
1737	34	4	70.94	309	5.15	56.67	625	312.5	313	0.16%	14	1
2626	32	4	38.34	316	5.27	53.33	655	327.5	328	0.15%	14	1
5129	32	6	19.97	303	5.05	53.33	615	307.5	308	0.16%	7	1
5228	34	3	63.35	321	5.35	56.67	679	339.5	340	0.15%	5	1
6681	31	5	20.06	314	5.23	53.33	630	315	315	0.00%	12	2
7244	30	3	29	311	5.18	51.67	630	315	315	0.00%	8	2
7612	32	9	49.84	289	4.82	53.33	595	297.5	298	0.17%	5	1
Average	32.1	4.9	36.89	299.8	5.00	54.00	616.7	308.35	308.6	0.08%	9	1.5
Min	30	3	16.84	269	4.48	51.67	570	285	285	0.00%	5	1
Max	34	9	70.94	321	5.35	56.67	679	339.5	340	0.17%	14	2

Table A-125: 2 Teams, 70 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	38	9	33.71	316	4.51	57.14	665	332.5	333	0.15%	10	1
1409	38	6	42.34	334	4.77	54.29	699	349.5	350	0.14%	7	1
1685	38	4	26.79	332	4.74	55.71	666	333	333	0.00%	11	2
1737	38	7	49.68	351	5.01	54.29	709	354.5	355	0.14%	7	1
2626	38	4	22.53	366	5.23	54.29	755	377.5	378	0.13%	10	1
5129	38	5	25.68	360	5.14	54.29	730	365	365	0.00%	10	2
5228	35	3	37.29	369	5.27	51.43	776	388	388	0.00%	8	2
6681	36	6	49.75	367	5.24	51.43	735	367.5	368	0.14%	7	1
7244	39	6	61.67	353	5.04	55.71	713	356.5	357	0.14%	5	1
7612	35	4	35.03	360	5.14	51.43	738	369	369	0.00%	8	2
Average	37.3	5.4	38.45	350.8	5.009	54.00	718.6	359.3	359.6	0.08%	8.3	1.4
Min	35	3	22.53	316	4.51	51.43	665	332.5	333	0.00%	5	1
Max	39	9	61.67	369	5.27	57.14	776	388	388	0.15%	11	2

Table A-126: 2 Teams, 80 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	43	5	39.19	372	4.65	55	778	389	389	0.00%	12	2
1409	42	5	44.12	389	4.86	53.75	809	404.5	405	0.12%	8	1
1685	44	7	56.39	400	5	56.25	802	401	401	0.00%	6	2
1737	44	8	38.16	414	5.18	55	835	417.5	418	0.12%	7	1
2626	43	4	64.02	422	5.28	53.75	867	433.5	434	0.12%	13	1
5129	42	6	31.43	398	4.98	52.5	805	402.5	403	0.12%	10	1
5228	43	4	34.09	431	5.39	55	899	449.5	450	0.11%	22	1
6681	40	7	55.25	430	5.38	51.25	862	431	431	0.00%	9	2
7244	42	6	41.1	402	5.03	52.5	811	405.5	406	0.12%	15	1
7612	41	5	41.88	397	4.96	51.25	812	406	406	0.00%	8	2
Average	42.4	5.7	44.56	405.5	5.07	53.63	828	414	414.3	0.07%	11	1.4
Min	40	4	31.43	372	4.65	51.25	778	389	389	0.00%	6	1
Max	44	8	64.02	431	5.39	56.25	899	449.5	450	0.12%	22	2

Table A-127: 2 Teams, 90 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	48	6	38.23	420	4.67	53.33	874	437	437	0.00%	12	2
1409	48	7	37.21	438	4.87	53.33	907	453.5	454	0.11%	22	1
1685	50	9	83.22	447	4.97	55.56	895	447.5	448	0.11%	6	1
1737	48	8	37.56	471	5.23	54.44	949	474.5	475	0.11%	12	1
2626	48	7	52.21	463	5.14	53.33	949	474.5	475	0.11%	12	1
5129	47	4	29.85	447	4.97	53.33	903	451.5	452	0.11%	18	1
5228	48	7	73	481	5.34	54.44	999	499.5	500	0.10%	7	1
6681	47	8	52.68	470	5.22	52.22	942	471	471	0.00%	9	2
7244	49	5	40.2	439	4.88	54.44	886	443	443	0.00%	9	2
7612	48	5	26.15	440	4.89	54.44	898	449	449	0.00%	14	2
Average	48.1	6.6	47.03	451.6	5.02	53.89	920.2	460.1	460.4	0.06%	12.1	1.4
Min	47	4	26.15	420	4.67	52.22	874	437	437	0.00%	6	1
Max	50	9	83.22	481	5.34	55.56	999	499.5	500	0.11%	22	2

Table A-128: 2 Teams, 100 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	54	6	40.54	463	4.63	54	959	479.5	480	0.10%	15	1
1409	54	6	37.37	469	4.69	54	969	484.5	485	0.10%	11	1
1685	52	7	47.23	512	5.12	52	1025	512.5	513	0.10%	15	1
1737	51	5	67.59	527	5.27	53	1061	530.5	531	0.09%	8	1
2626	53	5	70.87	508	5.08	53	1040	520	520	0.00%	7	2
5129	53	6	40.55	498	4.98	54	1006	503	503	0.00%	19	2
5228	52	7	28.85	536	5.36	52	1110	555	555	0.00%	25	2
6681	51	5	87.08	533	5.33	51	1067	533.5	534	0.09%	8	1
7244	51	7	79.59	494	4.94	52	996	498	498	0.00%	19	2
7612	51	5	15.08	496	4.96	52	1010	505	505	0.00%	21	2
Average	52.2	5.9	51.48	503.6	5.04	52.70	1024.3	512.15	512.4	0.05%	14.8	1.5
Min	51	5	15.08	463	4.63	51	959	479.5	480	0.00%	7	1
Max	54	7	87.08	536	5.36	54	1110	555	555	0.10%	25	2

A2.17. Genetic Algorithm (JIT Priority) – 3 Teams

Table A-121: 3 Teams, 30 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	21	6	19.14	87	2.9	70	310	103.33	104	0.64%	8	2
1409	20	4	9.6	87	2.9	66.67	307	102.33	103	0.65%	8	1
1685	20	6	20.35	100	3.33	66.67	302	100.67	101	0.33%	6	2
1737	15	4	9.6	92	3.07	60	287	95.67	96	0.35%	4	2
2626	20	4	21	92	3.07	66.67	312	104.00	104	0.00%	6	3
5129	20	6	20.3	89	2.97	66.67	282	94.00	94	0.00%	4	3
5228	20	5	17.05	82	2.73	66.67	303	101.00	101	0.00%	7	3
6681	16	6	22.88	110	3.67	56.67	332	110.67	111	0.30%	8	2
7244	20	4	18.2	97	3.23	66.67	303	101.00	101	0.00%	11	3
7612	19	7	24.89	98	3.27	63.33	319	106.33	107	0.62%	4	1
Average	19.1	5.2	18.301	93.40	3.114	65.00	305.7	101.9	102.20	0.29%	6.60	2.2
Min	15	4	9.6	82	2.73	56.67	282	94	94.00	0.00%	4.00	1
Max	21	7	24.89	110	3.67	70	332	110.67	111	0.65%	11.00	3

Table A-122: 3 Teams, 40 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	29	15	26.03	118	2.95	72.5	404	134.67	135	0.25%	11	2
1409	27	7	18.67	111	2.78	67.5	381	127.00	127	0.00%	10	3
1685	23	6	14.57	128	3.2	65	387	129.00	129	0.00%	6	3
1737	22	9	23.18	127	3.18	60	392	130.67	131	0.25%	5	2
2626	25	11	25.16	128	3.2	67.5	417	139.00	140	0.71%	8	2
5129	24	8	15.33	136	3.4	62.5	419	139.67	141	0.95%	12	2
5228	26	4	15.58	114	2.85	65	398	132.67	133	0.25%	10	2
6681	22	4	13.27	140	3.5	55	423	141.00	141	0.00%	9	3
7244	26	8	27.42	139	3.48	65	428	142.67	143	0.23%	3	2
7612	27	4	28.07	121	3.03	67.5	389	129.67	130	0.26%	6	2
Average	25.1	7.6	20.728	126.20	3.157	64.75	403.8	134.6	135.00	0.29%	8.00	2.3
Min	22	4	13.27	111	2.78	55	381	127	127.00	0.00%	3.00	2
Max	29	15	28.07	140	3.5	72.5	428	142.67	143	0.95%	12.00	3

Table A-123: 3 Teams, 50 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	32	8	16.66	151	3.02	64	503	167.67	168	0.20%	8	2
1409	32	7	26.34	141	2.82	66	470	156.67	157	0.21%	10	2
1685	33	7	21.67	158	3.16	66	476	158.67	159	0.21%	15	2
1737	34	9	28.09	174	3.48	72	532	177.33	178	0.37%	10	2
2626	32	14	46.34	172	3.44	64	549	183.00	184	0.54%	8	1
5129	28	5	18.54	176	3.52	58	540	180.00	181	0.55%	9	1
5228	34	7	26.65	161	3.22	68	540	180.00	180	0.00%	7	3
6681	31	7	23.65	168	3.36	64	505	168.33	169	0.39%	10	2
7244	29	5	24.48	178	3.56	62	544	181.33	182	0.37%	15	2
7612	33	6	13.94	160	3.2	66	504	168.00	169	0.59%	14	2
Average	31.8	7.5	24.636	163.90	3.28	65.00	516.3	172.1	172.70	0.34%	10.60	1.9
Min	28	5	13.94	141	2.82	58	470	156.67	157.00	0.00%	7.00	1
Max	34	14	46.34	178	3.56	72	549	183.00	184	0.59%	15.00	3

Table A-124: 3 Teams, 60 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	39	11	23.03	174	2.9	68.33	572	190.67	191	0.17%	11	2
1409	40	7	24.75	183	3.05	66.67	596	198.67	199	0.17%	14	2
1685	33	5	18.24	189	3.15	58.33	570	190.00	190	0.00%	16	3
1737	36	9	18.44	205	3.42	65	625	208.33	209	0.32%	8	1
2626	39	9	37.79	207	3.45	65	655	218.33	219	0.30%	8	1
5129	37	6	39.46	201	3.35	63.33	615	205.00	206	0.49%	10	2
5228	41	6	19.85	208	3.47	70	679	226.33	227	0.29%	18	1
6681	37	6	28	209	3.48	63.33	630	210.00	210	0.00%	17	3
7244	40	8	31.25	207	3.45	66.67	630	210.00	211	0.47%	17	1
7612	38	9	27.42	190	3.17	65	595	198.33	199	0.34%	9	1
Average	38	7.6	26.82	197.30	3.29	65.17	616.7	205.57	206.10	0.26%	12.80	1.7
Min	33	5	18.24	174	2.9	58.33	570	190	190.00	0.00%	8.00	1
Max	41	11	39.46	209	3.48	70	679	226.33	227	0.49%	18.00	3

Table A-125: 3 Teams, 70 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	43	11	27.95	205	2.93	62.86	665	221.67	222	0.15%	17	2
1409	47	8	40.11	218	3.11	67.14	699	233.00	234	0.43%	12	2
1685	44	8	35.32	221	3.16	62.86	666	222.00	222	0.00%	14	3
1737	44	11	27.27	233	3.33	65.71	709	236.33	237	0.28%	9	2
2626	43	7	19.56	240	3.43	62.86	755	251.67	252	0.13%	17	2
5129	45	7	19.44	239	3.41	64.29	730	243.33	244	0.27%	15	1
5228	43	7	22.12	240	3.43	62.86	776	258.67	259	0.13%	12	2
6681	43	7	27.16	245	3.5	62.86	735	245.00	246	0.41%	11	1
7244	45	4	10.64	234	3.34	64.29	713	237.67	238	0.14%	21	2
7612	39	5	15.85	237	3.39	58.57	738	246.00	246	0.00%	9	3
Average	43.6	7.5	24.54	231.20	3.303	63.43	718.6	239.53	240.00	0.19%	13.70	2
Min	39	4	10.64	205	2.93	58.57	665	221.67	222.00	0.00%	9.00	1
Max	47	11	40.11	245	3.5	67.14	776	258.67	259	0.43%	21.00	3

Table A-126: 3 Teams, 80 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	52	10	27.94	243	3.04	66.25	778	259.33	260	0.26%	23	2
1409	53	9	22.53	254	3.18	66.25	809	269.67	270	0.12%	17	2
1685	52	7	26.23	267	3.34	65	802	267.33	268	0.25%	20	1
1737	51	14	37.06	275	3.44	65	835	278.33	279	0.24%	11	1
2626	52	8	36	277	3.46	66.25	867	289.00	289	0.00%	19	3
5129	54	6	36.61	264	3.3	67.5	805	268.33	269	0.25%	13	1
5228	48	8	33.88	281	3.51	60	899	299.67	300	0.11%	21	2
6681	52	6	54.96	287	3.59	65	862	287.33	288	0.23%	10	1
7244	53	8	43.64	267	3.34	67.5	811	270.33	271	0.25%	9	1
7612	46	5	16.72	262	3.28	58.75	812	270.67	271	0.12%	12	2
Average	51.3	8.1	33.56	267.70	3.348	64.75	828	276	276.50	0.18%	15.50	1.6
Min	46	5	16.72	243	3.04	58.75	778	259.33	260.00	0.00%	9.00	1
Max	54	14	54.96	287	3.59	67.5	899	299.67	300	0.26%	23.00	3

Table A-127: 3 Teams, 90 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	58	5	21.16	275	3.06	64.44	874	291.33	292	0.23%	18	2
1409	64	11	26	287	3.19	71.11	907	302.33	303	0.22%	17	2
1685	58	8	36.78	298	3.31	64.44	895	298.33	299	0.22%	17	2
1737	55	10	35.29	313	3.48	63.33	949	316.33	317	0.21%	19	2
2626	59	9	64.08	305	3.39	65.56	949	316.33	317	0.21%	18	1
5129	48	6	17.4	296	3.29	54.44	903	301.00	301	0.00%	23	3
5228	49	9	19.18	314	3.49	56.67	999	333.00	333	0.00%	12	3
6681	54	11	42.56	313	3.48	60	942	314.00	314	0.00%	9	3
7244	57	9	28.56	292	3.24	63.33	886	295.33	296	0.23%	16	1
7612	57	7	22.4	291	3.23	64.44	898	299.33	300	0.22%	22	2
Average	55.9	8.5	31.34	298.40	3.316	62.78	920.2	306.73	307.20	0.15%	17.10	2.1
Min	48	5	17.4	275	3.06	54.44	874	291.33	292.00	0.00%	9.00	1
Max	64	11	64.08	314	3.49	71.11	999	333.00	333	0.23%	23.00	3

Table A-128: 3 Teams, 100 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	58	7	34.07	303	3.03	58	959	319.67	320	0.10%	14	2
1409	63	9	27.11	307	3.07	63	969	323.00	323	0.00%	16	3
1685	65	9	24.14	341	3.41	65	1025	341.67	342	0.10%	19	2
1737	57	6	21.12	350	3.5	59	1061	353.67	354	0.09%	15	2
2626	64	10	22.39	336	3.36	64	1040	346.67	348	0.38%	17	1
5129	62	6	20	331	3.31	62	1006	335.33	336	0.20%	30	1
5228	66	8	60.27	352	3.52	66	1110	370.00	371	0.27%	18	2
6681	58	11	26.53	355	3.55	58	1067	355.67	356	0.09%	16	2
7244	65	12	34.2	328	3.28	65	996	332.00	332	0.00%	21	3
7612	63	7	50.27	328	3.28	63	1010	336.67	337	0.10%	12	2
Average	62.1	8.5	32.01	333.10	3.331	62.30	1024.3	341.43	341.90	0.13%	17.80	2
Min	57	6	20	303	3.03	58	959	319.67	320.00	0.00%	12.00	1
Max	66	12	60.27	355	3.55	66	1110	370.00	371	0.38%	30.00	3

A2.18. Genetic Algorithm (Makespan/JIT) – 4 Teams

Table A-129: 4 Teams, 30 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	24	9	17.29	62	2.07	80	310	77.50	79	1.90%	5	2
1409	20	5	9.1	62	2.07	66.67	307	76.75	78	1.60%	6	2
1685	21	5	16.33	76	2.53	70	302	75.50	77	1.95%	11	1
1737	18	3	14.17	68	2.27	70	287	71.75	72	0.35%	2	3
2626	22	6	14.91	67	2.23	73.33	312	78.00	79	1.27%	7	2
5129	19	6	15.42	66	2.2	63.33	282	70.50	71	0.70%	8	2
5228	18	5	10.94	57	1.9	66.67	303	75.75	76	0.33%	6	3
6681	18	5	16.5	83	2.77	63.33	332	83.00	84	1.19%	4	2
7244	19	5	8.42	73	2.43	66.67	303	75.75	77	1.62%	6	3
7612	22	4	14.68	72	2.4	73.33	319	79.75	81	1.54%	7	3
Average	20.1	5.3	13.776	68.6	2.287	69.333	305.7	76.425	77.4	1.25%	6.2	2.3
Min	18	3	8.42	57	1.9	63.33	282	70.5	71	0.33%	2	1
Max	24	9	17.29	83	2.77	80	332	83	84	1.95%	11	3

Table A-130: 4 Teams, 40 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	24	4	6.54	85	2.13	60	404	101.00	102	0.98%	8	1
1409	31	6	16.61	80	2	77.5	381	95.25	96	0.78%	11	2
1685	28	6	14.5	97	2.43	70	387	96.75	98	1.28%	8	3
1737	27	13	26.04	95	2.38	70	392	98.00	99	1.01%	5	1
2626	26	5	15.96	94	2.35	65	417	104.25	106	1.65%	10	1
5129	29	8	17.76	101	2.53	72.5	419	104.75	106	1.18%	9	1
5228	27	7	16.52	81	2.03	67.5	398	99.50	100	0.50%	10	3
6681	23	8	18.26	106	2.65	57.5	423	105.75	107	1.17%	6	1
7244	29	8	22.17	104	2.6	72.5	428	107.00	108	0.93%	7	3
7612	23	6	8.48	89	2.23	62.5	389	97.25	98	0.77%	9	2
Average	26.7	7.1	16.284	93.2	2.333	67.5	403.8	100.95	102	1.02%	8.3	1.8
Min	23	4	6.54	80	2	57.5	381	95.25	96	0.50%	5	1
Max	31	13	26.04	106	2.65	77.5	428	107	108	1.65%	11	3

Table A-131: 4 Teams, 50 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	36	6	17.03	110	2.2	72	503	125.75	127	0.98%	12	1
1409	36	8	20.03	103	2.06	74	470	117.50	119	1.26%	8	2
1685	35	11	30.09	119	2.38	70	476	119.00	120	0.83%	12	1
1737	36	10	19.14	130	2.6	72	532	133.00	134	0.75%	8	2
2626	36	6	16.44	127	2.54	72	549	137.25	139	1.26%	15	1
5129	31	7	25.97	131	2.62	62	540	135.00	136	0.74%	11	1
5228	34	5	22.38	118	2.36	68	540	135.00	137	1.46%	14	2
6681	33	4	15.48	126	2.52	68	505	126.25	127	0.59%	12	2
7244	32	11	19.41	134	2.68	66	544	136.00	138	1.45%	9	1
7612	30	8	14	119	2.38	60	504	126.00	128	1.56%	12	1
Average	33.9	7.6	19.997	121.7	2.434	68.4	516.3	129.075	130.5	1.09%	11.3	1.4
Min	30	4	14	103	2.06	60	470	117.5	119	0.59%	8	1
Max	36	11	30.09	134	2.68	74	549	137.25	139	1.56%	15	2

Table A-132: 4 Teams, 60 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	38	7	18	127	2.12	65	572	143.00	144	0.69%	18	2
1409	46	18	26.72	134	2.23	76.67	596	149.00	150	0.67%	11	2
1685	45	14	25.6	144	2.4	75	570	142.50	145	1.72%	20	1
1737	37	8	28.84	154	2.57	65	625	156.25	158	1.11%	8	1
2626	42	10	33.74	154	2.57	70	655	163.75	166	1.36%	9	2
5129	35	6	15.03	151	2.52	61.67	615	153.75	156	1.44%	12	2
5228	45	12	25.18	153	2.55	75	679	169.75	172	1.31%	13	1
6681	41	7	15.63	160	2.67	68.33	630	157.50	161	2.17%	12	1
7244	43	8	17.72	156	2.6	71.67	630	157.50	160	1.56%	13	1
7612	43	8	19.51	141	2.35	73.33	595	148.75	150	0.83%	19	2
Average	41.5	9.8	22.597	147.4	2.458	70.167	616.7	154.175	156.2	1.29%	13.5	1.5
Min	35	6	15.03	127	2.12	61.67	570	142.5	144	0.67%	8	1
Max	46	18	33.74	160	2.67	76.67	679	169.75	172	2.17%	20	2

Table A-133: 4 Teams, 70 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	50	10	19.52	150	2.14	71.43	665	166.25	167	0.45%	15	2
1409	49	7	28.37	161	2.3	71.43	699	174.75	177	1.27%	18	2
1685	43	13	15.93	168	2.4	62.86	666	166.50	169	1.48%	18	1
1737	45	6	18.64	175	2.5	64.29	709	177.25	179	0.98%	12	1
2626	51	11	32.78	179	2.56	72.86	755	188.75	191	1.18%	17	1
5129	52	11	31.96	180	2.57	74.29	730	182.50	185	1.35%	17	1
5228	49	8	24.33	178	2.54	70	776	194.00	197	1.52%	13	1
6681	50	10	22.92	184	2.63	71.43	735	183.75	185	0.68%	16	1
7244	42	9	17.38	175	2.5	64.29	713	178.25	179	0.42%	10	3
7612	46	11	24.33	179	2.56	65.71	738	184.50	188	1.86%	20	1
Average	47.7	9.6	23.616	172.9	2.47	68.859	718.6	179.65	181.7	1.12%	15.6	1.4
Min	42	6	15.93	150	2.14	62.86	665	166.25	167	0.42%	10	1
Max	52	13	32.78	184	2.63	74.29	776	194	197	1.86%	20	3

Table A-134: 4 Teams, 80 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	57	10	22.32	180	2.25	71.25	778	194.50	197	1.27%	16	1
1409	50	5	20.66	188	2.35	62.5	809	202.25	204	0.86%	17	1
1685	53	7	27.06	203	2.54	68.75	802	200.50	204	1.72%	15	1
1737	58	9	30.9	208	2.6	73.75	835	208.75	212	1.53%	19	1
2626	48	6	13.85	206	2.58	62.5	867	216.75	218	0.57%	18	1
5129	53	7	20.83	199	2.49	66.25	805	201.25	204	1.35%	22	1
5228	58	14	32.86	208	2.6	72.5	899	224.75	227	0.99%	21	1
6681	50	12	20.28	217	2.71	62.5	862	215.50	218	1.15%	14	1
7244	44	8	21.7	200	2.5	55	811	202.75	204	0.61%	16	1
7612	52.33	8.67	23.38	201	2.51	66.11	829.78	207.44	209.78	1.12%	17.56	1
Average	44	5	13.85	180	2.25	55	778	194.5	197	0.57%	14	1
Min	58	14	32.86	217	2.71	73.75	899	224.75	227	1.72%	22	1
Max	57	10	22.32	180	2.25	71.25	778	194.50	197	1.27%	16	1

Table A-135: 4 Teams, 90 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	61	9	27.57	203	2.26	67.78	874	218.50	220	0.68%	20	1
1409	64	7	22.06	213	2.37	71.11	907	226.75	229	0.98%	20	1
1685	57	9	27.56	224	2.49	64.44	895	223.75	225	0.56%	19	2
1737	63	7	44.03	237	2.63	71.11	949	237.25	241	1.56%	22	1
2626	67	8	25.63	226	2.51	74.44	949	237.25	238	0.32%	19	1
5129	67	11	27.9	222	2.47	74.44	903	225.75	227	0.55%	16	2
5228	62	8	29.19	232	2.58	68.89	999	249.75	251	0.50%	31	1
6681	59	7	19.98	237	2.63	65.56	942	235.50	238	1.05%	24	1
7244	64	11	21.92	220	2.44	71.11	886	221.50	224	1.12%	23	1
7612	63	11	37.87	217	2.41	70	898	224.50	226	0.66%	20	1
Average	62.7	8.8	28.371	223.1	2.479	69.888	920.2	230.05	231.9	0.80%	21.4	1.2
Min	57	7	19.98	203	2.26	64.44	874	218.5	220	0.32%	16	1
Max	67	11	44.03	237	2.63	74.44	999	249.75	251	1.56%	31	2

Table A-136: 4 Teams, 100 Parts (GA – Makespan/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	74	11	28.42	225	2.25	74	959	239.75	242	0.93%	19	1
1409	72	6	23.06	230	2.3	72	969	242.25	246	1.52%	18	1
1685	72	13	39.86	257	2.57	73	1025	256.25	258	0.68%	23	2
1737	68	11	24.93	264	2.64	69	1061	265.25	268	1.03%	23	2
2626	69	11	34.25	250	2.5	69	1040	260.00	262	0.76%	13	1
5129	72	10	38.68	250	2.5	72	1006	251.50	255	1.37%	32	2
5228	70	11	29.07	262	2.62	70	1110	277.50	281	1.25%	18	1
6681	71	12	33.92	269	2.69	71	1067	266.75	270	1.20%	15	1
7244	69	8	22.29	249	2.49	70	996	249.00	253	1.58%	16	1
7612	73	10	22.97	246	2.46	73	1010	252.50	255	0.98%	24	1
Average	71	10.3	29.75	250.2	2.502	71.3	1024.3	256.08	259	1.13%	20.1	1.3
Min	68	6	22.29	225	2.25	69	959	239.75	242	0.68%	13	1
Max	74	13	39.86	269	2.69	74	1110	277.5	281	1.58%	32	2

A2.19. Genetic Algorithm (Makespan/Max in Stack/JIT) – 2 Teams

Table A-137: 2 Teams, 30 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	16	2	15.75	138	4.6	53.33	310	155.00	155	0.00%	6	2
1409	9	1	6	138	4.6	36.67	307	153.50	154	0.32%	2	1
1685	11	1	23.64	150	5	36.67	302	151.00	151	0.00%	7	2
1737	11	1	7.91	140	4.67	43.33	287	143.50	144	0.35%	1	1
2626	16	2	37.63	144	4.8	53.33	312	156.00	156	0.00%	6	2
5129	8	1	4.5	136	4.53	40	282	141.00	141	0.00%	8	2
5228	11	1	7	133	4.43	40	303	151.50	152	0.33%	5	1
6681	10	1	17.4	165	5.5	36.67	332	166.00	166	0.00%	2	2
7244	11	1	9.91	148	4.93	40	303	151.50	152	0.33%	2	1
7612	11	1	12.64	151	5.03	36.67	319	159.50	160	0.31%	1	1
Average	11.4	1.2	14.24	144.3	4.81	41.67	305.7	152.85	153.1	0.16%	4	1.5
Min	8	1	4.5	133	4.43	36.67	282	141	141	0.00%	1	1
Max	16	2	37.63	165	5.5	53.33	332	166	166	0.35%	8	2

Table A-138: 2 Teams, 40 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	17	2	12.12	185	4.63	47.5	404	202.00	202	0.00%	5	2
1409	19	2	20.89	175	4.38	47.5	381	190.50	191	0.26%	5	1
1685	17	2	8.41	193	4.83	45	387	193.50	194	0.26%	8	1
1737	17	2	50.29	192	4.8	47.5	392	196.00	196	0.00%	5	2
2626	20	2	36.55	197	4.93	50	417	208.50	209	0.24%	8	1
5129	11	1	11.18	205	5.13	27.5	419	209.50	210	0.24%	1	1
5228	10	1	6.7	180	4.5	30	398	199.00	199	0.00%	0	2
6681	9	1	4.67	211	5.28	25	423	211.50	212	0.24%	2	1
7244	18	2	11.5	210	5.25	45	428	214.00	214	0.00%	4	2
7612	13	1	6.08	186	4.65	32.5	389	194.50	195	0.26%	3	1
Average	15.1	1.6	16.84	193.4	4.84	39.75	403.8	201.9	202.2	0.15%	4.1	1.4
Min	9	1	4.67	175	4.38	25	381	190.5	191	0.00%	0	1
Max	20	2	50.29	211	5.28	50	428	214	214	0.26%	8	2

Table A-139: 2 Teams, 50 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	24	2	12.67	235	4.7	48	503	251.50	252	0.20%	7	1
1409	20	2	7.75	219	4.38	42	470	235.00	235	0.00%	13	2
1685	23	2	13.91	237	4.74	46	476	238.00	238	0.00%	12	2
1737	23	2	35.65	262	5.24	48	532	266.00	266	0.00%	5	2
2626	23	2	12.04	263	5.26	48	549	274.50	275	0.18%	8	1
5129	20	2	9.6	265	5.3	42	540	270.00	270	0.00%	4	2
5228	20	2	21	251	5.02	44	540	270.00	270	0.00%	7	2
6681	22	2	19.68	252	5.04	44	505	252.50	253	0.20%	9	1
7244	25	2	48.32	268	5.36	50	544	272.00	272	0.00%	6	2
7612	24	2	40.46	243	4.86	48	504	252.00	252	0.00%	10	2
Average	22.4	2	22.11	249.5	4.99	46.00	516.3	258.15	258.3	0.06%	8.1	1.7
Min	20	2	7.75	219	4.38	42	470	235	235	0.00%	4	1
Max	25	2	48.32	268	5.36	50	549	274.5	275	0.20%	13	2

Table A-140: 2 Teams, 60 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	26	2	11.38	269	4.48	45	572	286.00	286	0.00%	8	2
1409	23	2	8.43	282	4.7	41.67	596	298.00	298	0.00%	7	2
1685	29	2	42.79	284	4.73	48.33	570	285.00	285	0.00%	8	2
1737	23	2	8.43	309	5.15	41.67	625	312.50	313	0.16%	7	1
2626	26	2	23.23	316	5.27	43.33	655	327.50	328	0.15%	9	1
5129	24	2	12.67	303	5.05	43.33	615	307.50	308	0.16%	9	1
5228	25	2	14.12	321	5.35	46.67	679	339.50	340	0.15%	9	1
6681	26	2	24.96	314	5.23	43.33	630	315.00	315	0.00%	9	2
7244	26	2	15.19	311	5.18	45	630	315.00	315	0.00%	11	2
7612	22	2	12.59	289	4.82	41.67	595	297.50	298	0.17%	9	1
Average	25	2	17.38	299.8	5.00	44.00	616.7	308.35	308.6	0.08%	8.6	1.5
Min	22	2	8.43	269	4.48	41.67	570	285	285	0.00%	7	1
Max	29	2	42.79	321	5.35	48.33	679	339.5	340	0.17%	11	2

Table A-141: 2 Teams, 70 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	29	2	10.97	316	4.51	41.43	665	332.50	333	0.15%	10	1
1409	23	2	10.91	334	4.77	32.86	699	349.50	350	0.14%	6	1
1685	26	2	14.5	332	4.74	38.57	666	333.00	333	0.00%	6	2
1737	30	2	16.87	351	5.01	45.71	709	354.50	355	0.14%	10	1
2626	32	2	16.59	366	5.23	45.71	755	377.50	378	0.13%	8	1
5129	28	2	11.82	360	5.14	41.43	730	365.00	365	0.00%	7	2
5228	26	2	15.92	369	5.27	40	776	388.00	388	0.00%	8	2
6681	29	2	27.72	367	5.24	41.43	735	367.50	368	0.14%	9	1
7244	33	3	11.42	353	5.04	47.14	713	356.50	357	0.14%	10	1
7612	36	3	62.31	360	5.14	51.43	738	369.00	369	0.00%	6	2
Average	29.2	2.2	19.90	350.8	5.01	42.57	718.6	359.3	359.6	0.08%	8	1.4
Min	23	2	10.91	316	4.51	32.86	665	332.5	333	0.00%	6	1
Max	36	3	62.31	369	5.27	51.43	776	388	388	0.15%	10	2

Table A-142: 2 Teams, 80 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	37	3	14.03	372	4.65	47.5	778	389.00	389	0.00%	14	2
1409	31	2	11.81	389	4.86	40	809	404.50	405	0.12%	9	1
1685	38	3	18.11	400	5	47.5	802	401.00	401	0.00%	12	2
1737	35	3	36.06	414	5.18	46.25	835	417.50	418	0.12%	7	1
2626	28	2	13.68	422	5.28	35	867	433.50	434	0.12%	10	1
5129	37	3	28.19	398	4.98	47.5	805	402.50	403	0.12%	8	1
5228	28	2	10.32	431	5.39	35	899	449.50	450	0.11%	9	1
6681	37	3	14.78	430	5.38	47.5	862	431.00	431	0.00%	11	2
7244	29	2	14.45	402	5.03	37.5	811	405.50	406	0.12%	9	1
7612	38	3	27.55	397	4.96	47.5	812	406.00	406	0.00%	8	2
Average	33.8	2.6	18.90	405.5	5.07	43.13	828	414	414.3	0.07%	9.7	1.4
Min	28	2	10.32	372	4.65	35	778	389	389	0.00%	7	1
Max	38	3	36.06	431	5.39	47.5	899	449.5	450	0.12%	14	2

Table A-143: 2 Teams, 90 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	43	3	20.4	420	4.67	47.78	874	437.00	437	0.00%	16	2
1409	35	2	10.49	438	4.87	40	907	453.50	454	0.11%	10	1
1685	44	3	36.59	447	4.97	48.89	895	447.50	448	0.11%	10	1
1737	33	2	10.79	471	5.23	41.11	949	474.50	475	0.11%	17	1
2626	42	3	30.81	463	5.14	46.67	949	474.50	475	0.11%	21	1
5129	44	3	40.55	447	4.97	48.89	903	451.50	452	0.11%	19	1
5228	34	2	14.47	481	5.34	38.89	999	499.50	500	0.10%	12	1
6681	43	3	24.12	470	5.22	47.78	942	471.00	471	0.00%	16	2
7244	43	3	29.33	439	4.88	47.78	886	443.00	443	0.00%	19	2
7612	36	2	33.14	440	4.89	40	898	449.00	449	0.00%	9	2
Average	39.7	2.6	25.07	451.6	5.02	44.78	920.2	460.1	460.4	0.06%	14.9	1.4
Min	33	2	10.49	420	4.67	38.89	874	437	437	0.00%	9	1
Max	44	3	40.55	481	5.34	48.89	999	499.5	500	0.11%	21	2

Table A-144: 2 Teams, 100 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	41	2	11.61	463	4.63	43	959	479.50	480	0.10%	14	1
1409	45	2	16.31	469	4.69	46	969	484.50	485	0.10%	12	1
1685	45	3	15.02	512	5.12	45	1025	512.50	513	0.10%	24	1
1737	48	3	35.52	527	5.27	49	1061	530.50	531	0.09%	11	1
2626	48	3	27.52	508	5.08	49	1040	520.00	520	0.00%	13	2
5129	48	3	27.13	498	4.98	49	1006	503.00	503	0.00%	14	2
5228	47	3	19.11	536	5.36	47	1110	555.00	555	0.00%	18	2
6681	46	3	17.83	533	5.33	48	1067	533.50	534	0.09%	19	1
7244	40	3	32.9	494	4.94	42	996	498.00	498	0.00%	10	2
7612	46	3	20.61	496	4.96	46	1010	505.00	505	0.00%	17	2
Average	45.4	2.8	22.36	503.6	5.04	46.40	1024.3	512.15	512.4	0.05%	15.2	1.5
Min	40	2	11.61	463	4.63	42	959	479.5	480	0.00%	10	1
Max	48	3	35.52	536	5.36	49	1110	555	555	0.10%	24	2

A2.20. Genetic Algorithm (Makespan/Max in Stack /JIT) – 3 Teams

Table A-145: 3 Teams, 30 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	16	3	7.38	87	2.9	53.33	310	103.33	104	0.64%	5	2
1409	18	2	11.39	87	2.9	60	307	102.33	103	0.65%	6	1
1685	15	3	12.4	100	3.33	53.33	302	100.67	101	0.33%	6	2
1737	19	3	10.68	92	3.07	70	287	95.67	96	0.35%	7	2
2626	14	3	10	92	3.07	50	312	104.00	104	0.00%	6	3
5129	18	3	15.33	89	2.97	60	282	94.00	94	0.00%	8	3
5228	17	4	19.71	82	2.73	56.67	303	101.00	101	0.00%	4	3
6681	13	2	10.38	110	3.67	46.67	332	110.67	111	0.30%	4	2
7244	16	3	6.81	97	3.23	53.33	303	101.00	101	0.00%	3	3
7612	15	2	11.47	98	3.27	50	319	106.33	107	0.62%	4	2
Average	16.1	2.8	11.56	93.4	3.11	55.333	305.7	101.90	102.2	0.29%	5.3	2.3
Min	13	2	6.81	82	2.73	46.67	282	94.00	94	0.00%	3	1
Max	19	4	19.71	110	3.67	70	332	111	111	0.65%	8	3

Table A-146: 3 Teams, 40 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	25	4	19.96	118	2.95	62.5	404	134.67	135	0.25%	9	2
1409	17	2	7.41	111	2.78	47.5	381	127.00	127	0.00%	6	3
1685	19	4	10.89	128	3.2	52.5	387	129.00	129	0.00%	9	3
1737	23	3	19.43	127	3.18	60	392	130.67	131	0.25%	10	2
2626	27	5	19.78	127	3.18	67.5	417	139.00	139	0.00%	6	3
5129	23	3	9.22	135	3.38	60	419	139.67	140	0.24%	7	2
5228	20	4	9.9	114	2.85	57.5	398	132.67	133	0.25%	6	2
6681	24	5	14.42	140	3.5	60	423	141.00	141	0.00%	8	3
7244	24	3	33.46	139	3.48	60	428	142.67	143	0.23%	6	2
7612	23	5	13.91	121	3.03	57.5	389	129.67	130	0.26%	5	2
Average	22.5	3.8	15.84	126	3.15	58.5	403.8	134.60	134.8	0.15%	7.2	2.4
Min	17	2	7.41	111	2.78	47.5	381	127.00	127	0.00%	5	2
Max	27	5	33.46	140	3.5	67.5	428	143	143	0.26%	10	3

Table A-147: 3 Teams, 50 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	29	4	16.76	151	3.02	58	503	167.67	168	0.20%	7	2
1409	25	4	13.16	141	2.82	50	470	156.67	157	0.21%	6	2
1685	28	5	23.96	158	3.16	58	476	158.67	159	0.21%	8	2
1737	23	4	16.09	174	3.48	50	532	177.33	178	0.37%	6	2
2626	30	4	16.57	172	3.44	60	549	183.00	184	0.54%	13	1
5129	31	4	11.58	175	3.5	62	540	180.00	180	0.00%	14	3
5228	32	6	17.47	161	3.22	64	540	180.00	180	0.00%	14	3
6681	27	4	15.89	168	3.36	56	505	168.33	169	0.39%	8	1
7244	27	4	13.15	178	3.56	56	544	181.33	182	0.37%	8	2
7612	22	4	20.5	159	3.18	50	504	168.00	168	0.00%	8	3
Average	27.4	4.3	16.51	163.7	3.27	56.4	516.3	172.10	172.5	0.23%	9.2	2.1
Min	22	4	11.58	141	2.82	50	470	156.67	157	0.00%	6	1
Max	32	6	23.96	178	3.56	64	549	183	184	0.54%	14	3

Table A-148: 3 Teams, 60 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	32	5	20.16	174	2.9	55	572	190.67	191	0.17%	12	2
1409	37	6	31.22	183	3.05	63.33	596	198.67	199	0.17%	7	2
1685	30	5	13	189	3.15	51.67	570	190.00	190	0.00%	8	3
1737	31	5	8.9	205	3.42	55	625	208.33	209	0.32%	12	1
2626	32	4	12.91	207	3.45	53.33	655	218.33	219	0.30%	8	2
5129	32	7	29.75	200	3.33	53.33	615	205.00	205	0.00%	6	3
5228	33	4	24.15	208	3.47	55	679	226.33	227	0.29%	12	2
6681	36	5	30.42	210	3.5	60	630	210.00	211	0.47%	9	1
7244	35	4	19.57	207	3.45	58.33	630	210.00	211	0.47%	9	1
7612	37	6	23.59	190	3.17	65	595	198.33	199	0.34%	16	1
Average	33.5	5.1	21.37	197.3	3.29	56.999	616.7	205.57	206.1	0.25%	9.9	1.8
Min	30	4	8.9	174	2.9	51.67	570	190.00	190	0.00%	6	1
Max	37	7	31.22	210	3.5	65	679	226	227	0.47%	16	3

Table A-149: 3 Teams, 70 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	47	5	18.7	205	2.93	67.14	665	221.67	222	0.15%	17	2
1409	43	6	16.6	217	3.1	61.43	699	233.00	233	0.00%	12	3
1685	42	5	17.33	221	3.16	60	666	222.00	222	0.00%	16	3
1737	39	5	18.13	233	3.33	58.57	709	236.33	237	0.28%	13	1
2626	46	6	32.46	240	3.43	67.14	755	251.67	252	0.13%	15	2
5129	43	7	30.86	239	3.41	61.43	730	243.33	244	0.27%	23	1
5228	40	5	23.55	240	3.43	57.14	776	258.67	259	0.13%	12	2
6681	30	5	18.57	245	3.5	47.14	735	245.00	246	0.41%	12	1
7244	40	5	17.4	234	3.34	58.57	713	237.67	238	0.14%	15	2
7612	39	5	20.56	238	3.4	55.71	738	246.00	247	0.40%	16	1
Average	40.9	5.4	21.42	231.2	3.30	59.427	718.6	239.53	240	0.19%	15.1	1.8
Min	30	5	16.6	205	2.93	47.14	665	221.67	222	0.00%	12	1
Max	47	7	32.46	245	3.5	67.14	776	259	259	0.41%	23	3

Table A-150: 3 Teams, 80 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	45	4	20.16	243	3.04	56.25	778	259.33	260	0.26%	15	1
1409	44	4	16.82	254	3.18	60	809	269.67	270	0.12%	18	2
1685	47	7	29.85	267	3.34	60	802	267.33	268	0.25%	22	2
1737	48	5	41.19	275	3.44	61.25	835	278.33	279	0.24%	13	2
2626	48	5	26.46	278	3.48	61.25	867	289.00	290	0.34%	13	1
5129	46	4	16.54	264	3.3	58.75	805	268.33	269	0.25%	13	1
5228	45	5	12.67	281	3.51	56.25	899	299.67	300	0.11%	12	2
6681	43	3	22.4	287	3.59	53.75	862	287.33	288	0.23%	10	2
7244	49	6	27.06	267	3.34	61.25	811	270.33	271	0.25%	14	2
7612	48	5	17.06	262	3.28	61.25	812	270.67	271	0.12%	22	2
Average	46.3	4.8	23.02	267.8	3.35	59	828	276.00	276.6	0.22%	15.2	1.7
Min	43	3	12.67	243	3.04	53.75	778	259.33	260	0.11%	10	1
Max	49	7	41.19	287	3.59	61.25	899	300	300	0.34%	22	2

Table A-151: 4 Teams, 90 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	58	5	21.16	275	3.06	64.44	874	291.33	292	0.23%	18	2
1409	64	11	26	287	3.19	71.11	907	302.33	303	0.22%	17	2
1685	58	8	36.78	298	3.31	64.44	895	298.33	299	0.22%	17	2
1737	55	10	35.29	313	3.48	63.33	949	316.33	317	0.21%	19	2
2626	59	9	64.08	305	3.39	65.56	949	316.33	317	0.21%	18	1
5129	48	6	17.4	296	3.29	54.44	903	301.00	301	0.00%	23	3
5228	49	9	19.18	314	3.49	56.67	999	333.00	333	0.00%	12	3
6681	54	11	42.56	313	3.48	60	942	314.00	314	0.00%	9	3
7244	57	9	28.56	292	3.24	63.33	886	295.33	296	0.23%	16	1
7612	57	7	22.4	291	3.23	64.44	898	299.33	300	0.22%	22	2
Average	55.9	8.5	31.34	298.40	3.316	62.78	920.2	306.73	307.20	0.15%	17.10	2.1
Min	48	5	17.4	275	3.06	54.44	874	291.33	292.00	0.00%	9.00	1
Max	64	11	64.08	314	3.49	71.11	999	333.00	333	0.23%	23.00	3

Table A-152: 3 Teams, 100 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	58	7	34.07	303	3.03	58	959	319.67	320	0.10%	14	2
1409	63	9	27.11	307	3.07	63	969	323	323	0.00%	16	3
1685	65	9	24.14	341	3.41	65	1025	341.67	342	0.10%	19	2
1737	57	6	21.12	350	3.5	59	1061	353.67	354	0.09%	15	2
2626	64	10	22.39	336	3.36	64	1040	346.67	348	0.38%	17	1
5129	62	6	20	331	3.31	62	1006	335.33	336	0.20%	30	1
5228	66	8	60.27	352	3.52	66	1110	370	371	0.27%	18	2
6681	58	11	26.53	355	3.55	58	1067	355.67	356	0.09%	16	2
7244	65	12	34.2	328	3.28	65	996	332	332	0.00%	21	3
7612	63	7	50.27	328	3.28	63	1010	336.67	337	0.10%	12	2
Average	62.1	8.5	32.01	333.10	3.331	62.30	1024.3	341.43	341.90	0.13%	17.80	2
Min	57	6	20	303	3.03	58	959	319.67	320.00	0.00%	12	1
Max	66	12	60.27	355	3.55	66	1110	370.00	371	0.38%	30	3

A2.21. Genetic Algorithm (Makespan/Max in Stack/JIT) – 4 Teams

Table A-153: 4 Teams, 30 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	21	4	9.14	61	2.03	70	310	77.50	78	0.64%	5	2
1409	18	4	16.39	61	2.03	63.33	307	76.75	77	0.32%	7	3
1685	18	6	15.94	76	2.53	60	302	75.50	77	1.95%	6	1
1737	19	3	7.42	69	2.3	66.67	287	71.75	73	1.71%	3	1
2626	22	6	14.91	67	2.23	73.33	312	78.00	79	1.27%	7	2
5129	20	5	11.75	66	2.2	70	282	70.50	71	0.70%	9	3
5228	18	4	15.28	57	1.9	60	303	75.75	76	0.33%	5	3
6681	20	5	19.15	83	2.77	66.67	332	83.00	84	1.19%	6	2
7244	23	5	18.39	72	2.4	76.67	303	75.75	76	0.33%	7	3
7612	17	4	16.47	72	2.4	63.33	319	79.75	81	1.54%	4	3
Average	19.6	4.6	14.484	68.4	2.279	67.00	305.7	76.425	77.2	1.00%	5.9	2.3
Min	17	3	7.42	57	1.9	60	282	70.5	71	0.32%	3	1
Max	23	6	19.15	83	2.77	76.67	332	83	84	1.95%	9	3

Table A-154: 4 Teams, 40 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	30	7	13.83	85	2.13	75	404	101.00	102	0.98%	7	1
1409	29	7	14.24	80	2	72.5	381	95.25	96	0.78%	8	2
1685	27	5	14.04	97	2.43	67.5	387	96.75	98	1.28%	8	3
1737	27	4	23.44	95	2.38	72.5	392	98.00	99	1.01%	7	1
2626	23	4	15.65	93	2.33	62.5	417	104.25	105	0.71%	7	3
5129	27	6	17.78	101	2.53	67.5	419	104.75	106	1.18%	11	1
5228	24	3	11.79	82	2.05	60	398	99.50	101	1.49%	6	1
6681	20	3	13.65	107	2.68	50	423	105.75	108	2.08%	5	1
7244	29	4	19.31	104	2.6	72.5	428	107.00	108	0.93%	9	1
7612	23	6	12.48	89	2.23	62.5	389	97.25	98	0.77%	9	2
Average	25.9	4.9	15.621	93.3	2.336	66.25	403.8	100.95	102.1	1.12%	7.7	1.6
Min	20	3	11.79	80	2	50	381	95.25	96	0.71%	5	1
Max	30	7	23.44	107	2.68	75	428	107	108	2.08%	11	3

Table A-155: 4 Teams, 50 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	34	8	20.65	110	2.2	68	503	125.75	127	0.98%	11	2
1409	31	5	19.45	102	2.04	64	470	117.50	118	0.42%	14	2
1685	36	8	18.89	119	2.38	72	476	119.00	120	0.83%	17	3
1737	29	5	13.86	131	2.62	60	532	133.00	135	1.48%	8	1
2626	32	6	15.88	127	2.54	66	549	137.25	139	1.26%	10	2
5129	28	3	10.36	132	2.64	58	540	135.00	137	1.46%	5	1
5228	37	4	21.49	118	2.36	74	540	135.00	137	1.46%	16	1
6681	30	5	17.2	127	2.54	62	505	126.25	128	1.37%	8	1
7244	35	8	24.31	134	2.68	70	544	136.00	138	1.45%	15	1
7612	29	3	9.52	119	2.38	60	504	126.00	128	1.56%	11	2
Average	32.1	5.5	17.161	121.9	2.438	65.40	516.3	129.075	130.7	1.23%	11.5	1.6
Min	28	3	9.52	102	2.04	58	470	117.5	118	0.42%	5	1
Max	37	8	24.31	134	2.68	74	549	137.25	139	1.56%	17	3

Table A-156: 4 Teams, 60 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	38	7	18	127	2.12	65	572	143.00	144	0.69%	18	2
1409	41	9	14.05	135	2.25	70	596	149.00	151	1.32%	15	1
1685	44	8	22.16	142	2.37	73.33	570	142.50	143	0.35%	13	3
1737	37	6	17.14	155	2.58	65	625	156.25	159	1.73%	15	1
2626	43	7	17.19	153	2.55	73.33	655	163.75	165	0.76%	18	3
5129	37	5	18.27	150	2.5	65	615	153.75	155	0.81%	13	2
5228	41	9	39	153	2.55	68.33	679	169.75	172	1.31%	12	1
6681	39	4	15.74	157	2.62	65	630	157.50	158	0.32%	17	2
7244	41	9	17	155	2.58	68.33	630	157.50	159	0.94%	12	2
7612	40	6	19.85	141	2.35	68.33	595	148.75	150	0.83%	13	3
Average	40.1	7	19.84	146.8	2.45	68.17	616.7	154.18	155.6	0.91%	14.6	2
Min	37	4	14.05	127	2.12	65	570	142.5	143	0.32%	12	1
Max	44	9	39	157	2.62	73.33	679	169.75	172	1.73%	18	3

Table A-157: 4 Teams, 70 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	46	7	16.15	151	2.16	65.71	665	166.25	168	1.04%	17	1
1409	45	7	21.13	161	2.3	64.29	699	174.75	177	1.27%	13	2
1685	39	7	20.51	168	2.4	57.14	666	166.50	169	1.48%	8	1
1737	45	8	26.73	174	2.49	71.43	709	177.25	178	0.42%	14	2
2626	47	9	30.23	179	2.56	68.57	755	188.75	191	1.18%	20	2
5129	47	6	14.74	179	2.56	67.14	730	182.50	184	0.82%	17	3
5228	47	8	21.66	177	2.53	67.14	776	194.00	196	1.02%	14	1
6681	41	7	21.8	183	2.61	61.43	735	183.75	184	0.14%	12	3
7244	46	8	17.33	177	2.53	65.71	713	178.25	181	1.52%	22	1
7612	44	7	20.3	178	2.54	62.86	738	184.50	187	1.34%	11	2
Average	44.7	7.4	21.058	172.7	2.468	65.14	718.6	179.65	181.5	1.02%	14.8	1.8
Min	39	6	14.74	151	2.16	57.14	665	166.25	168	0.14%	8	1
Max	47	9	30.23	183	2.61	71.43	776	194	196	1.52%	22	3

Table A-158: 4 Teams, 80 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	57	8	23.04	179	2.24	71.25	778	194.50	196	0.77%	24	1
1409	56	13	36.04	188	2.35	70	809	202.25	204	0.86%	16	1
1685	57	7	18.63	202	2.53	71.25	802	200.50	203	1.23%	13	2
1737	56	7	35.25	206	2.58	71.25	835	208.75	210	0.60%	12	1
2626	52	6	15.52	207	2.59	65	867	216.75	219	1.03%	15	1
5129	57	7	23.6	199	2.49	71.25	805	201.25	204	1.35%	20	1
5228	52	6	24.12	208	2.6	67.5	899	224.75	227	0.99%	14	1
6681	54	10	30.37	216	2.7	68.75	862	215.50	217	0.69%	14	1
7244	58	9	33.1	201	2.51	72.5	811	202.75	205	1.10%	13	1
7612	54	8	16.28	196	2.45	67.5	812	203.00	205	0.98%	23	2
Average	55.3	8.1	25.595	200.2	2.504	69.63	828	207	209	0.96%	16.4	1.2
Min	52	6	15.52	179	2.24	65	778	194.5	196	0.60%	12	1
Max	58	13	36.04	216	2.7	72.5	899	224.75	227	1.35%	24	2

Table A-159: 4 Teams, 90 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	56	5	13.82	203	2.26	62.22	874	218.50	220	0.68%	16	2
1409	58	6	19.29	212	2.36	65.56	907	226.75	228	0.55%	14	2
1685	56	6	26.52	227	2.52	64.44	895	223.75	228	1.86%	15	1
1737	58	7	16.31	236	2.62	65.56	949	237.25	240	1.15%	15	3
2626	62	8	21.4	228	2.53	68.89	949	237.25	240	1.15%	21	1
5129	67	11	27.9	222	2.47	74.44	903	225.75	227	0.55%	16	2
5228	67	14	41.67	234	2.6	74.44	999	249.75	253	1.28%	19	1
6681	61	9	42.25	236	2.62	67.78	942	235.50	237	0.63%	11	2
7244	59	11	22.88	219	2.43	65.56	886	221.50	223	0.67%	25	2
7612	60	9	17.97	218	2.42	67.78	898	224.50	227	1.10%	21	1
Average	60.4	8.6	25.001	223.5	2.483	67.67	920.2	230.05	232.3	0.96%	17.3	1.7
Min	56	5	13.82	203	2.26	62.22	874	218.5	220	0.55%	11	1
Max	67	14	42.25	236	2.62	74.44	999	249.75	253	1.86%	25	3

Table A-160: 4 Teams, 100 Parts (GA – Makespan/Max in Stack/JIT)

Seed	No. In Stack	Max in Stack	Avg. Time in Stack	Total Waiting Time	Avg. Waiting Time	JIT %	Total Work	Optimum Makespan	Make span	% Diff	Reduced Double Handled	Teams = Make span
1256	73	9	43.23	227	2.27	73	959	239.75	244	1.74%	17	2
1409	63	7	17.68	229	2.29	66	969	242.25	245	1.12%	29	2
1685	66	7	30.62	257	2.57	66	1025	256.25	258	0.68%	27	1
1737	66	8	33.02	263	2.63	69	1061	265.25	267	0.66%	25	1
2626	70	9	35.03	250	2.5	71	1040	260.00	262	0.76%	25	1
5129	70	12	27.96	248	2.48	70	1006	251.50	253	0.59%	31	1
5228	70	8	27.9	260	2.6	70	1110	277.50	279	0.54%	27	1
6681	66	7	26.52	267	2.67	66	1067	266.75	268	0.47%	20	2
7244	74	17	30.04	247	2.47	74	996	249.00	251	0.80%	23	2
7612	67	5	20.72	246	2.46	67	1010	252.50	255	0.98%	19	1
Average	68.5	8.9	29.272	249.4	2.494	69.20	1024.3	256.075	258.2	0.83%	24.3	1.4
Min	63	5	17.68	227	2.27	66	959	239.75	244	0.47%	17	1
Max	74	17	43.23	267	2.67	74	1110	277.5	279	1.74%	31	2

A3. Screen Shots of Algorithmic Programs Output

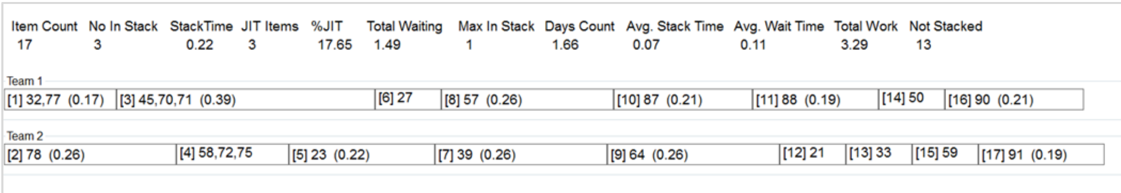


Figure A-1: Greedy Only (2 Teams)

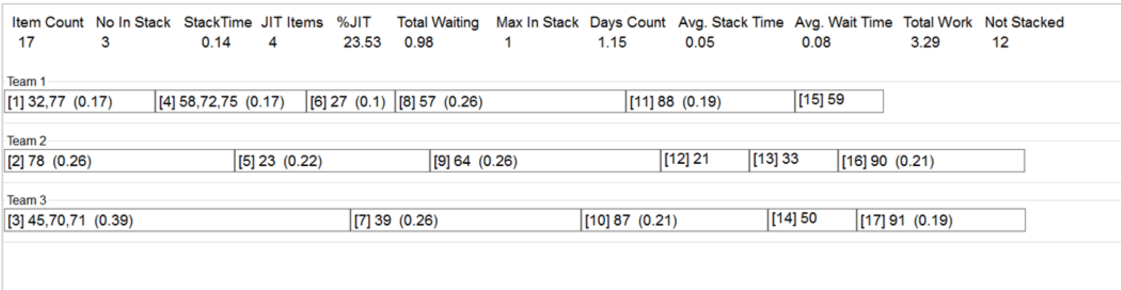


Figure A-2: Greedy Only (3 Teams)

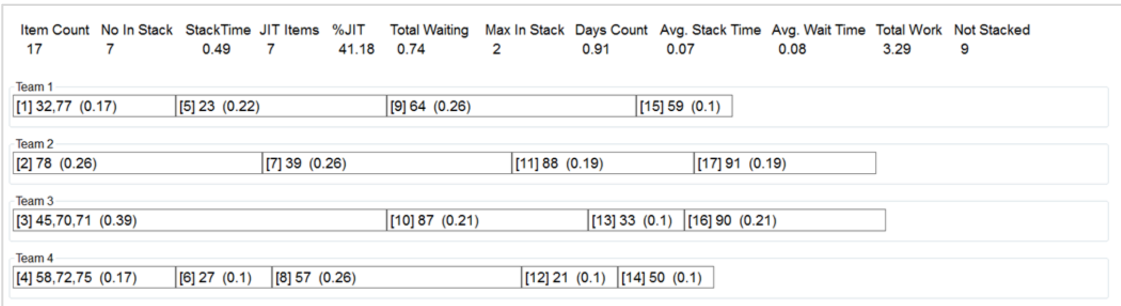


Figure A-3: Greedy Only (4 Teams)

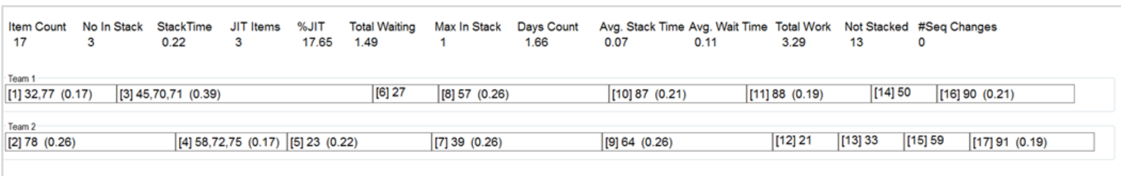


Figure A-4: Greedy with Post Ordering (2 Teams)

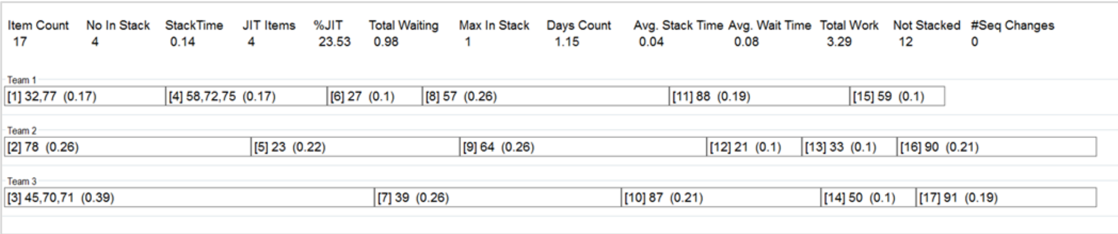


Figure A-5: Greedy with Post Ordering (3 Teams)

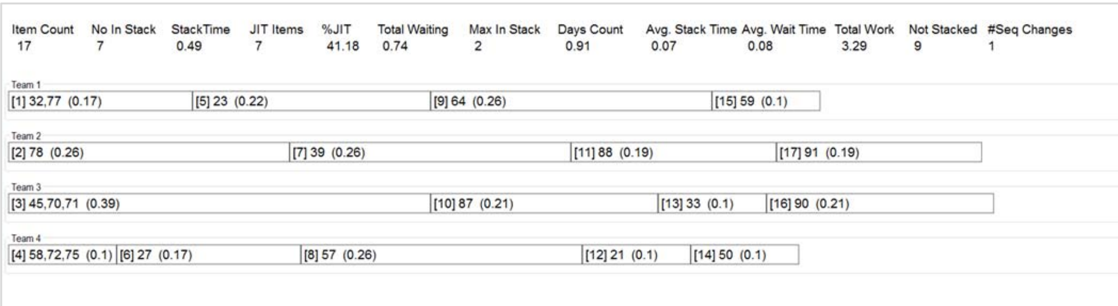


Figure A-6: Greedy with Post Ordering (4 Teams)

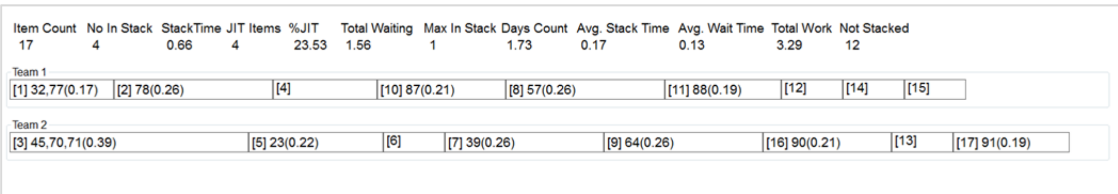


Figure A-7: Greedy with Backtracking (2 Teams)

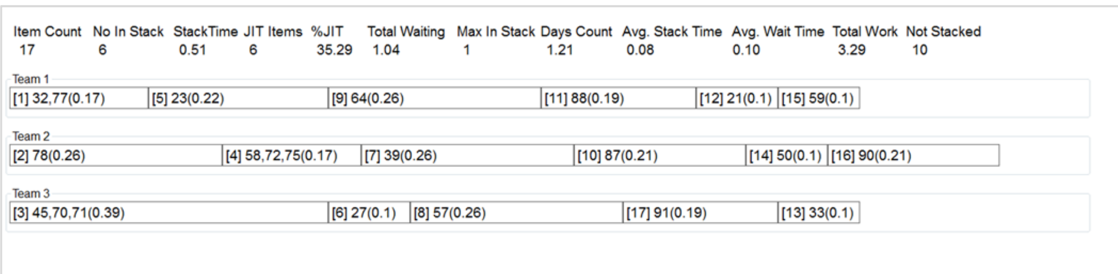


Figure A-8: Greedy with Backtracking (3 Teams)

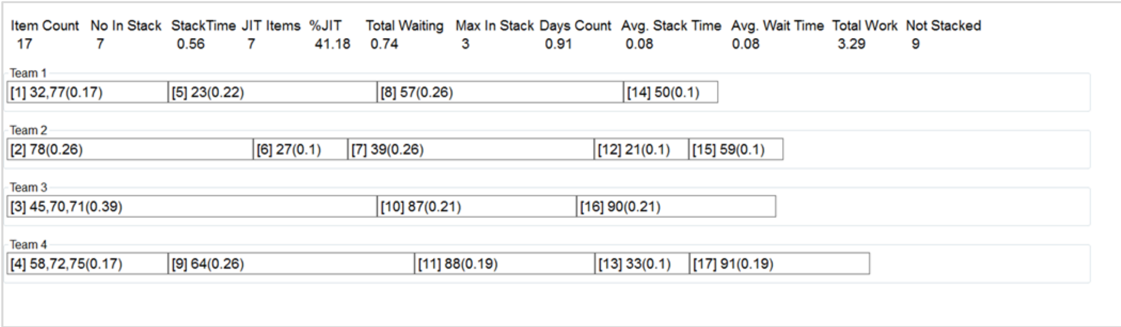


Figure A-9: Greedy with Backtracking (4 Teams)

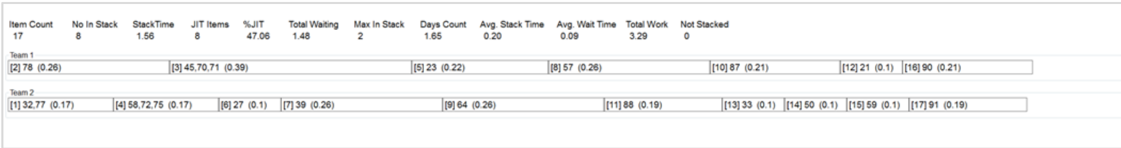


Figure A-10: Genetic Algorithm with Post Ordering (2 Teams)

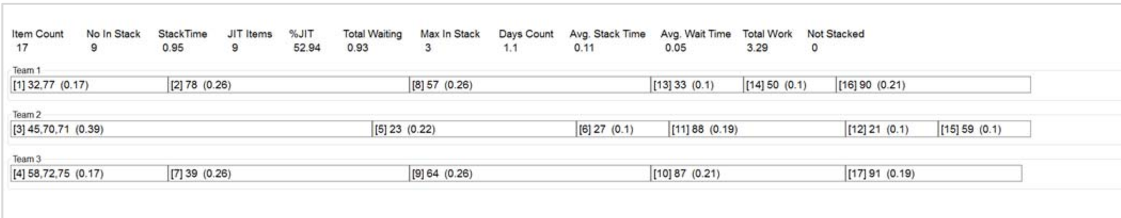


Figure A-11: Genetic Algorithm with Post Ordering (3 Teams)

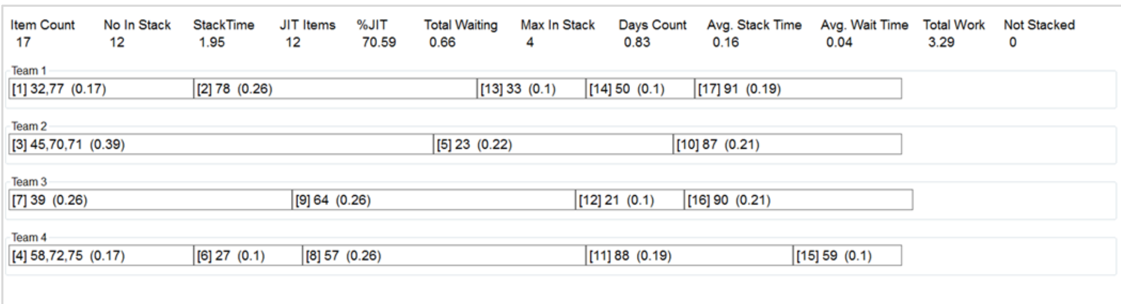


Figure A-12: Genetic Algorithm with Post Ordering (4 Teams)

A4 – Standard Welding Joints

Figure A-13: Fillet Joint Unequal Thickness < 19 mm

Figure A-14:Fillet Joint Equal Thickness > 19mm

Figure A-15: Butt Joint, Equal Thickness < 19mm

Figure A-16: Fillet Joint, Equal Thickness < 19mm

Figure A-17: Fillet Joint, Incomplete, >19mm

Figure A-18: Fillet Joint Unequal Thickness > 19mm

PUBLICATIONS

Liu, Z, K Yeoh, D K H Chua and E L S Abbott, "Optimization Of Resource Leveling In Shipbuilding With Variation-Float Tradeoffs". *6th International Conference on Innovation in Architecture, Engineering and Construction* 2010): 104-113. Loughborough: Loughborough University. (2010 AEC Innovation Conference, 9 - 11 Jun 2010, The Pennsylvania State University, Pennsylvania State University, United States)

Abbott, E L S, Z Liu, D K H Chua and CL Lim, "Extraction of Ship Product Design Data". *Proceedings of International Conference on Engineering, Project, and Production Management*, ed. David KH Chua and Thanwadee Chinda (2010): 31-40. Pingtung National Pingtung University of Science and Technology. (International Conference on Engineering, Project, and Production Management, 14 - 15 Oct 2010, National Pingtung University of Science and Technology, Pingtung, Taiwan)

Abbott, E L S, Z Liu, D K H Chua and F Jiang, "Integration of Extracted Data for Ship Block Construction". *Proceedings of International Conference on Engineering, Project, and Production Management* (2010): 71-80. Pingtung: National Pingtung University of Science and Technology. (International Conference on Engineering, Project, and Production Management, 14 - 15 Oct 2010, National Pingtung University of Science and Technology, Pingtung, Taiwan)

Liu, Z, E L S Abbott, D K H Chua and W.S Chang, "Integration of Digital Assembly Process Planning and Simulation-Aided Production Planning for Block Fabrication in Ship/Offshore Building". *Proceedings of 4th PAAMES and AMEC 2010* (2010). Singapore: Proceedings of 4th PAAMES and AMEC 2010. (Proceedings of 4th PAAMES and AMEC 2010, 6 - 8 Dec 2010, Singapore)

Abbott, E L S, Z Liu, D K H Chua and W. S. Chang, "Estimating Man-Hours To Construct a Ship's Block using Work Measurement". *Proceedings EPPM 2011*, ed. Hsiang- Hsi Huang, Kassim Gidado (2011). Singapore: National University of Singapore. (International Conference on Engineering, Project and Production Management 2011, 20 - 21 Sep 2011, National University of Singapore, Singapore)

Lim, C.L., E L S Abbott and D K H Chua, "Automatic Searching Of Lifting Eye Lugs Locations On A Steel Block". *Proceedings of 4th PAAMES and AMEC2010* (2011). Singapore: Proceedings of 4th PAAMES and AMEC2010. (Proceedings of 4th PAAMES and AMEC2010, 6 - 8 Dec 2010, NUS, Singapore)

Abbott, E L S and D K H Chua, "The Application of Polychromatic Set Theory in the Assembly Sequencing of Blocks for Offshore Rigs". *Proceedings of the 4th International Conference on Engineering, Project and Production Management* (2013): 59-74. Bangkok: Thammasat University. (4th International Conference on Engineering, Project, and Production Management (EPPM 2013), 23 - 25 Oct 2013, Sukosol Hotel, Bangkok, Thailand)

Nguyen, T Q, E L S Abbott, D K H Chua and Y M Goh, "Formalizing construction safety knowledge for intelligent bim-based review of design for safety". *Proceedings of CIB W099 International Conference on Achieving Sustainable Construction Health and Safety* (2014): 597-607. Lund: Ingvar Kamprad Design Centre (IKDC). (CIB W099 International Conference on Achieving Sustainable Construction Health and Safety, 2 - 3 Jun 2014, Ingvar Kamprad Design Centre (IKDC), Lund, Sweden)

Abbott, E L S and D K H Chua, "A JIT Algorithm for Offshore Rig Assembly Sequencing". *Conference Proceedings*, ed. J. Tamosaitiene, K. Panujwatwanich, N. Mishima, C.H. Ko, comp. John Smalllwood (2014): 172-181. Port Elizabeth: EPPM Association. (International Conference on Engineering, Project, and Production Management, 26 - 28 Nov 2014, Protea Marine Hotel, Port Elizabeth, South Africa)

Cheng, T L D, E L S Abbott, D K H Chua and T B S Ahmad, "Automated Master Project Schedule for Construction Incorporating Building Information Model and IFC". *Conference Proceedings*, ed. J. Tamosaitiene, K. Panujwatwanich, N. Mishima, C.H. Ko, comp. John Smalllwood (2014): 182. Port Elizabeth: EPPM Association. (International Conference on Engineering, Project, and Production Management, 26 - 28 Nov 2014, Protea Marine Hotel, Port Elizabeth, South Africa)