Optimization of Erection Network for Pre-Erection Load Management

선행 탑재장 부하를 고려한 탑재 일정계획의 최적화

February 2006

Graduate School of Chosun University

Department of Naval Architecture and
Ocean Engineering

Baek Dong Sik

Optimization of Erection Network for Pre-Erection Load Management

Advisor: Professor Yoon, Duck Young

Thesis submitted for the degree of Master of Engineering

February 2006

Graduate School of Chosun University

Department of Naval Architecture and
Ocean Engineering

Baek Dong Sik

Master Thesis approved by

Prof Yoon, Duck Young
Chosun University

Prof Kwon, Young Sub
Chosun University

Prof Park, Jae Woong
Chosun University

November 2005

Chosun University Graduate School

- Contents -

List of Figures

List of Tables

초 록

백 동 식
지도교수 : 윤덕영
조선대학교 일반대학원
선박해양공학과

선행 탑재장 부하를 고려한 탑재 일정계획의 최적화

조선 산업에서 경쟁력을 유지하기 위하여 첨단 정보기술의 적용을 통하여 새로운 시장 환경에 적응할 수 있는 최적화된 일정계획의 수립이 요구된다. 조선 산업에서 일정계획은 계획 기간이 길고 일정계획 수립의 대상이 되는 작업들의 종류가 다양한 관계로 매우 어려운 문제로 인식되고 있다. 조선 산업에서 수행되는 일정계획의 수립과정을 살펴보면 우선 가장 중요한 자원인 도크에서의 일정계획 즉 선각 조립과 관련된 '탑재일정계획'을 수립하고 이 결과를 바탕으로 선행탑재일정과 도크 및 안벽에서 의장일정을 결정한 뒤 선행도장 및 의장일정 가공 및 조립공장에 대한 일정계획등이 차례로 수립된다. (김기동 등,2001). 본 연구에서는 선행탑재장 에서의 일정문제를 Looking-Forward Algorithms (미리보기 알고리즘)의 일정계획과 여유일 기반의 탑재일정 결정을 통한 선행탑재장의 가용능력을 최대한 이용하면서 일정계획상에서 부하평준화가 이루어질 수 있는 시스템을 제시한다.본 연구에서는 선행탑재장의 일정계획에서 부하평준화를 고려한 Looking-forward Algorithms 를 사용하였으며 탑재일자 결정에 있어서 여유일기반의 탑재일자결정방법론을 사용하였다.

ABSTRACT

Optimization of Erection Network for Pre-Erection Load Management

Name : Dong Sik Baek
Thesis Advisor : Prof. Duck Young Yoon
Chosun University Graduate School
Department of Naval Architecture
And Ocean Engineering

In the ship building industry various problems of erection are merged due to formation of bottlenecks in the block erection flow pattern. This kind of problems gets accumulated problems in real-time erection at the pre-erection area.

When such a problem is encountered, a support data of the entire erection sequence should be available. Here planning is done by reasoning about the future events in order to verifies the existence of a reasonable series of actions to accomplish a goal. This technique helps in achieving benefits like handling search complications, in resolving goal conflicts and anticipation of bottleneck formation well in advance so as to take necessary countermeasures and boosts the decision support system. The data is being evaluated and an anticipatory function is developed. This function is quite relevant in day-to-day planning

operation. The system updates database with rearrangement of off-critical blocks in the erection sequence diagram. As a result of such a system, planners can foresee months ahead and can effectively make decisions regarding the control of loads on the man, machine and work flow pattern, culminating to an efficient load management. Such a foreseeing concept helps us in eliminating backtracking related adjustment, which is less efficient compared to the looking-Forward concept.

An attempt is made to develop a computer program to update the database of block arrangement pattern based on heuristic formulation.

Chapter 1 Introduction

1.1 Background of Research

Shipbuilding is an industry that produces products such as ships, offshore structures, floating plants, etc for customers (e.g., private owners, companies, governments, etc.) In most cases, the products are built to order and customized to the specific requirements of the purchaser. This applies even in cases where a similar series of ship is being built (Storch *et al.* 1995). The manufacturing process is likely to vary somewhat, but it generally involves three basic types of work: hull construction, outfitting, and painting. The hull construction forms the structural body of ship through the series of production stages such as parts fabrication, block assembly, and hull erection, while the outfitting handles all the parts of a ship that are not structural in nature (e.g., pipes, derricks, masts, rigging, engines, machinery, electrical cable, hotel service, etc.) Next, the painting applies paint or other coating materials for protective and decorative purposes by means of spray gun, brush, roller or immersion. Production schedule form the framework that assists the flow of information between the various shipyard functions. This information flow is necessary to ensure completion of a ship in an efficient and timely manner. (Storch *et al.* 1995) But, The shipbuilding industry has many constraints points of constraints in shipbuilding. We consider of space, time, crane, and equipment movement. This paper focuses on the Pre-Erection Area (P.E Area) constraints. In shipbuilding industry various problems of erection is counterfeited due to formation of bottlenecks in the block erection flow pattern. This kind of problems cause accumulated problems in real-time erection right on the floor.

Shipbuilding process last delivery is Hull Erection Hull step at erection level of hull construction All process in shipyard planning deemed basic. Because, P.E Area planning necessary optimization scheduling.

1-2 Previous Research

The previous research characterized to use heuristic or Development of Scheduling Expert System for shipbuilding Industry show the assembly scheduling on the whole require a multi-level hierarchical scheduling under the given resource constraints. This paper developed a hierarchical architecture for the shipbuilding scheduling, along with a constraint directed graph search technique for erection scheduling at the dock (Hyun-Rim, Choi. *et al.*1993). The erection process planning and scheduling using genetic algorithm paper suggest to method that generates the erection sequence. The load leveling should be done to all the ships in the same dock batch to get reliable results. This research achieved the load leveling system using Genetic Algorithm This system can made it possible to simulate various process plans to which scheduling is considered. (Jae-Won, Lee. *et al* 1993), and generation of Erection sequence in Shipbuilding process planning. The methodology suggested also includes some useful benefits of avoiding insert blocks or maintaining stability in erection process. (Yoon-Gee, Hong. *et al* 1997) A study on the application of resource leveling heuristic for ship erection scheduling. This paper shows that resource-leveling heuristic is developed for ship erection scheduling. The heuristic, which enable scheduling with limited resource, are composed of the gradual resource limit decrease and the resource allocation. At first, earliest start (ES) schedule is derived by PERT to get initial resource limit then, the

heuristics lower the resource limits by a given increment and derive a new schedule from day-by-day resource allocation procedure. The procedure is repeated until the project duration reaches at the given ship erection duration, called dock cycle. Through the application of the heuristic for real world ship erection projects, we can observe significant improvement leveling. (Tae-Hyun, Baek. *et al* 1999) A genetic Algorithm application for the load balancing of ship erection process, develop a genetic algorithm for erection scheduling in shipbuilding this paper propose a function as a minimization of load index which represented the load deviation over time horizon considering the yard production strategy. For the optimal parameter setting, we tried various experiments. We verified that the proposed approach was effective to deal with the erection-scheduling problem in shipbuilding. (Sang-Gyu, Min. *et al* 2000) Establishing Methodology for Simulation-based ship design and construction using Virtual Manufacturing technologies the concept and current status of digital manufacturing in general manufacturing industry will be reviewed. Then, related technologies, area of application and methods of digital manufacturing in shipbuilding and marine industries are presented. In addition, virtual assembly simulation system for shipbuilding, a tool for crane operability and block erection simulation in virtual dock based on 3D product model. A study on the Erection scheduling for shipbuilding considering resource constraints this paper, we model the erection scheduling problem for shipbuilding and develop a new problem solving method using CST (Constraints Satisfaction Technique) and ILOG scheduler. (Ki-Dong, Kim. *et al* 2001) Generation of ship hull blocks erection network in consideration of resource constraints in building dock. (Yonug-Tae, Kim. *et al* 2001) Establishing Methodology for simulation-based ship design and construction using virtual manufacturing technologies describe virtual assembly simulation

system for shipbuilding, a tool for crane operability and block erection simulation in virtual dock based on 3D product model (Hong-Tae, Kim. *et al* 2002) Load Balancing of block erection network using constraint satisfaction problem proposed in order to solve how to choose block erection data with load balancing. Characteristics of erection process had the non-fixed elements, and it requires a lot of calculation. So CSP, which demands less calculation and reflects more of the non-fixed elements, is the best profit solver in erection schedule problem. It was applicable in diminution method of maximum load through load balancing measure. This method seeks for a gradual lowering of the maximum load, and consequently aims for load balancing. It was effective for not only decrease of standard deviation, but also overtime work. (Ji-Sung, Ryu *et al* 2003)

1.3 Research objective and Contributions

The block Pre-erection process is placed between block paint process and Erection process, which are on critical path of shipbuilding process. The performance of Pre-Erection process influences the whole shipbuilding process such that its low performance directly results in work delays, accumulation of work-in-processes late delivery of a ship, etc The operation schedule of block Erection scheduling requires the closer control because its operation highly depends on environmental conditions (e.g., humidity and weather) and the limited workspace. The scheduling of Pre-erection process is very complicated task because its needs to consider the traditional scheduling problem as well as the spatial allocation of blocks within the limited workspace. We are focus on Pre-erection stage. In shipbuilding industry various problems of erection is counterfeited due to formation of bottlenecks in block erection right at the P.E Area. Such a problems is

approached by a support data of the future events in order to verify the existence of a reasonable series of actions to accomplish a goal. This technique helps in achieving benefits like handling search complications, in resolving goal conflicts and anticipation of bottleneck formation well in advance to take necessary countermeasures and boosts the decision support system. The data is being evaluated and an anticipatory function is to be developed. This function is quite relevant in day-to-day planning operation. The system updates database with rearrangement of off-critical blocks in the erection sequence diagram. As a result of such a system, planners can foresee months ahead and can effectively make decisions regarding the control of loads on the man, machine and work flow pattern, culminating to an efficient load management. Such a foreseeing concept helps us in eliminating backtracking related adjustment, which is less efficient compare to the look-ahead concept. An attempt is made to develop a computer program to up date of block arrangement pattern based on heuristic formulation.

The objectives of our research can be summarized as follows.

- "Looking-Forward" Algorithm for optimization of erection sequence diagram generation for shipbuilding industry.

- The block optimization scheduling in Pre-Erection Area

- Determinate of critical path of block sequence array leading to the description of strategic blocks.

- Evaluation of P.E Area ENT (starting day of erection) and LNT (Ending day of erection)

- Derive and develop the entire erection sequence diagram for shipbuilding.

1.4 Paper Organization

The organization of the paper is as follows.

Chapter 2 introduces Shipbuilding process and the status of shipbuilding planning. Relation process of pre-erection and block erection network method.

Chapter 3 describes Modern Heuristic scheduling algorithms, which is able to adjust shipbuilding scheduling. Also, Looking-Forward algorithm scheduling and probability based in Erection time method adjust Pre-Erection scheduling.

Chapter 4 explains the Float time based erection day and looking forward algorithm calculation program

Chapter 5 shows the simulation results and compared graph

Chapter 6 is the research summary and suggestions for future research issues.

Chapter 2 Pre-Erection Area Process Scheduling

2.1 Overview of Shipbuilding Process

2.1.1 Shipbuilding today

The shipbuilding process starting from order to delivery can be divided into design stage and manufacturing stage. Design stage can be further divided into contract design performed for the negotiation with ship owner, basic design to meet the requirements of ship owner, and detailed design performed in functional aspect. On the other hand, manufacturing process has been identified with those operation and process related with the transformation of sheet metal to steel hulls. These operation essentially consisted in marking and cutting the sheet metal prime material in elementary building pieces the assembly of which, in various stages (panel, sub-block, blocks, sections), finally yielded the complete steel hull. Thus, three basic classes of processes could be distinguished:

- Sheet (or beam) metal treatment processes (conditioning, marking, cutting, bending etc)
- Assembling (nowadays mostly welding)
- Handling (moving and positioning various blocks and sub-assemblies).
- In the context maximizing the efficiency of shipbuilding essentially meant`s
- Evaluation of P.E Area ENT (starting day of erection) and LNT (Ending day of erection)
- Derive and develop the entire erection sequence diagram for shipbuilding.
- Assuring the most cost-effective acquisition of the necessary prime materials and

components,

- Optimizing in terms of cost and quality each one of the process stated above and
- Planning the whole sequence of operation in such a way so as to ensure a seamless flow of material and an optimal utilization of the available resources (labors, equipment, space and time)

Today, the relative importance (at least in terms of cost) of the steel shaping and assembling has decreased substantially. In fact, due to the ever-increasing requirements in terms of passenger comfort, safety in navigation and environmental concerns, modern passenger and ferry vessels look more and more like huge navigating hotels or bug industrial complexes. Shipbuilding nowadays encompasses a range of processes and activities much broader than shaping and assembling sheet metal steel. Less 'traditional' shipbuilding activities, such as finishing and outfitting become increasingly important. Another very important change happening these days is on the role that CAD tools play in shipbuilding: the role of CAD/CAE tools in shipbuilding has traditionally been that of enhancing the planning and the start of the production activities. Nowadays, in view of the increasing complexity of the production operation and, consequently, the planning requirements and the stringent lead times to market, CAD/CAE tools are rapidly acquiring the role of the integrator between the 'traditional' design phases the planning and production processes. For all these reasons, design, be it basic or detailed, cannot be considered separately not only from production planning but also from most of the individual production processes. Nevertheless, for practical reasons, we distinguish the production from the design & planning activities in terms of

the different tools and technologies applied. While production deals with the handling and transformation material objects, design and planning activities have mainly to do with the generation and handling of information. As such, for the purpose of the shipbuilding understand, they are considered as fundamental components of the integration process and the corresponding information flows rather than parts of production flow.

Therefore, for the purpose of understands shipbuilding, we consider two types / class of process: The production process, dealing with material objects and the information processes, dealing with information.

2.1.2 Production processes

By the term, "production activities" or " production processes" we mean all the activities that essentially consist in transforming (machine, assembling, handling, transporting etc) material objects. In what follows, the "generic shipbuilding" is broken down to classes of processes hereafter referred as " production process" or "process". Although the meaning and the sequence of these processes can vary according to the type (tanker, container carrier, passenger ship) and size of the vessel as well as the particular methods and mentalities of each individual yard, it can be said that they represent quite well all but the very specialized constructions. So the generic shipbuilding production can essentially be broken down in the following production processes: as seen Figure 2.1

Processes 1: Raw material reception & preparation

Processes 2: Marking, cutting & conditioning of steel plates and profiles

Processes 3: Fabrication of 2D blocks: welding of flat and shaped Sub-assemblies (Panels and sub-blocks)

Processes 4: Fabrication of 3D blocks in workshop

Processes 5: Pre-erection: assembly of 3D blocks and subassemblies into Erection units

Processes 6: Prefabrication of pipes, supports, modules etc

Processes 7: Pre-outfitting

Processes 8: Blasting and Painting/Coating

Processes 9: Erection and Outfitting in the dry-dock or slip-way

Processes 10: Outfitting

Processes 11: Finishing & outfitting on board of the floating vessel

Processes 12: Commissioning & Sea trials

A first schematic representation of the genetic production process flow in shipbuilding is given in Figure 2.1 below

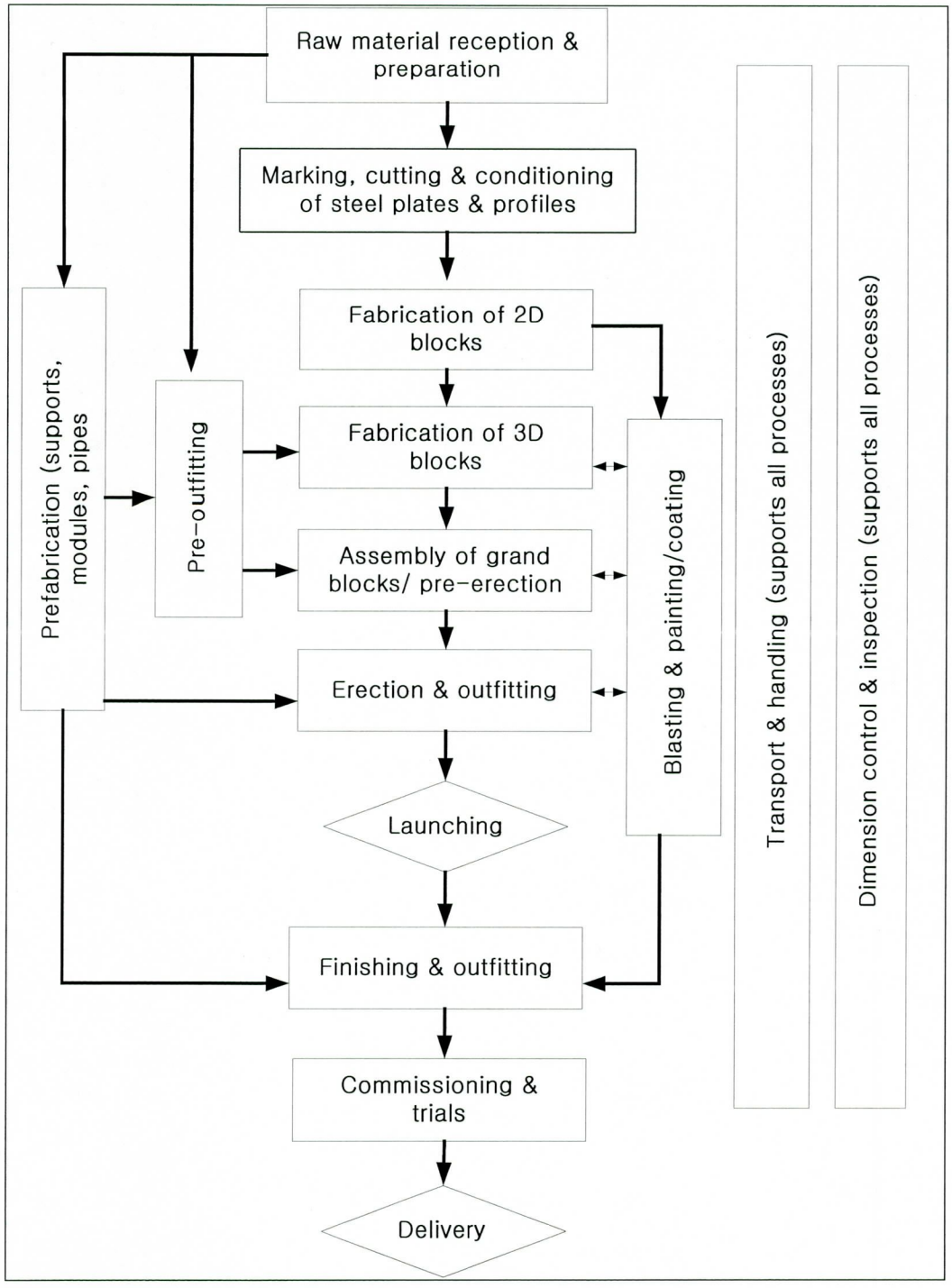


Figure 2.1 Shipbuilding Production Process Flow

2.1.3 Characteristics of shipbuilding processes and Modern shipbuilding method

Shipbuilding processes are different from other typical manufacturing processes in the following characteristics. Ship type and form is very diverse and it is difficult to standardize since the design process is done according to the user's request. Material procurement and manufacturing begin while the design stage is not complete, so engineering changes and materials replacement is expected in manufacturing stage. Shipbuilding is labor-intensive industry that is very difficult to mechanize and automate, so a lot of qualitative information is processed. While materials are big and heavy, required accuracy is high and structure is complex, so it is difficult to standardize manufacturing process. Ships with different specification are built at the same time, and a lot of information is required for management of each ship. Because of above reasons, building prototype for verification of product validity and quality assurance is practically impossible in the sense of manufacturing cost and time. Therefore, in order to increase efficiency in shipbuilding, extraction of detailed design and manufacturing information is required, and such information needs to be exchanged and integrated for intelligent scheduling process system. In modern shipbuilding system diagram shown Figure 2.2 characteristics

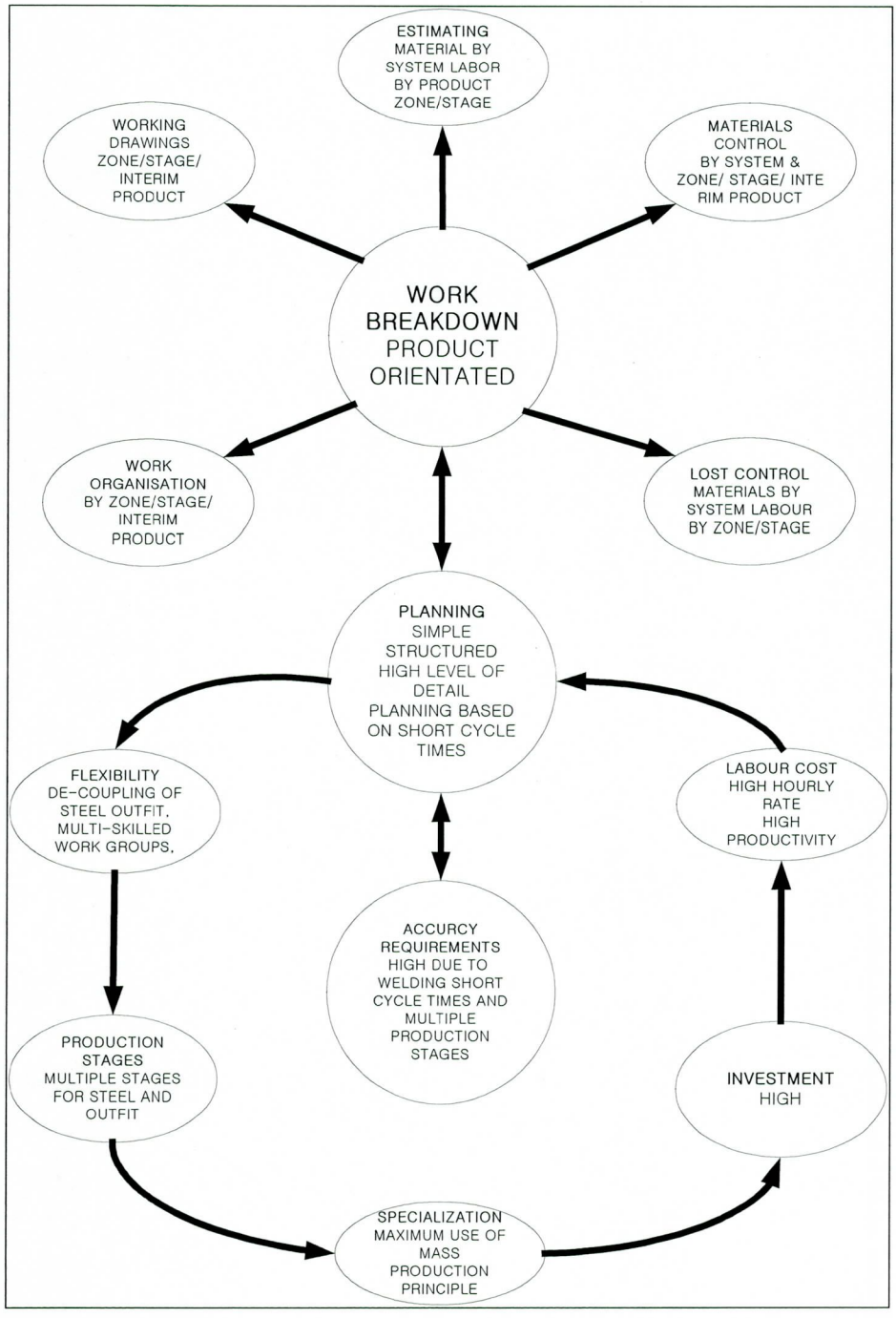


Figure 2.2 Characteristics of the Modern Shipbuilding method

2.2 The status of shipbuilding planning.

Planning and scheduling of all these activities within a specified time horizon subject to limited resource availability and limited space is a very difficult task. Such a large and complex production project could be chaotic if the activity planning is poor. Even the planned work will not usually progress as intended due to several unexpected events and delays and also probably due to difficulty in accurately estimating the activity times. This further complicates the overall work organization. The task of planning and scheduling in shipbuilding can be divided into three categories.

I Development of work breakdown structure, planning of major work components, and acquisition of raw materials during a specified time horizon.

II Planning and scheduling of major events of shipbuilding such as assembly completions, block formations. block joining, erection activities, and so forth, subject to constraints.

III Planning and scheduling of a collection of various tasks (which may belong to different ships or different work packages of a single ship) over short intervals at each work place.

The first and second categories mainly require a project management approach at the planning level, whereas the third category requires a resource-constrained job-shop scheduling approach

at the level of individual and somewhat independent work places. The latter approach, along with some project management principles, may also be useful for the second category. Planning and scheduling is a management problem. It is often carried out without considering the resource conflicts explicitly. The "worker" resource is relatively more flexible in the shipbuilding industry. The conflicts for this resource are usually sorted out by providing overtime or more workers. However, the conflicts for such resources as machines cannot be so easily sorted out. Such resource cause bottlenecks in production, and therefore the planning and scheduling must ensure their optimal utilization while meeting the due dates for various tasks. The planning and scheduling exercise must take into account all aspects, such as priorities, precedence relations and due dates of tasks, material availability, resource availability, level of work in process, and so forth.

2.2.1 Shop and Pre-erection shop for scheduling present

The planning and scheduling department focuses its attention primarily on the first two categories. The production control system of a shipyard is usually based on product work breakdown structures. However, the scheduling at lower levels that is, at the shop floor level is also important. Shop floor scheduling is basically an operation level scheduling. If this lower-level scheduling is not done efficiently, the resulting delays in task completions will seriously impact scheduling is addressing, apart from the workload, several other aspects, such as available times of input material, resource calendars, process flow (task precedence relationships), the priorities of end products, and so forth. Shop production supervisors are responsible for addressing these issues

at the shop floor. The shop floor scheduling of operation in various machines using input materials, workers, and tools (during the available times) subject to technical and precedence constraints is a highly complex decision-making problem. The complexity increases with heterogeneity among operations. Even simple cases of such scheduling problems are known to be computationally very hard in the scheduling literature (Pinedo 1995, Chao & Pinedo 1999). It is therefore very difficult for a shop production supervisor to regularly perform this complex exercise using only his/her knowledge and experience. This problem gets even more complex when schedules are to be revised frequently in response to unexpected events on the shop floor. Production personnel must revise an existing schedule in reaction to unexpected events on the shop floor, such as machine breakdown, worker absence, and delays in referred to as " reactive scheduling." (Szelke & Kerr 1993).

In addition to obtaining an optimal or suboptimal schedule of a given workload, the supervisor may also be interested in analyzing the impact of changing the priorities and due dates of tasks, material available times, resource availability, and so forth (as required sometimes by the management). Thus, shop floor scheduling is an important and complex decision-making problem regularly faced by the shop floor supervisors. A material requirement planning (MRP) does not address the resource-constrained scheduling problems due to the assumption of infinite capacity. Even MRP Ⅱ,which evolved from MRP through the inclusion of financial and engineering elements and capacity planning, does not explicitly deal with resource conflicts in shop floor operation. The theory of constraints (Goldratt & Cox 1986) mainly with scheduling of bottleneck resources assuming that no critical resources have infinite capacity. Therefore, it will not present a complete and explicit schedule of all workload in a shop.

In the shipbuilding industry, project management tools such as Microsoft project (Microsoft Corporation) and Primavera (Primavera Systems, Inc) are used for planning and scheduling. These tools are quite efficient for project management involving numerous related activities. However, they are not very appropriate for the complex scheduling of production activities at the shop floor level when the scheduling is subjected to technical constraints in terms of space, time, environment, and material movement in addition to resource constraints. The information on resource utilization and schedules, workload schedules,work order delivery times, and so forth is not presented in a customized manner. For these reasons, their implementation for shop floor scheduling is not very effective. A computer application based on a rigorous scheduling model and a sound solution technique will be an effective approach to solving the problem of scheduling shop floor workload subject to resource and technical constraints.

2.3 Pre-Erection Process and Activity

2.3.1 Define of Pre-Erection Problem

Shipbuilding has many block storage with multilevel probability of bottleneck formation during regular works flow pattern. The major problem arises at shipyard because of laxity in time for scheduling the building blocks for ships under construction. The real-time shipbuilding process plan follows Pre-Erection scheduling, and last flow material (Grand block, module, material etc.) combined in P.E Area. The P.E Area process must have exact schedule but its schedule have many constraints and minimal adjustment during Erection. The Pre-Erection Area is well laid out with

pre-determined location for blocks adjacent to the required erection point on the vessel. The transport equipment such as self-loading transporters is used to move assembly block from the assembly areas to the Pre-erection area. Pre-Erection Area Blocks are stored on adjustable stands or trestle for transporter access. Where large grand blocks or complete ship modules are erected, specialized rail mounted transfer systems are used with heavy lift platforms for lowering into a building dock for erection. There are activities occur in P.E Area. The entering & going block must follow erection network. All block have defined erection day and trace its processor and successor. More commonly addressed problems are the identification of erection sequence, the manpower works flow management, load balancing over man and machine, the spatial scheduling look ahead policy by the system to anticipate the flow blockade, productivity analysis, resource utilization aspects, and the like. The Pre-Erection Area scheduling is balanced among the other staged of block erection process. This problem leads to a minimization make span problem.

Up to these days, Hull Block Construction Method (HBCM) is wildly used in the shipyards. HBCM is method that divides hull into many blocks and builds them up in docks (Storch and Bunch 1995) Usually hull block is composed of several sub-assembly or part-assemblies that also consist of several pieces or units of pieces. Block division includes two stages. In the first stage, size of super blocks (pre-erection block) determined based on the key plan (general arrangement, mid-ship section, lines, construction profile and machinery arrangement). In the second stage, each super block is divided into assembly block considering assembly capacity. In HBCM, it is practical to plan hull construction in seven levels as shown in Figure 2.3

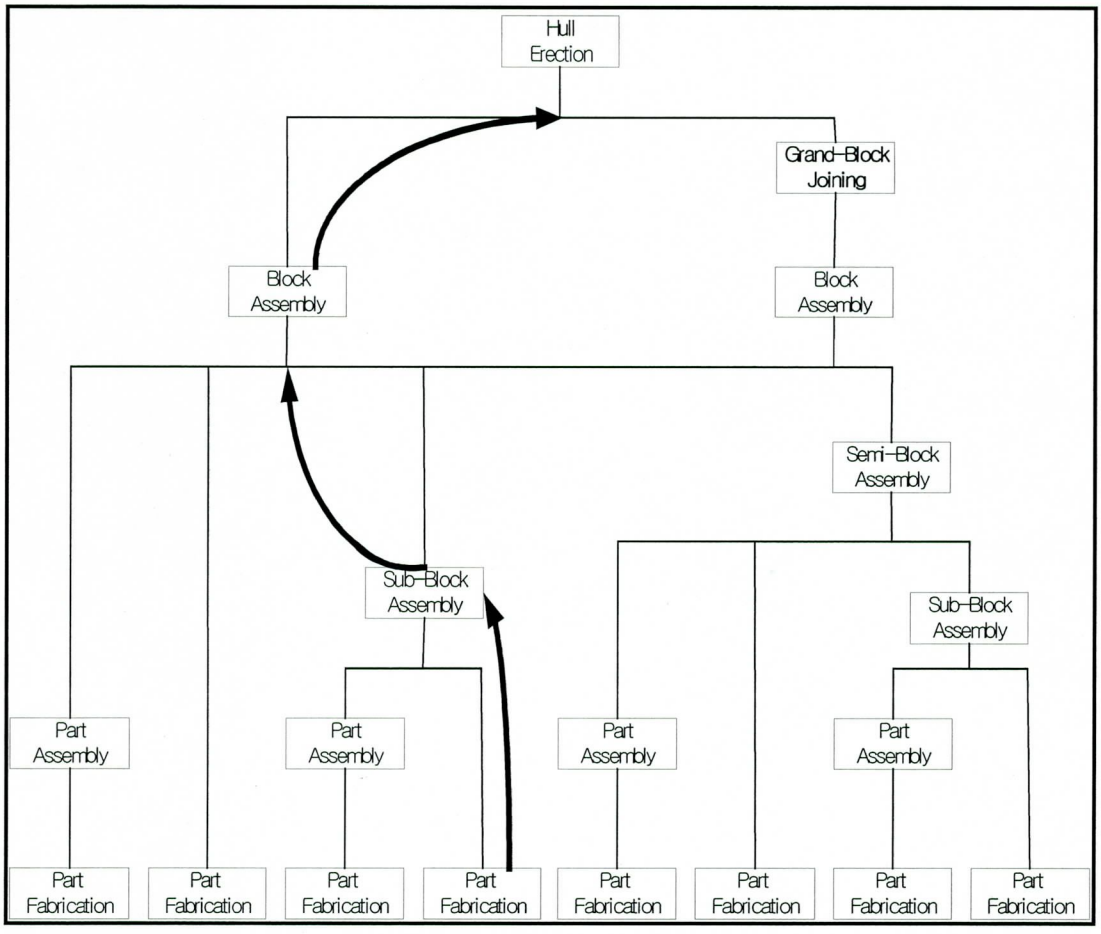


Figure 2.3 Hull Block Construction Method

2.3.2 Block Erection process

Block Erection process begins from on-block outfitting of blocks produces from assembly process. Then the blocks are painted, pre-erected, and finally erected on dock. In the viewpoint of manufacturing planning, first, building schedule (dock usage plan) is made based on long-term load balancing and master schedule. Then beginning date for each line and doc, launching date, on-board outfitting time is decided. Based on these timeline, block erection schedule is established. Generally, other manufacturing

process schedule is established based on this schedule. In other words, analysis and forecast of other process can be rapidly done using computer integration system based on erection process rather than considering on specific process. However, in execution and management, more detailed production tasks are required in short period of time. For example, expanding pre-erection and on-block outfitting cannot be done in conventional sequential process but by concurrent engineering process based on reduce lead time. Therefore, increased production activity in unit time increase the awareness of the sequence of production process, and a series of activities to decide the sequence can be called process planning or daily planning. If the sequential is not appropriate or when delay occurs, their effect on other process are far greater than in conventional production process.

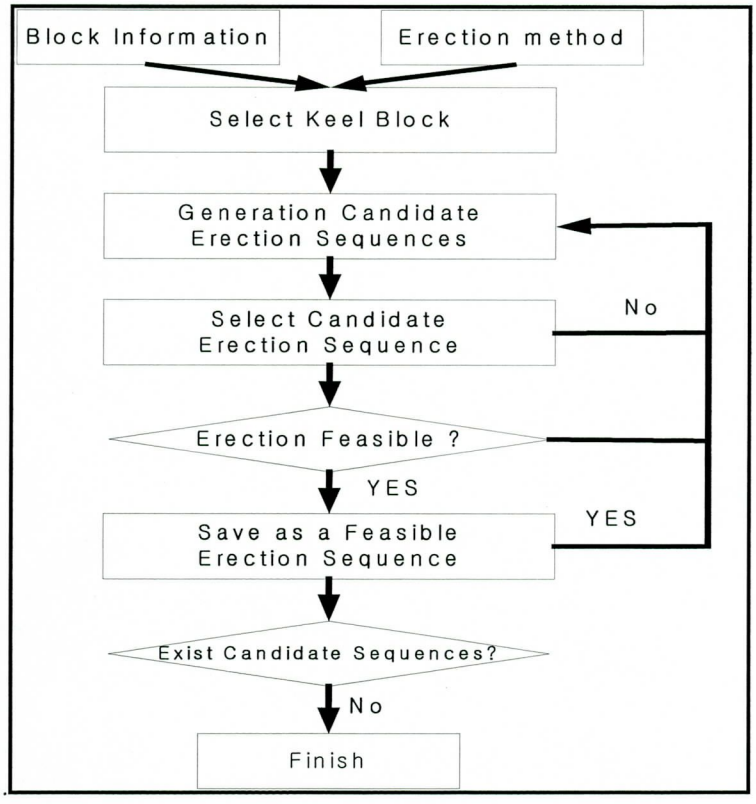


Figure 2.4 Generation of Erection sequence

Figure 2.4 shows the generation process of block erection sequence

2.3.3 Block Erection scheduling characteristics

Process planning in block erection can be summarized keel block and floating block, grouping of pre-erection blocks, and decisions on block erection sequence. Among these, the ones that are closed related to CAD systems are selection of floating block and grouping pre-erection blocks, and keel block selection and floating block selection are to be decided in production. In case of floating block selection and grouping of pre-erection blocks, selecting erection methods for efficient use of resources and based on launching method to minimize process time is decided during phase, but detailed erection plan for the selected erection methods should be decided in design phase. This paper focus on the evaluation of block erection sequence that is close related with production planning function. Since block erection process is used as a based for other planning, it should be the first to be established in intermediate schedule of ship manufacturing planning.

The important characteristics of erection scheduling can be summarized as follow

I Sequential Erection at Each Dock : Block and super-blocks in each dock have be erected one at a time because there is only one Goliath crane at each dock

II Large search block : Since the schedulers have to consider multiple ships, each composed of 400~500 blocks, a manual search for the best schedule is beyond mental processing capacity.

Ⅲ Utilization of Resource : Key resource in shipbuilding comprise human workforces, cranes and space. To utilize these resources effectively, a schedule should be established to balance the loads among block assembly shop, Pre-Erection Area and docks

2.4 Block Erection Network method

2.4.1. PERT/CPM methods

Program Evaluation and Review Technique (PERT) charts depict task, duration, and dependency information. Each chart starts with an initiation node from which the first task, or tasks, originates. If multiple tasks begin at the same time, they are all started from the node or branch, or fork out from the starting point. Each task is represented by a line that states its name or other identifier, its duration, the number of people assigned to it, and in some cases the initials of the personnel assigned. The other end of the task line is terminated by another node, which identifies the start of another task, or the beginning of any slack time, that is, waiting time between tasks. Each task is connected to its successor tasks in this manner forming a network of nodes and connecting lines. The chart is complete when all final tasks come together at the completion node. When slack time exists between the end of one task and the start of another, the usual method is to draw a broken or dotted line between the end of the first task and the start of the next dependent task. A PERT chart may have multiple parallel or interconnecting networks of tasks. If the scheduled project has milestones, checkpoints, or review points (all of which are highly recommended in any project schedule), the PERT chart will note that all tasks up to that point terminate at the review node. It should be noted at this point that the project review, approvals, user reviews, and so forth all take

time. This time should never be underestimated when drawing up the project plan. It is not unusual for a review to take 1 or 2 weeks. Obtaining management and user approvals may take even longer. When drawing up the plan, be sure to include tasks for documentation writing, documentation editing, project report writing and editing, and report reproduction. These tasks are usually time-consuming, so don't underestimate how long it will take to complete them. PERT charts are usually drawn on ruled paper with the horizontal axis indicating time period divisions in days, weeks, months, and so on. Although it is possible to draw a PERT chart for an entire project, the usual practice is to break the plans into smaller, more meaningful parts. This is very helpful if the chart has to be redrawn for any reason, such as skipped or incorrectly estimated tasks. Many PERT charts terminate at the major review points, such as at the end of the analysis. Many organizations include funding reviews in the projects life cycle. Where this is the case, each chart terminates in the funding review node. Funding reviews can affect a project in that they may either increase funding, in which case more people have to make available, or they may decrease funding, in which case fewer people may be available. Obviously more or less people will affect the length of time it takes to complete the project.

Critical Path Method (CPM) charts are similar to PERT charts and are sometimes known as PERT/CPM. In a CPM chart, the critical path is indicated. A critical path consists that set of dependence tasks (each dependent on the preceding one) which together take the longest time to complete. Although it is not normally done, a CPM chart can define multiple, equally critical paths. Tasks which fall on the critical path should be noted in some way, so that they may be given special attention. One way is to draw critical path tasks with a double line instead of a single line. Tasks

which fall on the critical path should receive special attention by both the project manager and the personnel assigned to them. The critical path for any given method may shift as the project progresses; this can happen when tasks are completed either behind or ahead of schedule, causing other tasks which may still be on schedule to fall on the new critical path.

- Network Critical Path Line

In a network, the path along which the project takes the maximum time from start to finish is called the critical path. The critical path determines the minimum expected time required for completion of the project. Activities along the critical path have the least total slack, which may be positive, zero or negative. When the due data (D) assigned for completion of the project is same as the expected date of completion (T) *i.e.* when $D=T$, the critical path has zero slack. In this case, any delay in performing any of the activities along the critical path will result in delay in the over-all completion of the project. Again, when 'a' due data is assigned to the project later than the expected date of completion *i.e* when $D>T$, the activities in the critical path will have positive slack. That means, any activity on the critical path could be delayed by a period (D-T), without delaying the due date of completion of the project. The same positive slack of (D-T) will also be available to all non-critical activities. Finally, when the assigned due date (D) is earlier than the expected due date (T), *i.e.* when D<T, The critical path will have a negative slack. In that case, the project has to be expected or 'crashed' as it is called, to achieve the estimated target data of completion. Activities on the non-critical paths may be delayed to the extent of the difference between Σ activity duration along critical path minus Σ activity durations along non-critical path. In case of inordinate delay in the performance of non-critical activities, a non-critical path may eventually become critical, mod-

way during execution of the project, necessitating urgent intervention of the project manager and re-orientation of resources to tide over the difficult situation. To avoid this, periodic review of the network is necessary and performances of both critical and non-critical activities are to be closely monitored regularly.

2.4.2 Creative Pre-Erection Block Erection Network

Figure 2.5 is a typical example block sequence diagram in which B1 is the seed with three children namely B3, B4 and B2 Similarly B3 has two children B6, B7 and B4,B6 has two children B9,B7 so on and so fort. Figure 2.5 is a typical example for block sequence diagram in which B1 is the seed with three children namely B3, B4. Similarly B3 has two children B6, B7 and B4, B6 has two children B9,B7 and so forth. This parent-child information is stored for each ship in the allocated database files. The starting day of erection (ENT) and ending day of erection (LNT) is stored in block data file. A Pitch is the time that is required for sequenced operation on the child block. Then we calculate ENT for all blocks and adding pitch to parent block's ENT does this, giving child's ENT this method is followed till the last block. But at junction nodes of block sequence (say block B7) we get an ENT value for the parent blocks B4, B3, and B6 but we consider only the highest value. Here an assumption is made that the first and the last blocks lie in critical path. A critical path is a series of linked tasks that must be done on time for the project to finish on time. If any task on a critical path is delayed, it can end up delaying the project's finish date. A block on the critical path is identified, as its ENT and LNT are same. In this case B10 is the last block and search in reverse direction is followed to find the parents to evaluate the LNT's of all

blocks. As discussed earlier in the case of ENT calculation at the junction here we take the lowest value after subtraction of pitch from the LNT value of the child to finalize the LNT of the concerned parent. In this way system stops at the block, as the system is not able to find more precedent parents.

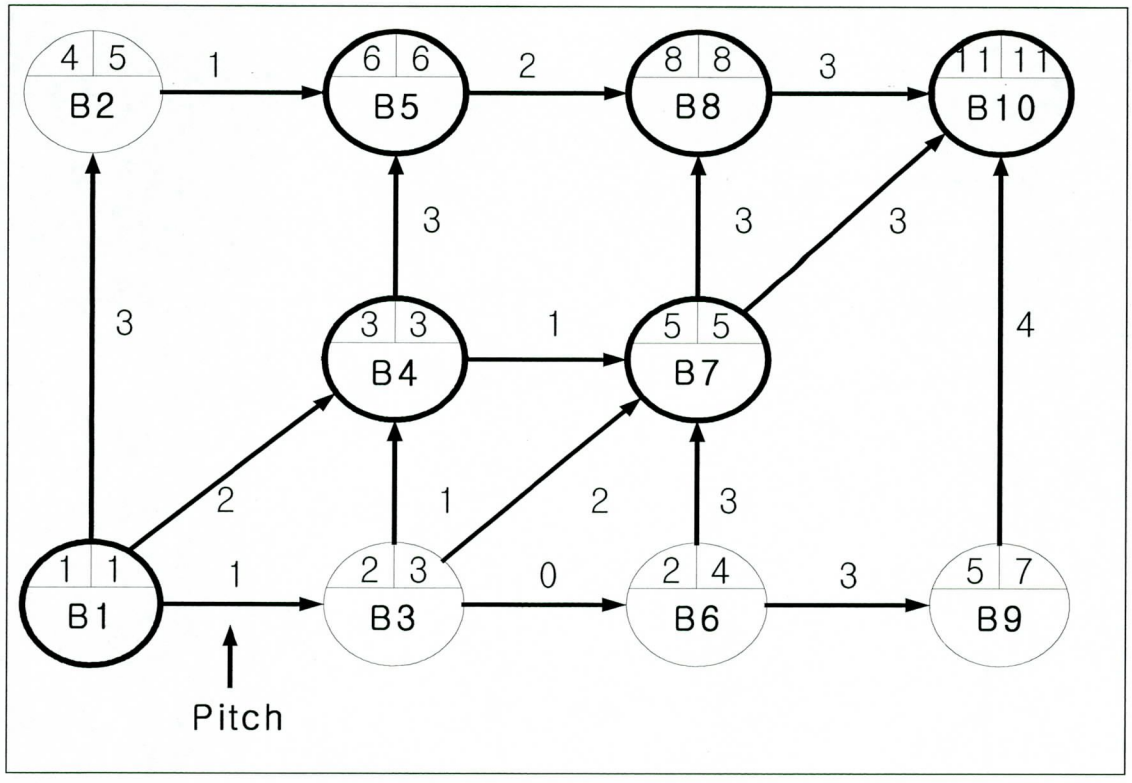


Figure 2.5 Block sequence diagram

Chapter 3 Background of Pre-Erection Area Scheduling Algorithms

3.1 Modern Heuristic Scheduling Method

What is Scheduling?

Shipbuilding occurs in a very wide range of economic activities. It always involves accomplishing a number of things that tie up various resources for periods of time. The resources are in limited supply. The things to be accomplished may be called "jobs" or "project" or " assignments" and are composed of elementary parts called "activities" or "operations" and " delays." Each activity requires certain amounts of specified resources for a specified time called the " process time." Resources also have elementary parts, called "machines," "cells," "transport," "delays," and so on. Scheduling problems are often complicated by large numbers of constraints relating activities to each other, resources to activities and to each other, and either resources or activities to events external to the system. For example, there may be precedence constraints connection activities that specify which activities must precede other activities, and by how much of a delay, or by how much allowed overlap. Or two particular activities may interfere with each other and be unable to us the same activities. Or a resource may be unavailable during specified intervals due to planned maintenance or planned use outside the system. Since these complex interrelationships can make exact or even approximate solutions of large scheduling problems very difficult, it is natural to attempt to solve simpler versions of a problems first to gain insight. Then one can test how sensitive the solution is to this complexity and find approximate solutions to difficult problems where the complexity proves central.

Material Requirements Planning (MRP) Scheduling

Material Requirements Planning (MRP) was initially developed without any capacity checks or input from other departments: thus, the production plan often was not believable to anyone outside of the production function, Closed loop MRP is an enhancement that include capacity checks, which are used iteratively with the master production scheduling(MPS) and the component sources planning or MRP II, contains an additional enhancement that converts a number of the outputs of production planning and control into financial terms-for example, inventories in dollars, labor, budget, shipping budget, standard hours of output in dollars, vendor dollars commitments, and so forth. It also draws on forecasts from other departments, such as marketing, so that its scheduling are more acceptable across the firm.

Master production scheduling is a complex task in an MRP assembly setting. For one thins, the bottleneck work center can shift depending on the changing nature of the workload and the labor force available in the short run. Thus, what appears on the surface to be a feasible master schedule may not be so when the detailed, implied needs at other production (and procurement) stages are worked out. Thus, although some bottleneck scheduling decision rules may be of assistance in master scheduling, a trial-and-error component, as well as some, will almost certainly be required.

Just-In-Time (JIT) Scheduling

Most of the Japanese success in international markets has been the result of the adoption of just-in-time, a concept first introduced by the Toyota Motor Corporation. JIT is a total manufacturing system encompassing product design, equipment selection, materials management, quality

assurance, line layout, job design, and productivity improvement. Inventories are reduced as much as possible to increase productivity, to improve quality, and to reduce production lead times (and hence customer response times). The Toyota Production System was developed over a period of twenty years and led to higher quality, lower cost, and substantially less labor time per vehicle than was achieved by Toyota`s international competitors, This success defied conventional wisdom, which assumed that the much larger plants of, say, the Americans should have exhibited the cost benefits of economies of scale. By the mid-1890s large number of major international manufacturing firms had developed a strong interest in understanding and replicating Toyota`s system. The JIT philosophy directly addresses some of the fundamental weaknesses of MRP. Specifically, the goal of JIT is to remove all waste from the manufacturing environment, so that the right time (not late or early), with zero inventories, zeros lead time, and no queues. Waste means anything –inventory, disruptions, or poor quality, that disrupts the flow of products and does not contribute to making or selling them, is waste, Common example are counting, scheduling, moving and sorting, In addition to eliminating waste, JIT seeks to eliminate all uncertainty, including machine breakdowns. Manufacturing is to be flexible, but in a flow process, For these ambitious goals to be achieved, the firm must pursue continuous improvement.

Theory of Constraints (TOC) Scheduling

The Theory of Constraints (TOC) is relatively recent development in the practical aspect of making organizational decisions in situations in which constraints exist. Dr. Eliyahu M Goldratt first described the theory in his novel, The Goal. In many organizations, TOC and TOC logic form

major portions of that organization, philosophy of continuous improvement. The TOC has been used at three different levels: Production Management-TOC was initially applied here to solve problems of bottlenecks, scheduling, and inventory reduction. TOC scheduling method is Drum-Buffer-Rope scheduling method. All manufacturing plants have dependent events and statistical fluctuations. The challenge is to manage these events and fluctuation so that the organization. achieves its goal. The emphasis in drum-buffer-rope scheduling (DBR) is the incorporation of the inevitable dependent events and statistical fluctuations within any systems in the development of a scheduling approach. The DBR methodology is a technique for developing a smooth, obtainable schedule for the plant and for maximizing and managing the productivity of a manufacturing facility from a global, not a local, perspective. It differs from other manufacturing technique in that it concentrates on determining the relationships among resources in resolving conflicts to create a smooth flow of product and is applicable to all types of processes whether they are repetitive, process, or job shop. Drum-buffer-rope also provides an improved method of focusing protection so that the impact of disturbances on smooth production flow can be minimized. The DBR methodology consists of three elements: the drum, the buffer and the rope. Three areas typically require protection, as illustrated in Figure 3.1

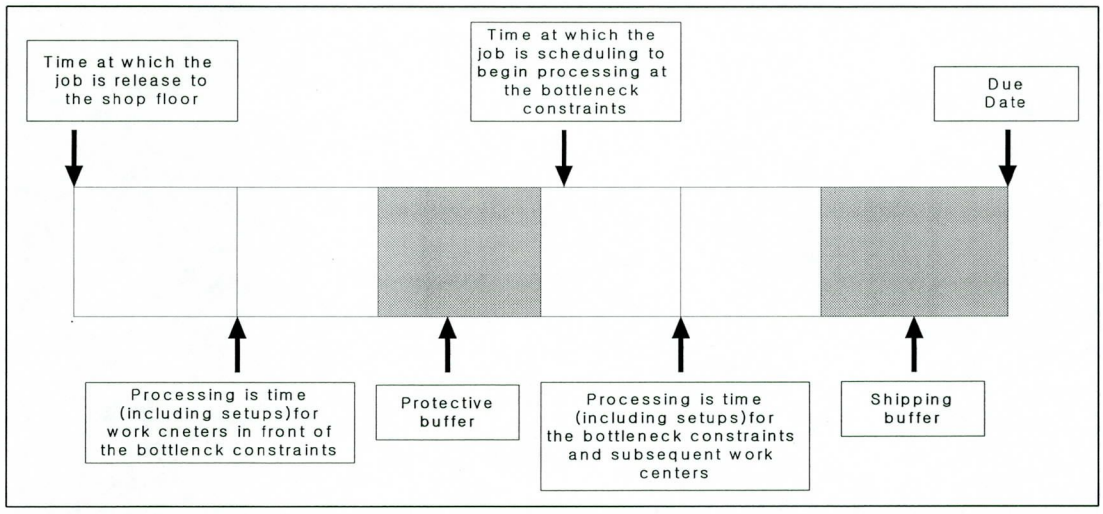


Figure 3.1 Buffers in a DBR Scheduling System

Forward and Backward procedure scheduling

Forward scheduling calculates job schedules forward from "today," to project the completion date. Backward scheduling calculates job schedules backward from the due date, to project the start date. Forward scheduling is necessary to calculate the start time and the finish time of each activity. By assuming the best possible scenario all activities start as early as they possibly can, we can estimate the Early Start (ES) and the Early Finish (EF) for each activity.

The calculation is: EF = ES + Activity Duration, and

ES = MAX (All Predecessor's EF)

[This is required when there is more than one predecessor to any activity].

The Backward Pass starts at the end of the project. We assume a number of days that we desire the project to be completed in. Typically this number is greater than, or equal to the Project

Duration obtained via the Forward Pass. A desired completion time shorter than the Forward Pass determination cannot be achieved, because the Forward Pass assumes the 'Best Case' scenario. The Backward Pass is used to determine the absolute latest that the project can start and still be completed within the time objective. In other words, the Backward Pass is used to determine the Late Start and Late Finish of each activity and the project as a whole. As noted, we start by calculating the Late Start (LS) and Late Finish (LF) of each activity, starting with the last.

The calculation is as follows: LS = LF – Activity Duration, and

LF = MIN (All Successor's LS)

[This is required when there is more than one successor to any activity].

Tomars and Lova develop an iterative forward-backward heuristic. In each iteration, either the serial or the parallel Schedule Generation Scheme is used to generate a schedule by regret-based sampling with the least finish time (LFT) priority rule. A backward-forward pass then improves the resulting schedule.

Looking -Forward in Scheduling

The term "Looking- Forward" means knowing the impact of current decision on future status. One of the human intelligent feature that Artificial Intelligence has difficult in imitating is looking-forward. In schedule problem solving, looking –Forward can be said as having a global perspective during the scheduling process. The looking-Forward schedule(LFS) represents an intermediate level of planning. In Looking-Forward approach the activities of schedule are decomposed into smaller activities where key resource and information constraints can be identified These activities

maintain the original precedence relationships but reveal greater details They will be further decomposed into tasks (or work assignments) shortly before construction work is executed. The schedule contains major work items that must be complete in order to meet the milestone dates in the master pull schedule. The major activities are " screened" prior to entry in the LFS Screening essentially means that all the constraints preventing the execution of the activity have been identified, and enough time remains prior to the activity's scheduled start date to eliminate those constraints. The exact duration of the looking-forward algorithm. screen is determined on a case-by-case basis by the lead-time required to eliminate constraints.

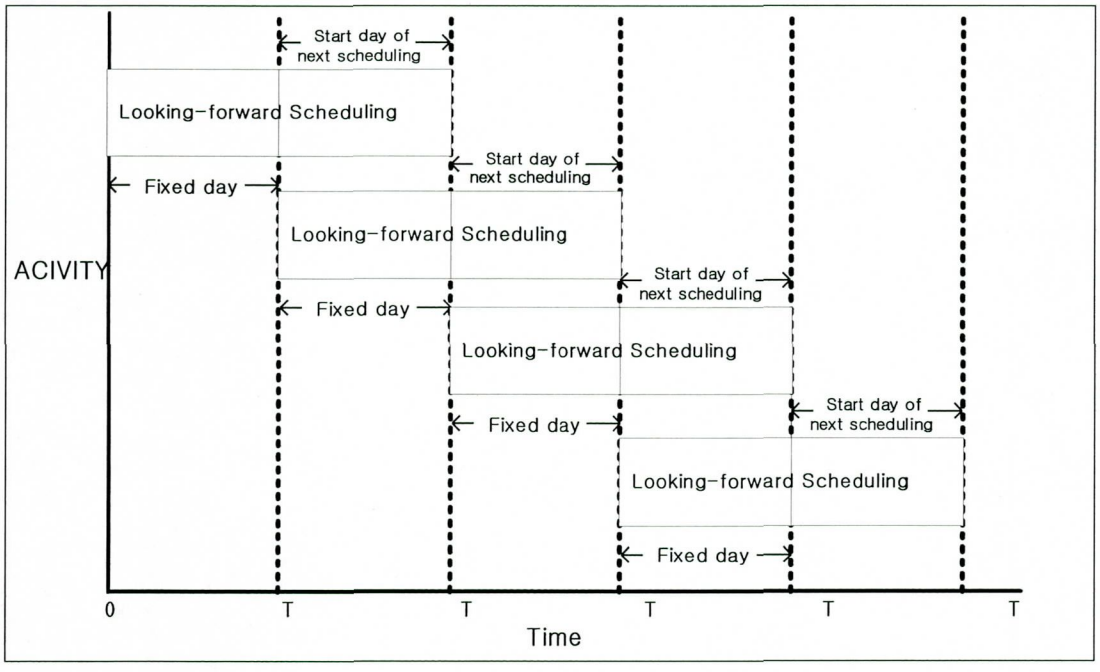


Figure 3.2 Looking-Forward Algorithms

The algorithm operates in two steps :A fixed day stage and a stary day of next scheduling. The

algorithm use focasting & load balance method called the looking-forward D. The algorithms primary scheduling start [T,T+D] in the decide day and forcast workload in the next interval [T+D,T+2D] in start day of next scheduling while collection the decide day in this primary schedule to be schedule in the next this process is illustrated by Figure 3.2

3.2 Development of Looking-Forward Scheduling

This concept is used for fixing the time for the off-critical blocks and it is done considering the percentage area occupation of the specific block in the Pre-Erection (P.E) Area and with respect to the total P.E Area and the period of the delay, that is, the difference of the dates between the ENT and the LNT. The weights are decided based on the available mathematical ideas such as weighted regression method is resolve the bottleneck caused by queuing situation.

In this paper time based constraint is approached though a serious spatial constraint is also cited in this problem. In the space based constraints the maximal occupancy is limited for 68 % because of the structural complications of the specialized vessels like LNG carriers. The time based constraint arises because of the surging of block flow into the PE Area on a Particular Period. Especially between the launching and keel laying period as the two activities has to happen simultaneously. The suggested LFS handles time based constraint significantly and easing the production crunch and surge situation. This also helps in the local load smoothening and on extrapolation this works on the global domain in stupendous improvement. The block erection sequence is decided by the optimal date as a result of making a flow diagram from the LFS. The calculation is made to happen from the erection network diagram and LFS in a hybrid formation.

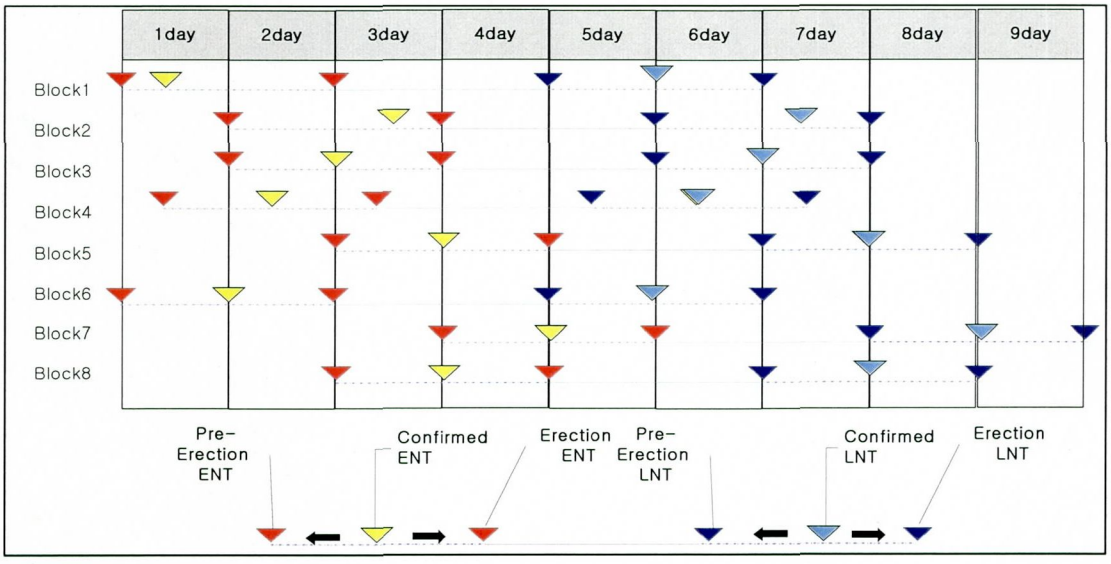


Figure 3.3 An Implementation of LFS algorithm

Let's understand the proposed algorithm, an implementation procedure is described in Figure3.3 Here legend describes the activity pattern of the blocks which is interpreted directly from the erection sequence diagram described in Figure 2.5 describing the possible fixing of the dates of

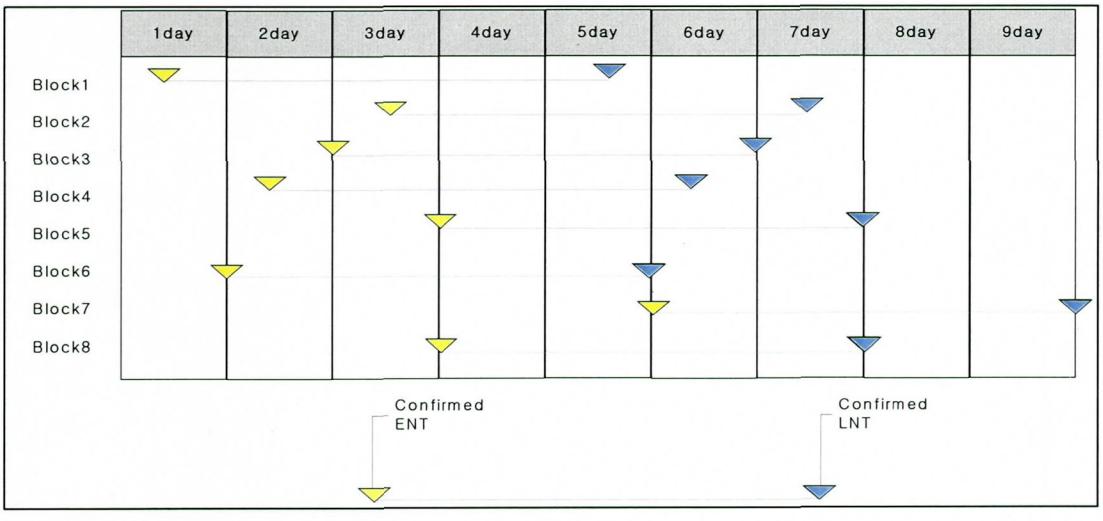


Figure 3.4 Decision making operation in LFS

erection. There is a serious decision making constraint always lies in the fixing the dates of off-critical blocks. The confirmed LNT and ENT is described in Figure 3.3 is as a result of Figure 3.4

In Figure 3.5 the forward domain is sighted to few days ahead to schedule for some days. The figure explains LF(2,1) and LF(3,1) typically explains the Looking Forward to three days to schedule one day. The limit of the forward sight is restricted by performing enough experiments to arrive at a conclusion. Graphically speaking, the X- direction describes days and Y- direction restricts activity name or in this case the block names. The addition of the activities on a particular day describes the percentage of the spatial occupancy on the ideal circumstances is based on these cases resulting in the date fixing. There is also a possibility of the queuing condition while making decisions for fixing up of dates especially on block surging conditions invoking the use of another sub-program

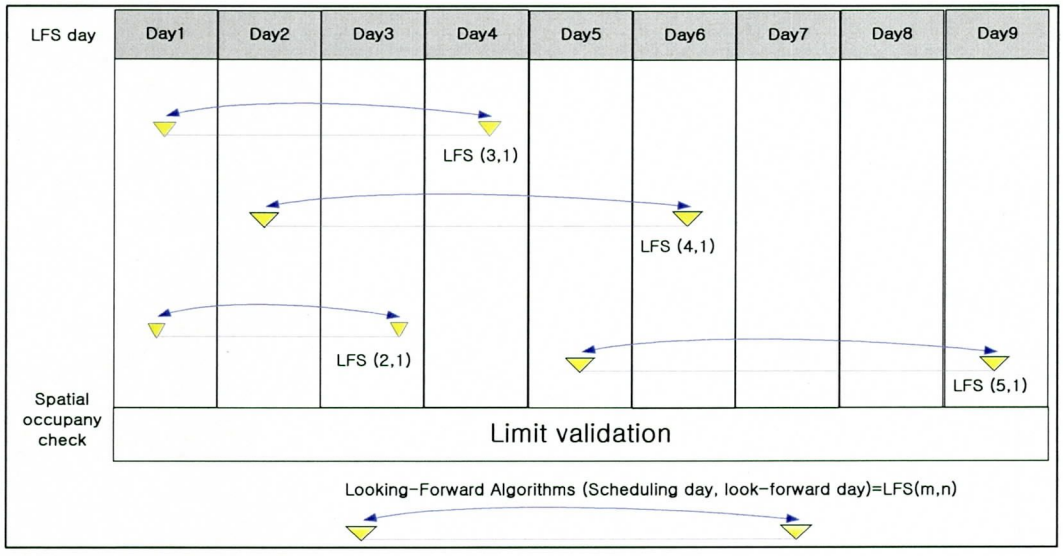


Figure. 3.5 LFS Result Estimation

dealing with the weighting regression algorithm. Once the Figure 3.4 problem is resolved dates are fixed as shown in the Figure 3.5. The two different color representations distinguish the ENT and the LNT specifically.

The "Looking-Forward" Algorithm presented in this section ellaborates the algorithm and the procedural execution stages. Inorder to test the algorithm for its verstility a computer code has been prepared in the VC++ compiler to test and obtain results. The prepared computer code is demonstrated using the Figure 3.6

```
procedure looking forward algorithm
begin
read erection sequence data
step-1
i=0
scan LF domain
evaluate spatial limitation
      if i=max block
      yes
      repeat
      else
      save minimal value addressed to
           candidate block set
      break;

      step – 2
      i=0
      scan LF domain
      evaluate PE Period load
      if i=max block
      repeat
      else
      save minimal value addressed to
           candidate block set
      break;

      step- 3
      i=0
      if  step 1 = step 2
      print results
      else
          run Weighted regression algorithm
          until  i= max
          fix erection dates
       print Candidate blocks = Results
                    end
```

Figure 3.6 "Looking-Forward" Scheduling Algorithm

3.3 Probability based on Fixation technique of Off-Critical Block

Basic concept of Probability

When there is an uncertainty in the occurrence of an envent, we talk of possibilities or probabilities of success or failure of occurrence of subc an event. In assessing the probability of occurrence of an event, all posibilities must be exhaustively examined and all such possibilities or outcomes must be equally likely and mutually exhaustive Thus,if we throw a perfect dice,it can show six outcomes 1,2,3,4,5 or 6 only As such the probability of having any one of the figures is 1/6 .Consequently, if there are 'n' possiblities and success occurs in 'p'cases, the probability of success is p/n. When p/n is equal to 1, it means a sure success or certainty and when p/n is zero,it denotes that occurrence of the event is impossible. Probability have two rule : Multiplication and addition rule.

The Mutltiple probabilty of simultance of two envents A and B is wqual to the product of their probabilities of occurrence separately. This is the multipliaction rule of probability. For example, if the probability of completion of a project under adverse weather condition is 0.80 and the probability of completion under 25% reduced labour force is 0.75, then the probability of the project being completed by the due date under adverse weather condition as well under 25% reduce labour force is $0.80 \times 0.75 = 0.60$ only. The additional rule probability of occurrence of any one or two or more events out of a number of independent occurrence of envents, is the sum of their individual probabilities. For example, if a box contins 12 samples of medicine, 4nos marked red,5nos marked blue and 3nos marked green, then then probability of drawing either a red or

blue sample from the box is $\frac{4}{12}+\frac{5}{12}=\frac{3}{4}$. It may be noted that, the sum of probabilities of all possible event occurences is always to unity.

Application of off-critical block erection scheduling

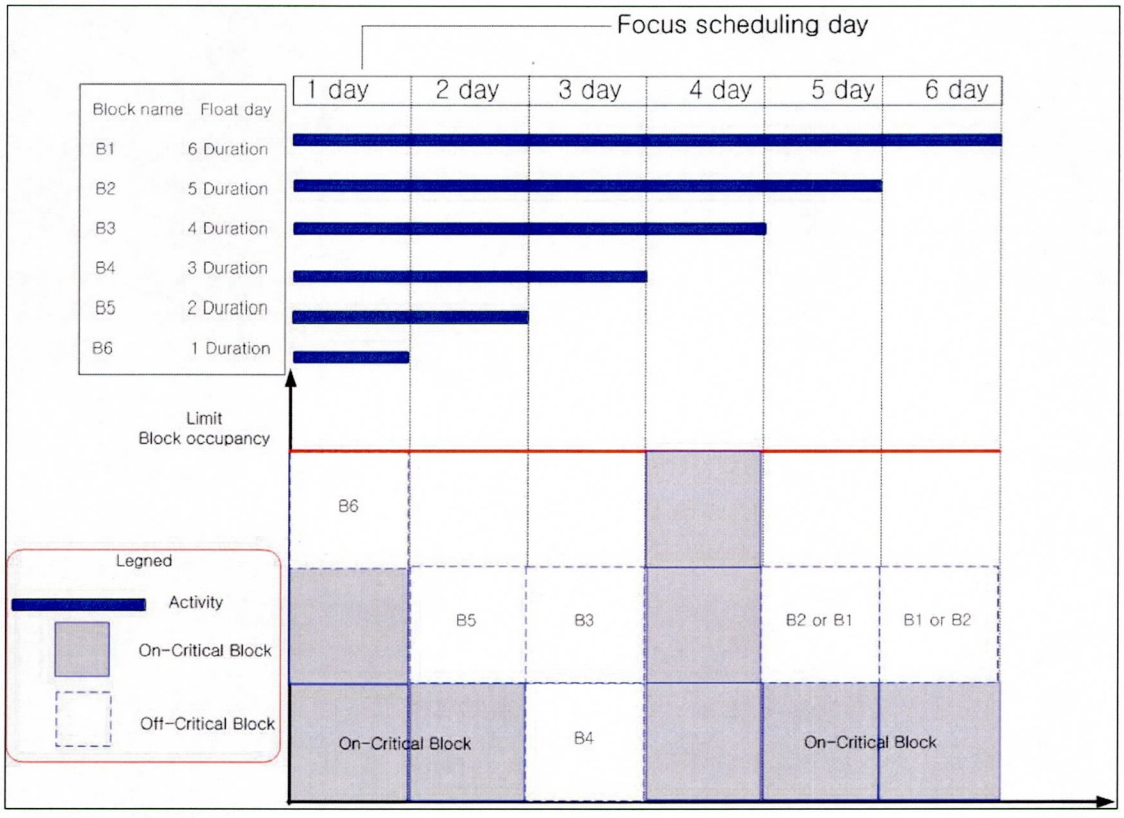


Figure 3.6 Probability based on Erection Day

The research suggest float time based Erection day. Table. 3.1 describe float time based Erection day diagram. The width axis express block name and float day following last Erection day.

	Probability day	Block condition status
Based 1days		
B6	1	Based day choice block
B5	1/2	Probability candidate block
B4	1/3	
B3	1/4	
B2	1/5	
B1	1/6	
Based 2days		
B5	1	Based day choice block
B4	1/2	Probability candidate block
B3	1/3	
B2	1/4	
B1	1/5	
Based 3days		
B4	1	Based day choice block
B3	1/2	Probability candidate block
B2	1/3	
B1	1/4	
Based 4days		
B2		occupancy area is full
B1		
Based 5days~6days		
B2	1	Probability candidate block ,choice number random concept
B1	1	

Table. 3.1 Probability Float Day Block Condition.

The decide scheduling 1day have one block space. Candidate block is B1~B6. This day must insert B1 based Probability method. This method use to decide Block Erection day. The decide scheduling 2day have candinate B5~B1. B5 have 1/5, B4 have 1/4,B3 have 1/3,B2 have 1/4,B1 have 1/5 probability calculation. The decide scheduling 3day follow same rule. The decide scheduling 4day already have full on-critical block in area. Candidate ENT Block in 3day and 4day recive Erection day. Also we consider total load blance scheduling and Erection day.

Chapter 4 Float time based Erection Day and Looking-Forward Algorithm Calculation Program.

4.1 P.E Area Scheduling Program Flow Diagram

P.E Area scheduling program follow STEP and Figure 4.1 flow chart

STEP 1: Reading Block Attribute and take Block Erection Network Activity.

STEP 2: Calculate New Erection Network extract: P.E ENT, P.E LNT, Erection ENT, Erection LNT

STEP 3: New Erection Network separate On-Critical Block information and Off-critical Information

STEP 4: Scheduling n day using look-ahead algorithm decides 'M' day scheduling. We consider P.E Area load

STEP 5: If block is allocated it day. Consider off-critical block and decide day (Re-Scheduling)

STEP 6: Scheduling data compare P.E Area overload.

STEP 7: Comply schedule for load distribution but uncomforting schedules go back Re-scheduling level.

STEP 8: Decide 'M' day scheduling and go back Primitive Scheduling. If arrive final day process is finish

STEP 9: We can optimization scheduling.

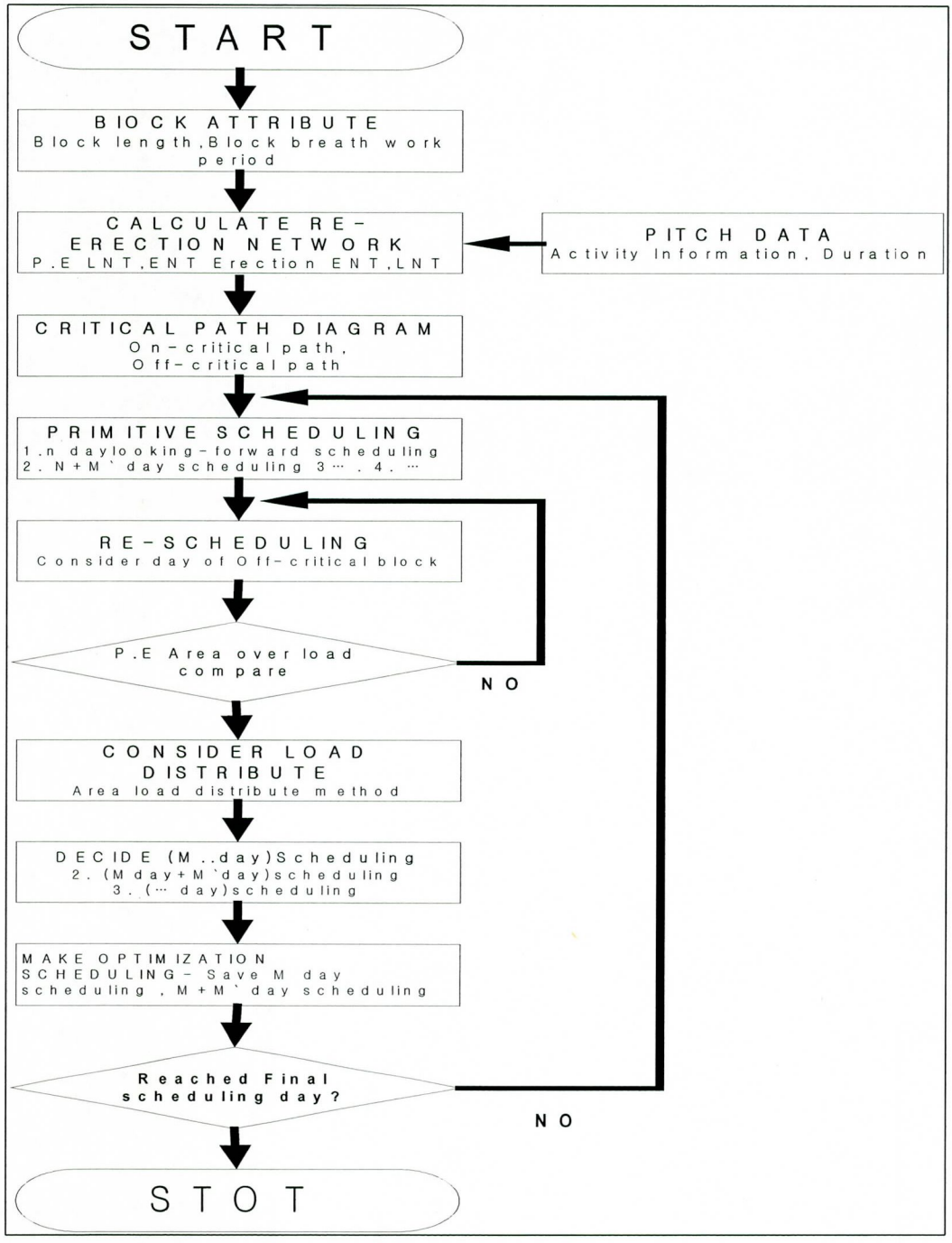


Figure 4.1 P.E Scheduling Program Flow

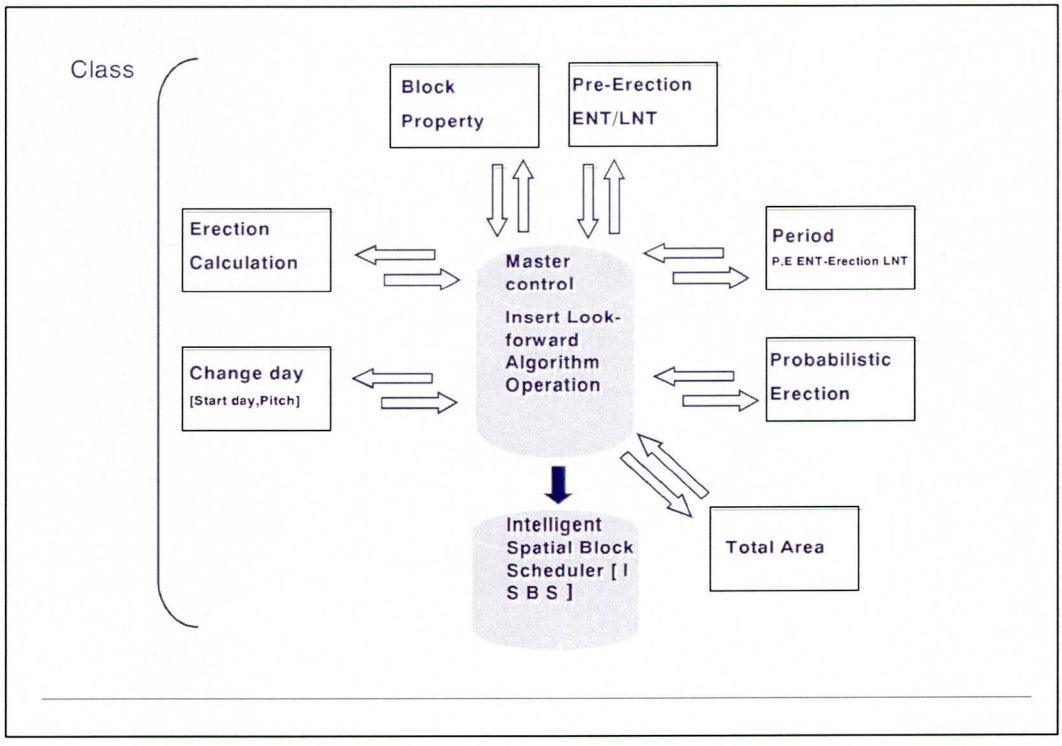


Figure 4.2 P.E Scheduling Program Information Calculation Chart

4.2 P.E Area Scheduling Program Chart

Implementations

The implementation is split up into part as seen in the Figure 4.2. The parts deals with the time constaints handing problem. This portion takes care of the LF and the requisite algorithms for fixing up of the data. The algorithms and the other calculation work and interact with the master controller and proper decisional signals are driven from the maste controller. This master controller takes care of the looking forward algorithm and fixes the appropriate dates of erection. The Change object reflection realworld time. The Erection calculation decide Erection ENT and

LNT using Block pitch data. The Block Property object sperate off-critical path block. The Pre-Erection ENT/LNT object print P.E ENT and P.E LNT. The Period object calculate Block period in P.E Area. The Probabilistic object can make Erection day based Probabilistic method.

4.3 P.E Area Scheduling Program Description.

The programs developed base on the Visual C++ class of windows programming generated by Microsoft Corporation. The generated software runs on Window platform. This problem involves the system. First stage a virtual shipyard with P.E Area information are created this area is considered as the maximum permissible area. Second stage it is scheduling part, where a standard scheduler and modifier are provided. In the standard scheduler scheduling is done from the existing database file. The database comprises of name of block, spatial aspects like length and breath, the starting day of erection, ending day of erection and the working period is being calculated, and then ship type, ship name and the starting day of erection of the first block specifically. Thus the scheduler works in the standard case, creating a standard erection sequence diagram. A modification option in the software is also developed so that, the scheduler can calls data from the previously scheduled erection sequence in order to tailor-make the spatial layout as per the requirements when there is not much of differences in the proposed and accomplished projects. This gives the entire block erection sequence. There is type1 dock information (Figure 4.3) consistent of Breath, length, and distance from bottom. P.E area information is breath and length, Crane height. All Type1 of yard information can input data. This mean is adjusting all shipbuilding company.

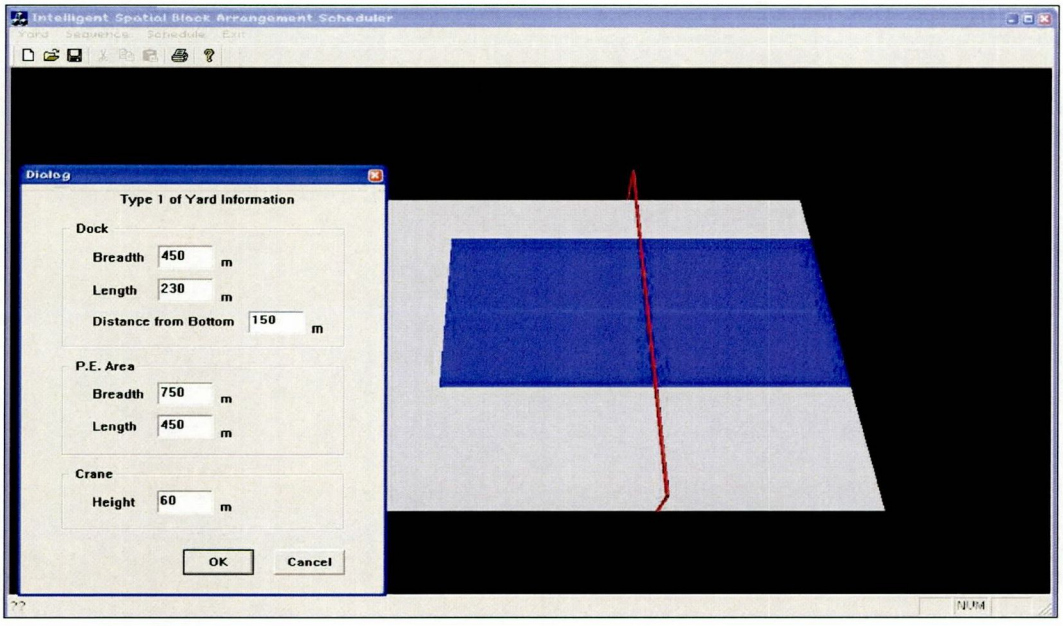


Figure 4.3 Dialogue box decide Dock and P.E Area Information.

Figure 4.4 show first input start block and day also can input final block and final day.

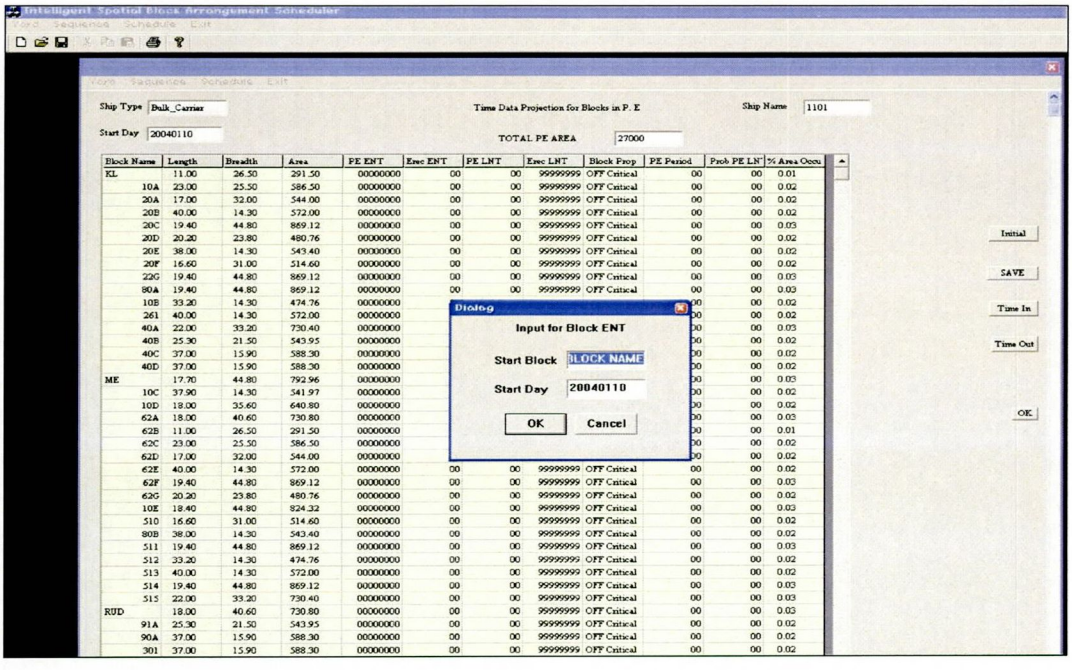
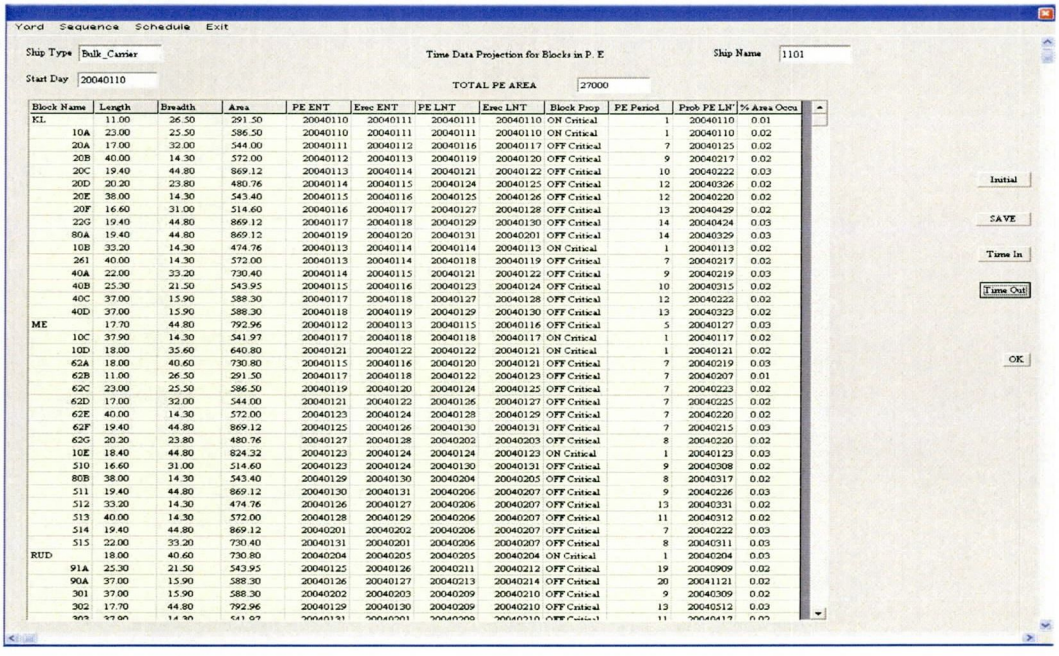


Figure 4.4 Program Input data.

Chapter 5 Simulation Results

5.1 P.E Area Scheduling Program Flow Diagram

Figure 5.1 show result calculation (P.E ENT, P.E LNT, Erection ENT, Erection LNT) also can separate off-critical block. The Looking-Forward based program fixes optimal erection day. This program follows network analysis method. When the Erection day is first finalized P.E Area spatial occupancy load and second give random number theory.



Block Name	Length	Breadth	Area	PE ENT	Erec ENT	PE LNT	Erec LNT	Block Prop	PE Period	Prob PE LNT	% Area Occu
KL	11.00	26.50	291.50	20040110	20040111	20040111	20040110	ON Critical	1	20040110	0.01
10A	23.00	25.50	586.50	20040110	20040111	20040111	20040110	ON Critical	1	20040110	0.02
20A	17.00	32.00	544.00	20040111	20040112	20040116	20040117	OFF Critical	7	20040125	0.02
20B	40.00	14.30	572.00	20040112	20040113	20040119	20040120	OFF Critical	9	20040217	0.02
20C	19.40	44.80	869.12	20040113	20040114	20040121	20040122	OFF Critical	10	20040222	0.03
20D	20.20	23.80	480.76	20040114	20040115	20040124	20040125	OFF Critical	12	20040326	0.02
20E	38.00	14.30	543.40	20040115	20040116	20040125	20040126	OFF Critical	12	20040220	0.02
20F	16.60	31.00	514.60	20040116	20040117	20040127	20040128	OFF Critical	13	20040429	0.02
22G	19.40	44.80	869.12	20040117	20040118	20040129	20040130	OFF Critical	14	20040424	0.03
80A	19.40	44.80	869.12	20040119	20040120	20040131	20040201	OFF Critical	14	20040329	0.03
10B	33.20	14.30	474.76	20040113	20040114	20040114	20040113	ON Critical	1	20040113	0.02
261	40.00	14.30	572.00	20040113	20040114	20040118	20040119	OFF Critical	7	20040217	0.02
40A	22.00	33.20	730.40	20040114	20040115	20040121	20040122	OFF Critical	9	20040219	0.03
40B	25.30	21.50	543.95	20040115	20040116	20040123	20040124	OFF Critical	10	20040315	0.02
40C	37.00	15.90	588.30	20040117	20040118	20040127	20040128	OFF Critical	12	20040222	0.02
40D	37.00	15.90	588.30	20040118	20040119	20040129	20040130	OFF Critical	13	20040323	0.02
ME	17.70	44.80	792.96	20040112	20040113	20040115	20040116	OFF Critical	5	20040127	0.03
10C	37.90	14.30	541.97	20040117	20040118	20040118	20040117	ON Critical	1	20040117	0.02
10D	18.00	35.60	640.80	20040121	20040122	20040122	20040121	ON Critical	1	20040121	0.02
62A	18.00	40.60	730.80	20040115	20040116	20040120	20040121	OFF Critical	7	20040219	0.03
62B	11.00	26.50	291.50	20040117	20040118	20040122	20040123	OFF Critical	7	20040207	0.01
62C	23.00	25.50	586.50	20040119	20040120	20040124	20040125	OFF Critical	7	20040223	0.02
62D	17.00	32.00	544.00	20040121	20040122	20040126	20040127	OFF Critical	7	20040225	0.02
62E	40.00	14.30	572.00	20040123	20040124	20040128	20040129	OFF Critical	7	20040220	0.02
62F	19.40	44.80	869.12	20040125	20040126	20040130	20040131	OFF Critical	7	20040215	0.03
62G	20.20	23.80	480.76	20040127	20040128	20040202	20040203	OFF Critical	8	20040220	0.02
10E	18.40	44.80	824.32	20040123	20040124	20040124	20040123	ON Critical	1	20040123	0.03
510	16.60	31.00	514.60	20040123	20040124	20040130	20040131	OFF Critical	9	20040308	0.02
80B	38.00	14.30	543.40	20040129	20040130	20040204	20040205	OFF Critical	8	20040317	0.02
511	19.40	44.80	869.12	20040130	20040131	20040206	20040207	OFF Critical	9	20040226	0.03
512	33.20	14.30	474.76	20040126	20040127	20040206	20040207	OFF Critical	13	20040331	0.02
513	40.00	14.30	572.00	20040128	20040129	20040206	20040207	OFF Critical	11	20040312	0.02
514	19.40	44.80	869.12	20040201	20040202	20040206	20040207	OFF Critical	7	20040222	0.03
515	22.00	33.20	730.40	20040131	20040201	20040206	20040207	OFF Critical	8	20040311	0.03
RUD	18.00	40.60	730.80	20040204	20040205	20040205	20040204	ON Critical	1	20040204	0.03
91A	25.30	21.50	543.95	20040125	20040126	20040211	20040212	OFF Critical	19	20040909	0.02
90A	37.00	15.90	588.30	20040126	20040127	20040213	20040214	OFF Critical	20	20041121	0.02
301	37.00	15.90	588.30	20040202	20040203	20040209	20040210	OFF Critical	9	20040309	0.02
302	17.70	44.80	792.96	20040129	20040130	20040209	20040210	OFF Critical	13	20040512	0.03

Figure 5.1 Optimization of Looking-Forward scheduling Result sample.

5.2 Result Process and Compare graph

The described LFS hybrid heuristic algorithm is implemented on the VC++ compiler and various experiments were conducted and one of the test client's truncated results is tabulated in Table. 5.1 and Table 5.2. The Table. 5.1 describes the experimental result of the tests performed on the prepared test client data and conducting the validation of the proposed algorithm. Here it calculates the erection sequence network computationaly, it determines whether the block is on critical or off-critical path, displays the percentage area occupied and the percentage of the area contribution per block, it calculates the pre-erection ENT, pre-erection LNT, erection ENT , erection LNT, and the LFS based fixed dates of erection as well. While moving forward the results are graphically ploted to analyse its capability. The program is subjected to various kinds of experimental tests and appreciable results have been found. The test client that is executed here displays the graphical results in Figure 5.3 and in Figure 5.4 These figures projects the load flutuations before and after the execution of "looking forward" algorithm. The second level of the programming has to handle the serious part of the described problem which deals with the spatial complication of the shipyard at the P.E Area

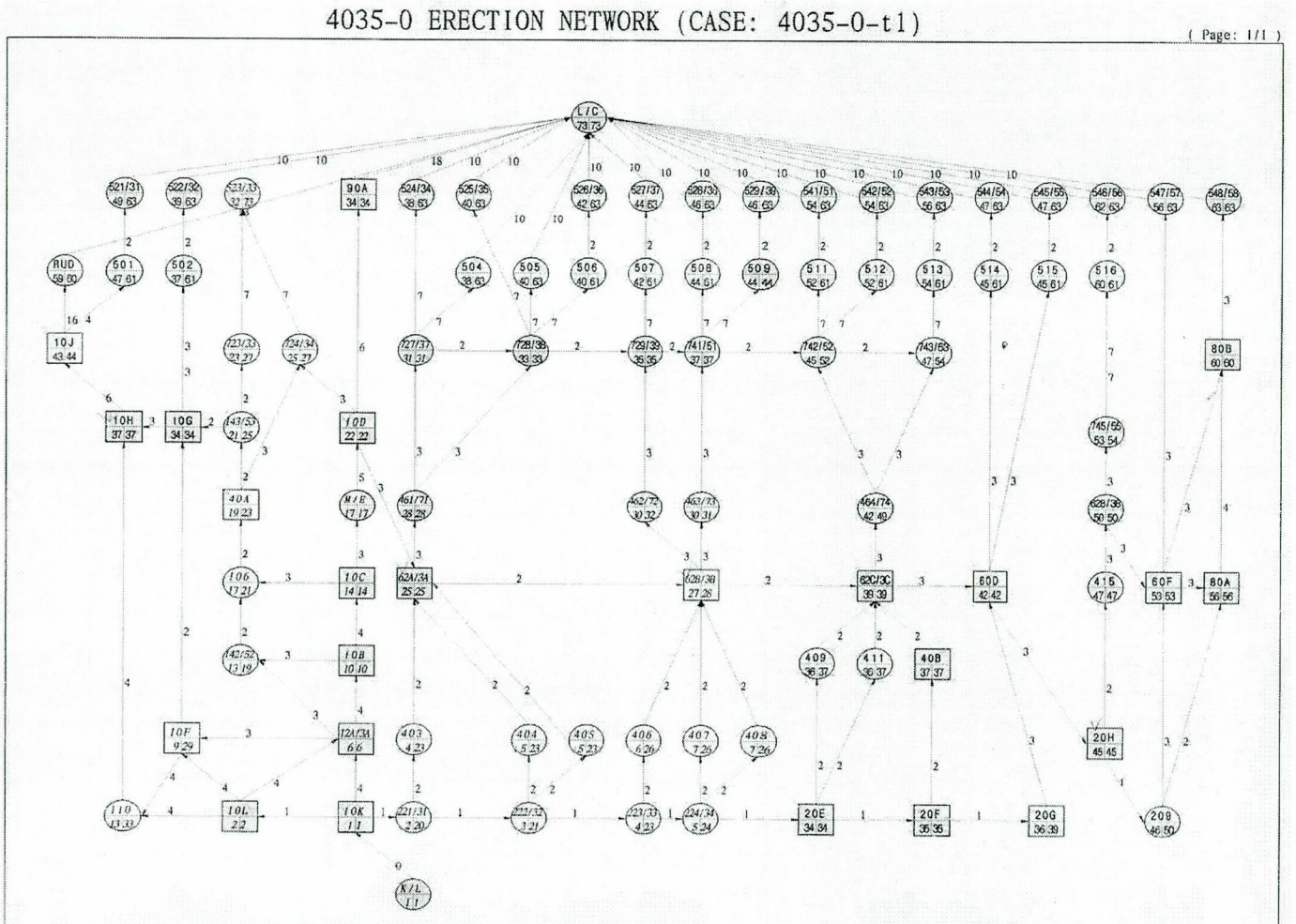


Figure 5.2 Live Shipyard of Block Erection Sequence for Testing

Ship Type Bulk_Carrier 1101

Start of Day : 2004-01-10

Block name	Length	Breath	Area	P.E ENT	Erection LNT	P.E Period	P.E LNT	Erection ENT	Prob P.E LNT	Block property
KL	11	26.5	291.5	20040110	20040110	1	20040111	20040111	20040110	On critical Block
10A	23	25.5	586.5	20040110	20040110	1	20040111	20040111	20040110	Off critical Block
20A	17	32	544	20040111	20040117	7	20040112	20040116	20040215	Off critical Block
20B	40	14.3	572	20040112	20040120	9	20040113	20040119	20040226	Off critical Block
20C	19.4	44.8	869.12	20040113	20040122	10	20040114	20040121	20040212	Off critical Block
20D	20.2	23.8	480.76	20040114	20040125	12	20040115	20040124	20040207	Off critical Block
20E	38	14.3	543.4	20040115	20040126	12	20040116	20040125	20040420	Off critical Block
20F	16.6	31	514.6	20040116	20040128	13	20040117	20040127	20040429	Off critical Block
22G	19.4	44.8	869.12	20040117	20040130	14	20040118	20040129	20040522	Off critical Block
80A	19.4	44.8	869.12	20040119	20040201	14	20040120	20040131	20040329	Off critical Block
10B	33.2	14.3	474.76	20040113	20040113	1	20040114	20040114	20040113	Off critical Block
261	40	14.3	572	20040113	20040119	7	20040114	20040118	20040127	On critical Block
40A	22	33.2	730.4	20040114	20040122	9	20040115	20040121	20040228	Off critical Block
40B	25.3	21.5	543.95	20040115	20040124	10	20040116	20040123	20040305	Off critical Block
40C	37	15.9	588.3	20040117	20040128	12	20040118	20040127	20040222	Off critical Block
40D	37	15.9	588.3	20040118	20040130	13	20040119	20040129	20040501	Off critical Block
ME	17.7	44.8	792.96	20040112	20040116	5	20040113	20040115	20040122	On critical Block
10C	37.9	14.3	541.97	20040117	20040117	1	20040118	20040118	20040117	Off critical Block
10D	18	35.6	640.8	20040121	20040121	1	20040122	20040122	20040121	Off critical Block
62A	18	40.6	730.8	20040115	20040121	7	20040116	20040120	20040219	Off critical Block
62B	11	26.5	291.5	20040117	20040123	7	20040118	20040122	20040214	On critical Block
62C	23	25.5	586.5	20040119	20040125	7	20040120	20040124	20040216	On critical Block
62D	17	32	544	20040121	20040127	7	20040122	20040126	20040218	On critical Block
62E	40	14.3	572	20040123	20040129	7	20040124	20040128	20040220	On critical Block

Block name	Length	Breath	Area	P.E ENT	Erection LNT	P.E Period	P.E LNT	Erection ENT	Prob P.E LNT	Block property
62G	20.2	23.8	480.76	20040127	20040203	8	20040128	20040202	20040315	Off critical Block
10E	18.4	44.8	824.32	20040123	20040123	1	20040124	20040124	20040123	Off critical Block
510	16.6	31	514.6	20040123	20040131	9	20040124	20040130	20040308	On critical Block
80B	38	14.3	543.4	20040129	20040205	8	20040130	20040204	20040309	Off critical Block
511	19.4	44.8	869.12	20040130	20040207	9	20040131	20040206	20040315	On critical Block
512	33.2	14.3	474.76	20040126	20040207	13	20040127	20040206	20040522	On critical Block
513	40	14.3	572	20040128	20040207	11	20040129	20040206	20040312	On critical Block
514	19.4	44.8	869.12	20040201	20040207	7	20040202	20040206	20040215	On critical Block
515	22	33.2	730.4	20040131	20040207	8	20040201	20040206	20040216	On critical Block
RUD	18	40.6	730.8	20040204	20040204	1	20040205	20040205	20040204	On critical Block
91A	25.3	21.5	543.95	20040125	20040212	19	20040126	20040211	20041017	Off critical Block
90A	37	15.9	588.3	20040126	20040214	20	20040127	20040213	20040614	Off critical Block
301	37	15.9	588.3	20040202	20040210	9	20040203	20040209	20040220	On critical Block
302	17.7	44.8	792.96	20040129	20040210	13	20040130	20040209	20040308	On critical Block
303	37.9	14.3	541.97	20040131	20040210	11	20040201	20040209	20040428	On critical Block
304	18.4	44.8	824.32	20040204	20040210	7	20040205	20040209	20040218	On critical Block
305	18	35.6	640.8	20040203	20040210	8	20040204	20040209	20040306	On critical Block
LC	11	26.5	291.5	20040217	20040217	1	20040218	20040218	20040217	On critical Block

Table 5.1 Data Calculation Results

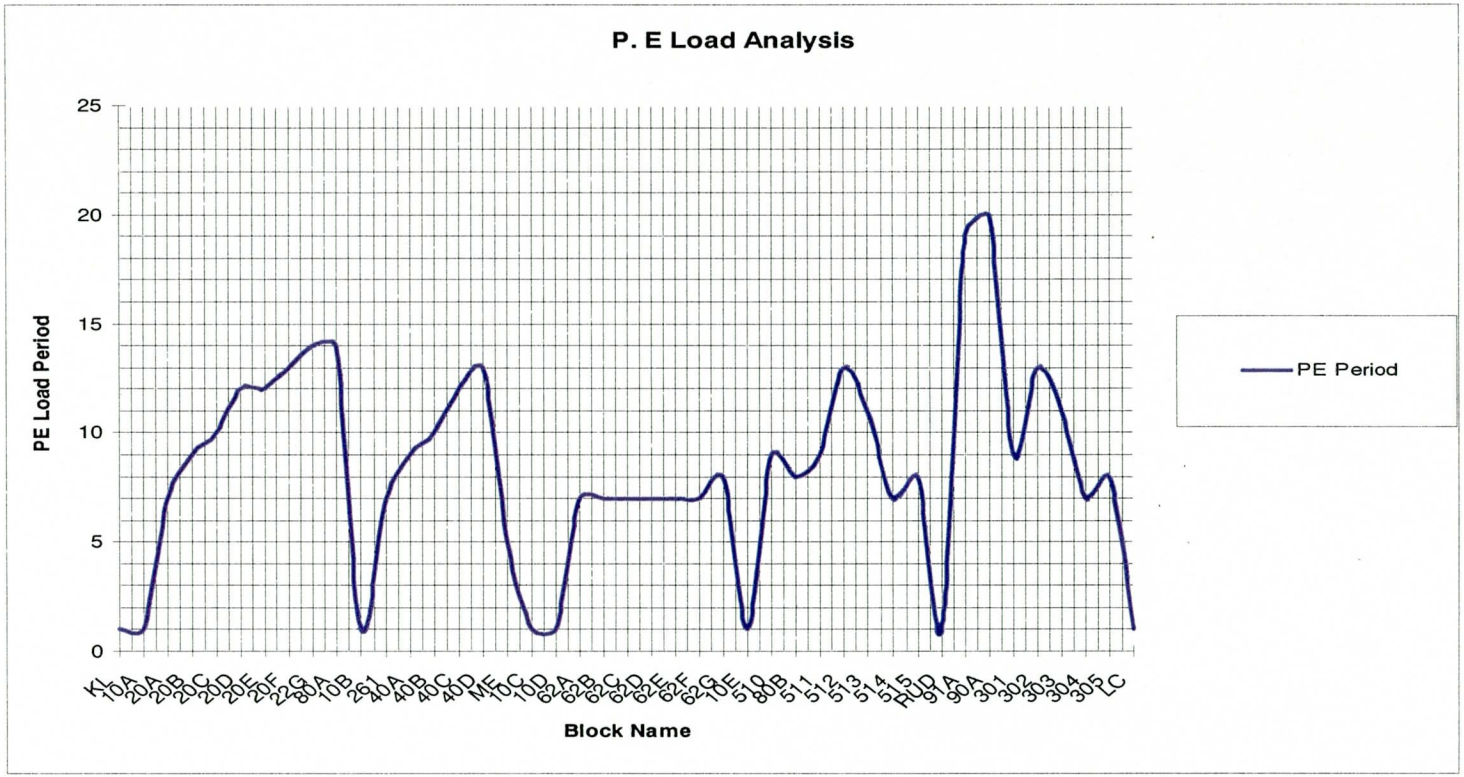


Figure 5.3 The Load Fluctuation of the Raw Data

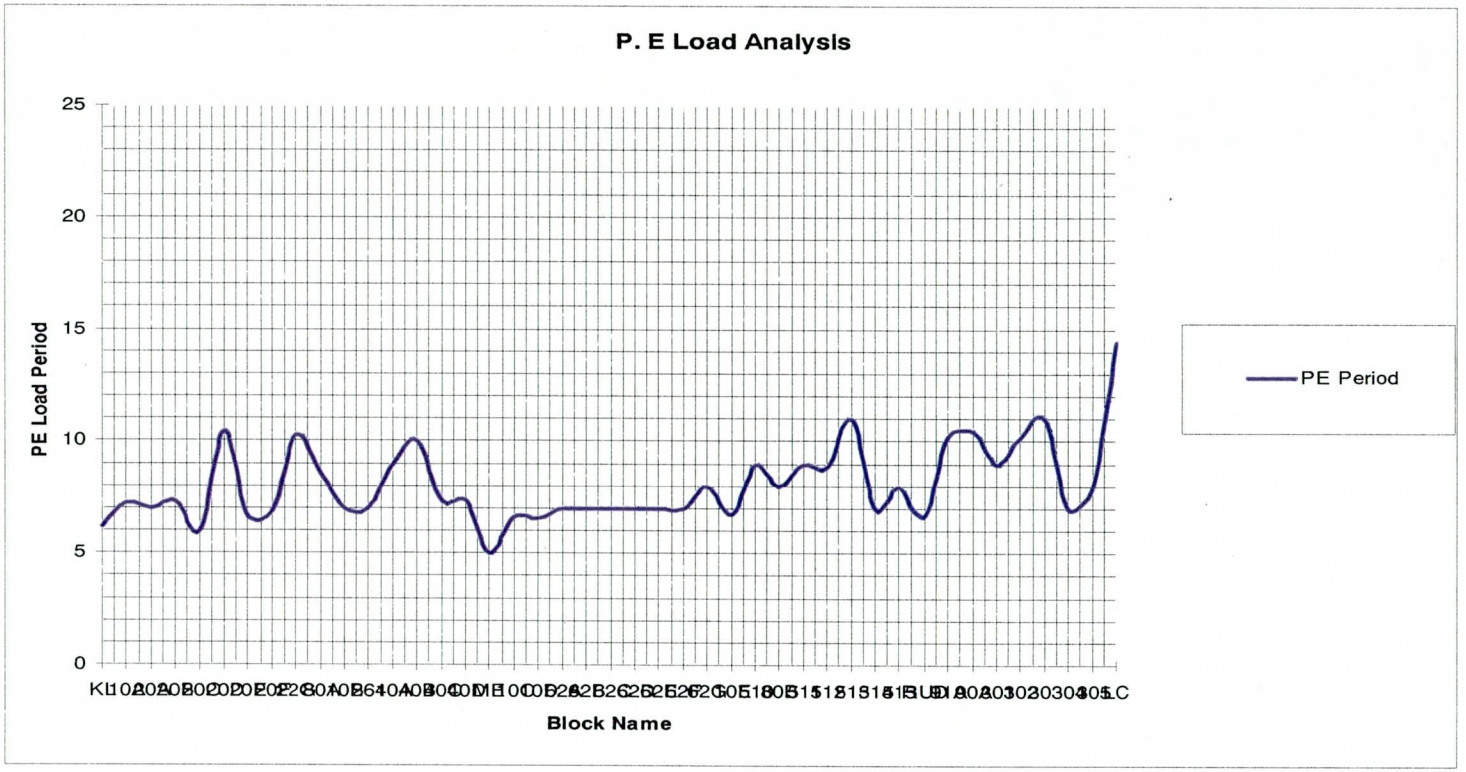


Figure 5.4 The Load Fluctuation of the Modified Data from the LFS Algorithm

1	80B	ME	10C	TE	10C	10A	10B	10D	62A	40B	62B	20D	62C	40D	62E
2	ME	10C	TE	10C	10A	10B	10D	62A	40B	62B	20D	62C	40D	62E	62G
3	10C	ME	10C	10A	10B	10D	62A	40B	62B	20D	62C	40D	62E	62G	10A
4	ME	10C	10A	10B	10D	62A	40B	62B	20D	62C	40D	62E	62G	10A	10X
5	10C	10A	10B	10D	62A	40B	62B	20D	62C	40D	62E	62G	10A	10B	10A
6	10A	10B	10D	62A	40B	62B	20D	62C	40D	62E	62G	10A	10B	10A	10B
7	10B	10D	62A	40B	62B	20D	62C	40D	62E	62G	10A	10B	10A	10B	KL
8	10D	62A	40B	62B	20D	62C	40D	62E	62G	10A	10B	10A	10B	KL	10A

Table 5.2 Generated Candidate Blocks Set

In order to solve the spatial problem it is proposed to use evolutionary algorithms and for those kinds of operations, Many candidate sets are required for the evaluation purpose to increase the search space for the better representation of the results. These results are projected in Table 5.2 and are described herewith the candidate block arrangement on a specific day.

The candidates are generated while the program is executed from start block to the end block between the given time limit and it is done by forward and reverse run of the program on the cyclic loop. Once these two candidate samples are emerged it can be used by the schedules for choosing the best candidate for execution into the other intelligent systems. Scheduling starts with first block 'KL' (Table 5.2) and ends with last block 'LC' in this test client's data set. We can get optimization of dates and fix the date of erection looking-forward method of scheduling. The generated results are the optimal scheduling path and this optimization scheduling path gives best load distribution over the P.E Area.

Chapter 6 CONCLUDING AND FUTURE REMARKS

The "Looking-Forward" scheduling heuristic algorithm has generated optimial schedule and candidate block set. This contribution has illustrated a forecasting scheduling application at the PE area. The P.E Area has been studied by performing various experiements and an observation has been made which shows the credible improvements in the load balancing with adjustment of the fluctuations in the load. This paper has developed the looking-forward algorithm for block sequence scheduling. The developed erection sequence generator program is capable of handling any desired number of blocks required in a shipbuilding process with multiple project and database modifications facilities. It is presumed that the shipbuilder will find this proposed algorithm for the application on their specific cases. The conclusion and future remarks from this paper can be summarized as follows:

1. Looking-Forward Algorithm generates candidate blocks for erection on a specific day.

2. This program balance P.E Area load.

3. The result produced will serve as input data for spatial block arrangement problem using Genetic Algorithm.

Reference

. Yoon, Duck Young, Ranjan Varghese, 2004, "Erection Sequence Generator for Ship Building" , 25-27 March; 33rd International conference on computers and industrial engineering, Jeju, Korea.

. 백동식, 윤덕영, 곽현호, 2004 "공간일정계획에서의 부하조정을 위한 방법론 연구", 한국해양공학회 춘계학술대회 논문집, pp96-100.

. 백동식, 윤덕영, 2004 " Constraints Evaluation for Shipbuilding Industry", 한국해양공학회 추계학술대회 논문집, pp40-44.

. 백동식, 윤덕영, 2005 " Eavaluation of time-in and time-out data in Pre-Erection Area", 한국해양과학기술협회 공동학술대회 논문집, pp487.

. 백동식, 윤덕영, 2005 " 여유일 기반의 탑재일정결정과 미리보기 알고리즘을 이용한 선행탑재일정계획. pp321-325.

. Dong Sik Baek, Ranjan Varghese, Duck Young Yoon, 2005, "Innovation in Ship Erection Network Optimization using Looking Forward Algorithm", TEAM Conference.

. Ranjan Varghese, Duck Young Yoon, 2005, " The Shipyard Spatial Bottleneck Resolving Using Evolutionary Algorithm ", TEAM Conference.

. Hong Tae Kim, "A Simulation-Based Shipbuilding System for Evaluation of Validity in Design and Manufacturing", Department of Industrial System and Information Engineering Korea University, July 2002.

. Chang Kyu Park, 2002, "A Spatial Scheduling application at the block paint shop in shipbuilding : the HYPOS porject" . Journal of Produection Planning & Control Vol. 13,No. 4, pp 342-354.

. Hyung Rim Choi, 1993, "Erection Scheduling at Shipbuilding Using Constraints Directed Graph Search" : DAS-ERECT Doctoral Thesis. KAIST Korea..

. 정귀훈,외 " 조선공업에서의 공간일정계획 시스템 개발 및 응용", IE Interface Vol.14,No.4 pp 394-402, December 2001.

. Kyoung, Jun Lee, 1995, "Development of Spatial Scheduling Expert Systems" Doctoral Thesis. KAIST Korea.

. Ji-Sung Ryu, Jong-Gye Shin, 2004, " Load leveling of block Erection Network Using Diminution of Maximum Load Based on Constraint Satisfaction Technique. Journal of The Society of Naval Architects of Korea.

. Myung Dal Kong, Jung Ja Kim, 2000, "A Heuristic Algorithms for Resource-Constrainted Multi-Project Scheduling". IE Interfaces Vol. 13, No.1, pp 110-119.

. 이재원, 김훈주, 1995 " 유전 알고리즘을 이용한 탑재 공정과 일정계획", 대한조선학회 논문집, Vol.32.

. Ok Ryul Yang, " A Knowledge-Based heuristic Search for Long Rang Production Planning in the Shipbuilding Industry", Master Thesis. KAIST Korea,1992.

. Jae Won Lee, Hoon Joo Kim, " Erection Process Planning & Scheduling using Genetic Algorithm", Transation of the Society of Naval Architecture of Korea, Vol.32, No, February, 9-16.

. Sang Bok Woo, Hyung Gon ryu, Hyung Sang Hahn, September, " Heuristic Algorithm for Resource Leveling in Pre-Erection Scheduling and Erection Scheduling of Shipbuilding", IE Interfaces, Vol.16, No.3,pp332-343,.

. Jeong Seung Lee, " Reactive Scheduling and Control for Shipbuilding :DAS-REACT", Master Thesis. KAIST Korea, 1992.

. V. Ranedra Prasad, etc " *Resiurec-Constrained Shop-Level Scheduling in* a shipyard", Journal of Ship Production.

. H.Poetschke, S.Rosner, " A Look-ahead Scheduler Capable of Trading Scheduling *Complexity against Application Performance*

. Caroline Gagne Marc Gravel, 2001, "A look-ahead additional to the ant colony optimization meta-heuristic and its application to an industrial scheduling problem MIC'2001- 4th Meta heuristics International Conference.

. Fivos Andritsos, Juan perez-Prat, "State-of-the-Art repor on: The Automation and Intergration of Production Processes in Shipbuilding", European Commission Joint Reserch Centre, June 2000

. R A Shenoi, "Ship Production Technology", School of Engineering Sciences The University of Southampton. Chapter 12-3, Chapter 9-1.

. S.K. Bhattacharjee, Fundamentals of PERT/CPM and Project Management, KHANNA PUBLISHERS, pp17. pp18. pp41.pp47.pp61,2002.

. Eliyahu M. Goldratt, Critical Chain, The North River Press, pp77,1997.

. Zbigniew michalewicz David B.Fogel, How to Solve It: Modern Heuristic,Springer,pp232,2000.

. S. N Chary, Production and Operations Management, The McGraw-Hill, Chapter 19, 2004.

. Michael Pinedo Xiuli Chao, Operation Scheduling with application in mnaufacturing and Serives, The McGraw-Hill , pp 59.pp137.pp245, 1999.

. Richard Lee Storch, Colin P. Hammon, Howard M. Bunch, & Richard C .Moore,1978"SHIP Production", pp60-70.

. 대한조선학회편, 선박건조공학,pp 94-103,1996.

. NSRP ,2000, " Process and Operation Technologies" State-of-the-Art-Report pp25-27.

< Appendix >

```
// SequenceDlg.cpp : implementation file
//
#include "stdafx.h"
#include "SSES.h"
#include "SequenceDlg.h"
#include "BlockData.h"
#include "BlockENTDlg.h"
#include "BlockLNTDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSequenceDlg dialog

CSequenceDlg::CSequenceDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CSequenceDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CSequenceDlg)
	m_strShipName = _T("");
	m_strShipType = _T("");
	m_strStartDay = _T("");
	m_strPercentPEArea = _T("");
	//}}AFX_DATA_INIT
}

void CSequenceDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CSequenceDlg)
	DDX_Control(pDX, IDC_SEQUENCE_GRID, m_SequenceGrid);
	DDX_Text(pDX, IDC_SEQUENCE_SHIP_NAME, m_strShipName);
	DDX_Text(pDX, IDC_SEQUENCE_SHIP_TYPE, m_strShipType);
	DDX_Text(pDX, IDC_SEQUENCE_START_DAY, m_strStartDay);
	DDX_Text(pDX, IDC_SEQUENCE_Percent_Area, m_strPercentPEArea);
	//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CSequenceDlg, CDialog)
	//{{AFX_MSG_MAP(CSequenceDlg)
	ON_BN_CLICKED(IDC_SEQUENCE_SAVE, OnSequenceSave)
	ON_BN_CLICKED(IDC_BLOCK_INITIAL, OnBlockInitial)
	ON_BN_CLICKED(IDC_BLOCK_ENT, OnBlockEnt)
	ON_BN_CLICKED(IDC_BLOCK_LNT, OnBlockLnt)
	ON_BN_CLICKED(IDC_PE_TIME_IN, OnPeTimeIn)
	ON_BN_CLICKED(IDC_TIME_OUT, OnTimeOut)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSequenceDlg message handlers

int CSequenceDlg::ChangeDay(BOOL Dir, int StartDay, int Pitch)
{
	int year, month, day, eday;
	CString text;

	text.Format("%d", StartDay);

	year = atoi(text.Mid(0, 4));
	month = atoi(text.Mid(4, 2));
	day = atoi(text.Mid(6, 2));

	if(!Dir)
		Pitch = -Pitch;

	day += Pitch;

	if(Pitch >= 0)
	{
		while(TRUE)
		{
			if(month == 2)
			{
				eday = (year%4 == 0) ? 29 : 28;
			}
			else
			{
				if(month == 1 || month == 3 || month == 5 || month == 7 || month == 8 ||
month == 10 || month == 12)
					eday = 31;
				else
					eday = 30;
			}

			if(day > eday)
			{
				month++;

				if(month > 12)
				{
```

```
                                                year++;
                                                month = 1;
                                        }

                                        day -= eday;

                                }
                                else
                                {
                                        break;
                                }
                        }
                }
                else
                {
                        while(TRUE)
                        {
                                if(day < 1)
                                {
                                        month--;

                                        if(month < 1)
                                        {
                                                year--;
                                                month = 12;
                                        }

                                        if(month == 2)
                                        {
                                                eday = (year%4 == 0) ? 29 : 28;
                                        }
                                        else
                                        {
                                                if(month == 1 || month == 3 || month == 5 || month == 7 ||
month == 8 || month == 10 || month == 12)
                                                        eday = 31;
                                                else
                                                        eday = 30;
                                        }

                                        day += eday;
                                }
                                else
                                {
                                        break;
                                }
                        }
                }

                return year*10000 + month*100 + day;
}

int CSequenceDlg::Period(int StartDay, int EndDay)
{
                int temp;
                int count;


                temp = StartDay;
                count = 0;

                if(StartDay == EndDay)
                        return 1;

                while(TRUE)
                {
                        temp = ChangeDay(TRUE, temp, 1);

                        count++;

                        if(temp == EndDay)
                                break;
                }

                return count + 1;
}

int CSequenceDlg::ErectionDay(int StartDay, int EndDay)
{
                int temp;
                int count;
                int t, tt;
                int temp1;

                temp = StartDay;
                count = 0;

                if(StartDay == EndDay)
                return StartDay;

                while(TRUE)
                {
                        temp = ChangeDay(TRUE, temp, 1);
                        count++;
```

```
		if(temp == EndDay)
			break;

		}

		t= count + 1;

   tt= t/2;

		for (int i= 1; i<=t; i++)

		{
		int temp2 = StartDay;
		temp1 = ChangeDay(TRUE, temp2, tt);
		}
		return temp1;

}

int CSequenceDlg::GetChild(char name[])
{
		int i, cNum;
		BOOL Find;

		Find = FALSE;

		for(POSITION pos=m_PitchList->GetHeadPosition();pos!=NULL;)
		{
			CBlockData *pData = (CBlockData*)m_PitchList->GetNext(pos);

			if(strcmp(pData->Name, name) == 0)
			{
				if(First)
				{
					pData->SetStartDay(Start);
					First = FALSE;
				}

				Find = TRUE;
				cNum = pData->PitchNum;

				// Final Block ////////////////////////////////////////////////
				if(cNum == 0)
				{
					strcpy(EndBlock, pData->Name);
					End = pData->StartDay;

					return 0;
				}
				/////////////////////////////////////////////////////////////

				Person = new CArray [cNum];
				for(i=0;i<cNum;i++)
				{
					strcpy(Person[i].Name, pData->Child[i].Name);

					// Calculate EST ////////////////////////////////////////
					for(POSITION cpos=m_PitchList->GetHeadPosition();cpos!=NULL;)
					{
						CBlockData *pChild =
(CBlockData*)m_PitchList->GetNext(cpos);

						if(strcmp(pChild->Name, pData->Child[i].Name)==0)
						{
							pChild->SetMaxPitch(ChangeDay(TRUE,
pData->StartDay, pData->Pitch[i]));
							break;
						}
					}
					/////////////////////////////////////////////////////////
				}

				break;
			}
		}

		if(Find == FALSE)
			return -1;

		return cNum;
}

int CSequenceDlg::GetParent(char name[])
{
		POSITION pos;
		int i, TempEnd, pNum;
		BOOL Find;

		pNum = 0;
		Find = FALSE;

		// Name ////////////////////////////////////////////////////////////
		for(pos=m_PitchList->GetHeadPosition();pos!=NULL;)
		{
			CBlockData *pData = (CBlockData*)m_PitchList->GetNext(pos);
```

```
                        if(strcmp(pData->Name, name)==0)
                        {
                                if(First)
                                {
                                        pData->SetEndDay(pData->StartDay);
                                        End = pData->EndDay;
                                        TempEnd = pData->EndDay;

                                        First = FALSE;
                                }

                                Find = TRUE;
                                TempEnd = pData->EndDay;
                        }
                }
                ////////////////////////////////////////////////////////////////

                // Escape ///////////////////////////////////////////////////////
                if(Find == FALSE)
                {
                        return -1;
                }
                ////////////////////////////////////////////////////////////////

                // Parent ///////////////////////////////////////////////////////
                for(pos=m_PitchList->GetHeadPosition();pos!=NULL;)
                {
                        CBlockData *pData = (CBlockData*)m_PitchList->GetNext(pos);

                        for(i=0;i<pData->PitchNum;i++)
                        {
                                if(strcmp(pData->Child[i].Name, name)==0)
                                {
                                        pNum++;
                                }
                        }
                }
                ////////////////////////////////////////////////////////////////

                Person = new CArray [pNum];
                pNum = 0;

                // Calculate LST ////////////////////////////////////////////////
                for(pos=m_PitchList->GetHeadPosition();pos!=NULL;)
                {
                        CBlockData *pData = (CBlockData*)m_PitchList->GetNext(pos);

                        for(i=0;i<pData->PitchNum;i++)
                        {
                                if(strcmp(pData->Child[i].Name, name)==0)
                                {
                                        pData->SetMinPitch(ChangeDay(FALSE, TempEnd, pData->Pitch[i]));

                                        strcpy(Person[pNum].Name, pData->Name);
                                        pNum++;
                                }
                        }
                }
                ////////////////////////////////////////////////////////////////

                return pNum;
}

void CSequenceDlg::OnSequenceSave()
{
                // TODO: Add your control notification handler code here
                // Save to Block Structure for EST/LST Calculation ////////////////////////
                FILE *fp;
                CString fname;
                UpdateData(TRUE);
                fname = m_strShipName;
                fp = fopen(fname + "(output).xls", "wt");
                fprintf(fp, "%s %s\n", ShipType, fname);
                fprintf(fp, "%d %s\n", BlockNum, StartDay);

                for(POSITION pos=m_PitchList->GetHeadPosition();pos!=NULL;)
                {
                        CBlockData *pData = (CBlockData *)m_PitchList->GetNext(pos);

                        pData->WorkingDay();

                        fprintf(fp, "%s\t%-8.2lf\t%-8.2lf\t%-8.2lf\t", pData->Name, pData->Length, pData->Breadth,
pData->Area);
                        fprintf(fp, "%d\t%d\t%d\t%d\t", pData->StartDay, pData->EndDay, Period(pData->StartDay,
pData->EndDay), pData->PE);
                        fprintf(fp, "%d\t%d\t%d\n", PEWorkStartDay(pData->StartDay, pData->EndDay),
PEWorkEndDay(pData->StartDay, pData->EndDay), ProbabilisticErectionDay(pData->StartDay, pData->EndDay));
                        fprintf(fp, "%d\n", ProbabilisticErectionDay(pData-> StartDay, pData-> EndDay));

                }

                fclose(fp);
}
```

```
void CSequenceDlg::OnBlockInitial()
{
	// TODO: Add your control notification handler code here
	BOOL m_bInitial;

	m_bInitial = TRUE;

	if(m_bInitial)

	{
		for(POSITION pos=m_PitchList->GetHeadPosition();pos!=NULL;)
		{
			CBlockData *pData = (CBlockData*)m_PitchList->GetNext(pos);
			pData->SetStartDay(00000000);
			pData->SetEndDay(99999999);
		}
		DisplayGrid();
		return;
	}
}

void CSequenceDlg::OnBlockEnt()
{
	// TODO: Add your control notification handler code here
	CBlockENTDlg dlg;

	int i, tempsize;

	FILE *fp;
	BOOL same;

	UpdateData(TRUE);

	dlg.m_strStartBlock = "Block Name";
	dlg.m_strStartDay = m_strStartDay;

	if(dlg.DoModal()!=IDOK)
		return;

	strcpy(StartBlock, dlg.m_strStartBlock);
	Start = atoi(dlg.m_strStartDay);

	same = FALSE;

	for(POSITION pos=m_PitchList->GetHeadPosition();pos!=NULL;)
	{
		CBlockData *pData = (CBlockData *)m_PitchList->GetNext(pos);

		if(strcmp(StartBlock, pData->Name)==0)
		{
			same = TRUE;
			break;
		}
	}

	if(!same)
	{
		CString BName;
		BName.Format("Can't find this Block(%s) please check !", StartBlock);
		AfxMessageBox(BName);

		return;
	}

	First = TRUE;

	hNum = 1;

	House = new CArray [hNum];
	strcpy(House[0].Name, StartBlock);

	while(TRUE)
	{
		tNum = GetChild(House[0].Name);

		// Save ////////////////////////////////////////////////////////////
		fp = fopen("GoCheck.txt", "at");

		fprintf(fp, "%d %d - ", hNum, tNum);
		for(i=0;i<hNum;i++)
		{
			fprintf(fp, "%s ", House[i].Name);
		}
		fprintf(fp, "\n");

		fclose(fp);
		//////////////////////////////////////////////////////////////////

		// Escape //////////////////////////////////////////////////////////
		if((hNum + tNum -1) == 0)
		{
			break;
		}

		if(tNum == -1)
```

```
                {
                        AfxMessageBox("Can't Find this Block Name!");
                        break;
                }
                ////////////////////////////////////////////////////////////////

                tempsize = hNum + tNum - 1;
                Temp = new CArray [tempsize];

                if(hNum==1)
                {
                        for(i=0;i<tempsize;i++)
                        {
                                strcpy(Temp[i].Name, Person[i].Name);
                        }
                }
                else
                {
                        if(tNum == 0)
                        {
                                for(i=1;i<hNum;i++)
                                {
                                        strcpy(Temp[i-1].Name, House[i].Name);
                                }
                        }
                        else
                        {
                                for(i=1;i<hNum;i++)
                                {
                                        strcpy(Temp[i-1].Name, House[i].Name);
                                }

                                for(i=hNum;i<tempsize+1;i++)
                                {
                                        strcpy(Temp[i-1].Name, Person[i-hNum].Name);
                                }
                        }
                }

                delete [] House;

                hNum = tempsize;
                House = new CArray [hNum];

                for(i=0;i<hNum;i++)
                {
                        strcpy(House[i].Name, Temp[i].Name);
                }

                delete [] Temp;

                if(tNum != 0)
                        delete [] Person;
        }

        delete [] House;

        // Reload Grid/// ////////////////////
        DisplayGrid();
        ////////////////////////////////////////
}
void CSequenceDlg::OnBlockLnt()
{
        // TODO: Add your control notification handler code here
        CBlockLNTDlg dlg;

        int i, tempsize;
        FILE *fp;

        dlg.m_strEndBlock = EndBlock;
        dlg.m_nEndDay = End;

        if(dlg.DoModal()!=IDOK)
                return;

        strcpy(StartBlock, dlg.m_strEndBlock);

        First = TRUE;

        hNum = 1;

        House = new CArray [hNum];
        strcpy(House[0].Name, StartBlock);

        while(TRUE)
        {
                tNum = GetParent(House[0].Name);

                // Save ////////////////////////////////////////////////////////
                fp = fopen("BackCheck.txt", "at");

                fprintf(fp, "%d %d - ", hNum, tNum);
                for(i=0;i<hNum;i++)
                {
                        fprintf(fp, "%s ", House[i].Name);
```

```
                        }
                        fprintf(fp, "\n");

                        fclose(fp);
                        ////////////////////////////////////////////////////////////////

                        // Escape ////////////////////////////////////////////////////
                        if((hNum + tNum -1) == 0)
                        {
                                break;
                        }

                        if(tNum == -1)
                        {
                                AfxMessageBox("Can't find this Block Name!");
                                break;
                        }
                        ////////////////////////////////////////////////////////////////

                        tempsize = hNum + tNum - 1;
                        Temp = new CArray [tempsize];

                        if(hNum==1)
                        {
                                for(i=0;i<tempsize;i++)
                                {
                                        strcpy(Temp[i].Name, Person[i].Name);
                                }
                        }
                        else
                        {
                                if(tNum == 0)
                                {
                                        for(i=1;i<hNum;i++)
                                        {
                                                strcpy(Temp[i-1].Name, House[i].Name);
                                        }
                                }
                                else
                                {
                                        for(i=1;i<hNum;i++)
                                        {
                                                strcpy(Temp[i-1].Name, House[i].Name);
                                        }

                                        for(i=hNum;i<tempsize+1;i++)
                                        {
                                                strcpy(Temp[i-1].Name, Person[i-hNum].Name);
                                        }
                                }
                        }

                        delete [] House;

                        hNum = tempsize;
                        House = new CArray [hNum];

                        for(i=0;i<hNum;i++)
                        {
                                strcpy(House[i].Name, Temp[i].Name);
                        }

                        delete [] Temp;

                        if(tNum != 0)
                                delete [] Person;
                }

                delete [] House;

                AllFinished = TRUE;

                // Reload Grid  /////
                DisplayGrid();
                ////////////////////

                AllFinished = FALSE;
}

void CSequenceDlg::DisplayGrid()
{
                POSITION pos;
                int i;

                BlockNum = m_PitchList->GetCount();

                m_SequenceGrid.SetRows(BlockNum + 1);

                m_SequenceGrid.SetTextMatrix(0, 0, "Block Name");
                m_SequenceGrid.SetTextMatrix(0, 1, " Length");
                m_SequenceGrid.SetTextMatrix(0, 2, "Breadth");
                m_SequenceGrid.SetTextMatrix(0, 3, " Area");
                m_SequenceGrid.SetTextMatrix(0, 4, " PE ENT");
                m_SequenceGrid.SetTextMatrix(0, 5, "Erec ENT");
  m_SequenceGrid.SetTextMatrix(0, 6, "PE LNT");
```

```
		m_SequenceGrid.SetTextMatrix(0, 7, " Erc LNT");
		m_SequenceGrid.SetTextMatrix(0, 8, " Block Prop");
		m_SequenceGrid.SetTextMatrix(0, 9, " PE Period");
		m_SequenceGrid.SetTextMatrix(0, 10, " Prob PE LNT");
		m_SequenceGrid.SetTextMatrix(0, 11, "% Area Occu");

//		m_SequenceGrid.SetTextMatrix(0, 7, "P.E. Area");
		m_SequenceGrid.SetTextMatrix(0, 7, "Erec LNT");
//		m_SequenceGrid.SetTextMatrix(0, 8, "Selected Date");

		// Display Grid Function ////////////////////////////////////////////
		CString temp;
		i = 1;

		for(pos=m_PitchList->GetHeadPosition();pos!=NULL;)
		{
			CBlockData *pData = (CBlockData *)m_PitchList->GetNext(pos);

			m_SequenceGrid.SetTextMatrix(i, 0, pData->Name);
			temp.Format("%8.2lf", pData->Length);
			m_SequenceGrid.SetTextMatrix(i, 1, temp);
			temp.Format("%8.2lf", pData->Breadth);
			m_SequenceGrid.SetTextMatrix(i, 2, temp);
			temp.Format("%8.2lf", pData->Area)
			m_SequenceGrid.SetTextMatrix(i, 3, temp);
			temp.Format("%08d", pData->StartDay);
			m_SequenceGrid.SetTextMatrix(i, 4, temp);
			temp.Format("%08d", pData->EndDay);
			m_SequenceGrid.SetTextMatrix(i, 7, temp);


	if(AllFinished)
			temp.Format("%d", PEWorkStartDay(pData->StartDay, pData->EndDay));
			else
			temp.Format("00");
			m_SequenceGrid.SetTextMatrix(i, 5, temp);


	if(AllFinished)
			temp.Format("%d", PEWorkEndDay(pData->StartDay, pData->EndDay));
			else
			temp.Format("00");
			m_SequenceGrid.SetTextMatrix(i, 6, temp);


			int CCC;
			CCC=pData->EndDay - pData->StartDay;
			temp.Format("%08d", CCC);

			if( CCC == 0)
			m_SequenceGrid.SetTextMatrix(i, 8, "ON Critical");
			else
			m_SequenceGrid.SetTextMatrix(i, 8, "OFF Critical");

		//	temp.Format("%8.2lf", pData->EndDay);
		//	m_SequenceGrid.SetTextMatrix(i, 9, temp);

		/*	if(AllFinished)
			temp.Format("%d", ErectionDay(pData->StartDay, pData->EndDay));
			else
			temp.Format("00");
			m_SequenceGrid.SetTextMatrix(i, 8, temp);
*/

	if(AllFinished)
			temp.Format("%d", Period(pData->StartDay, pData->EndDay));
			else
			temp.Format("00");

			m_SequenceGrid.SetTextMatrix(i, 9, temp);
		/*
			temp.Format("%08d",PEWorkEndDay(pData->StartDay, pData->EndDay));
			m_SequenceGrid.SetTextMatrix(i, 8, temp);
*/
			if(AllFinished)
			temp.Format("%d", ProbabilisticErectionDay(pData->StartDay, pData->EndDay));
			else
			temp.Format("00");
			m_SequenceGrid.SetTextMatrix(i, 10, temp);


			temp.Format("%8.2f",(pData->Area/TotalPEArea(pData->StartDay, pData->EndDay)));
			m_SequenceGrid.SetTextMatrix(i, 11, temp);


			i++;
		}
		//////////////////////////////////////////////////////////////////////
}

BOOL CSequenceDlg::OnInitDialog()
{
		CDialog::OnInitDialog();
```

```
// TODO: Add extra initialization here
First = TRUE;
AllFinished = FALSE;

strcpy(StartBlock, "Block Name");

CFileDialog pdlg(TRUE, "PDT", NULL, OFN_HIDEREADONLY, "Pitch file(*.pdt)|*.pdt;|");

if(pdlg.DoModal() == IDOK)
{
        char parent[10];
        char child[10];
        int pitch, NodeNum, i;
        BOOL same;
        FILE *fp;
        CString temp;
        CString fname;

        // First? ////////////////////////////////////////////////////////////
        if(!m_PitchList->IsEmpty())
                m_PitchList->RemoveAll();
        //////////////////////////////////////////////////////////////////////

        // Pitch Data Read ///////////////////////////////////////////////////
        fname = pdlg.GetPathName();

        fp = fopen(fname, "rt");

        fscanf(fp, "%s", temp);                 // Ship Type
        fscanf(fp, "%s", temp);                 // Ship Name
        fscanf(fp, "%s", temp);                 // PercentPEArea
        fscanf(fp, "%d", &NodeNum); // Pitch Number
        fscanf(fp, "%s", temp);                 // Start Day

        for(i=0;i<NodeNum;i++)
        {
                fscanf(fp, "%s %s %s %d", temp, parent, child, &pitch);

                // Pitch Data Generation ////////////////////////////////////
                same = FALSE;

                for(POSITION pos=m_PitchList->GetHeadPosition();pos!=NULL;)
                {
                        CBlockData *pData = (CBlockData*)m_PitchList->GetNext(pos);

                        if(strcmp(pData->Name, parent)==0)
                        {
                                pData->Add(child, pitch);
                                same = TRUE;
                                break;
                        }
                }

                if(!same)
                {
                        CBlockData *pData = new CBlockData;

                        strcpy(pData->Name, parent);
                        pData->Add(child, pitch);

                        m_PitchList->AddTail(pData);
                }
                ////////////////////////////////////////////////////////////
        }

        fclose(fp);
        //////////////////////////////////////////////////////////////////////

        // Block Data Read ///////////////////////////////////////////////////
        CFileDialog bdlg(TRUE, "BDT", NULL, OFN_HIDEREADONLY, "block file(*.bdt)|*.bdt;|");

        if(bdlg.DoModal()==IDOK)
        {
                int i;
                char BlockName[20];
                FILE *fp;
                BOOL same;

                fp = fopen(bdlg.GetPathName(), "rt");

                fscanf(fp, "%s %s %s %d %s", ShipType, ShipName, PercentPEArea, &BlockNum,
StartDay);

                m_strShipType = ShipType;
                m_strShipName = ShipName;// this function gives value to ship name edit block
                m_strStartDay = StartDay;
                m_strPercentPEArea = PercentPEArea;


                UpdateData(FALSE);

                for(i=0;i<BlockNum;i++)
                {
                        fscanf(fp, "%s", BlockName);
```

```
                                        same = FALSE;

                                        for(POSITION pos=m_PitchList->GetHeadPosition();pos!=NULL;)
                                        {
                                                CBlockData *pData =
(CBlockData*)m_PitchList->GetNext(pos);

                                                if(strcmp(pData->Name, BlockName)==0)
                                                {
                                                        fscanf(fp, "%lf %lf %d", &pData->Length,
&pData->Breadth, &pData->PE);
                                                        pData->BlockArea();
                                                        same = TRUE;
                                                        break;
                                                }
                                        }

                                        if(!same)
                                        {
                                                CString BName;
                                                BName.Format("Block(%s) Title Absent !", BlockName);
                                                AfxMessageBox(BName);

                                                double t1, t2;
                                                int t3;

                                                fscanf(fp, "%lf %lf %d", &t1, &t2, &t3);
                                        }
                                }

                                fclose(fp);
                        }
                        ////////////////////////////////////////////////////////////////////////////

                        DisplayGrid();
                }
                else
                {
                        OnCancel();
                }

                return TRUE;  // return TRUE unless you set the focus to a control
                        // EXCEPTION: OCX Property Pages should return FALSE
}

BEGIN_EVENTSINK_MAP(CSequenceDlg, CDialog)
    //{{AFX_EVENTSINK_MAP(CSequenceDlg)
                ON_EVENT(CSequenceDlg, IDC_SEQUENCE_GRID, 71 /* EnterCell */, OnEnterCellSequenceGrid, VTS_NONE)
                //}}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()

void CSequenceDlg::OnEnterCellSequenceGrid()
{
                // TODO: Add your control notification handler code here


}


void CSequenceDlg::OnPeTimeIn()
{
                // TODO: Add your control notification handler code here
                        // TODO: Add your control notification handler code here
                CBlockENTDlg dlg;

                int i, tempsize;
                FILE *fp;
                BOOL same;
                UpdateData(TRUE);
                dlg.m_strStartBlock = "Block Name";
                dlg.m_strStartDay = m_strStartDay;
                if(dlg.DoModal()!=IDOK)
                        return;
                strcpy(StartBlock, dlg.m_strStartBlock);
                Start = atoi(dlg.m_strStartDay);

                same = FALSE;

                for(POSITION pos=m_PitchList->GetHeadPosition();pos!=NULL;)
                {
                        CBlockData *pData = (CBlockData *)m_PitchList->GetNext(pos);

                        if(strcmp(StartBlock, pData->Name)==0)
                        {
                                same = TRUE;
                                break;
                        }
                }

                if(!same)
                {
                        CString BName;
```

```
                BName.Format("Can't find this Block(%s) please check !", StartBlock);
                AfxMessageBox(BName);
                return;
        }
        First = TRUE;

        hNum = 1;

        House = new CArray [hNum];
        strcpy(House[0].Name, StartBlock);
        while(TRUE)
        {
                tNum = GetChild(House[0].Name);

                // Save ////////////////////////////////////////////////////////
                fp = fopen("GoCheck.txt", "at");

                fprintf(fp, "%d %d - ", hNum, tNum);
                for(i=0;i<hNum;i++)
                {
                        fprintf(fp, "%s ", House[i].Name);
                }
                fprintf(fp, "\n");

                fclose(fp);
                ////////////////////////////////////////////////////////////////

                // Escape //////////////////////////////////////////////////////
                if((hNum + tNum -1) == 0)
                {
                        break;
                }

                if(tNum == -1)
                {
                        AfxMessageBox("Can't Find this Block Name!");
                        break;
                }
                ////////////////////////////////////////////////////////////////

                tempsize = hNum + tNum - 1;
                Temp = new CArray [tempsize];

                if(hNum==1)
                {
                        for(i=0;i<tempsize;i++)
                        {
                                strcpy(Temp[i].Name, Person[i].Name);
                        }
                }
                else
                {
                        if(tNum == 0)
                        {
                                for(i=1;i<hNum;i++)
                                {
                                        strcpy(Temp[i-1].Name, House[i].Name);
                                }
                        }
                        else
                        {
                                for(i=1;i<hNum;i++)
                                {
                                        strcpy(Temp[i-1].Name, House[i].Name);
                                }

                                for(i=hNum;i<tempsize+1;i++)
                                {
                                        strcpy(Temp[i-1].Name, Person[i-hNum].Name);
                                }
                        }
                }

                delete [] House;

                hNum = tempsize;
                House = new CArray [hNum];

                for(i=0;i<hNum;i++)
                {
                        strcpy(House[i].Name, Temp[i].Name);
                }

                delete [] Temp;

                if(tNum != 0)
                        delete [] Person;
        }

        delete [] House;

        // Reload Grid/// ////////////////////
        DisplayGrid();
        ////////////////////////////////////
}
```

```
void CSequenceDlg::OnTimeOut()
{
	// TODO: Add your control notification handler code here
		// TODO: Add your control notification handler code here
	CBlockLNTDlg dlg;

	int i, tempsize;
	FILE *fp;

	dlg.m_strEndBlock = EndBlock;
	dlg.m_nEndDay = End;

	if(dlg.DoModal()!=IDOK)
		return;

	strcpy(StartBlock, dlg.m_strEndBlock);

	First = TRUE;

	hNum = 1;

	House = new CArray [hNum];
	strcpy(House[0].Name, StartBlock);

	while(TRUE)
	{
		tNum = GetParent(House[0].Name);

		// Save ////////////////////////////////////////////////////////////
		fp = fopen("BackCheck.txt", "at");

		fprintf(fp, "%d %d - ", hNum, tNum);
		for(i=0;i<hNum;i++)
		{
			fprintf(fp, "%s ", House[i].Name);
		}
		fprintf(fp, "\n");

		fclose(fp);
		////////////////////////////////////////////////////////////////////

		// Escape //////////////////////////////////////////////////////////
		if((hNum + tNum -1) == 0)
		{
			break;
		}

		if(tNum == -1)
		{
			AfxMessageBox("Can't find this Block Name!");
			break;
		}
		////////////////////////////////////////////////////////////////////

		tempsize = hNum + tNum - 1;
		Temp = new CArray [tempsize];

		if(hNum==1)
		{
			for(i=0;i<tempsize;i++)
			{
				strcpy(Temp[i].Name, Person[i].Name);
			}
		}
		else
		{
			if(tNum == 0)
			{
				for(i=1;i<hNum;i++)
				{
					strcpy(Temp[i-1].Name, House[i].Name);
				}
			}
			else
			{
				for(i=1;i<hNum;i++)
				{
					strcpy(Temp[i-1].Name, House[i].Name);
				}

				for(i=hNum;i<tempsize+1;i++)
				{
					strcpy(Temp[i-1].Name, Person[i-hNum].Name);
				}
			}
		}

		delete [] House;

		hNum = tempsize;
		House = new CArray [hNum];

		for(i=0;i<hNum;i++)
		{
```

```
                                        strcpy(House[i].Name, Temp[i].Name);
                        }

                        delete [] Temp;

                        if(tNum != 0)
                                delete [] Person;
                }

                delete [] House;

                AllFinished = TRUE;

                // Reload Grid /////
                DisplayGrid();
                ////////////////////

                AllFinished = FALSE;
}

int CSequenceDlg::PEWorkStartDay(int StartDay, int EndDay)
{

                int temp;
                int count;
                int t;

                temp = StartDay;
                count = 0;

/*              if(StartDay == EndDay)
                        return StartDay;
/*                              for(POSITION pos=m_PitchList->GetHeadPosition();pos!=NULL;)
                {

                CBlockData *pData = (CBlockData *)m_PitchList->GetNext(pos);

                 t= pData->PE;

                        temp = ChangeDay(TRUE, temp, t);


                if(temp == EndDay)
                        break;
                return temp;
                }*/

                while(TRUE)
                {
                        for(POSITION pos=m_PitchList->GetHeadPosition();pos!=NULL;)
                {

                CBlockData *pData = (CBlockData *)m_PitchList->GetNext(pos);

                 t= pData->PE;

                        temp = ChangeDay(TRUE, temp, t);
                //              count++;

                if(temp == EndDay)
                        break;
                return temp;
                }

}

}

int CSequenceDlg::PEWorkEndDay(int StartDay, int EndDay)
{

                int temp;
                int count;
                int t, tt;
                int temp2;

                temp = StartDay;
                count = 0;

                if(StartDay == EndDay)
                        for(POSITION pos=m_PitchList->GetHeadPosition();pos!=NULL;)
                {

                CBlockData *pData = (CBlockData *)m_PitchList->GetNext(pos);

                 t= pData->PE;

                        temp = ChangeDay(TRUE, temp, t);
                //              count++;

                if(temp == EndDay)
                        break;
                return temp;
                }
```

```
		while(TRUE)
		{
			temp = ChangeDay(TRUE, temp, 1);

			count++;

		if(temp == EndDay)
			break;

		}

		t= count;

		int PE;

		for(POSITION pos=m_PitchList->GetHeadPosition();pos!=NULL;)
		{

		CBlockData *pData = (CBlockData *)m_PitchList->GetNext(pos);

		PE= pData->PE;

		tt= t - PE;

		for (int i= 1; i<=t; i++)

		{
		temp2 = StartDay;
		temp2 = ChangeDay(TRUE, temp2, tt);
		}
		return temp2;
		}
}

int CSequenceDlg::ProbabilisticErectionDay(int StartDay, int EndDay)
{

		int temp, temp2;
		int count;
		int t, to,tp,tm, te;


		temp = StartDay;
		count = 0;

		if(StartDay == EndDay)
		return StartDay;

		while(TRUE)
		{
			temp = ChangeDay(TRUE, temp, 1);

			count++;

		if(temp == EndDay)
			break;
		}

		tp = count+1;
		to = 1;


		tm = ((double)(rand()%1000)/1000)*(tp-to) + to;

		te = ((tp+to+(4*tm))/6);
		temp2 =	StartDay;
		for (int i= 1; i<=tp; i++)
		{

		temp2 = ChangeDay(TRUE, temp2, te);


		}
		return temp2;

}

float CSequenceDlg::TotalPEArea(float StartDay, float EndDay)
{
		int temp,t;
		float count;
		count = 0;

for(POSITION pos=m_PitchList->GetHeadPosition();pos!=NULL;)
{

CBlockData *pData = (CBlockData*)m_PitchList->GetNext(pos);

count = count +  pData->Area;

}

return count;
}
```

Acknowledgements

저를 있게 해주시고 지금까지 끊임없이 희생만 하시면서 저를 키워주신 어머님
언제나 밤낮을 가리지 않고 화학공단에서 땀을 흘리신 제가 그 누구보다
존경하는 아버님의 노고에 부족한 아들이 머리 숙여 제일 먼저 감사의 말을
들이고 싶습니다. 그리고 저를 학부 때부터 지금까지 못난 제자지만 꿋꿋이
인내로 지도해주시고 논문을 모든 것을 가르쳐 주신 윤덕영교수님께
감사들입니다. 교수님께서 학문뿐만 아니라 제가 힘들때나 기쁠때모두 저에게
힘이 되어주셨습니다. 그리고 선박에 대하여 너무나 재미있게 가르쳐주신
박제웅 교수님과 이귀주 교수님, 언제나 방에 불러서 고민을 들어주셨던 권영섭
교수님, 역학에 대하여 핵심을 가르쳐주신 방한서 교수님께 감사의 말씀을
드리고자 합니다. 과 교수님들 모두 이 논문의 심사와 완성까지의 모든 과정을
도와주셔서 논문이 완성될 수 있었습니다. 대우에서 열심히 일하시고 계시고 학
부때 너무나 많이 도움을 주신 경원이형 그리고 4년간이나 공부했던 진동소음실
험실 재형이 용접실험실에 재선이, 준기, 호경, 은영 과 CAD/CAM 실 멤버들과
유체실험실 멤버, 선박안전실 실험실멤버들에게도 감사의 인사를 드립니다.
그리고 동기면서 언제나 돌봐주신 인식형, 이리저리같이 다니신 남휘형, 같은
실험실에 현호형 에게도 같이 부대끼면서 정 많이 들었습니다. NURI & BK 사무
실에 친누나처럼 잘 돌봐주신 현정이 누나, 대 선배님이신 자희 누나, 후배면
서 무서운 오해진양, 이것저것 너무나 도움을 많이 받았던 김정미양, 그리고 친
구라서 너무나 힘이 돼어주었는데 그에 비하여 내가 많이 괴롭혔던 친구 박정미
모두다 너무나 고맙고 감사합니다. 항상 힘들때마다 실험실에 와서 힘이되어준
종현이형과 주연씨 얼른 결혼하세요. 그리고 준기형, 민환이형, 재훈이형 도 잊
지 못할 껍니다. 그 많은 이모들과 이숙들 그리고 외할머니, 사촌동생들, 친가
식구들 그리고 친동생 효주. 제겐 푸근한 곳이었습니다. 그리고 제 신앙과 같은
그 사람에게도 비록 같이 있지는 못했지만 내가 나태해질 때마다 생각나게 하여
서 열심히 해주 그 사람에게도 고개 숙여 감사들입니다.

마무리 못할 것 같았던 논문이고 교수님의 가르침에 비하여 너무나 보잘것없지만 이제 완성을 하였고 학부를 졸업하면서 선박해양이란 전공에 지식의 자만심을 가지고 있던 저에게 제가 얼마나 많이 부족한지를 얼마나 모르는 것이 많이 있다는 것을 알았다는 과정 이였다고 것을 깨달은 것만으로도 충분히 만족할만한 시간 이였다고 생각됩니다. and then Several people have directly affected this book, but none more than Ranjan Varghese, who patiently read the manuscript at each juncture, always improving the prose and the sense.

To Brain Korea21 and Nuri Shipbuilding Engineers Training and Education center I owe special thanks for the help I have received with the copying of the various versions of the manuscript and other technical assistance.