ABSTRACT

Title of Document: CONTAINERSHIP LOAD PLANNING WITH CRANE OPERATIONS

Masoud Hamedi, Doctor of Philosophy, 2010

Directed By: Professor Ali Haghani, Department of Civil & Environmental Engineering

Since the start of the containerization revolution in 1950’s, not only the TEU capacity of the vessels has been increasing constantly, but also the number of fully cellular container ships has expanded substantially. Because of the tense competition among ports in recent years, improving the operational efficiency of ports has become an important issue in containership operations. Arrangement of containers both within the container terminal and on the containership play an important role in determining the berthing time. The berthing time of a containership is mainly composed of the unloading and loading time of containers. Containers in a containership are stored in stacks, making a container directly accessible only if it is on the top of one stack. The task of determining a good container arrangement to minimize the number of re-handlings while maintaining the ship’s stability over several ports is called stowage planning, which is an everyday problem solved by ship planners.

The horizontal distribution of the containers over the bays affects crane utilization and overall ship berthing time. In order to increase the terminal productivity and reduce the turnaround time, the stowage planning must conform to the berth design. Given the configuration of berths and cranes at each visiting port, the stowage planning must take into account the utilization of quay cranes as well as the reduction of unnecessary shifts to minimize the total time at all ports over the voyage. This dissertation introduces an optimization model to solve the stowage planning problem with crane utilization considerations. The optimization model covers a wide range of operational and structural constraints for containership load planning.

In order to solve real-size problems, a meta-heuristic approach based on genetic algorithms is designed and implemented which embeds a crane split approximation routine. The genetic encoding is ultra-compact and represents grouping, sorting and assignment strategies that might be applied to form the stowage pattern. The evaluation procedure accounts for technical specification of the cranes as well as the crane split. Numerical results show that timely solution for ultra large size containerships can be obtained under different scenarios.

CONTAINERSHIP LOAD PLANNING WITH CRANE OPERATIONS

By

Masoud Hamedi

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2010

Advisory Committee:
Professor Ali Haghani, Chair
Professor Paul M. Schonfeld
Professor Gang-Len Chang
Assistant Professor Cinzia Cirillo
Associate Professor Philip T. Evers

Dedication

To my wife Nina, the love of my life,

and to my daughter Mana, the joy of our lives

Acknowledgements

During my long years at the University of Maryland I had the opportunity to work and collaborate with many wonderful people that without their help and support this journey would not have been possible. My deepest and sincerest thanks is devoted to my advisor, professor Ali Haghani, for his guidance, advice, support, and especially his extraordinary patience and trust in me. I would also like to thank my dissertation committee members for all their helpful guidance and discussions throughout the course of my research.

I am grateful to all my colleagues in the graduate research group for their friendship as well as their encouraging and constructive critiques especially in seminars and presentation dry runs.

Last but not least, I am deeply grateful to my parents, my family and my wife. My parents made many sacrifices to ensure I get the best possible education and lovingly guided me every step of the way. My wife is the most important reason in the success of my PhD. She has provided me with unconditional love, support, and understanding. I have been very lucky, beyond anything I could have wished for, to have her by my side. It would be impossible for me to express my gratitude towards her and my parents in mere words.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1. Containerization

Containerization is an intermodal transportation system in which the containers as cargo units can be loaded on containerships, railroad cars and trucks without handling the contents. Containerization revolution that began in 1950s increased ocean carries productivity dramatically. Prior to containerization all goods other than bulk cargo were carried in break bulk format. Pieces of cargo were loaded one by one into trucks and at the marine port they were unloaded and loaded into the hold of a ship. At the destination individual pieces were unloaded and put on truck or train for delivery. In addition to inefficiency cargo was exposed to potential damage. Loading numerous pieces of cargo into a standard sealed metal box carried by truck or train to the seaport where it would be lifted and stored aboard ship sped up the process. At the destination the process would be reversed. The simple solution improved the delivery time, decreased transportation cost and made intermodal transportation far more feasible. On April 26th 1956 Ideal-X, the first cargo ship carrying containers left the port of Newark to the Port of Houston (Cudahy 2006). Soon the concept of containerization proved to be faster, safer and cheaper than the existing methods. By making the exchange of commodities easier it opened new markets for import and export.

The vast majority of international trade travels by ship and over that past two decades, container utilization has grown dramatically, helping the idea of a global intermodal economy. “Today over 60% of the world’s deep sea general cargo is transported in containers whereas some routes, especially between economically strong and stable countries, are containerized up to 100%” (Steenkan et al. 2004). The globalization would have been impossible without containers. Because the demand for transportation is different across the ports, different sizes of containerships have been designed.

Container handling technology has also evolved over the years, not only in terms of size, type and capacity but also in ways that the containers are moved. In the beginning, ships were equipped with cranes to load and unload the containers themselves or traditional shore equipment was used. However as the container revolution went on, specialized equipment was developed allowing for a faster handling of containers. Nowadays, automated guided vehicles and cranes are in use in some ports to handle containers. The port of Rotterdam was the first one to develop and use this technology (Ben-Jaap 2005).

1.2. Container port terminal

A container terminal is the interface between land side and the quayside transshipment of the containers. Import containers arrive by containerships at the terminal where they are stored temporarily before being loaded onto the ground modes of transportation i.e. trains or trucks and dispatched to their final destination. The export containers arrive by rail or truck and are stored in a similar manner before being loaded to the ship and leave the port. Transshipment containers on the other hand are unloaded from the ship and stored in the yard, but eventually leave the port on a different containership. The container terminal is the point at which containers change their mode of travel.

Container terminals can be looked at as three relatively independent subsystems.

- Quay-side interface
- Storage yard
- Land-side interface

Figure 1.1 shows a schematic of a container terminal system.

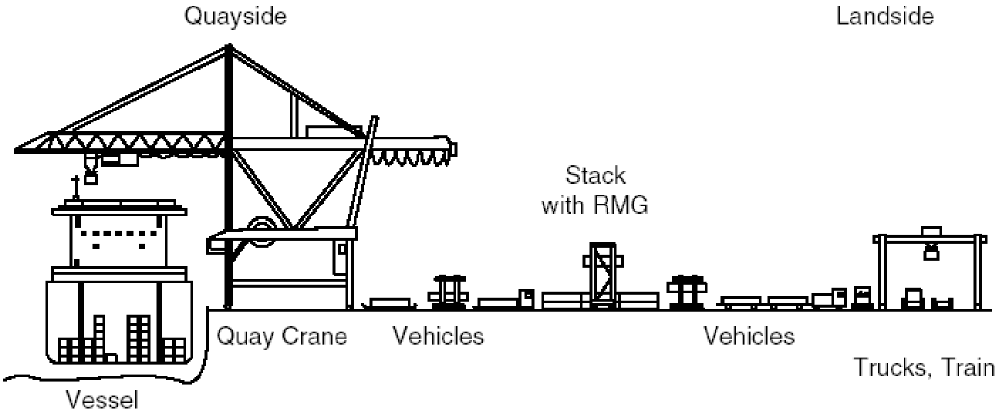


Figure 1.1: General schematic of a container terminal (Steenkan et al. 2004)

There are several decisions to be made in order to create a smooth and efficient flow of the containers in the system. The major tactical decision makers are terminal managers and the ship planners while at the operational level decisions might be made by crane operators or straddle carriers drivers. The hierarchy and timeline of the decisions for incoming and outgoing ships are depicted in figure 1.2.

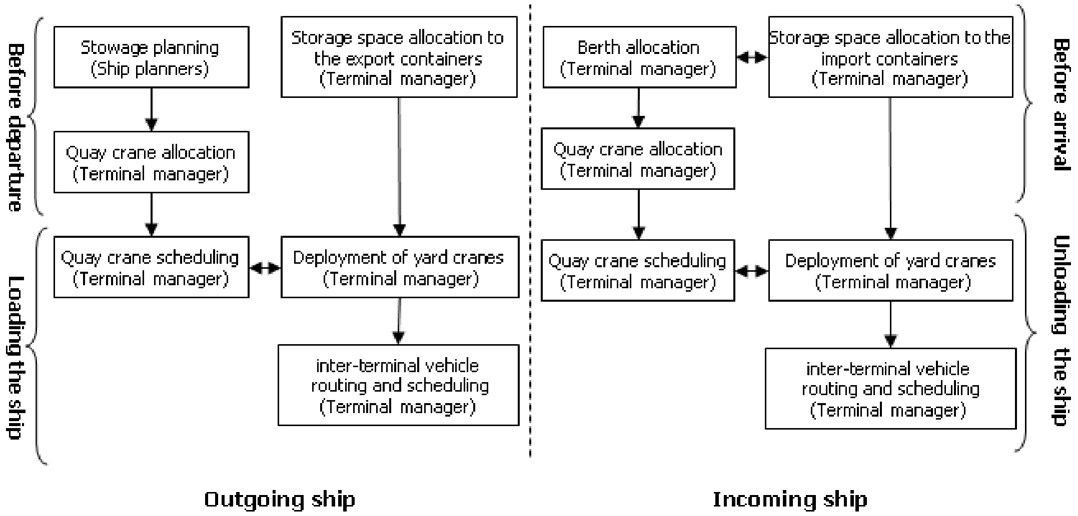


Figure 1.2: Hierarchy of the main decisions for incoming and outgoing ships

1.2.1. Quay-side interface

Before a ship calls for a port, her specifications and berth requirements as well as estimated time of arrival will be transmitted to the port. Based on the availability of the berths and the schedules of the incoming vessels, the terminal operators allocate a berth, berthing time and other resources (i.e. quay cranes) to the vessel. Although the shipping lines expect prompt berthing upon arrival, this might not be possible due to the limitation of the wharfs and congestion at the sea side of the port. Vessels of the priority customers might be granted berth-on arrival service if they have contracts with terminal operators.

After the containership docks at one of the available berths of the port, the containers will be loaded to, and unloaded from the containership using quay cranes. Quay cranes are the most expensive pieces of equipment at the container terminal and play a crucial role in loading and unloading operations. In the case of fully cellular containerships where no cranes are mounted on the vessel, quay cranes are the only means of moving containers to/from the ship. Quay cranes are rail mounted and can move horizontally alongside the ship. The required time for moving a crane from one bay to the next depends on the type of the equipment and the underlying technology, but typically it is on the order of a one container move which ranges from one to three minutes. The crane operator uses the spreader arm to handle the container to/from the vessel. Maximum performance of the quay cranes depends on the crane type. While the technical performance is in the range of 50-60 box/hr, the operational performance is in the range of 22-30 box/hr (Steenken et. al. 2004). A quay crane typically has four legs. The space between the legs accommodates up to five truck lanes which is used by the internal trucks. Trucks stop in the lanes under the cranes to either feed the export containers to the crane or take the import containers from the crane and transport them to the yard. Import containers might directly leave the port by road or rail without going to the storage yard; however this is not very common.

1.2.2. Storage yard

Storage yard is an intermediate system between the quay and the land side of the port system. Both import and export containers are stacked in the columns at the storage yard. The yard handling equipment retrieve the export containers within the yard and take them to the quay cranes and bring back the import containers unloaded at the berth for storage in the yard.

There are two types of storage yards: those that stack the containers on the ground and those that store the containers on chassis. Although the containers on chassis can be retrieved and moved quickly and easily, this option is only available to the ports that do not have space limitations. When the land becomes expensive or simply unavailable and the flow of the containers grows rapidly, similar to the situation in major Asian ports, stacking becomes the only viable option. Saving the space by using stacks comes at the price of increased time and effort for accessing the containers. The yard area in this case is divided into different blocks with several rows and tiers at each block. Some blocks or stack sections are reserved for special containers such as reefer containers which need electric plugs, hazardous cargo or overweight containers. Usually yard managers do not mix the import and export containers and store them in separate stacks. This is because the import containers arrive in the yard, in large batches but leave the port one by one in random order. On the contrary, the departure of export containers is predictable but their arrival happens in a random order. Empty containers also are stacked in different sections where they can be stacked higher than the normal containers.

There are several types of equipment for handling and transporting the containers within the storage yard. Straddle carriers are individual independent units that are capable of both lifting and transporting standard containers. When the maximum storage density at the container yard is required, a combination of Rubber Tyre Gantry Cranes (RTCG) and trucks is usually preferred to straddle carriers. Each storage block in this case consists of several rows of containers and a truck lane. RTCG's are caple of lifting a container from the truck waiting in the truck lane and store it in the stack or retrieve an outbound container and put it on the truck. They are very expensive equipment and planning for their proper utilization is crucial to the throughput of the yard. RTCG' are not fixed within the block and may move to adjacent blocks, their movement however is slow and it is even slower if making a 90^0 turn for reaching the adjacent block is necessary.

When using straddle carriers no other vehicles are necessary for horizontal transportation of the containers within the storage yard. On the other hand when RTGC's are employed, trucks with trailer, multi-trailers or Automatic Guided vehicles (AGV) are needed for moving the containers. AGV's are computer controlled robots which operate on a grid of pre-designed wired routes with sensors and transponders. The deployment of AGV's is driven by economic reasons where the labor costs are high. Although they call for high investment they are already in operation at ECT/Rotterdam and at the HHLA/Hamburg in combination with automatic gantry cranes (Steenkan et. al. 2004). Figure 1.3 shows an aerial photo of the port of Rotterdam.

Figure 1.3: Port of Rotterdam in Netherlands (www.zpmc.com)

1.2.2. Land-side interface

The land-side interface is the transshipment point between the storage yard and the inland transportation system which can be truck, rail or both. The landside operation starts at gates where two main activities happen: export delivery and import receiving. Export delivery begins with checking the documentation and inspection of the containers which is brought in by freight forwarders. A storage location will be assigned to the container and the truck will be routed to the destination area at the yard where the container will be lifted and stored. Import receiving process initiates by a request from the customer at the gate. The location of the container then is reported by the computer system and the truck will be guided to the specific yard area to load the container.

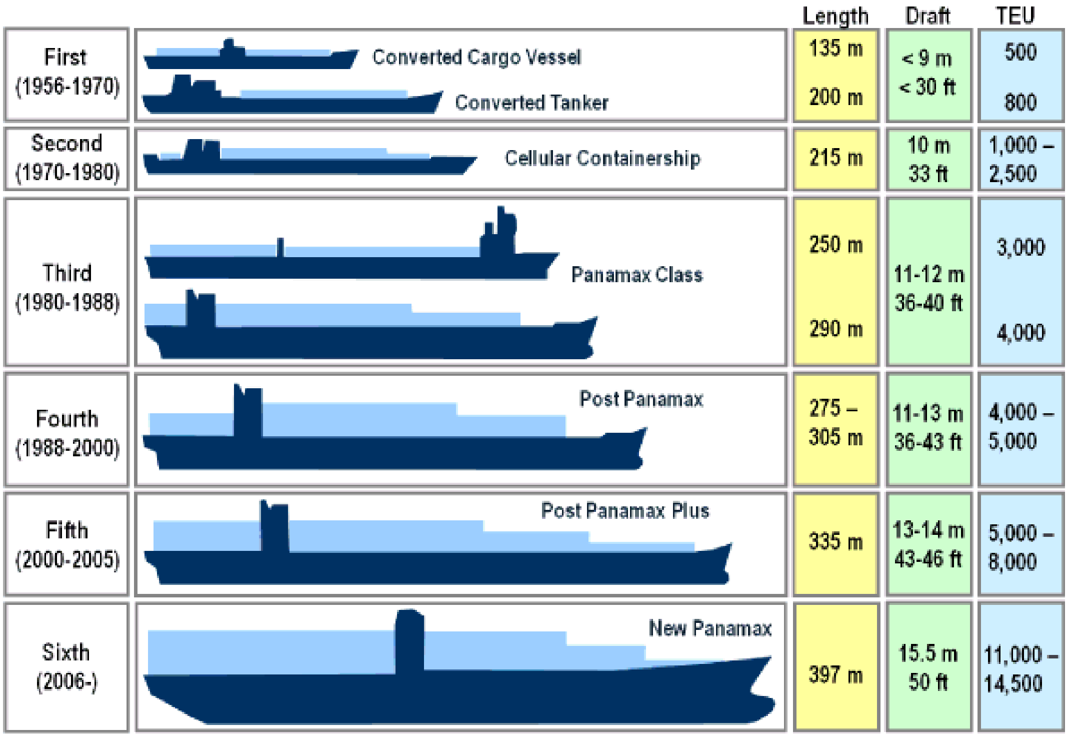
To maximize the throughput of the port system and to avoid congestion, the processes of the above subsystems must be synchronized and optimized.

1.3. The containership

The size of a containership is normally stated as the number of TEU sized containers that it can carry (TEU is the abbreviation for twenty foot equivalent unit which is the standard container size by International Organization for Standardization). The first containerships were built by modifying bulk vessels in order to accommodate containers. These ships had their own on board cranes to handle the containers. As containers became more popular in 1970's, a new generation of the fully cellular containerships were introduced to the market. On board cranes were removed from these vessels so they had more space to dedicate to the stack of containers. Until the mid 1980’s containership size was limited by the dimensional constraint of the Panama canal. The economies of scale (increased capacity at higher speeds with lower costs per TEU) encouraged ship builders to design larger vessels until the Panama Canal limit of 13 containers across a 32.2m wide deck was reached. The result was a new generation of containerships known as post Panamx that started in 1988. Figure 1.4 shows different generations of the containerships and Figure 1.5 shows the maximum containership size by the year of build as well as the projected trend of the size growth.

Since then, the development of the post-Panamax fleet has been dramatic. According to the Lloyd’s register fact sheets (Lloyd’s 2003) the world post-Panamax container fleet has risen to 25% of the total containership fleet by capacity in 2003 and with the current trend a jump to 58% is expected. The new Panamax vessels will fit the third line of docks of the Panama Canal which will be operational in 2014 (Rodrigue et al. 2009).

United Nations reports that average carrying capacity per ship for the world containership fleet has increased from 3,489 TEUs in 2008 to 4,016 TEUs in 2010 as a result of building larger vessels to achieve economies of scale. Data shows that well-defined trend towards large container vessels is continuing unabated. The largest fully cellular vessel in early 2010 had a nominal capacity of 14,770 TEU. The largest vessels delivered in 2009 were two 13,880 TEU ships for CMA CGM shipping lines (UNCTAD 2010).



		Length	Draft	TEU
First (1956-1970)	Converted Cargo Vessel Converted Tanker	135 m 200 m	< 9 m < 30 ft	500 800
Second (1970-1980)	Cellular Containership	215 m	10 m 33 ft	1,000 – 2,500
Third (1980-1988)	Panamax Class	250 m 290 m	11-12 m 36-40 ft	3,000 4,000
Fourth (1988-2000)	Post Panamax	275 – 305 m	11-13 m 36-43 ft	4,000 – 5,000
Fifth (2000-2005)	Post Panamax Plus	335 m	13-14 m 43-46 ft	5,000 – 8,000
Sixth (2006-)	New Panamax	397 m	15.5 m 50 ft	11,000 – 14,500

Figure 1.4: Containership sizes (Source: The geography of transport systems)

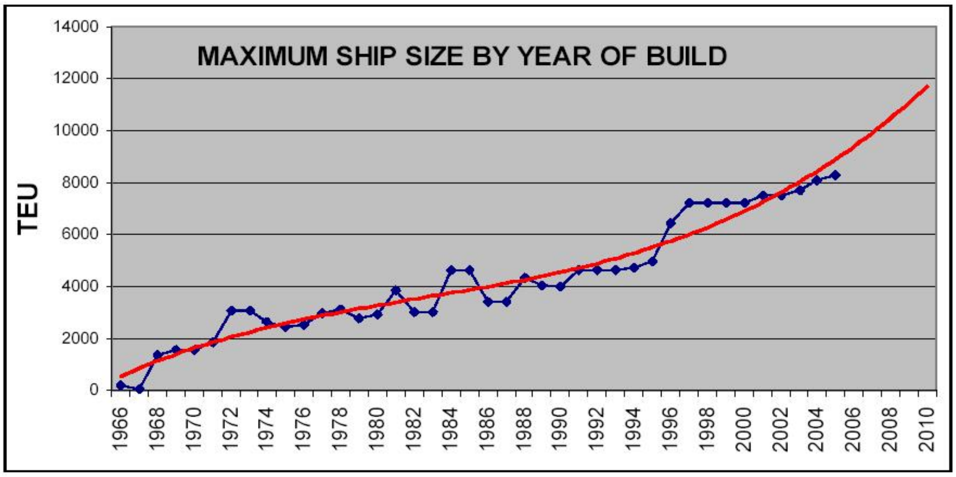
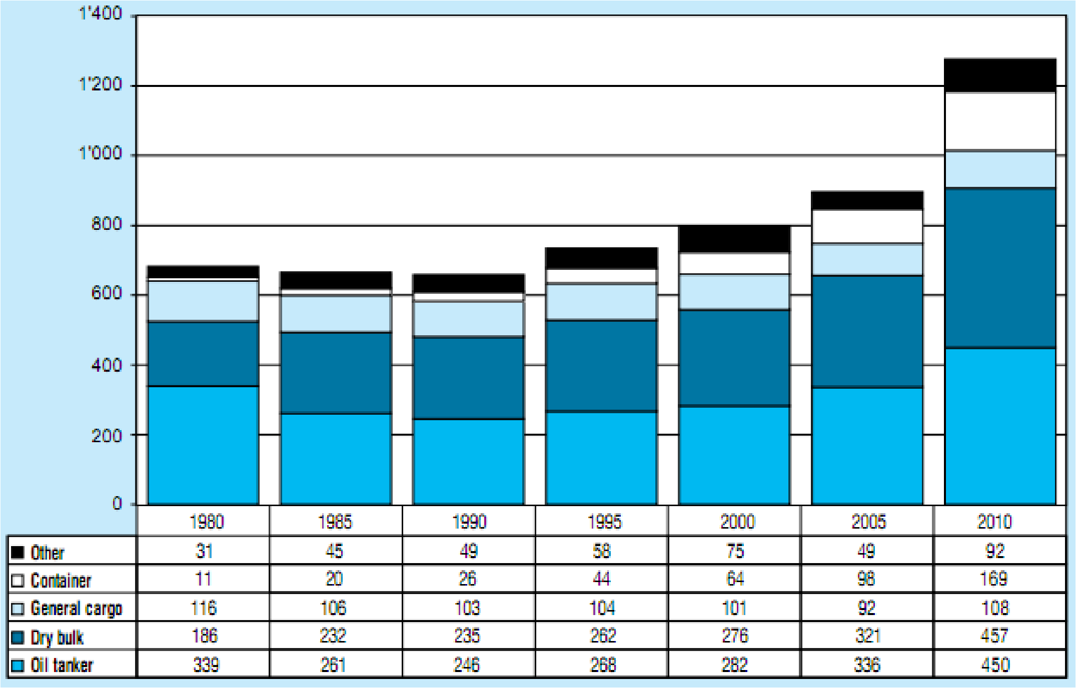


Figure 1.5: Maximum containership size by the year of build (Lloyd’s 2003)

Not only the TEU capacity of the vessels has increased, but also the number of fully cellular containerships has expanded substantially. Studies show that by the beginning of 2010 there were 4,677 ships with a combined total capacity of 12.8

million TEUs (UNCTAD 2010). Overall there was an increase from of 8.9 percent in the number of ships and 12.9 percent in TEU capacity over the previous year. Figure 1.6 demonstrates the world fleet by principal vessel types for selected years.



	1980	1985	1990	1995	2000	2005	2010
Other	31	45	49	58	75	49	92
Container	11	20	26	44	64	98	169
General cargo	116	106	103	104	101	92	108
Dry bulk	186	232	235	262	276	321	457
Oil tanker	339	261	246	268	282	336	450

Figure 1.6: World fleet by principal types of vessel (UNCTAD 2010)

Containerization has revolutionized cargo shipping. Today, approximately 90% of non-bulk cargo worldwide moves by containers stacked on transport ships; 26% of all containers originate from China. As of 2005, some 18 million total containers make over 200 million trips per year (Levinson 2006).

Samsung Heavy Industries compared the cost of ship building between two 6200TEU vessels and one 12000TEU. The results suggest approximately 16% reduction in costs by building the latter vessel rather than the formers. The fuel cost per TEU for

12000TEU vessel compared to two 6200TEU is also approximately 17% lower (Yang 2004).

Today deployment of 15,000 TEU vessels on the routes between both east and west coast of North America and Southeast Asia is justified by the economies of scale. The containership "Emma Maersk" which is 396 m long was launched in August 2006. A study conducted at the Delft University (Wijnolst et al 1999) suggests that the maximum size for future containerships would be 18,000 TEU with a draft of 21m, given depth restrictions in the Malacca Strait which is a major shipping route between Europe and Asia. According to Levinson (2006) this so-called Malaccamax size constrains a ship to dimensions of 470 m in length and 60 m wide (1542 feet * 197 feet).

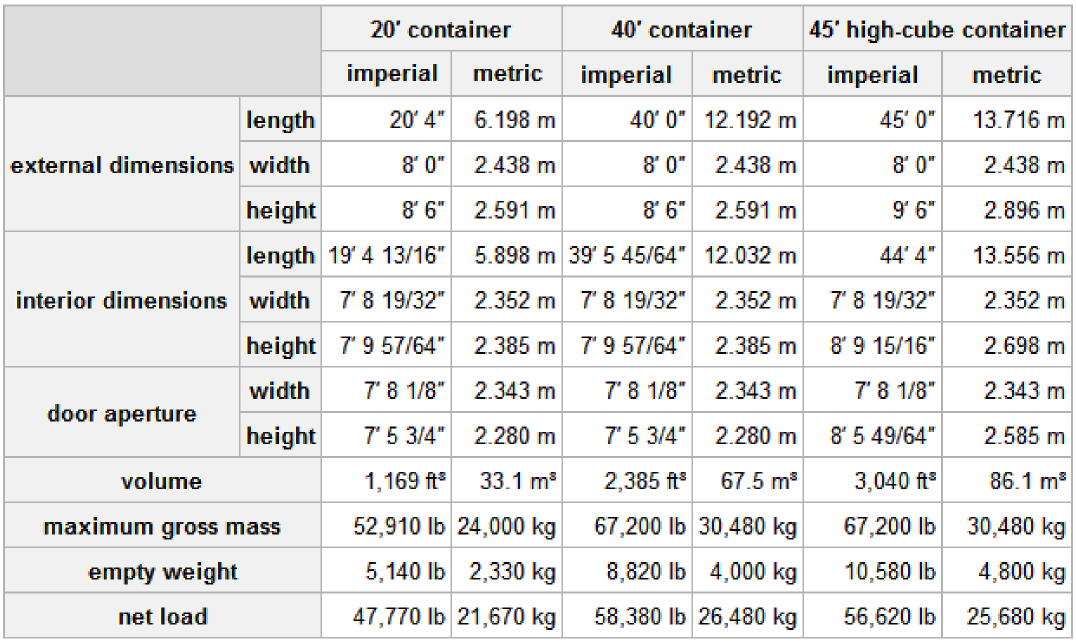
1.4. The container

Containers are large metal boxes used to transport commodities from one destination to another. The dimensions of the containers have been standardized. The term TEU[1] is used to refer to one container with a length of twenty feet, so a container of 40 ft is expressed by 2 TEU. Although the 20 foot containers are the most popular, the 40 footer are increasingly replacing them particularly since costs tend to be per container rather than per foot. The longer container types are also becoming more popular as the shorter containers (e.g. 10 foot containers) are rarely used. Table 1.1 shows the dimensions and weights for the three most common container types worldwide.

---

[1] Twenty feet equivalent unit

Table 1.1: Average dimensions and weights of popular container types[2]



		20′ container		40′ container		45′ high-cube container	
		imperial	metric	imperial	metric	imperial	metric
external dimensions	length	20′ 4″	6.198 m	40′ 0″	12.192 m	45′ 0″	13.716 m
	width	8′ 0″	2.438 m	8′ 0″	2.438 m	8′ 0″	2.438 m
	height	8′ 6″	2.591 m	8′ 6″	2.591 m	9′ 6″	2.896 m
interior dimensions	length	19′ 4 13/16″	5.898 m	39′ 5 45/64″	12.032 m	44′ 4″	13.556 m
	width	7′ 8 19/32″	2.352 m	7′ 8 19/32″	2.352 m	7′ 8 19/32″	2.352 m
	height	7′ 9 57/64″	2.385 m	7′ 9 57/64″	2.385 m	8′ 9 15/16″	2.698 m
door aperture	width	7′ 8 1/8″	2.343 m	7′ 8 1/8″	2.343 m	7′ 8 1/8″	2.343 m
	height	7′ 5 3/4″	2.280 m	7′ 5 3/4″	2.280 m	8′ 5 49/64″	2.585 m
volume		1,169 ft^3	33.1 m^3	2,385 ft^3	67.5 m^3	3,040 ft^3	86.1 m^3
maximum gross mass		52,910 lb	24,000 kg	67,200 lb	30,480 kg	67,200 lb	30,480 kg
empty weight		5,140 lb	2,330 kg	8,820 lb	4,000 kg	10,580 lb	4,800 kg
net load		47,770 lb	21,670 kg	58,380 lb	26,480 kg	56,620 lb	25,680 kg

1.5. Containerization challenges

The global demand for containerized services has shown an increasing trend for the past decade. Table 1.2 shows double digit percentages of growth in container flow for Trans-Pacific and Asia-Europe routes between the years 2003-2004. The flow estimates used in this table are based on UNCTAD 2010 report. The decline of container flow in 2009 is attributed to the global recession which reduced the flow of containers from Asia to the United States and Europe.

---

[2] Container Handbook

Table 1.2: Estimated cargo flow along major trade routes (millions of TEU)

	Trans-Pacific		Asia-Europe		Transatlantic		
	Asia-US	**US-Asia**	**Europe-Asia**	**Asia-Europe**	**US-Europe**	**Europe-US**	**Total**
2009	11.5	6.9	5.5	11.5	2.5	5.3	43.2
2008	14.5	5.6	10.5	16.7	2.9	4.3	54.5
2007	15.2	5.0	10.1	17.2	2.7	4.5	54.7
2006	15.0	4.7	9.1	15.3	2.5	4.4	51
2005	12.4	4.4	5.5	10.8	2.1	3.8	39
2004	10.2	4.2	5.2	8.9	1.7	3.2	33.4
2003	8.8	4.1	4.9	7.3	1.7	2.9	29.7
2002	7.2	3.9	4.2	6.1	1.5	2.6	25.5
2001	5.6	3.9	4.0	5.9	2.7	3.6	25.7
2000	5.2	3.3	3.6	4.5	2.2	2.9	21.7

This ongoing growth has put an enormous pressure on ports and terminal operators to increase productivity in order to handle all these containers in a fast and smooth way. To maintain rapid dwell time for large vessels, ports need to invest in high speed container handling equipments and to accommodate large volumes of containers per vessel, expansion of landside storage facilities is necessary. Currently mega containerships can be served from one side of the vessel. Developing new berthing systems such as indented berths which make it possible to handle the containers from both sides of the ship can also speed up the process.

The key factors that shipping lines use to choose among competitive ports include handling cost per TEU, ship dwell time (total time spent at port), performance of quay cranes, availability of the berths and the interface to the landside intermodal transportation system. To meet the challenges of today's competitive market it is crucial for each port to invest in state-of-the-art technologies and optimize the utilization of its expensive and limited facilities.

One of the major contributing factors in ship turnaround time is the pattern that the containers are stowed in the vessel. In general, a containership calls at a number of ports on her route and in each port, containers are unloaded and loaded. The containers will be stored in stacks that are only accessible from the top. In a favorable stowage pattern containers will be assigned to the positions in the ship such that the overall stability of the ship is maintained and the number of unnecessary movements is minimized. Unfavorable movements appear if at a certain port, containers have to be unloaded and reloaded again, since they are stored on top of containers destined for that port. Reducing the overall ship turnaround time by optimizing the stowage pattern and maximizing the utilization of the quay cranes during the load/unload process is the subject of this research.

1.6. Environmental issues of containerships

Stability of the vessel is a very important factor in cargo safety and maneuverability. Improper distribution and inadequate trimming of the containers causes horizontal as well as vertical imbalances to the vessel whether she is fully or partially loaded. One of the solutions to this problem is using ballast water to stabilize the vessel. Ballast in general is any material used to balance an object e.g. sandbags used to balance hot air balloons. Modern ships take ocean water into their ballast tanks instead of traditional solid ballasts like rocks and sands which have been used by old ships for a long time.

Figure 1.7 shows the ballast water status in a typical vessel at different points of the voyage.

Containerships discharge their existing ballast water as the cargo is loaded. Ballast is primarily composed of water but it also contains sediment and thousands of living species which will be disposed in foreign waters. These species known as alien or invasive species can affect the native marine food chain and cause environmental damages. Rigby et al. 1995 estimate that about 10 billion tones of ballast water is transported around the world each year. The role of ballast water in introducing exotic species has received extensive attention recently and governments have established guidelines for discharge and treatment of the ballast water.

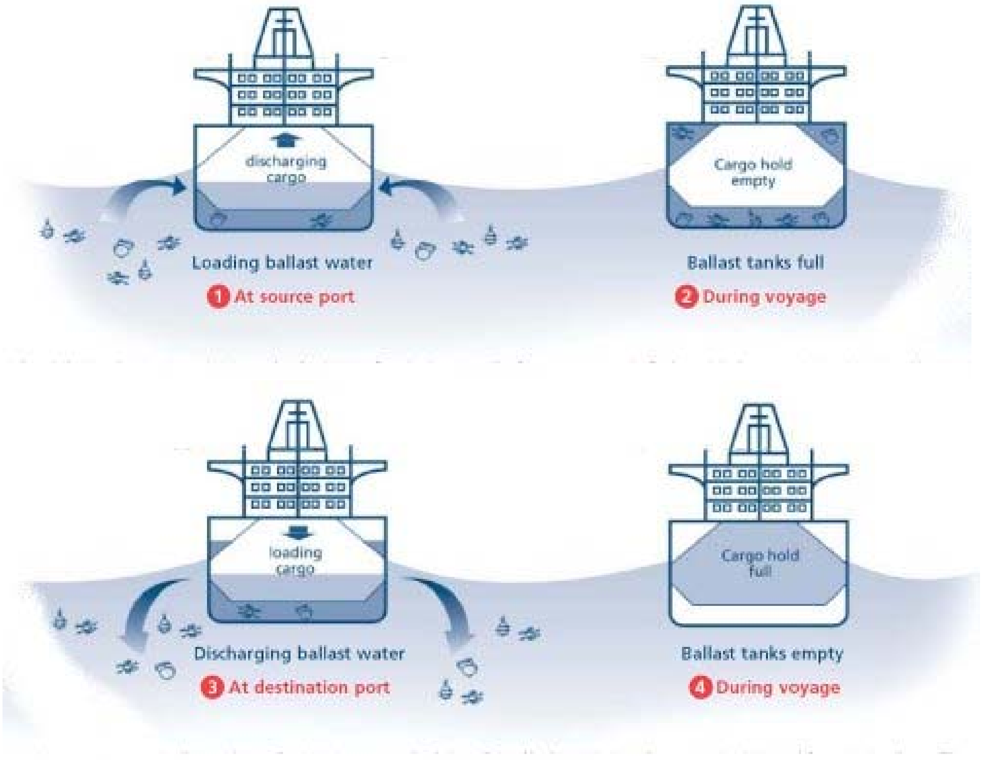


Figure 1.7: Cross section of a ship showing ballast tanks and ballast water cycle

(Source: International Marine Organization's website www.imo.org)

The more the imbalance in a containership, the more ballast water is required to stabilize it. In summary a proper stowage plan helps to make the containership operations more environmentally friendly in two ways. First it reduces the amount of ballast water needed to be carried which consequently lowers the risk of polluting the forthcoming ports with the exogenous species. Second it reduces the energy required for the thrust because the thrust is proportionate to the load of the vessel.

1.7. Motivation of this research

Because of the tense competition among ports in recent years, improving the operational efficiency of ports has become an important issue in containership operations. One of the major performance measures is the berthing time at a port. Arrangement of containers both within the container terminal and on the containership play an important role in determining the berthing time. The berthing time of a containership is mainly composed of the unloading and loading time of containers. Containers in a containership usually are stored in stacks. A container is directly accessible only if it is on the top of one stack (Last in First Out, LIFO). The ship visits several ports during a voyage and containers are loaded and unloaded at each port. The task of determining a good container arrangement to minimize the number of re-handlings while maintaining the ship's stability is called stowage planning, which is an everyday problem solved by ship planners.

Containers are loaded to and unloaded from the containership using quay cranes. The problem of allocating quay cranes to ship's sections is known as crane split. In some cases up to ten quay cranes might be allocated to a ship. Technical requirements determine the range in which each quay crane can operate. The ship's dwell time is determined by the time that the latest carne finishes its job. Since the distribution of the containers over the bays affects crane utilization and overall ship berthing time, crane split and stowage problem are interrelated. Given the configuration of cranes at each visiting port, the stowage planning must take into account the utilization of quay cranes as well as the reduction of unnecessary shifts to minimize the total time at ports over the voyage. It seems that integration of the stowage plan and the crane split results in a more efficient working instruction which ultimately increases port utilization; however the joint optimization of these processes has not been discussed in the literature.

1.8. Structure of the dissertation

This dissertation is organized as follows. An introduction to the problem and the motivation for the research are presented in Chapter 1. Chapter 2 summarizes the existing literature that focus on containership loading/unloading operations and some other related problems. A mathematical programming model for solving the problem is developed and discussed in Chapter 3. Solution results of the aforementioned model for some sample problems are reported in Chapter 4. A general discussion on optimization techniques as well as a genetic algorithm framework for solving the containership loading problem is presented in Chapter 5. In Chapter 6 the parameters of the proposed genetic algorithm are analyzed and the performance of the method is discussed. Application of the solution algorithm for solving several scenarios and the effect of different policies on the loading/unloading process are shown in Chapter 7. The final chapter includes the concluding remarks and directions for future research.

Chapter 2: Literature review

2.1. Containership stowage planning

The stowage planning for a containership deals with the arrangement of containers within the ship. In general, a containership calls at a number of ports on her route and in each port, containers are unloaded and loaded. The stowage problem considers the assignment of containers to the positions in the ship such that the overall stability of the ship is maintained and the number of unnecessary movements is minimized. These unfavorable movements appear if at a certain port, containers have to be unloaded and reloaded again, since they are stored on top of containers destined for that port.

Shields (1984) used a Monte Carlo method to solve the problem. Multiple parameters and constraints are considered in the research but the quality of the solution is not addressed. In this method the solution space is not searched systematically. Since then researchers have used mathematical programming and heuristics and artificial intelligence based optimization methods have been developed to solve large scale problems.

Aslidis (1990) solved a very special case of one uncapacitated column with constraints imposed on the vertical center of weight of the stack. He proved that exact optimal solution for the single column stack can be obtained in polynomial time.

Avriel and Penn (1993) and Avriel et al. (1998) formulated the problem as a binary linear programming model without considering stability constraints. All containers are assumed to have the same size. Since the stability is not taken into account the weight of the containers is ignored. Due to the large number of variables needed, it was impossible to find the optimal solution for large size problems. Small examples were solved to optimality and alternative heuristics were proposed for larger problems. In their heuristic the authors broke the original transportation matrix into a sum of two sub matrices, one included the whole column and the other was a matrix of remainders. In a separate work Avriel et al. (2000) investigated the relation between the stowage problem and the coloring of circle of graphs problem. They showed that finding the minimum number of columns for which there is a zero shifts stowage plan is equivalent to finding the coloring number of circle graphs and through that they proved that the general stowage planning problem is NP-Complete. Haghani and Kaisar (2001) developed a mixed integer programming model and a heuristic algorithm for the simplified stowage planning to minimize container loading cost while maintaining the ship's stability within an acceptable range. They took longitudinal moment, trim and metacentric height (GM) into account and assumed that all containers have the same dimensions. In their two step heuristic approach they first assigned containers to stations and then to the individual cells within the station. Giemsch and Jellinghaus (2003) proposed a mixed integer programming model and a three step heuristic. Stability constraints are not considered in their work and results are not reported clearly. Imai et al. (2006) developed a multi-objective mathematical model for simultaneous stowage and load planning of a containership. They simplified the stowage part of the problem by considering only loading related rehandlings and single size containers. Since the effects of the unloading related reahndlings are ignored during the load planning, the burden will be carried to the forthcoming ports. They used genetic algorithms to solve the joint problem of stowage and load planning to reduce the container rehandle in yard stacks.

In the aforementioned models an important simplification is the uniform container size. Multiple container sizes are considered in Ambrosino et al. (2004). They assumed that all the loading is done at the first port and the coming ports are only for discharge. This assumption reduces the stowage problem to an assignment problem with stability constraints at the master bay. They used a decomposition heuristic algorithm to solve the problem.

In the area of meta-heuristics, genetic algorithms and tabu search are used as solution approaches to stowage planning problem. Wilson and Roach (1999), (2000) used a hybrid heuristic composed of branch and bound and tabu search. They considered multiple type containers and stability and solved the problems in two steps. In the first step blocks or cargo are allocated to the bays in the vessel by branch and bound and in the second step tabu search assigns individual containers to each block. They reported that results are as good as the ones by human planners; however the size of the solved example was small and details of the solution approach were not presented. Later Wilson et al (2001) used genetic algorithms instead of tabu search to progressively refine the arrangement of containers within the cargo space of a containership until each container is specifically allocated to a stowage location. No mathematical model was presented in the papers by Wilson et al.

Todd and Sen (1997) developed a multi criteria genetic algorithm. They call their genetic encoding a complete encoding because the whole assignment pattern at each port is stored in chromosomes which is both memory consuming and computationally expensive. Besides that the crossover operator does not guarantee the feasibility of the resulting off-springs. To address this issue they apply a repair procedure to the results which interferes with the natural inheritance mechanism of genetic algorithm and destroys useful information which are crucial for evolution. Instead of saving the complete layout, Dubrovsky et al. (2002) used a genetic algorithm with a compact solution encoding by recording only the changes of the layout form port to port which result from loading and unloading the containers along the route. This method significantly reduces the search space and speeds up the convergence. However it does not take into account the effect of crane utilization and multiple containers sizes. To demonstrate the stability concerns they presented an example with horizontal equilibrium constraint. A parallel implementation of their genetic algorithm promises shorter running times when the number of CPU's is more than one.

Most recently Delgado et al. (2009) proposed a constrained programming approach for stowage planning. They assumed that containers must form a stack, 20-foot containers cannot be stacked on top of 40-foot containers, reefer containers must be assigned to reefer slots, and sum of the heights and weights of containers in each stack must stay within the stack limits. The objective of their approach was to minimize overstows, keep stacks empty if possible and avoid loading non reefer container into reefer cells. Results for some small scale cases shows that this method outperforms integer programming as well as column generation based approaches for the problem. Cranes are not considered in this research.

2.2. Container loading problem

Container loading problem is the problem of loading a subset of small items (e.g. rectangular boxes) into a large container. Depending on the field of application, the objective function and the side constraints, several variants of the container loading problem have been discussed in the literature. One classification is the problem of packing all given items into the least possible number of containers vs. packing as many items as possible into a given number of containers. Two, three and four dimensional packing models are discussed in the literature. Restrictions include but not limited to container capacity. Dyckhoff (1990) classifies such problems.

2.2.1. Bin packing

The problem of packing a set of boxes with different dimensions into a set of bins is called the bin packing problem. The objective is to use minimum number of bins to accommodate all the boxes. This problem has been discussed in the computer science and operations research literature extensively. Among them Scheithauser (1991) studied three-dimensional bin packing problem and Martello et al. (2000) considered exact methods for the solution. A review on the approximation algorithms for the bin packing problem can be found in Coffman et al. (1996). Giemsch and Jellinghaus (2003) discussed the possibilities of extending the three-dimensional bin packing problem to the containership stowage problem and Giemsch (2004) studied the stowage problem as a 4-D packing problem.

2.2.2. Strip packing

The strip packing problem also known as pallet loading problem involves the packing of a set of rectangles into a strip of given width and infinite height so that no rectangles are overlapping and the height of the strip is minimized. It is a generalization of bin packing because if we restrict all input boxes to be of the same height, then strip packing is equivalent to bin packing. It has applications in manufacturing industry, job scheduling, etc. The problem also has applications in multi-drop situations where the load should be divided into distinct sections for different destinations as it is discussed in Bischoff and Ratcliff (1995). A survey on two-dimensional packing problems is available in Lodi et al. (2002).

2.2.3. Multi-Container loading

Multi-Container Loading is a variation of the bin packing where the containers can have different dimensions. The objective is to choose a subset of the containers such that the shipping costs are minimized. An analytical model for the problem is described in Chen et al. (1995) and LP-based bounds are found in Scheithauser (1999).

2.2.4. Knapsack loading

Given a profit for each box, the knapsack loading problem is the problem of loading a subset of rectangular boxes into a container to maximize the loading profit subject to the container capacity. Minimization of the unused space can also be an objective function if the profit of each box is associated to its volume. Pisinger et al. (2004) covers many methods and techniques available for the solution of the Knapsack problem and its variations.

2.3. Stacking problem

A storage system in general is composed of the structure and the rules for adding and retrieving items in a storage area. Based on the application, a storage system can be anything like rail shunting yard, parking garage, computer memory, book library, a warehouse inventory, etc. The storage system may accommodate single type or multiple-type items. In the parking garage example the items are cars while in a hardware warehouse inventory items might be different kind of tools. The term stacking usually appears in the context of storage systems. Although it might be referred to the general situation in which items are stored on top of one another, it usually implies that the method of retrieval from the storage area is last in first out (LIFO). In a LIFO system the last item that is stored in the system is the first item to be retrieved. If the items can be retrieved regardless of their entering sequence the system will be randomly accessible. An Example is an inventory shelf where items can be stored vertically, but can be taken in any desired order. While stacks can be found in physical form in environments such as warehouses, their conceptual form is extensively used in the computer science and queuing theory. The storage rules in both forms are similar, but the purpose of stacking is different.

In physical systems stacking is usually used because the storage area is limited and also because it is cheaper to put the items of approximately the same size on top of each other rather than building shelving systems and cellular structures.

In computer systems however this is not the case. Stacks can be found in every level of a computer system not because it is cheaper to store data in a LIFO fashion, but because it is an efficient and powerful method to implement specific applications. In the low level layers of a computer they are used for interrupt handling and system function call management. In the higher levels stacks have applications in expression evaluation and syntax parsing, runtime memory management, backtracking, etc. Using stacks for expression evaluation was first proposed by the early German computer scientist Friedrich L. Bauer and patented in 1957. He received IEEE Computer Society Pioneer Award in 1988 for his work on Computer Stacks (Broy 2002).

2.3.1. Overstowage

Stacking the items might seem to be a cheap alternative at first, but it might come at the cost of overstowage. Overstowage happens when there is need to retrieve an item which is not located on the top of the stack. In that case the items must be temporarily retrieved one by one until the designated item becomes accessible. After that, the temporarily removed items must be put back into the stack. Overstowage happens in everyday life. For example in an overcrowded elevator some people might have to temporarily exit the elevator in order to let people who have reached their desired floor out of the elevator. While packing a suitcase for a trip, one usually tries to put the items that are needed more frequently on top to avoid overstowage. Drivers of the pick up and delivery service trucks like UPS and FedEx may experience overstowage if the packages that they want to reach far inside the truck are blocked by the recently loaded ones.

Another example is the multiple-car carrier truck which, based on the size of the vehicles, can transport up to 12 vehicles in their stack shaped structure. Figure 2.1 shows a schematic design of such trucks. There are two independent stacks in this example each having capacity of five cars. For unloading the leftmost car in the top tier it is necessary to first unload all the other cars in that tier. So if all the cars are not destined to the same destination and enough attention is not paid while loading the cars, the operator may have to go through the process of unloading and loading the overstowed cars. This is both time consuming and expensive. Overstowing is not always a result of bad planning; it might be inevitable due to technical and operational constraints. In the case of multiple-car carrier there might be a situation that for example only sedan cars are allowed on the leftmost position. In that case if the truck has to transport 9 SUV’s and 1 sedan to two dealerships and the sedan happens to belong to the first dealership, then 4 SUV’s on top ought to be overstowed.

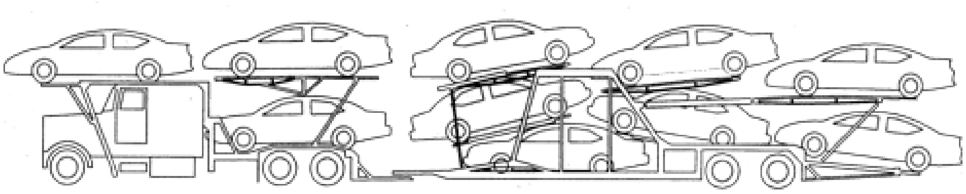
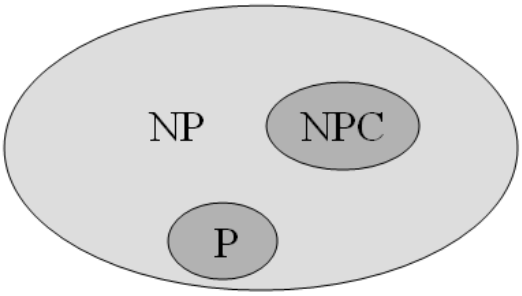


Figure 2.1: Schematic of a multiple-car carrier truck

Similar to the other stacking problems, stacking containers into the cellular columns of a containership or in the storage area of the container yard terminals may result in overstowage. The containers are stored in the yard before moving to the bays in the vessel. In order to prevent the situations like a container in the yard has to be moved so that the container below can be accessed; the terminal managers try to match the yard arrangement to the stowage plan. However some containers arrive while the loading process has already begun and that makes the overstowage in the yard inevitable. The overstowage in containership occurs mainly because of the stability, technical and operational considerations.

2.4. Complexity of the containership stowage problem

In computing theory, the complexity of an algorithm is measured by the number of required operations in term of the size of the problem. For long time computer scientists and mathematicians have struggled to find the common features for some problems that could determine whether a polynomial time algorithm for solving them does or does not exist. Polynomial time refers to the computation time of an algorithm, where the running time is less than a polynomial function of the problem size. Nondeterministic Polynomial (NP) problems are problems that their solutions are verifiable in polynomial time. NP-complete problems are a subset of NP problems that are considered the hardest in the sense that no NP-complete problem can be solved by any known polynomial time algorithm. It is also proven that if there is a polynomial time algorithm for any NP-complete problem then there are polynomial algorithms for all NP-complete problems. Thousands of computer scientists have been unsuccessful for decades to design polynomial time algorithms for this class of problems. Based on the overwhelming empirical evidence many researchers have conjectured that there can be no polynomial time algorithm for any NP-complete problem; however nobody has been able to prove this. Optimization problems whose decision versions are NP-complete are called NP-hard (Papadimitriou and Steiglitz 1998). Figure 2.2 shows a venn diagram of the conjectured relationships of different classes.



$P \subset NP, NPC \subset NP, P \cap NPC = \varnothing$

Figure 2.2: Conjectured relationships between P, NP, and NP-complete

Researchers have taken different approaches to prove or at least show that the containership stowage planning problem with stability constraints is NP-Complete. These methods are summarized in this section.

2.4.1. Connection to capacitated multi-stack overstowage problem

Aslidis (1989) developed an exact analytical algorithm for solving the simplified case of single column and single size stacking problem to optimality. He shows that multi-stack overstowage problems (MSOP) are much harder than their single stack counterparts and identifies two possible sources of difficulties for that. First it is the problem of assigning the containers to stacks to avoid overstowage. Since the time for finding optimal overstowage solution to the one stack problem is non-linear the assignment problem alone can make the problem very hard. Secondly the possibility of container switching among stacks through the voyage is another complexity factor. He presented a model for MSOP and transformed it into a minimum network cost flow problem with integrality constraints. By using a decision version of the problem he then proves that MSOP belongs to the class of NP problems. To prove that MSOP problem is NP-complete, a general method is to find a known NP-complete problem and transform it to MSOP in polynomial time. Although the author found strong connection between MSOP and some well known NP-complete problems, he could not find a polynomial transformation. Thus Aslidis (1989) failed to mathematically prove that MSOP is NP-complete. By bringing the stability and operational constraints into MSOP he concluded that there is a very high chance that the problem is NP-complete. Introducing different size containers, operational constraints and crane utilization considerations adds to the complexity of the problem and one can use the same reasoning to consider the extended version of the problem NP-complete as well.

2.4.2. Connection to tram dispatching problem

Given a set of arriving trams, a set of departure schedules and a set of depot positions consisting of horizontal capacitated stacks, the tram dispatching problem (TDP) is the problem of assigning the trams to the stacks such that the cost of operations is minimized. Winter (1999) proved that TDP is NP-complete. Using the binary programming model for containership stowage problem by Avriel and Penn (1993), the author established a connection between container stowage problem and the tram dispatch problem. It was assumed that shift operations for containers and shunting operations for trams are of the same nature but different. A transformation model from TDP to container stowage problem is presented. Results show that because of the NP-completeness of the problem if the stacks contain five or more positions, it is impossible to solve instances of more than fifteen trams in reasonable time. The binary model by Avriel and Penn (1993) is developed for problem with uniform container size without considering the stability constraints. So based on its connection to TDP, it can be concluded that solution to the more comprehensive version of containership stowage problem cannot be obtained in polynomial time.

2.4.3. Connection to the coloring of circle of graphs

Graph coloring is a well known classical problem is graph theory. In general it is an assignment of colors to certain objects of a graph (e.g. edges or vertices) subject to certain constraints. Vertex coloring as a special case of graph coloring is the problem of coloring vertices of a graph such that no two adjacent vertices share the same color. Chromatic number of a graph is the least number of colors needed to color the graph. The problem of finding the minimum coloring of a graph is NP-hard and its corresponding decision problem is NP-complete (Jensen and Bjarne 1995).

Avriel et al. (1999) considered a containership consisting of a single bay and that has C vertical columns and R rows. They called the bay capacitated if each column has a finite number of rows and uncapacitated otherwise. Given the transportation matrix and uniform size containers they defined the minimum shift problem as the problem of finding the stowage plan with the smallest number of shifts. The decision problem is the uncapacitated s-shift problem which indicates whether given a transportation matrix, a stowage plan with a cost of at most s shifts exists. They established a connection between the zero-shift problem and the coloring of overlap graphs. They proved that the uncapacitated zero-shift problem is NP-complete and finally concluded that uncapacitated shift problem is NP-complete. Since the simplified containership stowage planning problem is NP-complete, the more complicated variances which account for stability and other constraints are also NP-complete.

2.5. Crane scheduling and utilization

One of the important decisions to be made by terminal operators is the crane scheduling, also known as the crane split problem. Quay cranes are the most expensive single unit of handling equipment at container terminals. By improving quay crane efficiency, ports can increase their productivity and improve their throughput. Depending on the ship size, up to five cranes may simultaneously operate on the ship as this number may be doubled at the indented berth terminals. Crane scheduling problem is the problem of optimal assignment of quay cranes to the ships with respect to technical specifications of the cranes and the vessels. Daganzo(1989) proposed a MIP model for static crane allocation problem assuming that the berth length is not restricted. The objective function was to serve all the vessels and minimize their total delay cost. Exact and approximate solutions were presented. Furthermore, Peterkofsky and Daganzo (1990) used branch and bound to determine the departure time of multiple vessels and the number of cranes assigned to the bays while minimizing the total delay cost. Neither of the above works considered the interference among cranes and the precedence relationship among tasks. Lim et. al. (2004) introduced spatial constraints to the problem, assuming that cranes cannot cross each other. Dynamic programming algorithms, a probabilistic tabu search, and a heuristic was proposed to find a job to crane assignment that maximizes the throughput. Considered the quay cranes as processors and the vessels as jobs, Guan et. al. (2002) show a multiprocessor task scheduling model for berth allocation in which the total weighted completion time of the jobs is minimized. They presented a heuristic for the problem and analyzed the worst case instances. More recently, Kim and Park (2004) described a mathematical model to determine the sequence of discharging and loading operations that a quay crane will perform so that turnaround time of a single vessel is minimized. They used branch and bound to obtain the optimal solution and developed a lower bound. To overcome the computational difficulty, they proposed a greedy randomized adaptive search procedure. Moccia et al. (2006) proposed modifications to the model by Kim and Park (2004) and formulated the problem as a vehicle routing problem with side constraints. They used a branch and cut algorithm for solving large instances of the problem and compared their results with the work by former authors. Imai et al. (2007) addressed the berth allocation problem with a consideration of serving simultaneously multiple small ships at an indented berth terminal. They conclude that although turnaround time of mega-ships was faster in such terminals, the total service time for all ships was longer than the one in a conventional terminal.

Other researchers have considered crane scheduling jointly with other decision problems at port. Schonfeld and Sharafeldien (1985) developed a model for minimizing the total port costs which accounts for the delay costs, mutual interference among the cranes, minimum work shifts and storage yard constraints. The results showed that total costs can be reduced by increasing the number of cranes per berth and berth utilization. Bish (2003) considered the crane scheduling along with storage assignment determination and vehicle dispatching problem and developed a heuristic to minimize the maximum turnaround time of all the ships in the planning horizon. Park and Kim (2003) discussed an integer programming model for scheduling berth and quay cranes and presented a two-phase solution algorithm. In the first phase a near optimal solution for berthing times and positions of the vessels is determined and in the second phase the specific operating schedules for individual cranes are constructed.

All previous studies were based on the assumption that all relevant containers are first unloaded before any are loaded. Goodchild and Daganzo (2007) studied the benefits of crane double cycling where loading and unloading operations are performed simultaneously. They formulated the problem as a scheduling problem and solved it using commercial solvers for small instances. A fast greedy algorithm and a lower bound are developed for real size problems. Zhang and Kim (2009) proposed a mixed integer programming model and a gap-based local search approach to maximize the number of dual-cycle operations of quay cranes.

A comprehensive literature review on container terminal operations may be found in Steenkan et al.(2004). Previous useful literature reviews are presented in Iris and Rene (2003) and Meermans and Dekker (2001).

2.6. Conclusions

The containership stowage planning problem which is the problem of stacking containers into the cellular columns of a containership is an everyday problem solved by the ship planners. Overstowage which is both costly and time consuming occurs in containership loading and unloading operations because of inefficient planning, technical limitations or both. Researchers have approached the problem as a variation of bin packing problem with stability constraints, multi-column stacking problem and assignment problem and have applied interesting techniques to minimize the overstowage. However the actual objective of stowage planning is to minimize the total time that the vessel spends at all ports. Although reducing overstowage may contribute to this goal, the role of other players in the loading and unloading operations such as quay cranes should not be ignored. Considering the quay crane assignment in containership stowage planning problem has not been addressed in the literature. Maximizing the utilization of quay side equipment while minimizing the number of overstowed containers can produce a better stowage plan which directly translates into cost saving and congestion reduction. This dissertation looks at the containership load planning problem from this new perspective.

Chapter 3: Problem formulation

3.1. Problem statement

Container port system consists of different subsystems. Because of the complexity of the operations, subsystems have been studied and analyzed individually. Recently researchers have paid more attention to the joint optimization of two or more subsystems. Containership load planning is an important part of the container transportation logistics. The growing competition among ports and increasing capacity of the containerships has resulted in congestion in major terminals and has put pressure on container terminal managers and shipping companies to improve their operations. At the quay side interface of the container port system, berthing time is the most important performance measure. Quay cranes are the most expensive equipment at port and they play a major role in the terminal productivity. Depending on the size of the vessel and availability of quay cranes, usually more than one crane will be assigned to a vessel. Assigning more cranes to a vessel might not improve the berthing time if the stowage plan of the vessel does not match the crane assignment. This research will combine the quay crane assignment with the traditional stowage planning problem in order to generate more efficient stowage plans. Instead of focusing on minimization of overstowage, the real objective function of the containership load planning which is the minimization of overall berthing time at all ports will be used. This will be done through maximizing the utilization of quay cranes while minimizing the unproductive container moves. Realistic stability and operational constraints as well as individual container characteristics are taken into account.

3.1.1. Modeling contribution

No optimization model exists in the literature that addresses joint optimization of quay crane utilization and stowage planning. Very few mathematical models exist for containership stowage planning optimization. Each of these models has its own simplifications and shortcomings. More details can be found in section 2.1. To the best of our knowledge this is the first mathematical model to minimize total berthing time and accounts for containers of different weight, size and type as well as stability and real life operational constraints. It also introduces the assignment pattern and technical specifications of the quay cranes to the optimization framework. So far almost everybody has directly translated the minimization of shifts to the minimization of time at port and the ones who have mentioned the necessity of paying attention to the horizontal distribution of the containers during stowage planning have not considered it in their models (Giemsch, Jellinghaus 2003). This research fills this gap.

3.2. Problem description

A containership has a cellular structure. The containers are held in bays along the length of the ship. The containers are stacked in tiers in each bay. Each of these tiers is made up of a number of cells. The position of the container within the ship is entirely specified by three indices: bay-row-tier. The layout of bays, rows and tiers differs from ship to ship because the location of engine rooms, accommodation sections and hull shapes are different in each ship. Figure 3.1 shows the cellular structure of a sample containership.

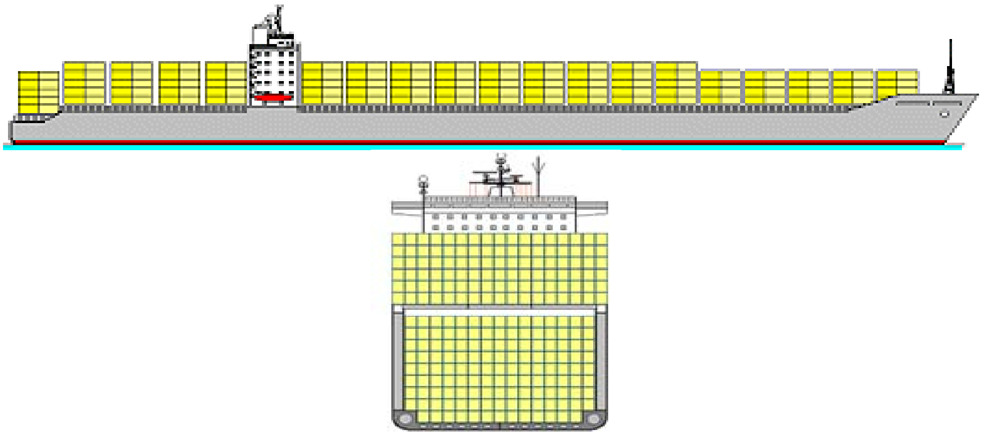


Figure 3.1: Cellular structure of a containership (www.containerhandbuch.de)

A general layout design can be looked at as a three dimensional matrix. Each element of the matrix corresponds to a cell in the vessel. This value might serve as the container number assigned to the corresponding cell or simply be a binary digit showing the availability of the cell. Specific hull shapes and design structures may be addressed by using predefined values in this way. Figure 3.2 shows the general layout.

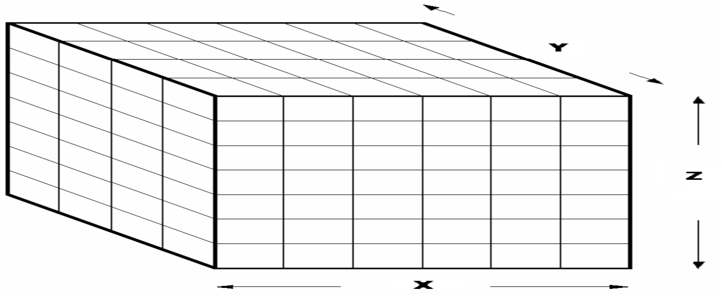


Figure 3.2: General cellular structure

Usually more than one quay crane operate on a containership at each port. Quay cranes move horizontally along the ship and load/unload containers to/from the vessel using a spreader arm. Horizontal moves are both slow and expensive, so they should be avoided as much as possible. These moves are also restricted by technical constraints such as no two cranes may work on the same bay simultaneously. Quay cranes are the most expensive single unit handling equipment in container terminals. Therefore, by improving crane utilization, ports can reduce ship dwell time, increase throughput of the system and improve port productivity. Utilization of each crane is determined by dividing the crane busy time over ship dwell time. Distribution pattern of containers along the bays plays a crucial role in crane utilization.

3.3 Containership stability

Safety of the sea vessels, whether they are cruise ships or cargo ships, goes hand in hand with their stability. For the cargo and containerships it is crucial that the weight is properly distributed through the ship so that the structure is not overstressed and the standard criteria of stability are met. A brief summary of Hydrostatic as well as experimental rules of stability for containerships is given in this section.

3.3.1. Hydrostatic rules of stability

Stability of a vessel is the ability to return to its upright position when disturbed, after the disturbing force is eliminated. Archimedes principle says that a body floating or submerged in a fluid is buoyed up by a force equal to the water it displaces. So a ship sinks if weight of water displaced by the underwater volume is less than the weight of the ship. One way to check for the stability of a ship is by measuring center of gravity (G) and the center of buoyancy (B) force. The former is the aggregation of all gravity forces acting downward through ship’s geometric center and the latter is all the buoyancy forces acting upward as on force through underwater geometric center. Location of G remains the same unless weight is added, removed or shifted. The location of B changes as the ship heels. Depending on the location of G and B there exist a righting moment which tends to return the ship to the upright position and an upsetting moment which tends to overturn the ship (Barrass and Derrett 2005). The meta-center of the ship (M) is the intersection of different lines of buoyancy as the sheep heels through small angles. The relationship between M and G determines the stability status of the ship. According to the position of M and G three cases exist:

1. **G under M:** Ship is in stable equilibrium meaning that when inclined, it tends to return to the initial upright position
2. **G above M:** Unstable equilibrium exists. In this situation if the ship is inclined to a small angle, it tends to heel over even further.
3. **G coincides with M:** Ship is in neutral equilibrium and if inclined to a small angle, it will tend to stay in that angle until another external force is applied.

Figure 3.3 shows the forces on a sample vessel.

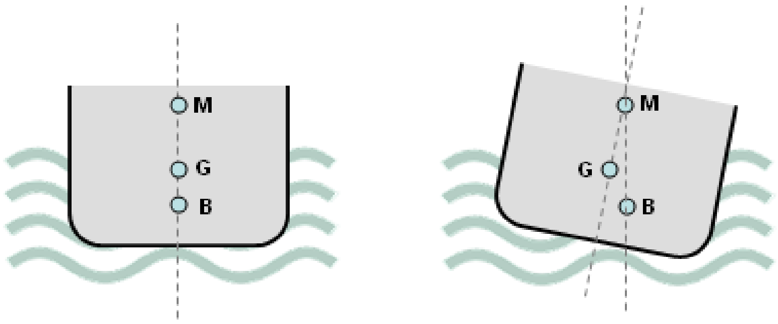


Figure 3.4: Gravity forces, Buoyancy forces and Meta-center of a ship

Another tool to measure the stability is the meta-centric height (GM) which is the distance between the meta-center and the center of gravity. The typical working value for GM for containerships is approximately 1.5 m. If the GM falls below this threshold the vessel will be unstable. Generally speaking the higher density containers must be stored in the lower holds of the vessel in order to increase the meta-centric height.

3.3.2. Experimental rules of stability

Since calculating the meta-centric height requires detailed information of the containership structure, experimental rules of stability have been created by ship planners which are applicable to typical containerships. These rules can be categorized as longitudinal, cross and vertical equilibrium.

Longitudinal equilibrium

Containers stowed at the bow side of the ship create a tilt which acts as an opposite force to the tilt created by the containers at the stern side. If these forces cancel out each other the bow and stern will have the same waterline height. Longitudinal equilibrium requires that the difference in the height of waterline between bow and stern does not exceed a given threshold. Besides safety considerations the longitudinal equilibrium affects the required propulsion and the fuel consumption by the engine. Figure 3.5 shows this equilibrium.

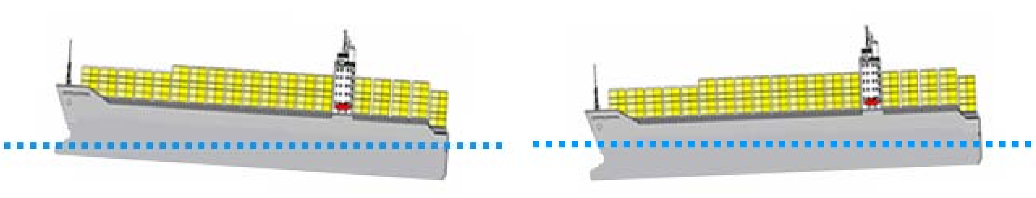


Figure 3.5: Longitudinal equilibrium

Cross equilibrium

Relating to the axis of symmetry going through bow to stern, the containers at the left side create a tilt opposite to the one by the containers at the right side. If these tilts are not equal the vessel will heel toward the heavier side. To ensure the stability the weight difference between the two sides must be kept within a predetermined range. Figure 3.6 illustrates this situation.

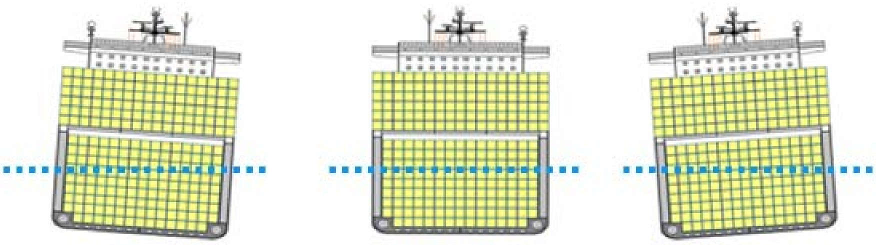


Figure 3.6: Cross equilibrium

Vertical equilibrium

According to the hydrostatic rules the location of center of gravity changes with the vertical shift of the weight in the vessel. However the vertical shift does not affect the center of buoyancy because the underwater portion of the containership does not change. This means that shifting the heavier containers to the lower compartments of the vessels increases the GM and improves the stability. This is the reason that empty containers are mostly stored above the deck area. The experimental rule of vertical stability requires that the total weight of each tier of containers to be less than or equal to the total weight of the tier underneath.

3.4 Operational consideration

In addition to the vessel stability constraints, a number of operational constraints must be satisfied. Generally these constraints relate to the placement restrictions with respect to the size, type, content and strength of the containers. Some of these constraints are presented below.

1. Weight of a single column: depending on the structural specification of the containership, there is a limit to the maximum weight of a single column of the containers that the deck structure can bear. So the weight of individual stacks may be restricted.
2. Racking strength: The containers below deck are stored in cells, however above deck there are no cell guides. In this case the containers on the lower tiers hold the containers stowed above them. The planners must make sure that the weight of the upper containers does not exceed the strength of the base containers. This is one of the reasons that empty containers are usually stored above deck.
3. Container support: standard cells are generally designed for twenty feet containers and 40 feet containers require two contiguous 20 feet cells. Each container needs to be fixed by four twisters to the upper corners of the containers below it, so smaller containers cannot be placed above larger ones.
4. Refrigerated containers: refrigerated containers (reefers) are used for transporting perishable goods. They need to stay connected to the electricity outlet for the safety of their contents. There are also containers that require ventilation. These containers must be placed in certain areas of the vessel where their requirements can be met.
5. Hazardous containers: the placement of the containers containing hazardous materials is governed by the hazardous materials safety regulations. According to Code of Federal Regulations hazmat containers must be separated from other hazmat

and reefer containers by a minimum distance (CFR, Title 49, Transportation, Parts 100-185).

On June 22nd 2007, a large number of containers carried by the Ital Florida – a 3450 TEU fully cellular containership - were damaged at the Port of Trieste because of improper lashing and weight distribution. Figure 3.7 shows the incident.

Figure 3.7: Damaged containers on Ital Florida (www.cargolaw.com)

3.5 Assumptions

Using the general cellular layout for the containership, each cell is identified using three indices: bay-row-tier. This address is a system of numerical coordinates relating to length, width and height of the containership. The route which the ship takes in her voyage and the sequence of the ports at which she stops are fixed and known. At each port a set of containers must be picked up and some containers must be unloaded. The number of containers to be loaded/unloaded at each port as well as the complete relevant information of the containers including weight, size, type and destination are also known. More than one quay crane may be assigned to a vessel at each port. The number of available quay cranes, the range of bays in which they can operate on the ship as well as the technical parameters of the cranes are known.

3.6. Mathematical model

Based on the assumptions in the former section, a mathematical model that minimizes the ship's berthing time at all ports by minimizing the number of re-handlings and maximizing crane utilization is developed. The model is a binary integer programming model which observes the stability and operational constraints.

3.6.1. Parameters

C	Set of all containers
TF	Set of 20 ft containers
R	Set of refrigerated containers
H	Set of hazmat containers
N	Set of all ports
$O(c)$	Origin of container c
$D(c)$	Destination of container c
$W(c)$	Weight of container c
$T(c)$	Size of container c in TEU
NQ_t	Number of quay cranes at port t
$S(Q_{k,t})$	Start bay of crane k at port t
$E(Q_{k,t})$	End bay of crane k at port t
$W_{\max}(t)$	Ship weight capacity at port t
W_{column}	Maximum weight allowed for a column
P_k	Handling time of a container by crane k
LRB	Left-Right balance threshold
BSB	Bow-Stern balance threshold

CWL	Crane workload balance threshold

Total weight limit of the ship might be different at different ports since the berthing depth limit is port specific and the ship draft must meet that limit.

3.6.2. Decision variables

$$\delta_{c,t}^{x,y,z} = \begin{cases} 1 & \text{If cell (x,y,z) is assigned to container c at port t;} \\ 0 & \text{Otherwise} \end{cases}$$

$$U_{c,t} = \begin{cases} 1 & \text{If container c is unloaded at port t;} \\ 0 & \text{Otherwise} \end{cases}$$

$$I_{c,d,t} = \begin{cases} 1 & \text{If container c is on top of container d at port t;} \\ 0 & \text{Otherwise} \end{cases}$$

$$J_{c,k,t} = \begin{cases} 1 & \text{If container c is handled by crane k at port t;} \\ 0 & \text{Otherwise} \end{cases}$$

δ's serve as assignment variables. They determine whether a given cell is occupied by a given container at a given port. The specific hull shape and design of a given containership can be addressed by assigning dummy containers to the virtual cells which do not physically exist. Index variables I's keep track of the relative location of each two containers at any port. Decision variables U's show if the container has been unloaded at a port either because of rehandling or simply because it has reached the final destination. J's are crane assignment variables and show the cranes that handle a container at each port.

3.6.3. Objective function

The difference between the time of which the ship is released and the arrival time at port is called dwell time. The objective function minimizes the total dwell time at all ports which is as follows:

$$Min\sum_{t=1}^{N}(T_t = Max\{P_{k_1}\sum_{c=1}^{C}J_{c,k_1,t}, P_{k_2}\sum_{c=1}^{C}J_{c,k_2,t}, \ldots, P_{k_{NQ_t}}\sum_{c=1}^{C}J_{c,k_{NQ_t},t}\}) \quad (3.1)$$

Since this objective function is nonlinear we transfer the crane utilization considerations to the constraints and use (3.2) as objective function.

$$Min\sum_{c=1}^{C}\sum_{t=1}^{N}U_{c,t} \quad (3.2)$$

This objective function minimizes the total number of unloading and rehandling activities over the voyage.

3.6.4. Cell assignment constraints

The cell assignment constraints are written as follows.

$$\sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z}\delta_{c,t}^{x,y,z} = 1 \qquad \forall c, t \in [O(c)..D(c)-1] \quad (3.3)$$

$$\sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z}\sum_{t=D(c)}^{N}\delta_{c,t}^{x,y,z} = 0 \qquad \forall c \quad (3.4)$$

$$\sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z}\sum_{t=1}^{O(c)-1}\delta_{c,t}^{x,y,z} = 0 \qquad \forall c \quad (3.5)$$

$$\sum_{c=1}^{C} \delta_{c,t}^{x,y,z} \leq 1 \qquad \forall x, y, z, t \qquad (3.6)$$

$$\sum_{c=1}^{C} \delta_{c,t}^{x,y,z} - \sum_{c=1}^{C} \delta_{c,t}^{x,y,z+1} \geq 0 \qquad \forall x, y, z \in [1..Z-1], t \qquad (3.7)$$

Constraints (3.3) force a container to be assigned to a cell at its origin port and stay aboard up to its destination. Constraints (3.4) and (3.5) prohibit a container to be assigned to a cell before its origin or after its destination port. Constraints (3.6) ensure that an individual cell will be assigned to no more than one container at each time. Constraints (3.7) ensure a container will not be put on top of an empty cell.

3.6.5. Stability constraints

The stability constraints based on experimental rules of stability are written as follows.

$$\left| \sum_{c=1}^{C} \sum_{x=1}^{X} \sum_{y=1}^{\lfloor Y/2 \rfloor} \sum_{z=1}^{Z} \delta_{c,t}^{x,y,z} \bullet W(c) - \sum_{c=1}^{C} \sum_{x=1}^{X} \sum_{y=\lceil Y/2 \rceil}^{Y} \sum_{z=1}^{Z} \delta_{c,t}^{x,y,z} \bullet W(c) \right| \leq LRB \qquad \forall t \qquad (3.8)$$

$$\left| \sum_{c=1}^{C} \sum_{x=1}^{\lfloor X/2 \rfloor} \sum_{y=1}^{Y} \sum_{z=1}^{Z} \delta_{c,t}^{x,y,z} \bullet W(c) - \sum_{c=1}^{C} \sum_{x=\lceil X/2 \rceil}^{X} \sum_{y=1}^{Y} \sum_{z=1}^{Z} \delta_{c,t}^{x,y,z} \bullet W(c) \right| \leq BSB \qquad \forall t \qquad (3.9)$$

$$\sum_{c=1}^{C} \sum_{x=1}^{X} \sum_{y=1}^{Y} \delta_{c,t}^{x,y,z} \bullet W(c) \leq \sum_{c=1}^{C} \sum_{x=1}^{X} \sum_{y=1}^{Y} \delta_{c,t}^{x,y,z-1} \bullet W(c) \qquad \forall t, z \in \left[2..Z\right] \qquad (3.10)$$

$$\sum_{c=1}^{C} \sum_{x=1}^{X} \sum_{y=1}^{Y} \sum_{z=1}^{Z} \delta_{c,t}^{x,y,z} \bullet W(c) \leq W_{Max}(t) \qquad \forall t \qquad (3.11)$$

$$\sum_{c=1}^{C} \sum_{x=1}^{X} \sum_{y=1}^{Y} \delta_{c,t}^{x,y,z} \bullet W(c) \leq W_{Column} \qquad \forall z, t \qquad (3.12)$$

The stability of the vessel must be maintained through the entire voyage. Constraints (3.8) and (3.9) are the horizontal and cross equilibrium stability showing that the weight difference between the right and the left and between bow side and stern side bays are within acceptable thresholds. Constraints (3.10) indicate that the weight of each tier must be equal to or lighter than the weight of the tier underneath. Constraints (3.11) and (3.12) limit the total weight of the containers on board for the vessel and for each column respectively.

To be more accurate (3.8) and (3.9) can be rewritten based on torque rather than weight as (3.13) and (3.14).

$$\left|\sum_{c=1}^{C}\sum_{x=1}^{X}\sum_{y=1}^{\lfloor Y/2\rfloor}\sum_{z=1}^{Z}\delta_{c,t}^{x,y,z}\bullet W(c)\bullet\left(\lfloor Y/2\rfloor-y\right)-\sum_{c=1}^{C}\sum_{x=1}^{X}\sum_{y=\lceil Y/2\rceil}^{Y}\sum_{z=1}^{Z}\delta_{c,t}^{x,y,z}\bullet W(c)\bullet\left(y-\lfloor Y/2\rfloor\right)\right|\leq LRB \quad \forall t \tag{3.13}$$

$$\left|\sum_{c=1}^{C}\sum_{x=1}^{\lfloor X/2\rfloor}\sum_{y=1}^{Y}\sum_{z=1}^{Z}\delta_{c,t}^{x,y,z}\bullet W(c)\bullet\left(\lfloor X/2\rfloor-x\right)-\sum_{c=1}^{C}\sum_{x=\lceil X/2\rceil}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z}\delta_{c,t}^{x,y,z}\bullet W(c)\bullet\left(x-\lfloor X/2\rfloor\right)\right|\leq BSB \quad \forall t \tag{3.14}$$

The horizontal torque by each container is calculated in relation to the axis of symmetry going through the ship from the bow to the stern, while the cross torque is evaluated in relation to the mid line of the ship. The total left side and right side torque could differ only within the given threshold. The same argument is valid for the total bow side and stern side torques. The minor imbalance caused by the accepted threshold will be corrected using the ballast water.

3.6.6. Shift constraints

The shift constraints are written as follows.

$$U_{c,D(c)} = 1 \qquad \forall c \tag{3.15}$$

$$\delta_{c,t}^{x,y,z} + \sum_{z'=z+1}^{Z} \delta_{i,t}^{x,y,z'} \le 1 + I_{i,c,t} \qquad \forall x,y,z,t\ \forall c,i \ne c \in [1..C] \tag{3.16}$$

$$I_{i,c,t-1} + I_{c,i,t} \le 1 \qquad \forall t\ \forall i,c \ne i \in [1..C] \tag{3.17}$$

$$U_{c,t} - U_{i,t} \le 1 - I_{i,c,t-1} \qquad \forall t\ \forall i,c \ne i \in [1..C] \tag{3.18}$$

$$\delta_{c,t+1}^{x,y,z} - \delta_{c,t}^{x,y,z} \le U_{c,t+1} \qquad \forall x,y,z,c\, \forall t \in [O(c)..D(c)] \tag{3.19}$$

$$\delta_{c,t}^{x,y,z} - \delta_{c,t+1}^{x,y,z} \le U_{c,t+1} \qquad \forall x,y,z,c\, \forall t \in [O(c)..D(c)] \tag{3.20}$$

Constraints (3.15) force a container to be unloaded at its destination port. Constraints (3.16) determine whether a container is on top of another one at certain port. Constraints (3.17) enforce that a container cannot be positioned both under and above another container at the same time. Constraints (3.18) ensure a container to be unloaded if the container underneath has to be unloaded at a port. Constraints (3.19) and (3.20) imply that if the position of a container in the vessel changes from one port to another, the container must be rehandled in order to shift the position.

3.6.7. Different size containers constraints

The most common container sizes in business are 20 feet and 40 feet. In this formulation a container of size S TEU (20 feet equivalent unit) is treated as S individual 20’ containers. Having $c = \{c_1, c_2, ..., c_S\}$:

$$\sum_{s=1}^{S} \delta_{c_1,t}^{X-s+1,y,z} = 0 \qquad \forall y,z,t,c \in C/TF \qquad (3.21)$$

$$\delta_{c_i,t}^{x,y,z} = \delta_{c_{i+1},t}^{x+1,y,z} \qquad \forall x,y,z,t,c \in C/TF, i \in [1..S-1] \qquad (3.22)$$

$$U_{c_i,t} = U_{c_{i+1},t} \qquad \forall x,y,z,t,c \in C/TF, i \in [1..S-1] \qquad (3.23)$$

When assigning a multi-section container to the bay located at the end side of the vessel (3.21) makes sure that there is enough room available for all the sections. Constraints (3.22) force all the cells occupied by a multi-section container to stick together horizontally and (3.23) implies that all cells occupied by such a container must be unloaded should one of the sections be unloaded. This definition expands the flexibility of the formulation to address different stowing policies. Similar equations can be written for containers with irregular heights.

With the presence of multi-section containers, the objective function in equation (3.2) must be changed as follow.

$$Min \sum_{c=1}^{C} \sum_{t=1}^{N} U_{c,t} / T(c) \qquad (3.24)$$

In (3.24) $T(c)$ is the TEU size of container c. Having this parameter as the denominator avoids double counting of the container moves for multi-section containers because all the sections are moved together as one piece.

3.6.8. Crane utilization constraints

The crane utilization constraints are as follows.

$$J_{c,k,O(c)} \ge \sum_{x=S(Q_{k,O(c)})}^{E(Q_{k,O(c)})} \sum_{y=1}^{Y} \sum_{z=1}^{Z} \delta_{c,O(c)}^{x,y,z} \quad , \forall k \in [1..NQ_{O(c)}], \forall c \tag{3.25}$$

$$\sum_{k=1}^{NQ_{O(c)}} J_{c,k,O(c)} = 1 \qquad \forall c \tag{3.26}$$

$$J_{c,k,D(c)} \ge \sum_{x=S(Q_{k,D(c)})}^{E(Q_{k,D(c)})} \sum_{y=1}^{Y} \sum_{z=1}^{Z} \delta_{c,D(c)-1}^{x,y,z} \quad , \forall k \in [1..NQ_{D(c)}], \forall c \tag{3.27}$$

$$\sum_{k=1}^{NQ_{D(c)}} J_{c,k,D(c)} = 1 \qquad \forall c \tag{3.28}$$

$$J_{i,k,t} + 2 \ge U_{c,t} + J_{c,k,t} + I_{i,c,t} \ , \ \forall c, i \ne c, \forall t \notin \{O(i), D(i)\}, \forall k \in [1..NQ_t] \tag{3.29}$$

$$\sum_{k=1}^{NQ_t} J_{c,k,t} = U_{c,t} \qquad \forall c, \forall t \notin \{O(c), D(c)\} \tag{3.30}$$

$$\left| \frac{\sum_{c=1}^{C} \sum_{k=1}^{NQ_t} J_{c,k,t}}{NQ_t} - \sum_{c=1}^{C} J_{c,q,t} \right| \le \frac{\sum_{c=1}^{C} \sum_{k=1}^{NQ_t} J_{c,k,t}}{CWL} \quad , \forall t, \forall q \in [1..NQ_t] \tag{3.31}$$

Given the number of quay cranes at each port and the range of bays on which the cranes will operate, constraints (3.25) through (3.28) find the cranes that perform the load/unload for each container at its origin and destination port, and ensure that only one crane will perform the handling. Should a container be shifted at any port, constraints (3.29) and (3.30) find the crane that does the shifting at that port. Constraints (3.31) balance the load among available cranes at each port. It is required that the difference between the workload of any crane and the average workload over all cranes at a port does not exceed a given threshold.

3.6.9. Operational constraints

The formulation is able to embrace other technical and operational considerations. For example in case of having containers of different sizes the operator may forbid putting large size containers on top of smaller ones (e.g. 40ft containers are not allowed on top of 20ft containers). Equation (3.32) formulates this rule.

$$\delta_{c_i,t}^{x,y,z+1} \leq 1 - \delta_{h,t}^{x,y,z} \qquad \begin{array}{l} \forall x, y, z \in [1..Z-1], t \in [O(h)..D(h)-1] \\ \forall h \in TF, c \in C / TF, i \in [1..S-1] \end{array} \tag{3.32}$$

In a reverse situation the following constrains can be used.

$$\delta_{h,t}^{x,y,z+1} \leq 1 - \delta_{c_i,t}^{x,y,z} \qquad \begin{array}{l} \forall x, y, z \in [1..Z-1], t \in [O(h)..D(h)-1] \\ \forall h \in TF, c \in C / TF, i \in [1..S-1] \end{array} \tag{3.33}$$

It is important to note that based on the operational rules only one set of constraints (3.32), (3.33) or neither of them should be in effect.

As an example to specific design constraints consider a vessel that allows 40 ft containers only in specific bays while no such restriction is imposed on 20 ft containers. If a vessel can accommodate 40 ft containers only in bays with even indices the following constraint may be used.

$$\delta_{c_i,t}^{x,y,z} = 0 \qquad \begin{array}{l} \forall\, odd\ x, y, z, t \in [O(c_i)..D(c_i)] \\ \forall\, c \in C / TF, i \in [1..S-1] \end{array} \tag{3.34}$$

This basically forces all the sections of a multi-section container to avoid odd bays.

The regulations regarding special type containers can be formulated in a similar fashion. For example perishable commodities are loaded into refrigerated containers. These containers should be plugged into electricity outlets which are available only in some sections of the vessel. Equation (3.35) implies this regulation.

$$\delta_{c,t}^{x,y,z} = 0 \qquad \begin{array}{l} \forall x \notin \{electrified\ bays\}, y, z \\ \forall t \in [O(c_i)..D(c_i)] \\ \forall c \in R \end{array} \tag{3.35}$$

When having hazmat containers mixed with other cargo, appropriate rules must be observed. For example if a special safety standard forbids storing a hazmat container adjacent to a refrigerated container the following constraints should be added to the model.

$$\delta_{c,t}^{x,y,z} + \delta_{h,t}^{x+1,y,z} \le 1 \qquad \begin{array}{l} \forall x \in [1..X-1], y, z, t \\ \forall (c,h) \in R \times H \end{array} \tag{3.36}$$

$$\delta_{c,t}^{x,y,z} + \delta_{h,t}^{x,y+1,z} \le 1 \qquad \begin{array}{l} \forall x, y \in [1..Y-1], z, t \\ \forall (c,h) \in R \times H \end{array} \tag{3.37}$$

$$\delta_{c,t}^{x,y,z} + \delta_{h,t}^{x,y,z+1} \le 1 \qquad \begin{array}{l} \forall x, y, z \in [1..Z-1], t \\ \forall (c,h) \in R \times H \end{array} \tag{3.38}$$

Constraints (3.36) and (3.37) avoid horizontal adjacency and (3.38) forbid vertical adjacency of each two hazmat and refrigerated containers.

3.7. Summary and conclusion

A binary integer programming model is proposed to solve the containership loading problem which is the problem of assigning containers to the cells of a containership that calls multiple ports. The objective function minimizes the total turnaround time of the vessel at all ports which is different from the objective function of stowage planning. The containers have different types, sizes and weights. Stability considerations are addressed in the form of cross and horizontal equilibrium, tier equilibrium and single column constraints. These are experimental rules of stability; however more accurate forms of stability such as meta-centric height calculations can be modeled using the given notation as long as they are linear or can be approximated by linear functions. Operational rules regarding the placement of different size containers as well as special purpose containers (e.g. hazmat) are modeled as constraints. This optimization model tries to maximize the utilization of the quay cranes while minimizing the number of shifts in order to minimize the overall time that the vessel spends at all visiting ports. The model is flexible and can easily embrace new operational rules and constraints.

Chapter 4: Formulation validation

4.1. Generating sample problems

To validate the mathematical model in chapter 3, sample problems have been generated and solved using commercial solver CPLEX 12.0. Each problem consists of following elements:

1. Number of ports to be visited
2. Number of cranes at each port plus the operating range of each crane
3. Dimensions of the containership $(X \times Y \times Z)$
4. List of containers to be transported. Each container has an identification number, origin port, destination port, size, type and weight

The list of the containers is randomly generated for each example such that the basic feasibility requirement is met. To define basic feasibility a transportation matrix T is built based on the list of the containers:

$$Tij \ : \ Number \ \ of \ \ containers \ \ going \ \ from \ \ port \ i \ \ to \ port \ \ j$$

$$Tij \ = \ 0, \forall j \ \geq \ i, \qquad 1 \leq \ i, \ j \ \leq \ N$$

If L_p and U_p are the lists of containers to be loaded and unloaded at port *p* respectively, then:

$$\left|L_p\right| = \sum_{j=p+1}^{N} T_{pj} \ \ , \ \left|U_p\right| = \sum_{i=1}^{p-1} T_{ip}$$

Since the unloading is done before the loading starts, the problem is feasible only if at each port there are at least $\left|L_p\right|$ cells available in the vessel after all $\left|U_p\right|$ containers are removed. A program source code is developed to generate such problems.

4.2. Model verification

To verify the accuracy of the mathematical model, formulations are generated for several sample problems and solved using CPLEX solver. A computer program is developed for analyzing the output by calculating some performance measures and visualizing the results.

4.2.1. Sample problem from Avriel and Penn (1993)

As it was mentioned in the literature review Avriel and Penn (1993) developed an integer programming model to minimize number of shifts in stowage planning with single size containers. Stability constraints are not considered in this model. A sample problem is reported in their paper and solved to the optimality. To make sure that the model in chapter 3 is able to produce optimal solution for the same problem, a formulation is generated by relaxing crane utilization and stability constraints. The containership in this example calls five ports. The ship has one bay consisting of two rows and five columns. Table 4.1 shows the transportation matrix.

Table 4.1: Transportation matrix for Avriel and Penn (1993)

From/To	2	3	4	5
1	4	4	2	0
2	0	2	0	1
3	0	0	0	5
4	0	0	0	1

Although the total number of containers is 19, the total number of unload operations is reported to be 20. This means that at least one shift is necessary. The output from the new model confirms this. Figure 4.1 is a graphical representation of port by port view of the solution. Each rectangle represents a container painted in two colors. The narrow color bar shows the origin port of the container and the wide bar is color coded to show the port of destination. The container assignment in this figure shows the stowage planning upon leaving the port. Some containers may be marked with a black or a red dot on their top right corner. The black dot means that the container will be unloaded at the next port while the red dot means that the container will be shifted at the next port. The black frame surrounding the container means that the container has been shifted in that port. In this example container 13 must be shifted at port 4.

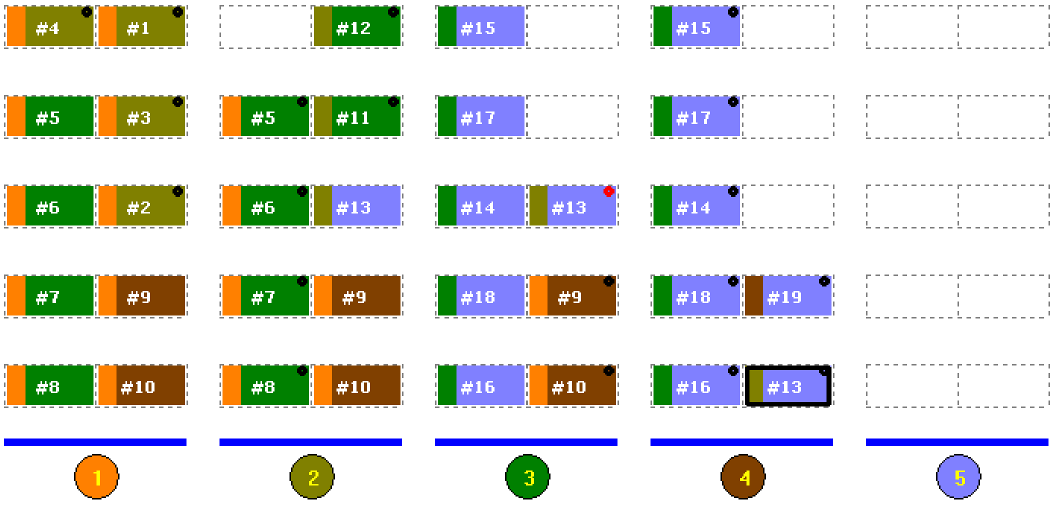


Figure 4.1: Solution to the original sample problem

4.2.2. Sample problem from Avriel and Penn (1993) with stability constraints

Assume that all containers in the previous example are of the same weight and each weigh 1 unit. It can be observed that the previous solution is not conforming with stability constraints since at ports 2, 3 and 4 total weight on the right side is not equal to the total weight on the left. If we allow 1 unit tolerance in the weight difference only ports 3 and 4 will violate the stability. We solve the problem again considering stability constraints with 1 unit threshold. The optimal solution can be seen in Figure 4.2.

Figure 4.2: Optimal solution with stability constrains

Number of shifts has increased to 2 in this case. This means that observing stability constraints may come at cost of extra shifts. Had we set the stability weight threshold to zero the problem would have been infeasible. This is because at ports 2 and 4 no arrangement of the containers will result in such balance. In the real operations imbalances are taken care of by using ballast water.

Consider the case that in the above example all containers with even numbers weigh twice as much as the ones with odd numbers. If we formulate and solve the problem based on these new assumptions and set the stability tolerance to 1 weight unit, the number of shifts in the optimal solution will be 3.

4.2.3. Effect of stack size on computational time

In this example, a containership will visit five ports. The transportation matrix is shown in Table 4.2, with a total of 38 containers of the same size and weight.

Table 4.2: Transportation matrix for the sample problem

From/To	2	3	4	5
1	8	8	4	0
2		4	0	2
3			0	5
4				2

Mathematical models are generated for five hypothetical containerships with approximately the same capacity (20-21 TEU) but different structures. To investigate the effect of stack size on computational time, the objective function minimizes the number of shifts at all ports while the crane and stability constraints are relaxed. Optimal solutions are obtained for all hypothetical containerships using the CPLEX solver. The ships’ structures and the corresponding computational times are summarized in Table 4.3.

Table 4.3: Ship configuration and running time for optimal solution

Ship	Bay	Row	Tier	Capacity (TEU)	Number of binary variables	Total number of constraints	Number of shift constraints	Running time (Second)
1	5	2	2	20	11020	86174	85674	521
2	4	1	5	20	11020	128384	127854	2011
3	3	1	7	21	11210	142595	142050	93465
4	2	1	10	20	11020	142454	141914	386142
5	1	1	20	20	11020	149489	148944	NA

The number of constraints in the formulation rises as the size of the stack increases. Most of the constraints relate to shift operations. It can be observed that for this example the running time grows dramatically with the increase in the number of tiers. For ship 5 the solver could not reach optimality in one week.

To investigate the pattern of the running time growth, four additional transportation matrices were generated and the solution was collected for ship structures 1 through 4 (optimal solution could not be obtained for containership 5). For each ship, average running time was calculated using the five recorded running times. The results are shown in Figure 4.3. The graph suggests that the running time rises exponentially as the height of the stack increases. This was expected from the literature since the problem is NP-Complete.

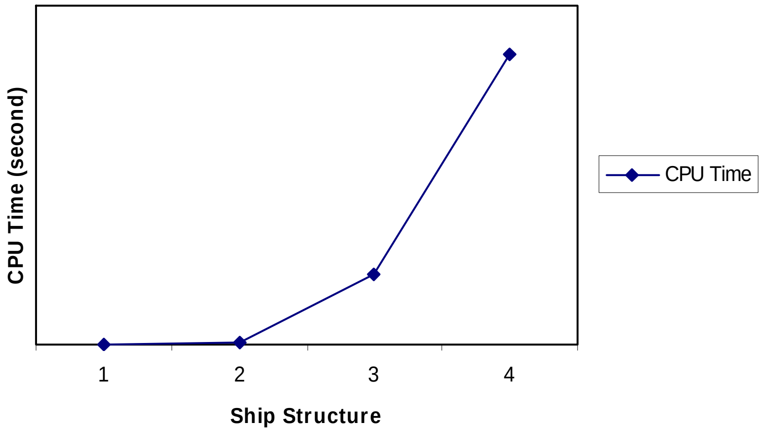


Figure 4.3: Average running time vs. ship structure

4.2.4. Sample problem for different size containers

This example aims to verify the situation of having a mix of 20’ and 40’ containers and the related operational policies. A containership with 5 bays, 1 row and 3 tiers is visiting three ports and is transporting 14 containers, 3 of which are 40’ and the rest are 20’. The stability constraints are in place with threshold set as 1 weight unit. All containers are of the same weight. If 40’ containers are not allowed on top of 20’ units, the optimal solution will look like what is shown in Figure 4.4.

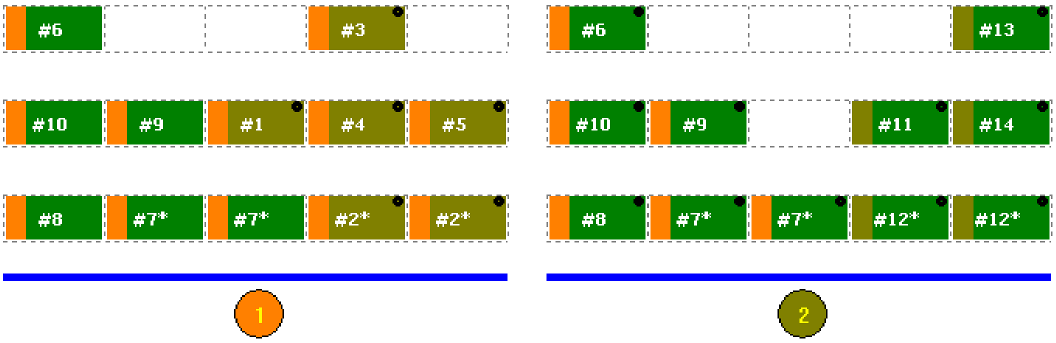


Figure 4.4: Results for no 40’ container on top of 20’ policy

Containers 2, 7 and 12 are 40’ and are marked with asterisk. Since this figure shows the stowage planning upon leaving each port, there is no need to display the results for port 3 because the ship is empty then. No shift is necessary in this case. However if the regulation is changed such that 20’ containers are not allowed on top of 40’ containers, shifting of container 7 at port 2 will be inevitable. Figure 4.5 shows the optimal solution based on this regulation.

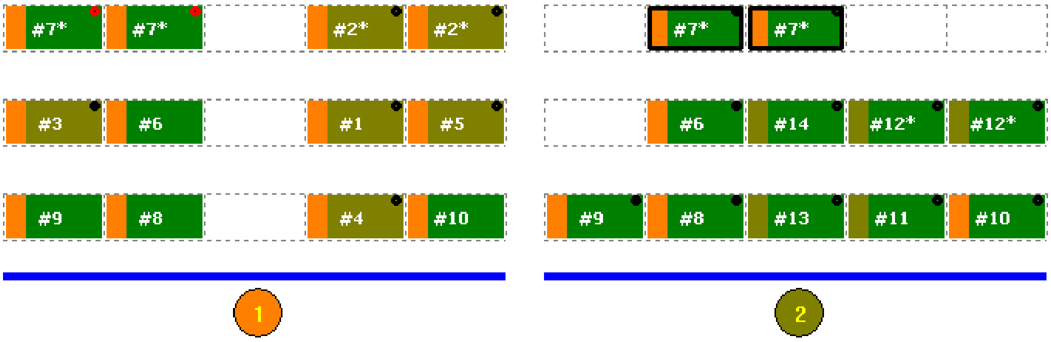


Figure 4.5: Results for no 20’ container on top of 40’ policy

4.3. Crane workload balancing

As it has been mentioned in the problem statement, the objective function of the stowage planning must be to minimize the total ship turnaround time at all ports. Part of this may be achieved by minimizing the shifts; however that is not the only factor. Knowing the assignment of quay cranes at teach port, efficient use of this equipment must be considered in the stowage planning. The next two examples try to highlight the difference between using minimizing total time at ports as objective function and the traditional stowage planning objective function which is the minimization of the shifts.

4.3.1. Crane workload balancing with single size containers

The containership in this example has 4 bays, 1 row and 5 tiers. There are five ports to be visited and 38 containers of the size 20' to be transported. The stability tolerance is set to a maximum of 1 weight unit. This example explores the effect of crane split on the solution. It is assumed that two cranes are available at each port, and each crane can handle one container per unit time. Handling includes loading a container at its origin, unloading at destination or shifting the position at an intermediate port. Four scenarios are compared in this example. In the first two scenarios all containers are assumed to have the same weight, while in scenarios 3 and 4, the containers departing from port 2 are four times heavier than the others. Scenarios 2 and 4 are the cases with the optimization of total turnaround time as the objective function while scenarios 1 and 3 are the classic stowage planning problems. Figure 4.6 shows the stowage plan for the second scenario.

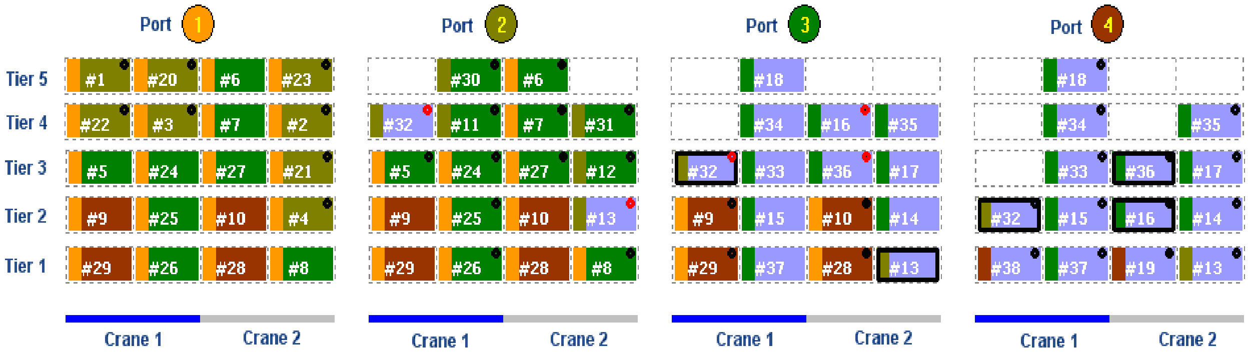


Figure 4.6: Stowage planning results for scenario 2

The operational range of each quay crane is shown in Figure 4.6. The last port is not displayed since the ship is empty after the operations are completed at that port. Total number of container handlings, total time spent at all ports, utilization of each crane and average carne utilization for each scenario are presented in Table 4.4. Utilization of each crane at each port is calculated by dividing the crane busy time over the total time that the vessel spends at the port. For scenarios 1 and 3 total time at all ports is a direct output of the model and the total number of handlings is calculated using the final value of the decision variables. The reverse happens in scenarios 2 and 4 in which the total number of handlings is obtained by the objective function and total time at all ports is calculated using the output variables.

Table 4.4: Summary of the results for four scenarios

	Port 1		Port 2		Port 3		Port 4		Port 5		Total number of container handlings	Total time at all ports	Average crane utilization %
	Crane utilization %		Crane utilization %		Crane utilization %		Crane utilization %		Crane utilization %				
	1	2	1	2	1	2	1	2	1	2			
Scenario 1	100	100	100	56	73	100	20	100	100	100	42	46	84.9
Scenario 2	100	100	100	100	100	85	100	80	100	100	43	40	96.5
Scenario 3	100	100	100	100	100	77	100	60	100	100	41	42	93.7
Scenario 4	100	100	88	100	92	100	100	100	100	100	42	41	98

The number of handlings in scenario 2 is slightly greater than scenario 1, however the average crane utilization is 14% higher than that of scenario 1. By distributing workload between cranes properly, 13% improvement in total berthing time is achieved in this example. Similar comparison between scenarios 3 and 4 shows 2.5% improvement in berthing time. Solutions details including the input and output for scenario 3 are provided in Appendix B.

4.3.2. Crane workload balancing with different size containers

This example illustrates a case with mixed size containers. A containership with six bays, two rows and three tiers will visit four ports. There are 44 containers to be transported of which 16 are 40’ and 28 are 20’ containers. It is assumed that there are two cranes available at ports 1 and 4, and three cranes at ports 2 and 3.

Similar to previous example two scenarios are tested. The summary of the results is shown in Table 4.5.While total number of handlings is equal in both cases, optimizing the crane utilization has improved the total berthing time by 7% in this example.

Table 4.5: Summary of the results for two scenarios

	Port 1		Port 2			Port 3			Port 4		Total number of container handlings	Total time at all ports	Average crane utilization %
	Crane utilization %		Crane utilization %			Crane utilization %			Crane utilization %				
	1	2	1	2	3	1	2	3	1	2			
Scenario 1	100	93	40	40	100	80	80	100	100	100	61	57	83.3
Scenario 2	93	100	100	86	71	100	93	86	100	100	61	53	92.9

4.3. Summary

The examples in this chapter demonstrate the potential for saving in the total ship turnaround time if an appropriate objective function is used in stowage planning optimization. While minimizing shifts helps to reduce the turnaround time, it should not be used as the objective function. The results show that concurrent maximization of crane utilization and minimization of shifts improves the overall turnaround time at all ports. The model balances the tradeoff between the crane utilization and extra container movements.

The results also show that the model in its current form is capable of handling required operational and technical constraints. Unfortunately since the problem is NP-Complete even the small size problems can be very computationally expensive. Running time is highly correlated with the height of the stack and grows exponentially as the stack size increases. No analytical method exists for solving multi-column stacking problem with stability constraints according to the literature. Thus heuristics are needed to deal with the real size containership loading problem.

Chapter 5: An algorithm for containership load planning optimization

5.1. Introduction to optimization

According to the Merriam-Webster dictionary the optimization is defined as “an act, process, or methodology of making something (such as a design, system, or decision) as fully perfect, functional, or effective as possible”. Optimization techniques have been used in a broad range of engineering applications in order to find the best possible solution within the limits and constraints of a problem. Differentiation and gradient based optimization, hill climbing and linear programming are among the well known traditional mathematical approaches for solving engineering problems. However in many real optimization problems the corresponding mathematical function is not well-behaved, the solution space is discrete or the problem is multiple criteria. In such cases these conventional methods will either fail to cope with the complexity of the problem or simply need extensive computational resources. Solution techniques such as evolutionary algorithms will be helpful in this kind of situation.

5.1.1. Evolutionary algorithms

Conventional optimization techniques are often incapable of dealing with non-linear multi-criteria optimization problems. In such cases a random search in the solution space in hope of finding the optimal feasible point is an alternative method. However performing a random search in an unsystematic manner can be extremely inefficient. Many efforts have been made to add intelligence to the random search procedures in the past decades. Evolutionary Algorithms as a class of intelligent search methods are results of such efforts.

Evolutionary Algorithms imitate the mechanisms inspired by biological evolution namely reproduction, mutation, recombination, natural selection and survival of the fittest. Individuals of the population are represented by candidate solutions to the optimization problem, and the fitness function determines the environment within which the solutions live. Evolution of the population is then simulated by applying the above operators iteratively. Genetic Algorithm is the most popular variant in this class.

Evolutionary Algorithms are not the only intelligent search methods inspired by ideas from the nature. Simulated Annealing, Tabu Search, Ant Colony and Harmony Search are examples of meta-heuristics which work based on the behavior of natural systems.

Simulated Annealing is based on the process of heating and controlled cooling of a material. It basically traverses the search space by replacing the current solution with a random nearby solution. The neighbor will be accepted if it is superior. For an inferior neighbor the acceptance probability depends on the difference between the corresponding function value and a global temperature parameter. Altering the temperature parameter during the process modifies the nature of the search.

Tabu Search is similar to the Simulated Annealing with the difference of generating more than one neighboring solution at each step. It moves to a better mutated solution by picking the neighbor with best fitness of those generated. Cycles are prevented by maintaining a tabu list of solutions which is being updated throughout the process.

Moving to the solutions that contains elements of the tabu list is prohibited. The idea came from observation of human behavior which appears to operate with a random element leading to inconsistent behavior given similar circumstances (Glover and Laguna 1997).

Ant Colony Algorithm is a probabilistic technique for optimization that hires a large number of artificial ants to incrementally build the final solution. It works by mimicking the movements of the ants in the real world. While searching for food or returning to their colony, ants lay down pheromone trails on their path which will be used by later ants to guide their search. As the time goes by, however, the pheromone trail starts to evaporate, lowering the attractiveness of the trail. This technique usually outperforms other meta-heuristics in routing problems such as traveling salesman problem when the graph changes dynamically (Dorigo Marco, Thomas Stützle 2004).

Harmony Search is another meta-heuristic which simulates the improvisation process by a musical band. While improvising each musician plays a note until finding the best harmony all together. Based on this idea decision variables in an optimization problem will accept different values and interact with each other to find the best solution vector all together.

5.1.2. Genetic Algorithms

Genetic Algorithm (GA) is an adaptive heuristic search method based on the evolutionary idea of natural selection which represents processes in natural system for evolution, specifically the principle of survival of the fittest by Charles Darwin. As such it performs an intelligent directed random search within a defined search space to optimize a problem. Genetic Algorithms use a vector of numbers to represent decision variables. They pursue an iterative process in which several solution points are being explored simultaneously at each step. The only information required for search is a fitness assessment. An illustration of the general process is shown in Figure 5.1.

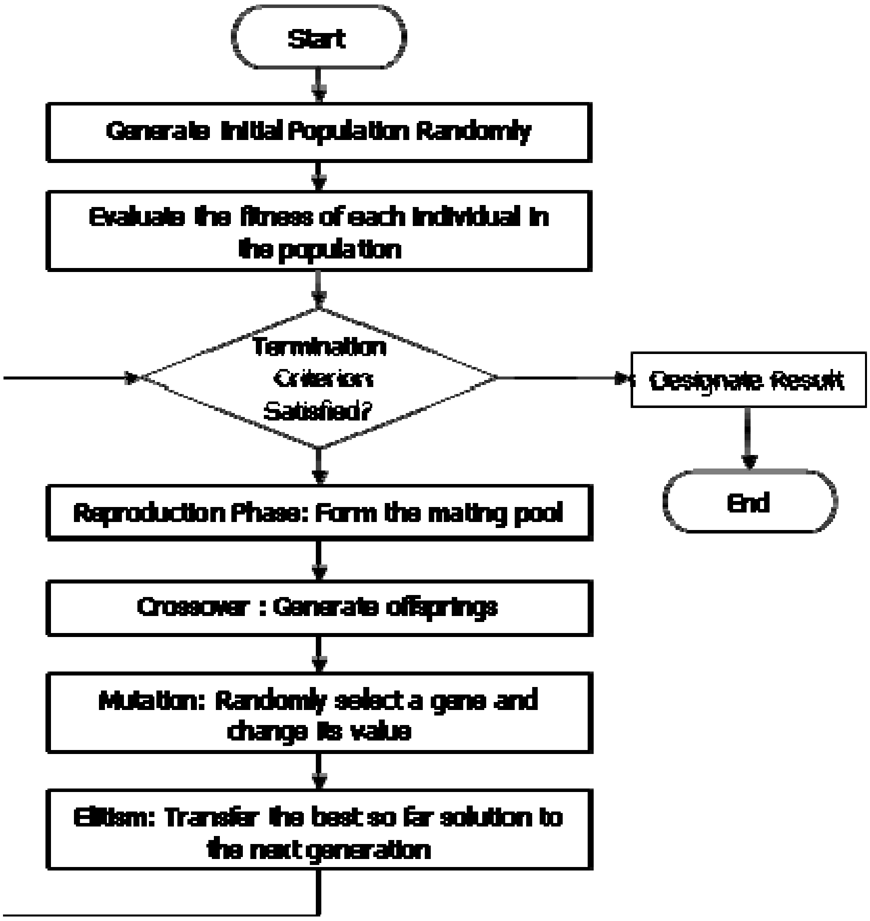


Figure 5.1: Flowchart of Genetic Algorithm

In summary genetic algorithms have been widely used to solve optimization problems where the analytical and other evolutionary methods fail. Some advantages and features of the genetic algorithms are as follow:

- They run simultaneous search over multiple regions of the search space rather than a unique point since a population is investigated at each step. This makes genetic algorithm suitable for parallel computing.
- They work with both continuous and discrete parameters as well as combination of them.
- They can handle a large number of parameters.
- They do not need a profound knowledge about the mathematical structure of the solution space in order to solve the problem.
- They can optimize multi-objective optimization problems to provide a list of solutions instead of a single one.
- Genetic algorithms are stochastic, not deterministic.
- They can work with incomplete information and noisy data.
- Genetic algorithms are flexible in cooperating with other techniques and can be part of hybrid methods.
- They have been successfully applied to a large number of complex problems in different fields of engineering.
- Genetic algorithms perform a very large number of objective function evaluations. Hence if such evaluations are computationally expensive, then the convergence might take a long time.

The following definitions are often used in genetic algorithm literature:

- **Chromosome:** The data structure that holds a potential solution.

- **Gene:** Fraction of a chromosome that represents a parameter in a potential solution.
- **Individual:** Collection of a chromosome and its fitness value.
- **Alleles:** The set of values that a gene can accept.
- **Locus:** The position of the gene on the chromosome.
- **Genotype:** In Biology a genotype is the total genetic information of an organism or phenotype and it can consist of one or more chromosomes. In most genetic algorithm applications however, a unique chromosome contains the total genetic information of the organism (solution), so this unique chromosome also represents the genotype of the organism. Because of that the terms genotype and chromosome are often used interchangeably in genetic algorithms context.
- **Phenotype:** The solution or organism built based on a genotype. For example if a chromosome represents the location of the containers in a containership stowage plan, the encoded locations will represent the genotype of the stowage plan while the actual vessel loaded based on that stowage plan makes the respective phenotype.

The complete terminology of the genetic algorithms can be found in Rawlins (1991).

5.2. Genetic Algorithm for containership load planning

As it was mentioned in the literature review in Chapter 2, containership stowage planning with uniform size containers is NP-Complete. Introducing multiple-size containers and crane assignment to the stowage planning adds to the complexity of the problem and because of that the mathematical model described in chapter 3 is incapable of solving the real size problems. To overcome the large number of variables and constraints, the model can be reformulated to assign groups of containers to the bays of the vessel rather than individual ones. Grouping can be done based on a given property such as size, type or destinations and this will reduce the problem size significantly and such model can be solved using branch and bound methods. The assignment of individual containers into the cells in each bay is done in another step. However the drawback of the simplification by decomposition is the inflexibility in dealing with constraints specially while trying to optimize crane utilization. This method accompanied by a tabu search was developed by Wilson and Roach (1999), (2000) to solve stowage planning. One of the most challenging issues in combinatorial optimization is to deal with the combinatorial explosion effectively, such that the algorithm can generate solutions to the real world size problems in a timely manner. Genetic Algorithms have been successfully applied to the containership stowage planning problem before by Todd and Sen (1997) and Debrovsky et al. (2002). They solved instances of the containership where all the containers are of the same size. The former researchers used transverse as well as vertical center of gravity to address the stability while the latter group only used the horizontal equilibrium.

5.2.1. Genetic encoding

In a genetic algorithm each potential solution is represented by a string with a fixed bit-length known as chromosome that encodes the decision variables. This representation is a key part of the genetic algorithm because the genetic operators directly manipulate the chromosomes as representatives of the solutions. “The complexity of a problem largely depends on the interactions between variables of a solution. A stochastic search process like evolution will perform well on a complex problem only when the search distribution is adapted to these interactions, i.e., when the search distribution obeys these dependencies between variables (Toussaint 2005)". To have a successful and efficient use of the genetic algorithm, it is crucial to find a proper representation of the problem, also known as genetic encoding and develop appropriate operators that conform to the characteristics of the problem. In other words the efficiency of the natural evolution process to perform an intelligent or learned exhaustive search for optimal or near optimal solution within a complex structure in which the fitness is measured by the interaction among variables, highly depends on a genetic representation that is both expressive and evolvable. In most genetic algorithms the individuals are represented by fixed-length binary strings that consist of genes with values of 0 or 1. The genetic encoding does not have to be binary, other types of encoding such as real-number encoding, integer or literal permutation encoding, and general data structure encoding can be used for different optimization problems. According to Collins and Eaton (1997) there does not exist a single encoding strategy that performs well on all optimization problems.

Infeasibility and Illegality are two common issues while developing the genetic encoding. There are two categories of spaces in each genetic algorithm: genotype space and phenotype space. Genetic operators work on genotype space where they manipulate different parameters of the problem. Evolution and selection on the other hand are done in phenotype space where the chromosomes are being evaluated. The mapping from genotype to phenotype space is a major contributing factor in the performance of the genetic algorithm. Infeasibility happens when a solution decoded from a chromosome falls outside of the feasible region of the problem. Illegality refers to the case that a chromosome does not correspond to any solution of given problem at all. Infeasibility originates from violation of constraints and by penalizing the unsatisfied constraints the algorithm will direct the search toward the feasible region. Illegality however is a result of genetic operators where a generated offspring does not represent a valid solution to the problem. A good genetic encoding and proper design of genetic operators decreases the illegality. “Because an illegal chromosome cannot be decoded to a solution, the penalty techniques are inapplicable to this situation. Repair techniques are usually adopted to convert an illegal chromosome to a legal one (Cheng and Gen 2000)”. Figure 5.2 shows the phenotype and genotype space for a typical problem.

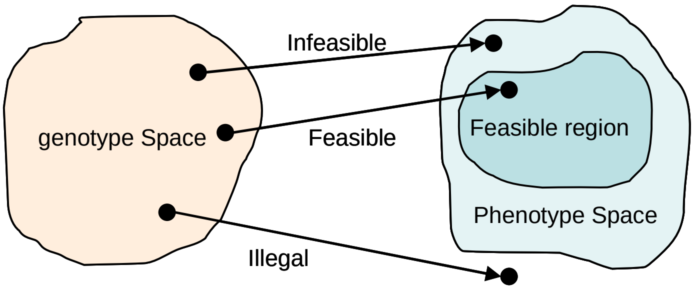


Figure 5.2: Infeasibility and Illegality

A new genetic encoding for containership loading problem is designed in this dissertation. Before going through the details of the new encoding, existing genetic representations including complete and compact encoding for containership stowage planning problem are reviewed. The terms complete and compact encoding in stowage planning was first introduced by Debrovsky et al.(2002).

Complete encoding by Todd and Sen (1997)

In a complete encoding the whole layout of the vessel at each port is encoded into the chromosomes. In other words each gene represents a cell and alleles are the set of all the container numbers to be transported. So for an X-TEU containership visiting N ports the genotype consists of a string of integers with a length equal to $N \times X$. This encoding is trivial and easy to implement but it has some shortcomings. Evaluating the fitness of the chromosomes requires the processing of the complete layout for four criteria: unloading, proximity, transverse center of gravity and vertical center of gravity. This is very time consuming especially when the population size goes up. But the main drawback is the illegality of the resulting offspring by the crossover operator. The crossover operator is designed to restrict an individual to mate only with the individuals who are located in its close surroundings in the criteria space. Omission and duplication of containers in the resulting strings can occur as a result of such restriction. To fix the inconsistency in the results, a repair procedure is used which manipulates the offsprings after the crossover is done. This repair routine is not only time consuming, but also will partially destroy some of the inherited information that are accumulated during previous iterations.

Compact encoding by Debrovsky et. al. (2002)

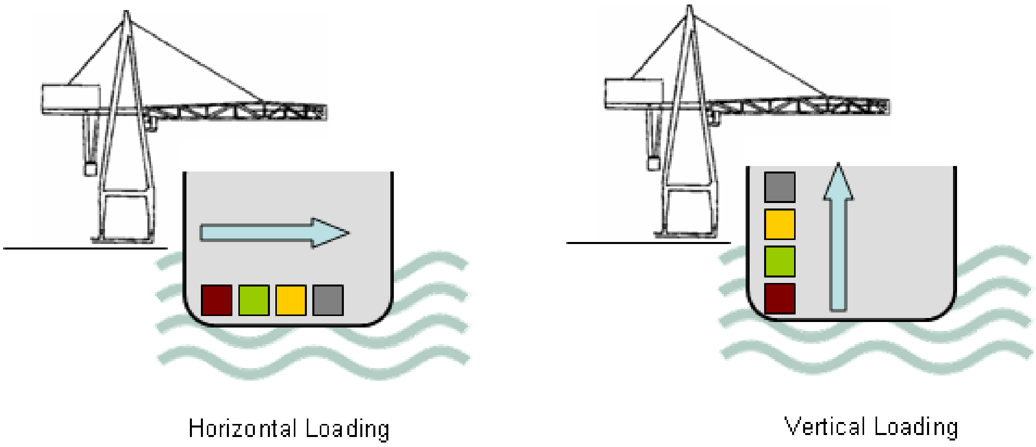
To overcome the disadvantages of the complete encoding, the compact encoding introduces a new representation which stores only the changes in the layout that result from the loading and unloading along the route instead of the complete layout. It reduces the processing time for evaluating the chromosomes, preserves the consistency of the layout, insures the legitimacy of the crossover operator and allows convergence to good solutions within a reasonable time by decreasing the search space and storage resource consumption. For a containership visiting N ports, the genotype is divided into N sections. Each section consists of four lists: (1) list of columns for loading the containers originated at the corresponding port, (2) list of columns for loading the containers that were unloaded due to necessary shift, (3) list of columns for loading the containers that were unloaded due to voluntary shift, (4) list of columns from which the containers should be unloaded because of voluntary shifts. To simulate the ship unloading and loading operations two auxiliary vectors are hired for each port. One of these vectors contains the destinations of the loading containers which can initially be acquired from the transportation matrix. The other vector which is two dimensional is a column waiting list which keeps the column information for the containers to be loaded at the port and is obtained from decoding the corresponding solution chromosome. The total number of shifts will be known only after running the solution decoding procedure.

Although this encoding has reported to be more efficient than the complete encoding, it does not account for multiple-size containers. Both complete and compact encodings presented above must undergo significant changes to be able to handle mix of 20 and 40 ft containers. The reported real size problem solved using this encoding had single size and uniform weight containers. Handling the stability was demonstrated by keeping the horizontal tilt within a threshold. However to enforce the vertical stability constraints which are a major source of mandatory shifts would be challenging for this method because of the genotype structure.

Assignment policy based encoding

As it was mentioned before the containers differ in weight, size and type. Export containers arrive to the container yard before the containership arrives, although some of them might arrive while the loading of the vessel has already started. Due to the shortage of space the terminal managers usually stack the containers in the terminal. It is a common practice to group the containers based on properties such as weigh, size or destination and then allocate the groups to the yard stacks .If the configuration of the stacks conforms to the stowage plan of the containership the unnecessary reshufflings of the containers at the yard can be minimized.

For loading the containers into the vessel on the other hand different strategies exist. The quay crane drivers may load the containers into the cells at one row and then move to the adjacent row, or they may fill up one column and then move to the next column in the row. Figure 5.3 illustrates these strategies.



Horizontal Loading

Vertical Loading

Figure 5.3: Horizontal and vertical loading strategies

The key idea for assignment policy based genetic representation comes from the combination of above grouping and loading strategies in the container terminal. In fact instead of searching for the favorite container to cell assignment pattern the search can be done to find the best combination of sorting, grouping and loading strategy at each port. Each chromosome string consists of several sections each of which corresponding to a visiting port. Each section then is divided into two subsections to represent sorting method and assignment strategy respectively. These subsections are represented by binary strings.

Four basic properties of the containers are size, weight, destination and type. One can sort a list of containers based on each of these criteria or any combination of them. For example the list can be sorted by destination only or by destination first, then weight and then size. Furthermore each criterion can be applied ascending or descending. Total number of sorting possibilities can be calculated as follow:

Total sorting combinations: $\sum_{r=1}^{4} P_r^4 \times 2^r$ = 632

There might be cases that none of the above sorting methods will suit the problem. To address this situation another sort based on random key is added to the pool of sorting options, so that containers can be retrieved and allocated in a predetermined random order.

Containers in the sorted list then can be retrieved and assigned to the available cells in the vessel. Initially at the first port all the cells are available. After berthing at each forthcoming port, first the containers that have reached their destinations will be unloaded. Because the containers are only accessible from the top of the stack, all the containers that block the access to container that should be unloaded must be removed first. These are the containers that have not reached their final destination yet, but must be shifted in order to allow access to the container below them. These containers must be loaded back into the vessel, along with the containers that originate from the current port. They may be given a location that is different from their former location in the ship. The recently emptied cells will be included in the set of available cells in the vessel. The allocation of export containers can be done horizontally or vertically as illustrated in Figure 5.3. In vertical policy the cells of each column are assigned from bottom to the top due to the fact that excluding the most bottom cell, a container can be stored in a cell only if the cell under it is not empty. For horizontal policy cells can be picked row-wise or bay-wise. In other words the cells can be picked horizontally either from bow side to the stern side and then from shore side to the water side or vice versa. In addition to that there are plenty of other orders in which the bays and rows can be picked. For example in selecting the bays one strategy can be picking one bay after another from bow to stern while an alternative strategy is to pick every other bay from the opposite direction. Combination of vertical and horizontal assignment strategy and the orders in which cells can be picked in each strategy creates a pool of possibilities to choose from. Stowage plan at each port can be constructed by allocating the sorted list of containers to the containership cells according to the encoded sorting method and assignment strategy for that port. Figure 5.4 is a visual representation of this concept. The generated stowage plan then can be analyzed to evaluate the fitness of the solution.

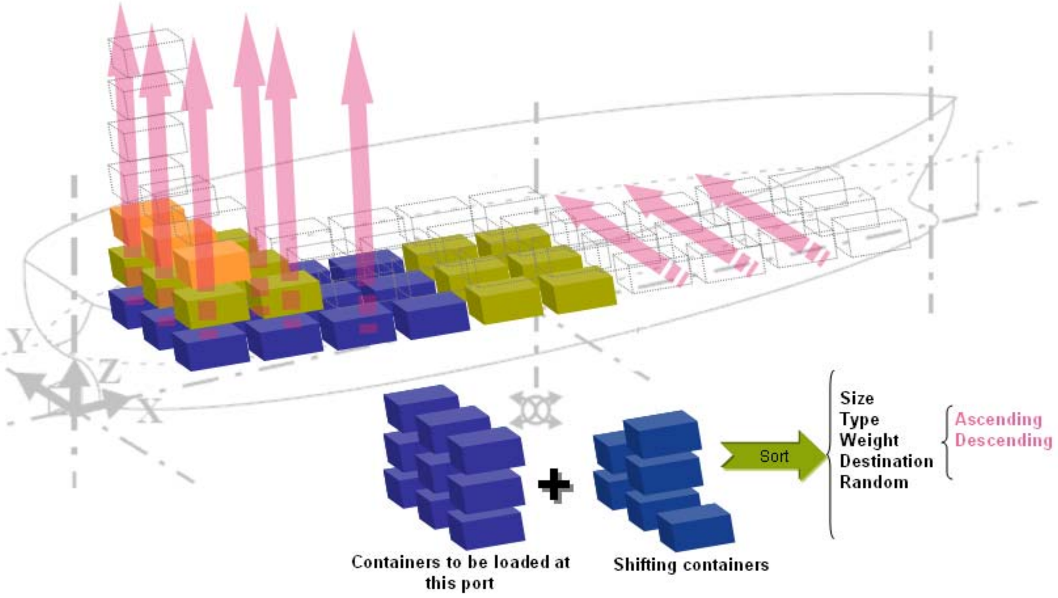


Figure 5.4: Different sorting and assignment policies for loading a containership

Usually a major part of the stowage pattern remains intact between two consecutive ports. Its implication for the new encoding is that the stowage plan for the next port will be built based on the remaining stowage pattern at the current port after unloading is complete. However based on the structure of the program and the characteristics of the containers no feasible or competitive solution might be generated if the existing stowage plan is fixed. To address that issue and to make sure that the genetic algorithm can exploit the solution space effectively an extra bit is added to the genotype which determines whether the existing stowage plan at the current port should be relaxed. In other words if the aforementioned gene has the value of 1, all the containers already loaded in the vessel will be subject to cell allocation along with the export and shift containers. Although a large number of these containers might end up staying in the same positions, it is important to keep the options open for the algorithm to look for optimal allocation. Figure 5.5 shows an illustration of assignment based genetic encoding.

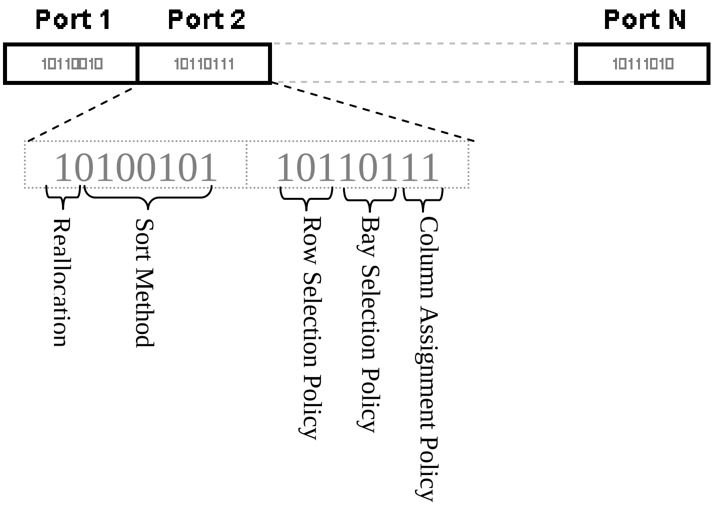


Figure 5.5: Illustration of assignment policy based genetic encoding

It can be observed that the length of the solution string is a function of the number of ports and is independent of the number of containers and the layout of the vessel. The sort and assignment policy information for each port can be stored in 4 bytes, so the length of each chromosome is $4*N$ byte which is very compact and memory efficient.

5.2.2. Evaluation of solution

The selection mechanism of genetic algorithm looks at the fitness value of each chromosome to decide its fate. The better the value of the fitness is, the higher is the chance of advancing to the next generation. The objective function of the containership loading problem with quay crane assignment consideration is to minimize total turnaround time at all ports. This is an appropriate objective function if the value of time at all ports is equal. Otherwise by multiplying the turnaround time at each port by its cost parameter we can build an economical objective function to minimize total cost of the operations at all ports. However the objective function value is one of the components of the fitness function. With the presence of stability and operational constraints, more components must be added to the fitness evaluation function to penalize the violation of the corresponding constraints. Imposing penalty to the solutions that do not satisfy the constraints helps the genetic algorithm and its operators to move toward the feasible area of the solution space. Evaluation of a chromosome needs four steps: first is to decode the solution, second is to transform the decoded genotype to the corresponding phenotype, third is the analysis of the resulting phenotype and the final step is to calculate the final fitness value by summation of normalized objective function and the penalties.

Decoding a solution

Since all the information is in binary string format, bitwise operators are hired for decoding the solution. For every section of the chromosome, the locus (position of genes in the genotype) is known and fixed. By using binary shift operator, the desired section can be repositioned to the rightmost part of the chromosome string. In a unary right shift operation, all the bits in the binary string are shifted to their immediate right position, the first bit will be lost and a 0 will fill the empty position of the leftmost bit. A binary mask is needed which has the same length as the chromosome. Every bit in the mask equals 0 except for the k rightmost bits where k is the length of the section to be decoded. The shifted chromosome and the mask serve as the operands to a binary “AND” operator which extracts the binary value of the section. This binary number then is converted to decimal to represent the gene value. The example below shows how to extract the vertical assignment policy from the given chromosome. We already know that this policy is stored at the two rightmost bits of the string so no binary shift is necessary in this case:

Solution:	100101011011011110100110
Mask:	0000000000000000000000**11**

Result (Binary):	000000000000000000000010
Decimal value:	2

Assuming bits are numbered from right to left starting from 0; if the bay selection policy is stored in bits 2 to 5 then the following example shows how to decode that information:

Solution:	100101011011011110100110
Binary shifted solution:	001001010110110111101001
Mask:	00000000000000000000**1111**

Result (Binary):	000000000000000000001001
Decimal value:	9

After all sections of the chromosome are decoded the corresponding stowage pattern can be constructed.

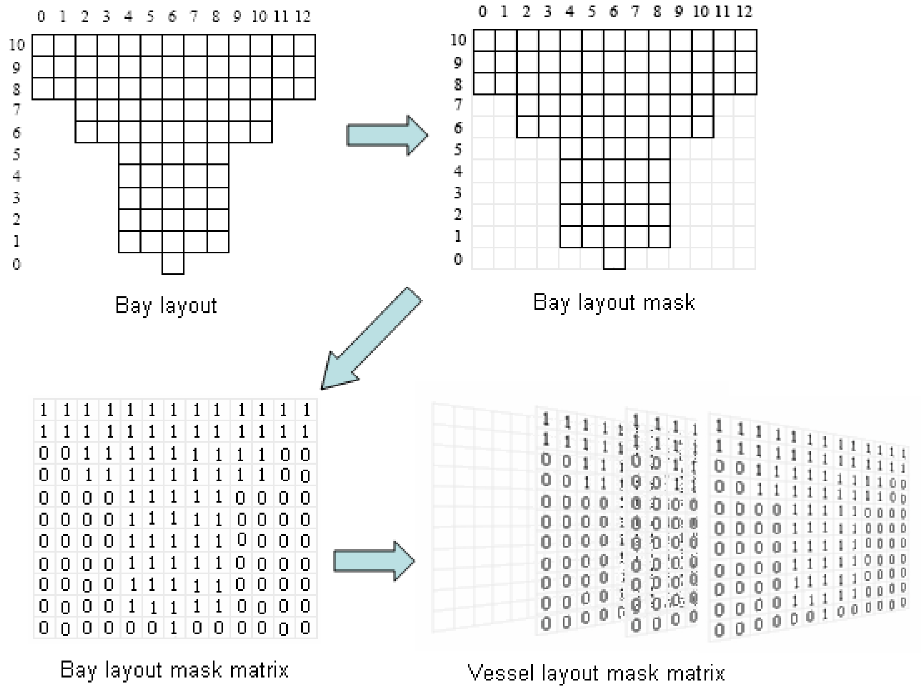
Creating the stowage pattern

As it was shown in Figure 3.2, the general layout of a containership can be looked at as a three dimensional matrix. This however is not true in the real world since the hull shape of the vessels and the location of the engine and accommodation rooms are different at each vessel. To keep the generality and without losing flexibility, a four dimensional data structure is designed to save the produced stowage pattern by a decoded solution. This data structure is referred to as the allocation matrix. The fourth dimension in the allocation matrix corresponds to the ports of visit and the three dimensions correspond to the cellular structure of the vessel. Each element in this data structure holds an integer value which shows the container number to which the cell is allocated to. In order to account for the containership design an extra three dimensional matrix of integers called layout mask matrix is introduced. The layout mask matrix has the same dimensions as the allocation matrix and is constructed based on the specific design of the vessel. Prior to allocating a container to a cell, the counterpart element in the layout mask matrix is checked, if it holds a value of zero it shows that the cell does not physically exist and thus cannot be allocated. Figure 5.6

demonstrates a sample bay from the cross section of a sample vessel and the construction of the vessel layout mask matrix.

The layout mask matrix is an input to the algorithm. If the layout design of the vessel is available in electronically interchangeable format (e.g. XML[3]), the mask matrix can be automatically created based on that.

The allocation matrix is empty at the beginning of the evaluation and will be filled up by the container numbers based on the decoded solution and the layout mask. Let $\Pi(p)$ be the stowage pattern at port p and N be the total number of visiting ports. Also let U_p, L_p and S_p be the set of containers that must be unloaded, loaded and shifted at port p respectively.



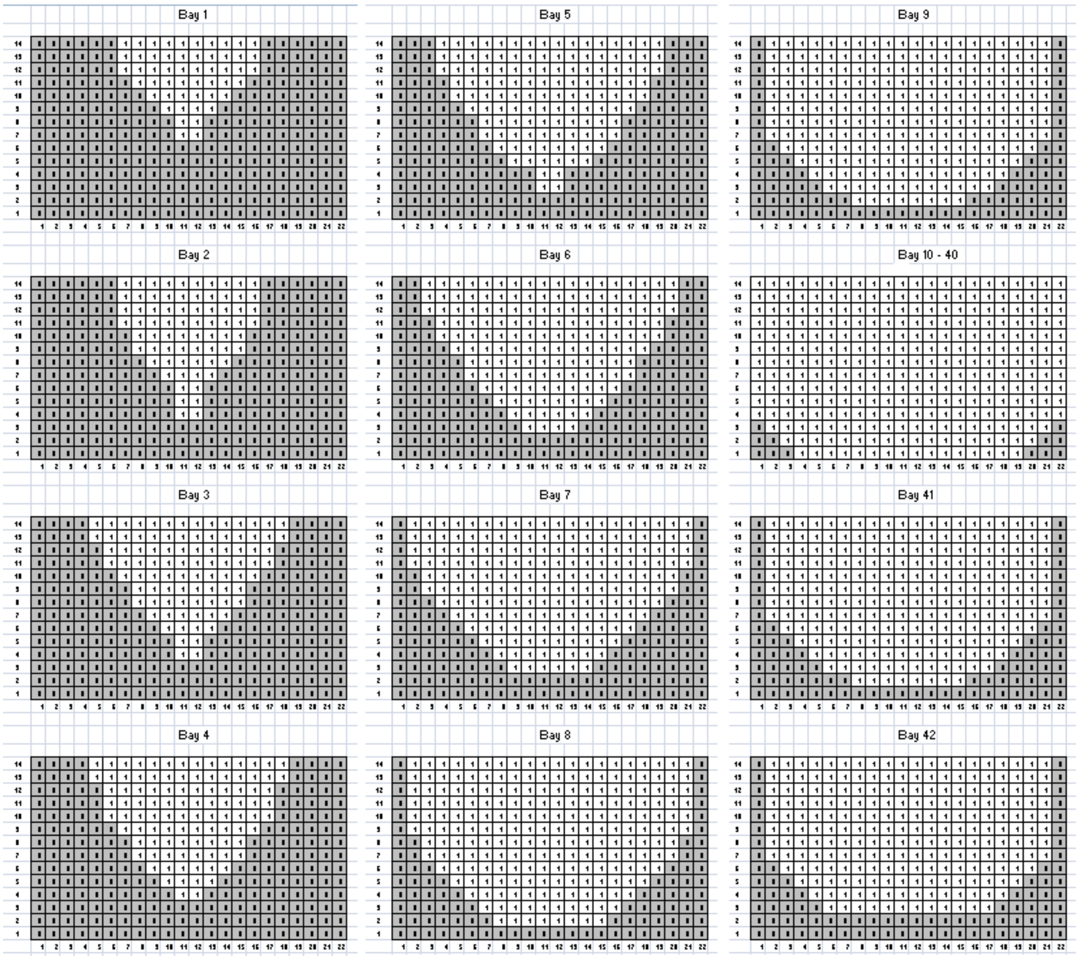
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 5.6: Construction of the vessel layout mask matrix

---

[3] Extensible Markup Language

Figure 5.7 shows the vessel layout mask matrix designed for a 10500 TEU containership with 42 bays, 14 tiers and 22 rows. The layout can be further customized to account for the restricted areas of the vessel that may either be not be accessible temporarily or permanently.



Bay 1 Bay 5 Bay 9

Bay 2 Bay 6 Bay 10 - 40

Bay 3 Bay 7 Bay 41

Bay 4 Bay 8 Bay 42

Figure 5.7: Sample of vessel layout mask matrix for a mega containership

The following procedure shows the main steps for creating the stowage pattern.

Procedure: Create Stowage Pattern

```
Initialization: $\Pi(p) \leftarrow \Phi$
For $P \leftarrow 1$ to $N$
{
  Decode the gene values for port p.
  Set: $\Pi(p) \leftarrow \Pi(p-1)$.
  Un-assign all the containers in $U_p$ from $\Pi(p)$ and calculate $S_p$
  If reassignment bit is 0 then
        $T = \{L_p, S_p\}$
  Else
        $T = \{L_p, S_p, all \quad containers \quad aboard\}$
        $\Pi(p) \leftarrow \Phi$
  Sort T according to the decoded sorting method
  Allocate all the containers in T to $\Pi(p)$ with respect to the decoded assignment
  policy and the layout mask matrix
}
```

Calculating the crane operations

By now the stowage pattern for all ports from the solution has been generated and saved in the allocation matrix. This matrix must be analyzed to measure different contributing factors in the fitness function. A simple procedure is developed to count the number of container loading, unloading and shifting at each port and for each crane. Since the loading starts only after all the unload containers are processed, this procedure starts with the containers that are to be unloaded. Those containers are the containers that either have reached their final destination, or must be unloaded temporarily in order to allow access to the containers under them. If containers of different size are mixed, the procedure accounts for the proper number of crane operations to retrieve a container. Figure 5.8 depicts an example.

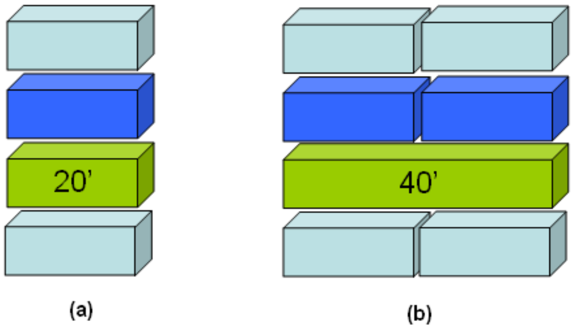


Figure 5.8: Stowage of different size containers

In figure 5.6 the two designated containers have reached their destinations while the other containers are to be transported to the forthcoming ports. In case (a), the two containers on the top must be also unloaded before then the 20' container can be retrieved and then must be put back to the vessel which accounts for a total of 5 crane operations. The total number of crane operations for unloading the 40' container in (b) will be 9 since all four containers on top must be shifted. This is based on the assumption that quay cranes can handle only one container per move regardless of the size of the container. This assumption however is not restricting since the procedure can be modified to support different crane characteristics such as double lifting. Total number of bays on the vessel is denoted by B. Consider $C: N \times B$ an array of integers for storing the total number of required crane operations per each bay. The create stowage pattern procedure must be called and $\Pi(p)$ should be populated before the number of crane operations can be counted. The crane operations counting procedure is as follow.

Procedure: Count crane operations

```
Initialization: $C \leftarrow \Phi$
For $P \leftarrow 1$ to $N$
{
    For $b \leftarrow 1$ to $B$
    {
        For each container $j$ in $U_p$ that belongs to bay $b$ in $\Pi(p)$
            If there is no container on top $j$ then
                $C[p,b] = C[p,b] + 1$
                Remove $j$ from $\Pi(p)$
            Else
                For each container $k$ blocking access to $j$
                    Remove $k$ from $\Pi(p)$
                    $C[p,b] = C[p,b] + 1$
                    If $P$ is the final destination of $k$ then
                        Remove $k$ from $U_p$
                    Else
                        Add $k$ to $S_p$
        For each container $j$ in $L_p \bigcup S_p$ that belongs to bay $b$ in $\Pi(p)$
            $C[p,b] = C[p,b] + 1$
    }
}
```

Calculating turnaround time at each port

After creating the stowage pattern and counting total number of crane operations at each bay, the ship turnaround time can be calculated. Turnaround time of a vessel at a port also known as dwell time is the difference between the time that the vessel berths, and the time that she is released and may leave the port. Since the time at port is very valuable we assume that unloading the vessel starts right after berthing and the vessel leaves the port after last container is loaded and the crane operations do not suspend in the middle of operation. This requires careful coordination and planning at the container yard between the yard cranes and internal trucks, straddle carriers, AGV’s or any other form of in yard equipment used for horizontal transportation of the containers. While unloading the vessel, movement of the import containers from the vessel to the yard should be planned properly such that quay cranes do not waste any time waiting for horizontal transportation. Also at the loading time a smooth flow of the export containers from the yard to the vessel will keep the quay cranes busy during the operations.

Quay cranes have different technical specification which determines their performance and capacity. Other factors such as weather conditions and visibility may also affect the performance of the cranes. Although the quay cranes at a specific port are usually homogenous, to generalize the solution algorithm we assume that not only the quay cranes at ports of call could be different, but also individual cranes at each port may be non-homogenous. We also assume that all the cranes that are assigned to a vessel work side by side and the vessel can be released only when the last remaining container is loaded onto the vessel by the respective crane. The horizontal divide and the operational range of the cranes are determined by the technical constraints. Generally a crane may not interfere with a neighboring crane on a common bay, so the minimum unit of horizontal separation between two adjacent cranes is one bay. Utilization of a crane can be defined as the percentage of the time that the crane is busy and is calculated by the total crane busy time over the turnaround time at port. The crane busy time is composed of two components: horizontal movement time and container handling time. The horizontal movements do not occur very frequently and are required only when the crane has finished the job on a bay and needs to proceed to the next bay. So the major fraction of the crane busy time is the container handling time. Container handling is performed using the spreader arm and is composed of one horizontal and two vertical moves by the spreader arm as shown in Figure 5.9.

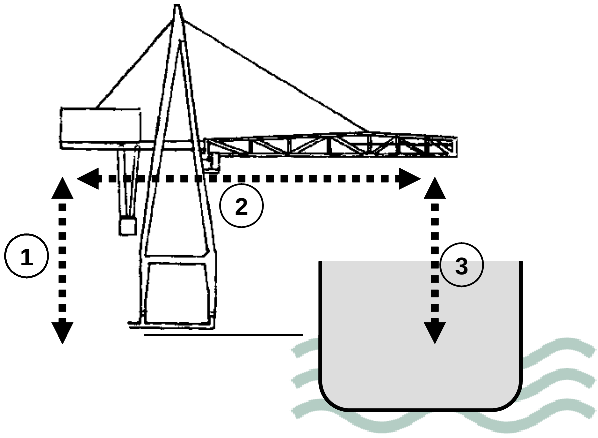


Figure 5.9: Three steps of spreader arm movement for handling a container

The time required for step 1 is the same for all the containers handled by the crane.

The completion time of steps 2 and 3 however depend on the position of the container in the vessel. The farther the container is from the berth line and the deeper the container is in the vessel, the higher is the time to handle the container.

The assignment of quay cranes to the vessel and the technical specifications of each crane at all ports is assumed to be known in this study. Given the stowage pattern and the set of available cranes, the solution to the crane split problem determines the ship turnaround time at port. As it was mentioned in section 2.5, crane split problem is in the class of scheduling problems and has been discussed as an independent problem or jointly with berth allocation problem in the container literature. To evaluate a candidate solution by the genetic algorithm we need to have the ship turnaround time at each port and for that matter we need to solve the crane split problem for each port. A large number of chromosomes are to be evaluated in each iteration of GA and to reach the convergence many iterations are required. Kim and Park (2004) report an average of 457 seconds CPU time for solving a problem with three homogenous cranes and 25 tasks. The computational burden of the crane split solution methods makes it impossible to use them in the evaluation procedure of our proposed genetic algorithm. To overcome that difficulty a heuristic procedure is proposed to approximate the crane split and total ship turnaround time. This procedure uses the output provided by the "Count Crane Operations" procedure in the former section as an input. Having B bays and Q_p quay cranes available at port p, the procedure begins with dividing the bays into Q_p subsets with $\left\lfloor \frac{B}{Q_p} \right\rfloor$ bays at each of sets. If B is not divisible by Q_p then the number of bays in the last subset will be equal to

$B - \left\lfloor \frac{B}{Q_p} \right\rfloor . (Q_p - 1)$. Then starting from the first crane it considers each crane with the crane right after it. It looks for any potential improvement in the turnaround time that might be achieved by separating a bay from the operational range of one crane and appending it to the other one. As it was mentioned before the number of crane operations at each port are known at this time and given by matrix C. The following notation is used in the proposed procedure.

$R^f_{p.q}$: Index of the first bay in the operating range of crane q at port p

$R^l_{p.q}$: Index of the last bay in the operating range of crane q at port p

$BT_{p.q}$: Busy time of crane q at port p

$\alpha_{p.q}$: Required time to handle one container by crane q at port p

$\beta_{p.q}$: Required time for horizontal movement to adjacent bay for crane q at port p

TT_p : Turnaround time at port p

<u>Procedure: Calculate crane split and turnaround time</u>

```
Initialization: $BT_{p,q}, R^f_{p,q}, R^l_{p,q} = 0 \quad \forall p, q$
For $P \leftarrow 1$ to $N$
{
          $i = 1$
          For $q \leftarrow 1$ to $Q_p - 1$
          {
```

```
        $R^f_{p.q} = i,\ R^l_{p.q} = R^f_{p.q} + \left\lfloor \frac{B}{Q_p} \right\rfloor$
        $i = R^l_{p.q} + 1$
    }
    $R^f_{p.Q_p} = i,\ R^l_{p.Q_p} = R^f_{p.Q_p} + \left\lfloor \frac{B}{Q_p} \right\rfloor$
    For $q \leftarrow 1$ to $Q_p$
    {
        $BT_{p,q} = \sum_{b \in [R^f_{p,q}..R^l_{p,q}]} C[p,b].\alpha_{p,q} + (R^l_{p,q} - R^f_{p,q}).\beta_{p,q}$
    }
    For $q \leftarrow 1$ to $Q_p - 1$
    {
        Crane range adjustment for cranes $q$ and $q+1$:
        If $BT_{p,q} > BT_{p,q+1}$ then
            $\gamma = BT_{p,q} - BT_{p,q+1},\ R^l_{p.q} = R^l_{p.q} - 1,\ R^f_{p.q} = R^f_{p.q} - 1$
            Recalculate $BT_{p,q}, BT_{p,q+1}$
            If $|BT_{p,q} - BT_{p,q+1}| \geq \gamma$ then $R^l_{p.q} = R^l_{p.q} + 1,\ R^f_{p.q} = R^f_{p.q} + 1$
        Else
            $\gamma = BT_{p,q+1} - BT_{p,q},\ R^l_{p.q} = R^l_{p.q} + 1,\ R^f_{p.q} = R^f_{p.q} + 1$
            Recalculate $BT_{p,q}, BT_{p,q+1}$
            If $|BT_{p,q} - BT_{p,q+1}| \geq \gamma$ then $R^l_{p.q} = R^l_{p.q} - 1,\ R^f_{p.q} = R^f_{p.q} - 1$
    }
    $TT_p = Max\{BT_{p,1}, BT_{p,2}, ..., BT_{p,Q_p}\}$
}
```

After calling this procedure crane utilization at each port and total turnaround time at all ports can be calculated as follow.

$$U_{p,q} = BT_{p,q} / TT_p$$

$$Total\,Turnaround\,Time = \sum_{p=1}^{N} TT_p$$

Figure 5.10 shows how the crane split approximation procedure works for a small example. The ship in this case has 6 bays and there are 3 cranes available. The number on each bay indicates the total number of crane operations required for the bay. Bays are numbered from stern to bow in increasing order.

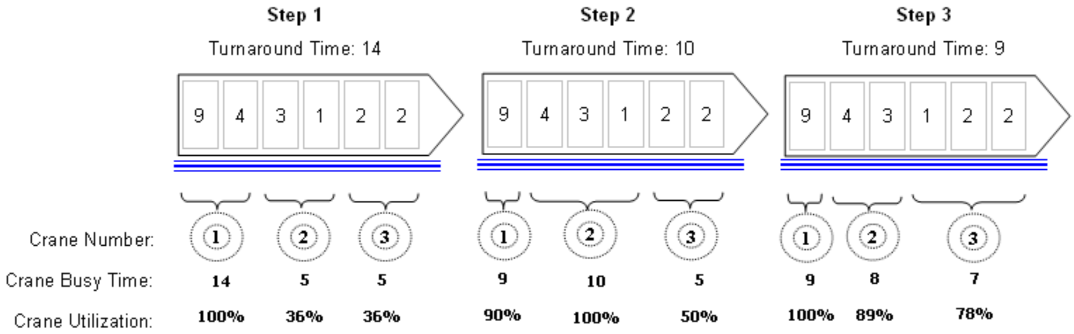


Figure 5.10: Illustration of crane split approximation procedure

At the first step two bays are assigned to each crane. It is assumed that each container can be handled in one time unit and also the crane can move horizontally to the adjacent bay in one time unit. Crane 1 for example must handle 13 containers in bays 1 and 2 and will need to do one horizontal displacement, which makes its total busy time equal to 14. The crane busy time for each crane as well as turnaround time and crane utilization for each step is shown in the figure. After the initial assignment the procedure tries to reassign the bays to the cranes if doing so decreases the turnaround time. In step 2 by separating bay 2 from crane 1 and assigning it to the operational range of crane 2, turnaround time will decrease by 4 units and finally in the last step by assigning bay 4 to crane 3, turnaround time is improved by 1.

Objective function

As it was mentioned before, the conventional objective function of the containership stowage planning is to minimize the total number of container movements at all ports. However the solutions results in chapter 4 show that this objective function does not represent the true objective function of the problem which is the minimization of total turnaround time at all ports. By using the suggested procedures we can calculate the value of either of these objective functions for each solution. In next chapters we will use the conventional objective function for verification of the genetic algorithm and creating base case scenarios. Another advantage of these procedures is the capability of creating economical objective functions. If the monetary value for each unit of time per port is given we can easily apply the time value parameter to the total turnaround time equation in order to minimize the total cost of operations.

$$Total\,Operational\,Cost = \sum_{p=1}^{N} TT_p \times \delta_p$$

$$\delta_p\ :Value\ of\ one\ time\ unit\,at\ port\ p$$

Similar cost minimization equation may be written for the situation that per TEU cost of handling the containers at each port is given and the cost parameter varies over ports or is different for different cranes.

5.2.3. Handling the constraints

The solution space for a constrained combinatorial problem can be divided into two regions: feasible and infeasible. Searching for optimal solution must be performed inside the feasible region. The assignment based genetic encoding for containership loading problem resolves the issue of illegality, however it may generate infeasible solutions. Infeasibility occurs as a result of violating the constraints. There are three ways to deal with infeasible chromosomes: reject, repair and penalty. Rejecting involves the elimination of infeasible chromosome from the generation. Repair methods try to find the genes that cause the infeasibility and modify their values to satisfy the feasibility criteria. Penalty methods identify the violated constraints and impose a penalty to the fitness value of the corresponding chromosome for each violation. The penalty can be fixed or be proportional to the deviation of the constraint from the acceptable range. The rejection method takes away any possibility of the chromosome for appearing in the next generations and may increase the convergence time dramatically. Choosing among the repair and penalty methods depends on the nature of the problem. A drawback of the repair method is that it interferes with the learning mechanism of the genetic algorithm by changing parts of the information. This might affect the convergence of the algorithm even more if repairing the infeasible chromosome asks for extensive modifications. The penalty method does not change the chromosomes; however it has the disadvantage of introducing the penalty term to the fitness function. Finding proper value for the penalty terms may be a complicated task especially if several constraints of different types are to be considered. In the proposed assignment policy based encoding it is very difficult and time consuming to modify the genes in order to remove the infeasibility because the genes hold the policy and sorting codes instead of actual stowage of the vessel. So the penalty method is used in this research for enforcing the constraints.

Instability penalty

As it was discussed in 3.3.1, GM is a proper measure of stability for the containerships. The deviation of GM from the permitted threshold can be imposed as penalty to the fitness function. Measuring the GM however needs detailed information about the structural design of the specific vessel. To generalize the solution approach the experimental rules of stability are used in this research. After decoding the solution and generating the stowage pattern for each port, different stability measures may be easily computed according to the containers weight information and their position in the vessel. Cross equilibrium can be measured based on the axis of symmetry going from bow to the stern. The total weight or momentum of the containers at either side must be calculated and the difference must not exceed a given value. The penalty is calculated by multiplying the penalty parameter by the magnitude of the violation, so that the solutions that are farther from the feasible region receive higher penalties. Similarly, longitudinal and vertical equilibrium will be measured and the evaluation function will receive penalty for their violation. Theoretically speaking tightening the constraints will shrink the feasible region and will make it more difficult for the algorithm to converge to possible optimal or suboptimal solutions if any.

Violation of operational constraints

Similar to the stability constraints, the necessary information for checking the operational constraints is retrievable from the stowage pattern and the container information list. A very important constraint in the mathematical problem presented in chapter 3 is that the cell under a container cannot be empty. This constraint is naturally taken care of in the assignment policy based encoding because all the policies start assigning cells to containers from the first unassigned cell at the bottom of the column. Another operational constraint prohibits storing a heavy container on a lighter one. As it was shown in Table 1.1 average standard weight of an empty container ranges from 2.3 to 4 tons, while the maximum average weight of a full container to be stowed in a containership is 24 and 30 tons for 20' and 40' containers respectively. For implementing the container weight constraint a simple procedure goes through the stowage plan and compares the weight of each container with the container immediately on top. If the difference between the two weights cannot be tolerated it will be counted as one violation. The tolerance is a parameter that can be decided by the ship planner, i.e. 5% weight difference might be considered acceptable. Total number of container weight violations is then multiplied by its penalty parameter and will be applied to the fitness functions. Similar approach is used to account for other operational constrains such as regulations for stowing special containers or rules for mixing the containers of different sizes.

Similar cost minimization equation may be written for the situation that per TEU cost of handling the containers at each port is given and the cost parameter varies over ports or is different for different cranes.

5.2.4. Genetic operators and chromosome selection

Genetic algorithm is an iterative process which works on a generation of chromosomes. It starts with a population of randomly generated individuals. At each step of the algorithm genetic operators manipulate the current generation and then a selection procedure chooses the individuals that will advance to the next generation based on their fitness value. The two classical operators of the genetic algorithm are crossover and mutation. Crossover is a binary operator that performs the exchange of information between two individuals that are randomly selected for breeding to create new individuals. Mutation on the other hand is a unary operator which modifies a randomly selected individual. Both of these operators are important to the genetic algorithm. Crossover enables the algorithm to extract the best genes from different individuals and combine them to create potentially superior offsprings whereas mutation introduces diversity to the population and thereby decreases the possibility of converging into local optima. Without mutation it is very likely that the algorithm could only produce individuals whose genes are a subset of the combined genes in the initial population. The genetic operators and the mechanism for selecting the individuals are discussed in this section.

Crossover

Recombination in genetic algorithm is done by crossover operator. After randomly selecting two parents, there are several methods that crossover can be applied.

<u>One-point crossover</u>: the classical one point crossover splits each parent into two parts from a randomly selected crossover point and recombines the swapped sections

of the parent to create two new children. This operator is more likely to keep together the neighboring genes, but it can never keep together genes from opposite ends of the chromosome. Figure 5.11 (a) shows and example.

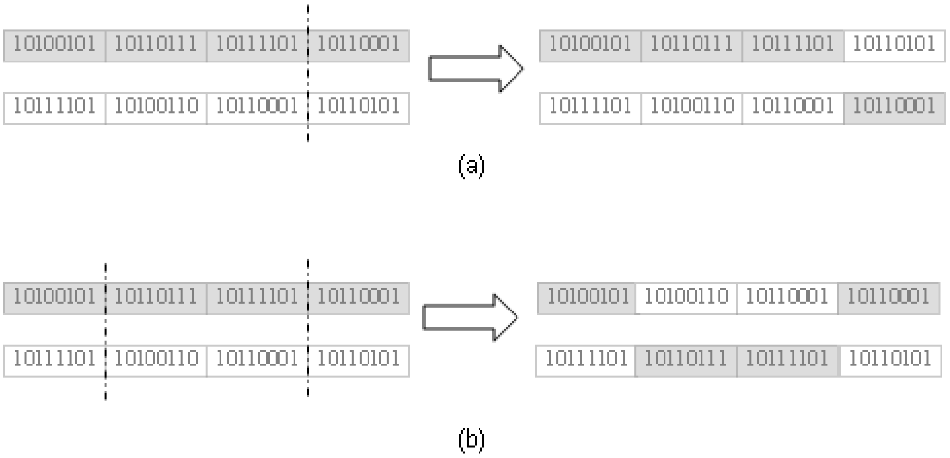


Figure 5.11: (a) one-point crossover (b) two-point crossover

Two-point crossover: this operator requires two random crossover points to be selected. Each parent then is split into three parts according to the crossover points and the children are created by recombining the sections of the parents as it is shown in Figure 5.11 (b). Unlike one-point crossover it is possible to keep the genes from both ends of the chromosome together while maintaining the block structure of the chromosomes.

k-point crossover: it is a generalization of the two-point crossover and applies the same idea to k randomly generated crossover points. De Jong (1975) and Goldberg (1989) conclude that the two-point gives an improvement, but adding further crossover points may reduce the performance of the algorithm.

In all above cases the crossover points will be located at the starting position of the gene values for a particular port. In other words the crossover points do not break the encoded sorting and assignment policy information for any particular port and because of that the integrality and legality of the generated children is guaranteed.

Uniform crossover: in this method for each bit position on the two children, corresponding bits in the parents are swapped with a fixed probability. The probability value determines the degree in which each parent contributes to each child and is typically 0.5. This method has the disadvantage of destroying the building blocks of the genes.

Cut and splice crossover: this method is a variant of the two-point cross over with the difference that crossover points on both parents are independent. This results in chromosomes with variable length. The proposed chromosome structure in this research is fixed length so this operator will not be a candidate.

The crossover operator has a probability P_c which determines the percentage of the chromosomes that undergo crossover. At the crossover step of the GA, a random number in the range of [0, 1] is generated for each chromosome. If the number is below P_c then the individual will be selected for crossover. The type of the crossover operator and the value for P_c will be discussed in chapter 6.

Mutation

After applying the crossover and adding the new children to the generation the enlarged population undergoes mutation with probability P_m. Mutation is applied by randomly selecting a bit on the candidate chromosome and switching its value. In the proposed genetic encoding for containership loading problem, mutation can be interpreted as modifying the sorting or assigning strategy at a port from one strategy to another. The value for the mutation parameter P_m will be analyzed in chapter 6.

Selection and migration

In the selection phase of the genetic algorithm a set of chromosomes from the current generation are chosen for breeding in the next generation. The selection process works in the favor of the individuals with better quality by giving them a higher chance for migration. The two well known classical methods for selection are roulette wheel and tournament methods.

Roulette wheel method: this is the most common method in which the selection chances are proportional to fitness. It starts with normalizing the evaluated fitness of all the chromosomes by multiplying the fitness value of each individual by a fixed number such that the sum of all fitness values over the population equals 1. Each individual then will be assigned a circular sector of the roulette wheel. The angle of the sector is equal to $2\pi\ \boldsymbol{f}_i / \bar{\boldsymbol{f}}$ where $\boldsymbol{f}_i$ is the fitness of chromosome *i* and $\bar{\boldsymbol{f}}$ is the fitness of the whole population (Holland 1975). Individuals can be chosen randomly one after another by spinning the wheel. To implement the wheel spinning process the population is sorted by descending normalized fitness values. A random number r is generated in the range of 0 to 1 and the first individual whose normalized fitness value is greater than r is chosen.

Tournament method: the simplest form of this method is binary tournament selection. Randomly picked pairs of individuals are selected from the population and the one with higher fitness value is copied to the mating pool. The process continues until the population is full.

Elitism

The idea behind elitism is to ensure that the current generation is at least as fit as the previous generation. It is done by directly copying the fittest individual to the next generation. Elitism is believed to speed up the search, that is, to converge faster toward the optimal point. However, there are some arguments which criticize this strategy, since it can prematurely limit the search in local optima.

Termination criteria

The process of creating generations continues until the termination criterion is met. There are several criteria to stop the genetic algorithm and selecting the proper criteria usually depends on the application. The most common termination conditions are as follow.

- Maximum number of iterations has reached
- Maximum time for running the algorithm has reached
- The average fitness of the population has reached a steady state, that is, the absolute difference between the average fitness of the two most recent generations is below ε. Another variant of this criterion is when the percentage of the improvement in average fitness compared to that of previous generation is below a certain percentage value.

The termination condition can also be a combination of the above items.

5.3. Summary

In this chapter after a brief review on the evolutionary algorithms, a genetic algorithm for optimizing containership load planning considering crane split is proposed. A new genetic representation based on combination of sorting and assignment policy is proposed which is very compact and expressive. The idea behind the proposed encoding comes from the grouping and storing practices in the container yard in which the containers are sorted and grouped based on their common property such as weight, size, type or destination. A data structure called *vessel layout mask matrix* is designed to represent the cell availability of the containership. Required procedures for decoding a solution, generating stowage pattern based on the decoded solution and analyzing the generated stowage pattern for calculating the components of the evaluation function are discussed. It is shown that the algorithm is capable of dealing with different objective functions for the problem including the conventional objective function of the stowage planning, temporal objective function and economical objective function. The algorithm uses penalty method for handling the constraints. The designed crossover and mutation operators ensure the legality of the generated offsprings. The roulette wheel method is used for selecting the individuals to build the next generation. Different termination conditions are defined and will be implemented.

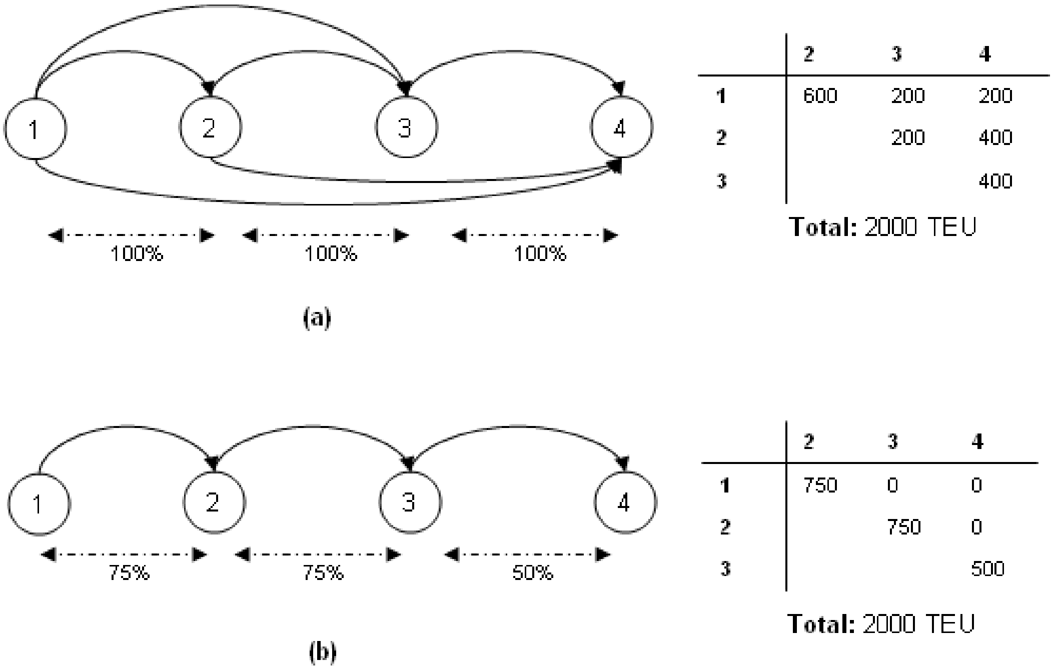
Chapter 6: Analysis of genetic algorithm parameters and lower bound

The proposed genetic algorithm in chapter 5 has been implemented using C++ language. To analyze the performance of the solution method and to find the appropriate set of parameters for the genetic algorithm a large set of sample problems have been generated and solved.

6.1. Generating sample problems

To estimate the value of the parameters and analyze the performance of the proposed genetic algorithm we need to run the algorithm on variety of sample problems with different sizes and structures. As it was mentioned in 4.1 the generated problems must be compatible with the capacity of the containership, that is, neglecting the stability and operational constraints the container to vessel assignment problem must be feasible. Besides that, it is important to generate scenarios to resemble the situations in which the containership operates under capacity or the emphasis is put on certain ports. The underlying structure of the transportation matrix plays a crucial role in the complexity of the containership loading problem. Figure 5.1 depicts an example. In this figure a 2000 TEU containership is considered to visit four ports under two scenarios. Each solid arc shows the flow of the containers between the corresponding ports and the demand can be seen in the transportation matrices. The numbers at the bottom show the percentage of the ship’s capacity that is full while moving from one port to another. As it is shown in figure 6.1 (a) the vessel in this case is fully utilized to transport a total of 2000 containers. This number is the same for the case in figure

6.1 (b), however the vessel could have transported more containers as it is not full all the time. Even though both cases have equal number of ports and containers with the exact same containership, the complexity of their stowage planning problem is different. In the second case the issue of overstowage does not exist because the containers of different destinations are not present in the vessel simultaneously. The stowage planning at this case is reduced to a simple container to cell assignment with stability and operational constraints. This however is different for the first case as overstowage might be inevitable.



	2	3	4
1	600	200	200
2		200	400
3			400

Total: 2000 TEU

	2	3	4
1	750	0	0
2		750	0
3			500

Total: 2000 TEU

Figure 6.1: Transportation matrices for two different scenarios

Besides the structure of the demand, the property of the containers i.e. weight, size, and type may contribute to the level of complexity. Consider two set of containers for the case presented in figure 6.1 (b) where one set has 2000 containers of uniform weights as opposed to the other set where weights vary across the containers. The case with different weights is more challenging since it has more binding operational constraints. The above example shows that the complexity of the containership loading problem is not governed only by the size of the problem, but also it depends on the structure of the demand matrix and the characteristics of the containers.

The following pseudo-code is designed to generate random test problems of different sizes for the containership loading problem with respect to the ship capacity.

Procedure: Generate test problem

```
Input the capacity of the ship, number of ports, maximum load ratio, containers
weight range, percentage of containers of different size and type.
For $P \leftarrow 1$ to $N$
{
    Available capacity = Capacity * maximum load ratio - Sum of the containers
                  that have been loaded at the ports preceding $P$ + Sum of the
                  containers that have been unloaded at port $P$ and its
                  preceding ports
    Generate ($N - P$) random demands, one for every port succeeding $P$, such
    that sum of the generated demand is less than or equal to the available
    capacity.
    Apply the percentage of different size containers to the demand and adjust the
    numbers.
}
$D$ = total generated demand (total number of containers)
```

```
For $c \leftarrow 1$ to $D$
{
    Generate random weight for container $c$ within the specified weight range.
    Store the origin, destination, weight, size and type of the container $c$ .
}
```

6.2. Genetic algorithm parameters

There are several parameters in each genetic algorithm that contribute to the accuracy and the performance of the algorithm. The population size, crossover and mutation ratio and termination criteria are the main parameters to be decided. Like other evolutionary algorithms, the value of the genetic algorithm parameters depends on the nature of the problem and has to be tuned for the specific problem representation. Alander (1992) studied the optimum population size of the genetic algorithms as function of problem complexity and concluded for problems coded as bit-strings, the length of the string in bits for sequential machines is a good approximation for the optimum population size. De Jong (1975) recommended population size, the mutation rate and the crossover rate to be 100, 0.001 and 0.6 respectively. We set the population size to 100 and analyze the value for crossover and mutation parameters based on this assumption.

The relative importance of crossover and mutation in genetic algorithm has been subject of discussion in the genetic algorithm literature. Although some researchers suggest that crossover operator alone is sufficient for evolving the solution, other studies confirm that the power of mutation operator cannot be neglected (Schaffer et al. 1989) and crossover has no general advantage over mutation (Fogel and Atmar 1990). In other words crossover is explorative and discovers promising areas in the search space by gaining information on the problem while mutation is exploitative which uses the information to improve the optimization within a promising area. In fact cooperation and competition coexist between these two operators (Eiben and Smith 2003).

6.2.1. Crossover and mutation parameter

To capture the effect of different mutation and crossover values on the performance of the genetic algorithm and quality of the solution, a set of test problems have been generated using the procedure described in 6.1. Four containerships with the capacities of 500, 1000, 2000 and 3000 TEU are considered, each of which visiting five ports.

After running some pilot experiments we decided to analyze P_c, P_m in the ranges of [0.1.0.8] and [0.005...0.5] respectively. Starting from the lowest value we increase P_c by 0.1 at each step and increase P_m by 0.05. This creates 90 (P_c, P_m) combinations. The stability rules of cross and longitudinal equilibrium are enforced. Each of the generated test problems were solved for every (P_c, P_m) pair which makes 360 cases. To make the results comparable the termination criteria was set to 500 iterations of the genetic algorithm. For each case total number of container handling at all ports was chosen as an indicator of the solution quality. Figure 6.2 shows the results for the

1000 TEU containership. In this case (P_c, P_m) = (0.6, 0.2) has generated better quality solutions in 500 iterations.

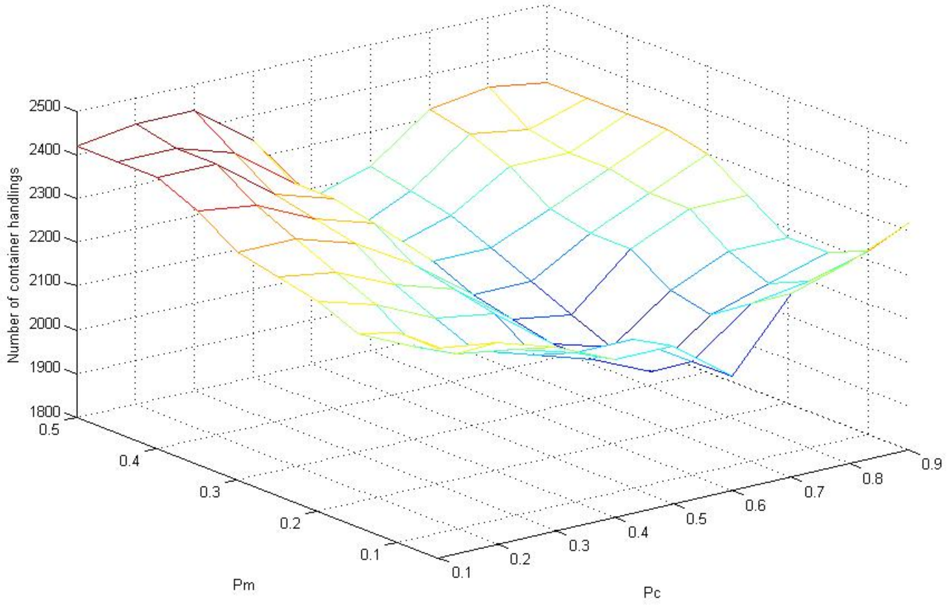


Figure 6.2: Crossover and mutation ratio analysis for 1000 TEU case

Since the containerships in the four considered cases are of different capacity, the result for each case has been converted to a scale of 1 to 100. The best solution among 90 combination of (P_c, P_m) has been given the score 100 and the other solutions have been given a score proportional to their deviation from the best solution. The average score of results over the four cases for each (P_c, P_m) is presented in figure 6.3 and table 6.1. The results show that by setting P_c, P_m equal to 0.6 and 0.15 respectively, better quality solutions are obtained within 500 iterations.

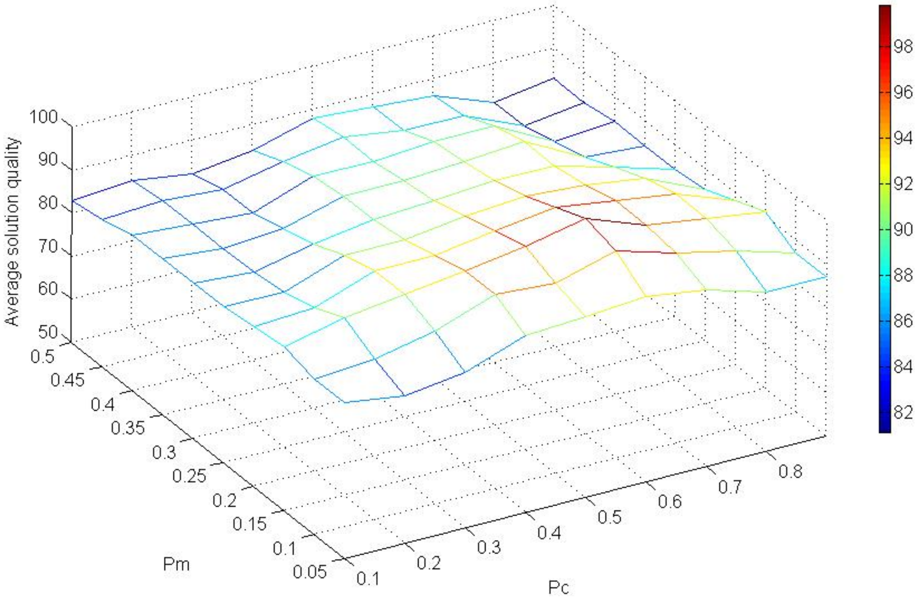


Figure 6.3: Overall crossover and mutation ratio analysis

Table 6.1: Normalized solution quality for overall crossover and mutation ratio analysis in four containership classes

		Pc								
		0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Pm	**0.05**	36	56	79	73	87	86	82	71	60
	0.1	43	63	82	80	90	93	89	78	67
	0.15	50	70	85	87	97	100	96	80	70
	0.2	46	66	81	83	93	96	92	81	71
	0.25	42	62	77	79	89	92	88	77	66
	0.3	38	58	73	75	85	88	84	73	62
	0.35	35	55	70	72	82	85	81	70	59
	0.4	32	52	67	69	79	82	78	67	56
	0.45	31	47	62	64	74	77	73	62	51
	0.5	30	42	57	59	69	72	68	57	46

6.2.2. Population size

Genetic algorithm needs memory to store the population. The larger the population size, the more memory is required. Besides the memory requirement concerns, increasing the population size usually results in increase of running time at each step of the algorithm. This happens because more chromosomes will be subject to crossover, mutation and fitness evaluation in a larger population. But at the same time if the population size is not large enough, then the algorithm might converge to a suboptimal solution prematurely due to limiting the diversity of the solutions. Having fixed the parameters for crossover and mutation, we can analyze the effect of the population size on the solution quality. The 1000TEU containership case from the previous section is used. To investigate the effect of population size on the running time, the problem is solved for three cases. The objective function in all cases is the minimization of the total turnaround time, but the constraints are different. In case 1 only horizontal and cross equilibrium constraints are enforced. In case 2 the vertical equilibrium is added and finally in case 3 placing any heavy container on a lighter one is also prohibited. The average weight of the 1896 containers in this problem is 25.55 tons with the standard deviation equal to 14.03. The weight threshold for the equilibrium constraints is 10% (e.g. if the weight difference between two containers is less than 10% they are treated equally).

Figure 6.4 illustrates the increase of CPU time based on the population size for fixed crossover and mutation ratio parameters and fixed number of iterations in each case. Since the number of iterations is fixed the quality of solution for each population size is different from the others. Investigate the mutual impact of the population size and the number of iterations on the solution quality, the convergence criteria is modified for figure 6.5. For this experiment the algorithm is set to stop when the average fitness value of the population does not improve by a margin of 0.001 compared to the prior iteration. The 1000 TEU containership described in section 6.2.1 is used and the vertical and horizontal equilibrium constrains are enforced. As figure 6.5 shows the results of this example, when the population size is small a higher number of iterations are needed to reach the targeted solution quality. Although the number of iterations is less as the population size grows, the CPU time goes up as a result of having a larger number of chromosomes. A population size of 100 in this case seems to be an acceptable tradeoff point between population size and number of required iterations.

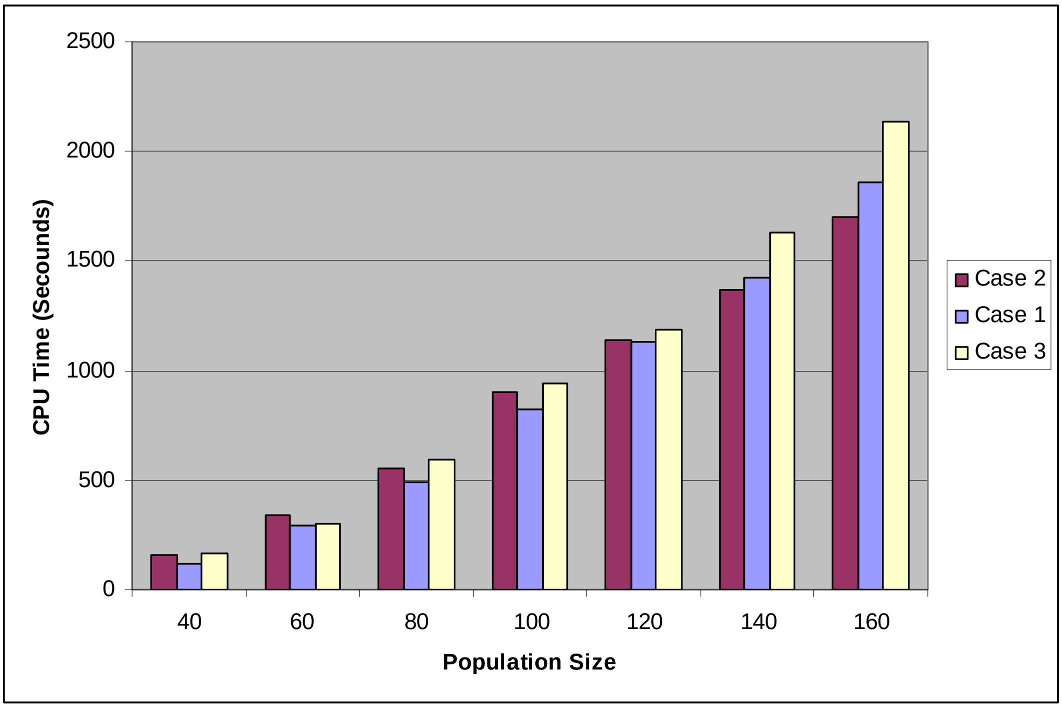


Figure 6.4: Effect on population size on CPU time

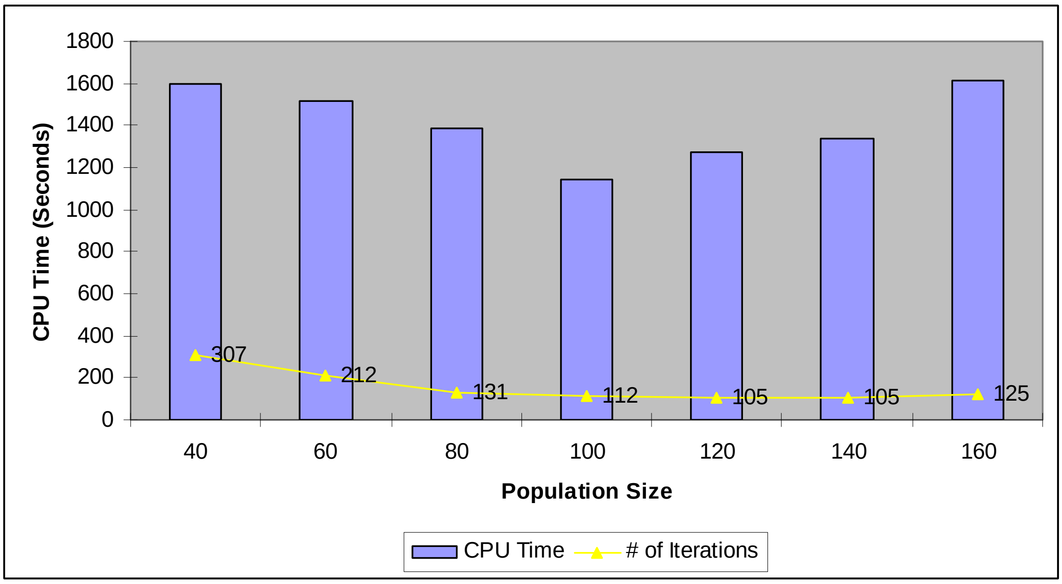


Figure 6.5: Effect on population size on CPU time and number of GA iterations

6.2.3. Two-point crossover vs. classic crossover

Classic and two-point crossover operators have been implemented. To evaluate their performance we applied the variation of the genetic algorithm using each operator to the same 1000 TEU containership problem having fixed the number of genetic iterations. If the number of iterations is large enough the results from test problems show that the quality of the solution by both methods are approximately the same, however the two-point crossover slightly speeds up the convergence. Figure 6.6 shows the comparison graph between the best fitness value of each generation versus the iteration number for the two operators.

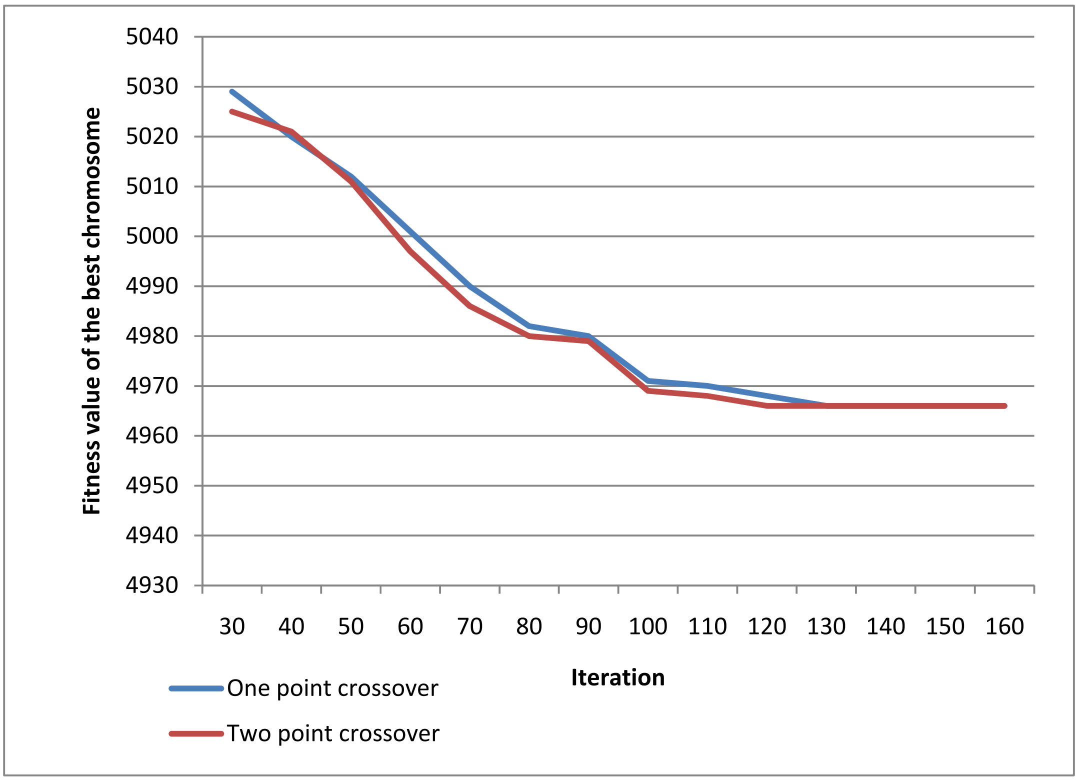


Figure 6.6: Comparison between one and two point cross over operators for a sample problem

6.2.4. Containership capacity and solution time

Although the capacity of containership and number of visiting ports are contributing factors to the problem complexity, the structure of the transportation matrix and the structural design are the vessel will define the complexity. In other words solving the containership load planning for a vehicle with larger capacity is not necessarily more challenging that solving the problem for a smaller vessel. In order to study the capability of the proposed solution approach for dealing with different classes of containership, and also the relationship between the solution time and size of the vessel, 40 sample problems for eight class of containerships were generated. Ten ports are considered in these problems and for each class of vessels five origin-destination matrixes were generated. Stability constraints are enforced such that the maximum horizontal and vertical imbalance is limited to maximum 5%. The containers are of different weight generated based on a uniform random distribution of U(3,25). Table 6.2 shows the characteristics of the problems and the average CPU time for each class.

Table 6.3: Sample problems for different classes of containerships

Vessle Class	Number of generated problems	(Bay,Row,Tier)	Capacity (TEU)	Average number of containers	Average CPU time (Minute)
1	5	(10,10,10)	1000	2444	2
2	5	(20,10,10)	2000	5938	11
3	5	(30,10,10)	3000	9001	21
4	5	(40,10,10)	4000	10210	28
5	5	(50,10,10)	5000	12662	49
6	5	(60,10,10)	6000	16640	75
7	5	(70,10,10)	7000	18641	126
8	5	(80,10,10)	8000	22507	137

Figure 6.6 illustrates the increase in the average computational time with respect to the containership capacity. It can be observed that the solution to real size problems can be obtained in a relatively short time. The solution time for 8000 TEU containerships is less than 2.5 hour.

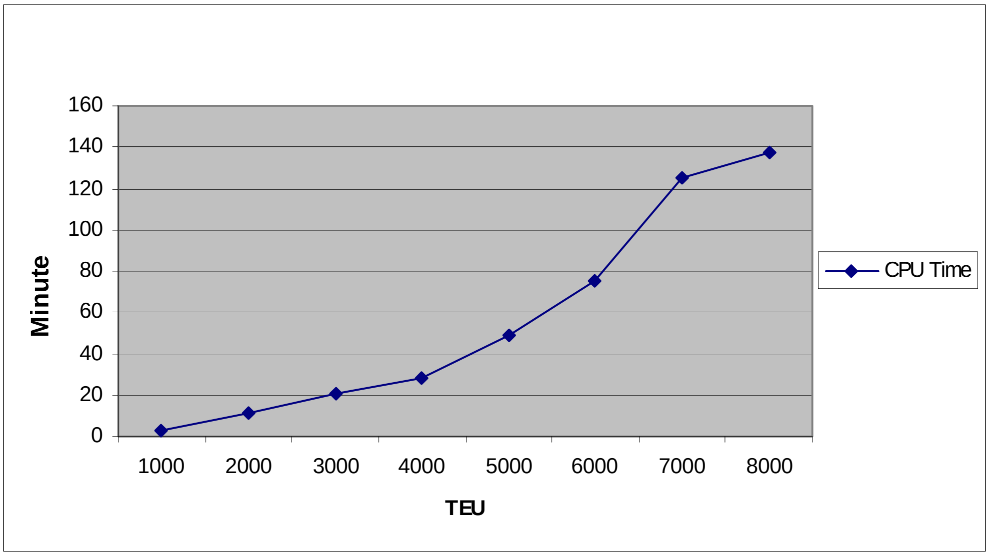


Figure 6.6: Average CPU time vs. containership class for 10 ports

6.2.5. Parallel processing

One of the methods to improve the efficiency of an algorithm is to take advantage of the parallel processing. As opposed to the sequential processing, parallel processing is simultaneous execution of the code or pieces of the code on several processors. On a multi-processor computer, parallelism can be done implicitly or explicitly. In implicit parallelism the operating system distributes the tasks over the available processors automatically whereas in the implicit method the programmer must design the algorithm in a way that different tasks or threads are assigned to the processors. Many parameters and factors govern the performance of the parallel algorithms, however if the architecture of the algorithm is not parallel then the benefits of parallel processing will not be significant. Genetic algorithms are parallel in nature and that makes them good candidates for parallel processing. The parallel parts of the program however need to be synchronized using programming techniques. In the proposed genetic algorithm for containership load planning, evaluation of the chromosomes makes a big fraction of the total CPU time. After crossover and mutation operators are applied to the current generation, there are several chromosomes to be evaluated before building the next generation. In the presence of multiple processors, the evaluation of the chromosomes can be done simultaneously and will reduce the total evaluation time of the generation. The implementation of the genetic algorithm in this research uses *threads* and the *synchronization* classes in C++ to distribute the chromosome evaluation tasks over the available processors.

If the speed of an algorithm is defined as its execution time, then the speedup can be defined as the speed of the serial algorithm (execution time on a single processor) over the parallel speed (execution time of the parallel algorithm on a multi-processor). Efficiency of the parallel processing is measured by the speedup divided by the number of processors (Lewis and El-Rewini 1992). Amdahl's law is a model for the relationship between the expected speedup of parallelized implementations of an algorithm relative to the serial algorithm. If F is the fraction of the algorithm that is sequential and N is the available number of processors, then according to Amdahl’s law the maximum expected speedup by parallelization is $\frac{1}{F+(1-F)/N}$.

Having set the P_c, P_m and generation size to 0.6, 0.15 and 150 respectively, on average there will be 90 new offspring and 22 mutated chromosomes to be evaluated in each generation. So the fraction of the algorithm that is parallel can be approximated as follow:

$$1-F=\frac{90+22}{150}=0.75\ ,\ F=0.25$$

According to Amdahl's law the maximum expected speedup by parallelism on a two processor computer is:

$$Max\ Speedup = \frac{1}{0.25 + (1 - 0.25)/2} = 1.6$$

Figure 6.7 shows the comparison of the results for the sequential and parallel runs of the algorithm on a computer with two processors.

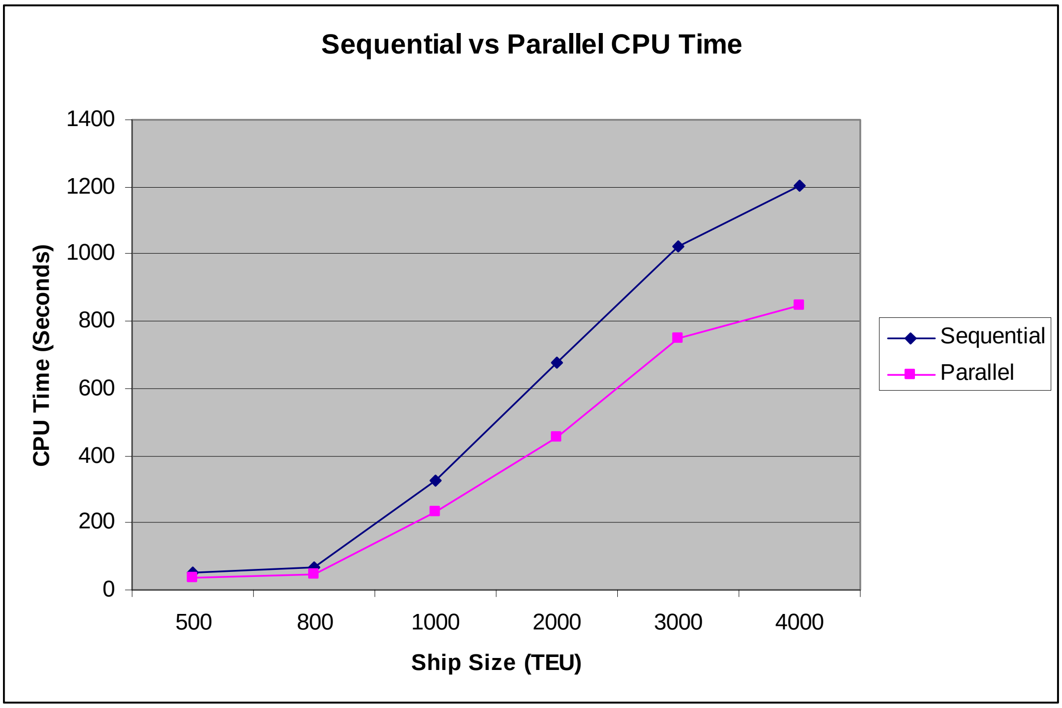


Figure 6.7: Sequential vs. parallel CPU time

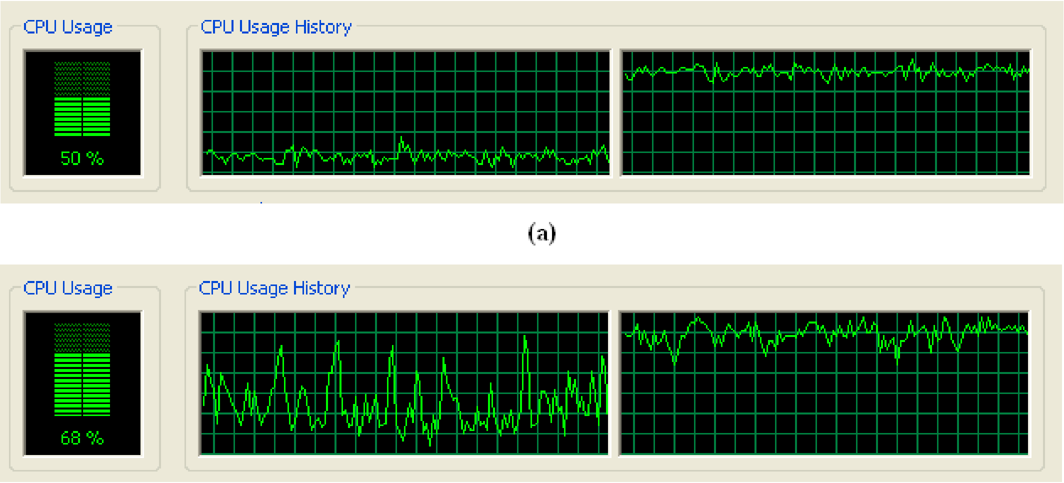
Table 6.3 shows and average speedup of 1.44 for these cases. This is 0.16 less than the suggested value by the Amdahl's law. Debrovsky et al (2002) reported a speedup factor of 2.7 on a parallel computer with three processors.

Table 6.3: Summary of the results for sequential and parallel algorithms

	CPU Time (Sec)		
Ship Size	Sequential	Parallel	Speedup
500	54	36	1.50
800	67	46	1.46
1000	324	232	1.40
2000	677	455	1.49
3000	1021	749	1.36
4000	1204	847	1.42

Average Speedup = **1.44**

Figure 6.8 illustrates a snapshot of the CPU usage history in two cases. In 6.8 (a) a the sequential algorithm is executed on a two processor machine and in 6.8 (b) the parallel algorithm has run on the same machine. The graphs are obtained from the task manager program in Microsoft Windows XP. It can be seen that in the second case some load is assigned to the additional CPU.



(a)

(b)

Figure 6.8: Snapshot of CPU usage history (a) sequential (b) parallel

6.3. Lower bound and algorithm performance

While developing heuristic algorithms to solve a minimization problem, having a lower bound helps to measure the deviation of the solution from that of the exact method.

There exist some traditional methods for developing lower bounds:

- Linear relaxation: the integrality constraints for some or all the integer variables are relaxed while the objective functions and other constraints remain intact.
- Lagrangian relaxation: some constraints especially the tight ones are moved to the objective function with a penalty term, where the value and sign of the penalty coefficient depend is determined by the nature of the optimization problem.
- Branch and bound: in the *branching* stage the solution space of the problem is systematically split into smaller sections whose union creates the original space and in the *bounding* stage upper and lower limits are assigned to the split node.

The complexity level of the containership stowage planning comes from the shift constraints to a great extent. The linear relaxation method is not applicable to the containership stowage planning problem since all the key variables in the mathematical model are binary and relaxing the integrality requirements is not an option. The Lagrangian relaxation method was not found to be useful for this problem either. This method requires moving the shift constraints as hard constraints (Eq. 3.15

to 3.20) to the objective function which creates two major issues. First because the shift constraints play a fundamental role in determining the value of the unloading variables as the building blocks of the objective function, their elimination oversimplifies the problem by transforming it to an assignment problem and generates a very loose bound. Second bringing these constraints to the objective function with a penalty term makes the objective function value of the lower bound problem non-comparable with that of the main problem. The branch and bound method cannot be applied because the large number of binary variables even for small problem instances results in an exponential number of branches.

Since no contender method among the traditional methods exist, a lower bound formulation based on the specific structure of the containership stowage planning is developed.

NL-LB: A non-linear formulation

The idea of reducing unnecessary shifts in containership stowage planning can be looked at as minimizing *the change in the value of the container location variable* at two consecutive ports for all containers and over all the ports. It means that wherever the container is located in the vessel, it is most desirable to remain at the same location thorough the voyage. Therefore in an ideal situation each container will be loaded and unloaded exactly once. So the objective function can be built by summation of the changes in the location variables for each container between each two immediate port. Modeling this objective function calls for calculating the absolute value of the difference between the binary location variables of the containers. Since the absolute value function is non-linear, the model will be a nonlinear programming model with quadratic objective function and linear constraints. The shift constraints can be removed from the formulation because the concept of shifts is taken care of in the proposed objective function. However the total number of unloading operations is not accurate and may be less than the actual number. Figure 6.9 depicts an example.

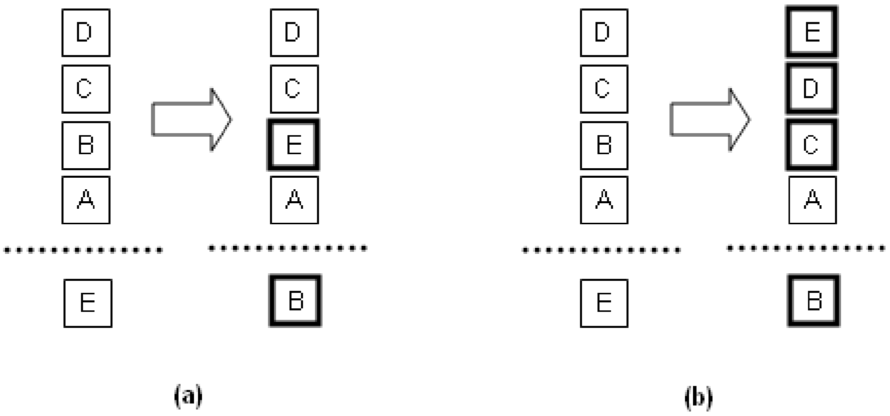


Figure 6.9: Number of unloading operations for the lower bound

Figure 6.8.a and 6.8.b show a single column of a containership in two different situations. Assume that container B has reached its destination, containers A, C and D must stay aboard and container B must be loaded at this port. To unload B, first containers D and C must be unloaded and then loaded back to the vessel. In fig 6.8.a stability and operational constraints have force the new container E to be loaded into the cell that belonged to B. Comparing the before and after snapshot shows that only two containers have changed position and hence the value of the lower bound objective function will be increased by two. On the other hand if like figure 6.8.b the new container is to be loaded at the top of the stack, the comparison of the two

snapshots shows change of the position for four containers. So the total number of container position changes is an estimate of the actual number of unloadings which gives a lower bound to the main problem. In an ideal scenario containers are loaded at the origin port, travel to their destination and get unloaded. So the total number of unloading operations is equal to total number of containers. In this situation the change of the position of each container happens only once at the unloading port and the gap between the lower bound and the optimal problem is nonexistent.

This lower bound model is as follow:

$$Min\sum_{c=1}^{C}\sum_{t=O(c)}^{N-1}\sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z}\left|\delta_{c,t}^{x,y,z}-\delta_{c,t'}^{x,y,z}\right| \tag{6.1}$$

Since the variables are binary and because mathematical solver does not recognize the absolute value, the eq. (6.1) can is written as follow:

$$Min\sum_{c=1}^{C}\sum_{t=O(c)}^{N-1}\sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z}\left(\delta_{c,t}^{x,y,z}-\delta_{c,t'}^{x,y,z}\right)^{2} \tag{6.2}$$

If the objective function of the main problem is considered to minimize the total time at all ports, the value of the lower bound objective function cannot be directly compared with that, so a post processing routine is hired to generate total time at all ports based on the results obtained by solving the non-linear lower bound model.

The constraints for the lower bound model are as follow:

$$\sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z}\delta_{c,t}^{x,y,z}=1 \qquad \forall c,t\in[O(c)..D(c)-1] \qquad (6.3)$$

$$\sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z}\sum_{t=D(c)}^{N}\delta_{c,t}^{x,y,z}=0 \qquad \forall c \qquad (6.4)$$

$$\sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z}\sum_{t=1}^{O(c)-1}\delta_{c,t}^{x,y,z}=0 \qquad \forall c \qquad (6.5)$$

$$\sum_{c=1}^{C}\delta_{c,t}^{x,y,z}\le 1 \qquad \forall x,y,z,t \qquad (6.6)$$

$$\sum_{c=1}^{C}\delta_{c,t}^{x,y,z}-\sum_{c=1}^{C}\delta_{c,t}^{x,y,z+1}\ge 0 \qquad \forall x,y,z\in[1..Z-1],t \qquad (6.7)$$

$$\left|\sum_{c=1}^{C}\sum_{x=1}^{X}\sum_{y=1}^{\lfloor Y/2\rfloor}\sum_{z=1}^{Z}\delta_{c,t}^{x,y,z}\bullet W(c)-\sum_{c=1}^{C}\sum_{x=1}^{X}\sum_{y=\lceil Y/2\rceil}^{Y}\sum_{z=1}^{Z}\delta_{c,t}^{x,y,z}\bullet W(c)\right|\le LRB \qquad \forall t \qquad (6.8)$$

$$\left|\sum_{c=1}^{C}\sum_{x=1}^{\lfloor X/2\rfloor}\sum_{y=1}^{Y}\sum_{z=1}^{Z}\delta_{c,t}^{x,y,z}\bullet W(c)-\sum_{c=1}^{C}\sum_{x=\lceil X/2\rceil}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z}\delta_{c,t}^{x,y,z}\bullet W(c)\right|\le BSB \qquad \forall t \qquad (6.9)$$

$$\sum_{c=1}^{C}\sum_{x=1}^{X}\sum_{y=1}^{Y}\delta_{c,t}^{x,y,z}\bullet W(c)\le\sum_{c=1}^{C}\sum_{x=1}^{X}\sum_{y=1}^{Y}\delta_{c,t}^{x,y,z-1}\bullet W(c) \qquad \forall t,z\in\left[2..Z\right] \qquad (6.10)$$

$$\sum_{c=1}^{C}\sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{z=1}^{Z}\delta_{c,t}^{x,y,z}\bullet W(c)\le W_{Max}(t) \qquad \forall t \qquad (6.11)$$

$$\sum_{c=1}^{C}\sum_{x=1}^{X}\sum_{y=1}^{Y}\delta_{c,t}^{x,y,z}\bullet W(c)\le W_{Column} \qquad \forall z,t \qquad (6.12)$$

The solver CPLEX is capable of solving non-linear optimization models if the objective function is quadratic and the constraints are linear. The generated lower bound model was solved by CPLEX and experiments show that although this method creates very promising lower bounds for small size problems, it is not useful to find lower bound to larger problems because the solver is not capable of reaching a solution in a timely manner for such problems. Table 6.3 shows the optimal value of the objective function, the gap compared to the lower bound generated by linear relaxation and the gap measured from the lower bound generated by nonlinear formulation for selected problems. Examples 4.2.1 and 4.2.2 and 4.2.4 that were discussed earlier in chapter four are used.

Table 6.3: Comparison between lower bound and optimal objective function value for sample problems

Problem	**Objective function**	**Lower bound by linear relaxation**	**Lower bound by NL-LB formulation**
4.2.1	20	19 (5% gap)	20 (0% gap)
4.2.2	21	19 (10% gap)	21 (5% gap)
4.2.4 case 1	42	38 (9.5% gap)	39 (7.1 % gap)
4.2.4 case 3	41	38 (7.3 %gap)	39 (4.8% gap)

Chapter 7: Sample containership load planning problems

During the course of this research many attempts were made to obtain real containership stowage planning and origin destination data from the shipping lines. Unfortunately the industry refused to share data and thus measuring the actual benefit of applying the proposed algorithm to the real world practice cannot be made at this point. However the algorithm can be applied to simulated data of realistic size and the results can be compared with base case scenarios to demonstrate the potential benefits of using the solution approach. This chapter provides several examples all of which generated using the procedure discussed in 6.1.

7.1. Indented berth terminal operations

There are two fundamental approaches to deal with the challenge of handling large container vessels. The first approach is to increase the capacity of quay cranes which can be done through methods such as double cycling or double lifting or both. The second approach is to increase the capacity of the quay side interface. This requires implementation of indented berth or use of floating cranes which both allow handling containers from either side of the vessel. A floating crane is a quay crane mounted on a pontoon that can be self propelled. A number of barges or feeders are needed to support the floating cranes in moving the containers from/to the vessel. Therefore structural change in the berth system is not necessary in the case of floating cranes. The indented berth on the other hand has twice as much quay wall as compared to a traditional berth and can deploy up to twice as many cranes on a vessel. Figure 7.1

shows the layout of an indented berth. Since the vessel is handled from both sides the chance of quay side congestion decreases significantly. The system is designed to accommodate large vessels, so if small vessels are served, the infrastructure will be underutilized.

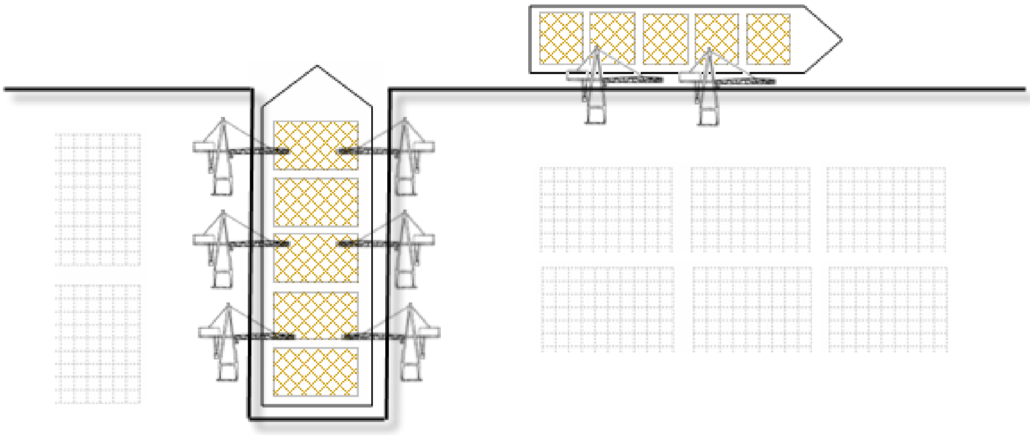


Figure 7.1: Indented berth vs. traditional berth

Figure 7.2 shows a containership at the Ceres Paragon container terminal at the port of Amsterdam. To increase the efficiency of the port system, besides optimal berth allocation decisions that must be made prior to the arrival, it is important to arrange the containers in the vessel in accordance with the configuration of the allocated cranes. Previous stowage planning algorithms do not account for the configuration of the cranes and thus they do not treat the indented berth terminal planning differently. An example is used to demonstrate the advantages of using the proposed solution approach in an indented berth operation.

Figure 7.2: A vessel entering the Ceres Paragon terminal at port of Amsterdam (photo courtesy http://www.portofamsterdam.nl)

To show the effect of considering crane split into stowage planning a 2000 TEU container ship is used as an example. This ship visits five ports to transport 3385 containers. The third port is equipped with an indented berth system. Configuration of the cranes and the transportation matrix are presented in figure 7.3 and table 7.1 respectively. The weights of the containers are generated randomly in the range of 1-20 tone. Trim, tilt and vertical stability constraints with a maximum 1% tolerance are observed. “The technical performance of the cranes is in the range of 50-60 box/hr, while in operation the performance is in the range of 22-30 box/h” (Steenkan et al. 2004). For simplicity and without lack of generality it is assumed that all cranes are identical and each can handle 25 containers per hour.

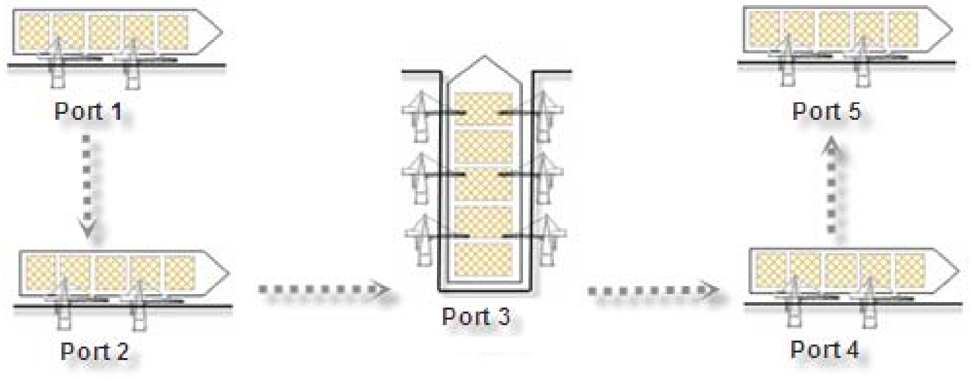


Figure 7.3: The shipping route and configuration of cranes at marginal and indented berth terminals

Table 7.1: Transportation matrix for indented berth operations problem

		Destination			
		2	**3**	**4**	**5**
Origin	**1**	463	141	308	685
	2		217	155	319
	3			270	155
	4				672

This example is solved for two scenarios. The objective function of scenario I and II are minimization of total berthing time and minimization of total shifts respectively. Table 7.2 shows the results for each scenario. According to this table, because of the proper horizontal distribution of the containers in scenario I, the utilization of individual cranes - which is the crane busy time over total time at port – and also average utilization of the cranes at each port is improved in 4 out of 5 ports. In scenario I, the vessel spend more time for rearranging the containers in the indented berth at port 3 compared to scenario II, however significant savings in berthing time at port 4 will be achieved. These improvements may come at cost of more container movements.

Table 7.2: Solution results for indented berth operations problem

		Port															
		1		2		3						4		5	
Scenario		Crane		Crane		Crane						Crane		Crane	
		1	**2**	**1**	**2**	**1**	**2**	**3**	**4**	**5**	**6**	**1**	**2**	**1**	**2**
I	# Containers	597	1000	651	641	415	391	359	375	366	343	723	743	918	913
	Utilization (%)	60	100	100	98	100	94	87	90	88	83	97	100	100	99
	Port Time (hr)	40		26.04		16.6						29.72		36.72	
	Avg. utilization	80.0%		99.0%		90.3%						98.5%		99.5%	
II	# Containers	1000	597	631	661	175	144	265	107	256	214	808	1048	927	904
	Utilization (%)	100	60	95	100	66	54	100	40	97	81	77	100	100	98
	Port Time (hr)	40		26.44		10.6						41.92		37.08	
	Avg. utilization	80.0%		97.5%		73.0%						88.5%		99.0%	

Table 7.3: Summary of the results for two indented berth operations scenarios

	Total Container Handling	Total Berthing Time (hr)	Overall Crane Utilization
I	5050	149.08	93.5%
II	4352	156.04	83.4%

Although total number of container handlings has gone up by 16% in scenario I compared to II, 4% reduction in total berthing time is gained which is the result of 12% improvement in overall crane utilization.

7.2. Technical and economical considerations in container load operations

In all the examples discussed so far, it was assumed that all cranes have similar performance. All approaches in the literature have either made the same assumption or have not accounted for the impact of crane operations on the stowage planning at all. However in the real world operations this is not the case meaning that technical specifications of the quay cranes differ over container terminals. Although it is not very common but practically the performance of the quay cranes on a given terminal may also be different. Our proposed formulation and solution algorithm is capable of accounting for the performance of quay cranes. Consider a containership that visits five ports and two cranes are available at each port. If all cranes have the same performance, the objective function of the problem will minimize the total turnaround time at all ports by optimizing the arrangement of the containers into the vessel with respect to the crane utilization and other operation constraints as discussed earlier. Now let us assume that the second port upgrades its cranes with the latest technology that makes each crane twice as fast. This means that handling time of each container will be cut into half at this port. The solution algorithm can take advantage of this feature by rearranging more containers at the second port to reduce the time needed for the rearrangement at the forthcoming ports if this contributes to overall improvement of total turnaround time. Similarly if in any of the given terminals, some of the cranes have a higher performance compared to the other ones there may be possibility of reducing total turnaround time by assigning more work to the higher performance cranes.

Based on the same argument, by converting time into money one can change the objective function of the problem to a function of economic value. So if the time unit spent at a given port is more expensive than others the ship planners may save money by spending time to rearrange containers in cheaper ports in order to spend less time and money in expensive ports. So although the total time spent at all ports and the total number of container handlings may be higher than the original solution, however the total cost of all operations could be lower. The following examples investigate these scenarios.

The 2000 TEU containership introduced in section 7.1 is used in this example. The vessel visits five ports, each of which make two quay cranes available. The transportation matrix is based on table 7.1. Stability constraints including horizontal and longitudinal equilibrium with 5% threshold are enforced. Also, weight of each tier must not exceed the weight of its underlying tier with an acceptable 1% tolerance. First, we set the objective function to minimize total turnaround time. We assume that cranes at all ports are of the same type and it takes four time units to move a container by each crane. Table 7.4 shows the summary of results including number of container shifts mandated by overstowage and turnaround time. As the table shows containers will only be shifted at the intermediate ports, because at the first and last port only loading and unloading happens respectively.

Table 7.4: Summary results for universal crane characteristics

	Port 1		**Port 2**		**Port 3**		**Port 4**		**Port 5**		**Total**
Crane	1	2	1	2	1	2	1	2	1	2	
No of containers	797	800	648	667	454	431	721	742	916	915	
Time unit	3188	3200	2592	2668	1816	1724	2884	2968	3664	3660	
Container shifts	0		161		102		58		0		321
Turnaround time	3200		2668		1816		2968		3664		14316

Now we assume that the two cranes at the third port are upgraded, such that each crane can handle a container in two time units. In other words cranes at port three are twice as fast as the rest of the cranes. Table 7.5 shows the summary of results for this case. As shown in the table, the number of container moves by the cranes at port 3 has increased drastically by 174%, however the total turnaround time is reduced by 3.4%.

Table 7.5: Summary results for different crane types at port 3

	Port 1		Port 2		Port 3		Port 4		Port 5		Total
Crane	1	2	1	2	1	2	1	2	1	2	
No of containers	800	797	593	626	776	790	721	715	914	917	
Time unit	3200	3188	2372	2504	1552	1580	2884	2860	3656	3668	
Container shifts	0		65		783		31		0		879
Turnaround time	3200		2504		1580		2884		3668		13836

If we change the cranes at port three again, and make them four times as fast as the rest of the cranes, each container handling will take one time unit by those cranes. With all other assumptions remaining intact, the results for this case are reported in table 7.6.

Table 7.6: Summary results for deployment of improved cranes at port 3

	Port 1		Port 2		Port 3		Port 4		Port 5		Total
Crane	1	2	1	2	1	2	1	2	1	2	
No of containers	800	797	589	605	827	814	719	702	916	915	
Time unit	3200	3188	2356	2420	827	814	2876	2808	3664	3660	
Container shifts	0		40		858		16		0		914
Turnaround time	3200		2240		827		2876		3664		12987

Comparison with the case that all cranes are uniform shows 9% improvement in total turnaround time that comes with 187% in container shifts. As it is shown in tables 7.4 through 7.6, the algorithm tries to reshuffle the containers at the port with faster cranes, in exchange of saving container shifts at the second and the fourth port. From the economical perspective if the shipping line is charged by time, having extra container moves at port three in the cases that upgraded cranes are available is advised. Otherwise if the ports charge per container handling basis, solution for minimizing total container moves should be used. In any case, the algorithm is flexible to create appropriate stowage and loading plan with respect to technical and economical considerations.

7.3. Container load planning for mega containerships

Like many other industries, transportation companies benefit from economies of scale in maritime shipping. Since the cost per TEU reduces with the increase of the capacity, there is a powerful trend to build larger vessels. The growth in capacity comes with increasing challenges to cope with large amount of containers to be transshipped in short periods of time, as the port time is an expensive resource. According to the Journal of Commerce, 42 ultra large container carriers (with a capacity of more than 10,000 TEU) are in operation in the world’s seas by August 2010[4]. One of the day to day problems to be solved is stowage planning for mega containerships. To show the capability of the proposed algorithm for dealing with real-size instances of stowage planning for mega containerships, a sample problem for a 12,000 TEU containership in a five port voyage is generated. There are 21,191

---

[4] 100th mega containership in Rotterdam, The Journal of Commerce Online, Aug 2010, http://www.joc.com/press-release/100th-mega-container-ship-rotterdam last visited: 3/27/2011

containers in total with different weights to be transported in this example. The vessel has 40 bays, 20 rows and each stack can hold 15 containers. There are four quay cranes assigned at the first and second port, six quay cranes at the third and forth port and five cranes will empty the ship at the last port. Real life operational constraints are imposed in this example. Horizontal and longitudinal stability must be met and total weight of each tier must not exceed the weight of the immediate tier underneath. A threshold of 5% is applied for both horizontal and longitudinal imbalance. Finally to make the problem challenging, in each column the weight of the top containers must be less than or equal to the weight of the bottom containers with a 5% acceptable weight difference. It is assumed that handling a container by each crane takes 1.5 minutes. Table 7.7 shows the summary of the transportation matrix. Table 7.8 shows a portion of the origin, destination and weight data for the containers. Since the complete data is more than 100 pages, complete data file is made available online for download[5].

Table 7.7: Transportation matrix for a 12,000 TEU containership

		Destination			
		2	**3**	**4**	**5**
Origin	**1**	2041	3212	1188	3156
	2		645	750	3048
	3			1355	1731
	4				4065

In total 48,329 crane moves are done in this example which involves 5,974 container moves mandated by overstowage. Overall utilization rate for cranes at all ports is

---

[5] http://www.eng.umd.edu/~masoud/dissertation_data

Table 7.8: Part of container data for a mega containership example[6]

No	O	D	W	No	O	D	W	No	O	D	W	No	O	D	W	No	O	D	W
1	1	2	27	2042	1	3	45	5254	1	4	78	9168	1	5	19	9964	2	3	39
2	1	2	15	2043	1	3	90	5255	1	4	94	9169	1	5	25	9965	2	3	95
3	1	2	11	2044	1	3	41	5256	1	4	63	9170	1	5	21	9966	2	3	92
4	1	2	73	2045	1	3	46	5257	1	4	89	9171	1	5	35	9967	2	3	55
5	1	2	32	2046	1	3	71	5258	1	4	37	9172	1	5	18	9968	2	3	27
6	1	2	95	2047	1	3	47	5259	1	4	42	9173	1	5	59	9969	2	3	18
7	1	2	47	2048	1	3	56	5260	1	4	60	9174	1	5	38	9970	2	3	35
8	1	2	53	2049	1	3	82	5261	1	4	72	9175	1	5	20	9971	2	3	86
9	1	2	26	2050	1	3	41	5262	1	4	94	9176	1	5	37	9972	2	3	53
10	1	2	99	2051	1	3	27	5263	1	4	66	9177	1	5	48	9973	2	3	3
11	1	2	60	2052	1	3	63	5264	1	4	74	9178	1	5	91	9974	2	3	62
12	1	2	27	2053	1	3	50	5265	1	4	24	9179	1	5	52	9975	2	3	97
13	1	2	56	2054	1	3	5	5266	1	4	84	9180	1	5	37	9976	2	3	27
14	1	2	16	2055	1	3	34	5267	1	4	50	9181	1	5	30	9977	2	3	39
15	1	2	15	2056	1	3	79	5268	1	4	5	9182	1	5	23	9978	2	3	68
16	1	2	51	2057	1	3	15	5269	1	4	14	9183	1	5	36	9979	2	3	48
17	1	2	41	2058	1	3	15	5270	1	4	55	9184	1	5	21	9980	2	3	42
18	1	2	93	2059	1	3	74	5271	1	4	11	9185	1	5	85	9981	2	3	10
19	1	2	86	2060	1	3	94	5272	1	4	47	9186	1	5	98	9982	2	3	71
20	1	2	82	2061	1	3	18	5273	1	4	83	9187	1	5	99	9983	2	3	20
21	1	2	62	2062	1	3	81	5274	1	4	26	9188	1	5	19	9984	2	3	2
22	1	2	12	2063	1	3	45	5275	1	4	50	9189	1	5	44	9985	2	3	54
23	1	2	13	2064	1	3	58	5276	1	4	12	9190	1	5	52	9986	2	3	15
24	1	2	95	2065	1	3	8	5277	1	4	20	9191	1	5	60	9987	2	3	10
25	1	2	22	2066	1	3	15	5278	1	4	4	9192	1	5	68	9988	2	3	21
26	1	2	61	2067	1	3	36	5279	1	4	71	9193	1	5	76	9989	2	3	12
27	1	2	43	2068	1	3	60	5280	1	4	50	9194	1	5	91	9990	2	3	8
28	1	2	52	2069	1	3	90	5281	1	4	87	9195	1	5	76	9991	2	3	65
29	1	2	94	2070	1	3	52	5282	1	4	68	9196	1	5	77	9992	2	3	69
30	1	2	23	2071	1	3	45	5283	1	4	10	9197	1	5	59	9993	2	3	95
31	1	2	78	2072	1	3	18	5284	1	4	78	9198	1	5	55	9994	2	3	1
32	1	2	39	2073	1	3	55	5285	1	4	47	9199	1	5	36	9995	2	3	71
33	1	2	35	2074	1	3	56	5286	1	4	41	9200	1	5	58	9996	2	3	65
34	1	2	31	2075	1	3	41	5287	1	4	96	9201	1	5	43	9997	2	3	44
35	1	2	85	2076	1	3	18	5288	1	4	34	9202	1	5	47	9998	2	3	90
38	1	2	8	2079	1	3	10	5291	1	4	23	9205	1	5	23	10001	2	3	46

---

[6] No: container identification number, O: origin port, D: destination port, W: container weight

83%. Table 7.9 shows the details of crane split solution based on the operating range of each crane, the number of containers handled by each crane and the crane utilization rates. In total the vessel spends 291.7 hour at all ports. The solution for this case was obtained in 62957 seconds (17.5 hours) on a computer running Microsoft Windows XP 32 bit with two Core Duo 2.66 GHz CPUs and 2 GB of RAM. The parallel implementation of the algorithm was used.

Table 7.9: Crane split and utilization rates for 12,000 TEU containership

		Crane number					
Port		**1**	**2**	**3**	**4**	**5**	**6**
1	Operation bay range	1-11	12-20	21-29	30-40	N/A	N/A
	No. of containers handled	2100	2700	2700	2097		
	Utilization rate per crane	78%	100%	100%	78%		
	Average utilization rate	89%					
	Time at port (hours)	67.5					
2	Operation bay range	1-10	11-20	21-30	31-40	N/A	N/A
	No. of containers handled	1200	2305	1794	1199		
	Utilization rate per crane	52%	100%	78%	52%		
	Average utilization rate	70%					
	Time at port (hours)	57.6					
3	Operation bay range	1-6	7-13	14-18	19-24	25-30	31-40
	No. of containers handled	1043	904	1980	2058	2079	1161
	Utilization rate per crane	50%	43%	95%	99%	100%	56%
	Average utilization rate	74%					
	Time at port (hours)	52					
4	Operation bay range	1-6	7-13	14-18	19-24	25-29	30-40
	No. of containers handled	1281	1703	1883	2184	1877	2081
	Utilization rate per crane	59%	78%	86%	100%	86%	95%
	Average utilization rate	84%					
	Time at port (hours)	54.6					
5	Operation bay range	1-8	9-16	17-24	25-32	33-40	N/A
	No. of containers handled	2400	2400	2400	2400	2400	2400
	Utilization rate per crane	100%	100%	100%	100%	100%	100%
	Average utilization rate	100%					
	Time at port (hours)	60					

The program also generates the details for placement of individual containers as well as operations of each crane including loading, shifting and unloading a container. Table 7.10 shows a portion of the output for the above example. Since complete output is over 500 pages, the complete solution for interested reader is made available online[7]. In table 7.10, the first column provides container information including container identification number, origin and destination port of the container and the cell assignment information including bay, row and column. In the second column when one of the letters “L”, “U” or “S” are followed by a number it means that the container will be “Loaded”, “Unloaded” or “Shifted” by the crane number that appears after the indicating letter. The port in which crane operation happens is shown by letter “P” followed by the port number. So the first row in table 7 shows that container number 10864 which originated from port 2 and is destined to port 4, is assigned to the column located at bay 3, row 1 and the vertical position of the container in the column is 5 from bottom. This container is loaded to the assigned cell by crane 1 in port 2. Later on in port 4 the container is unloaded by crane 1. This container does not need to be shifted throughout the voyage. On the other hand container 10865 that goes from port 2 to port 4, is assigned to the column at bay 3 and row 7 which is the third container in the column. This container is loaded by crane 1 at port 2, and must be shifted by the crane 1 at port 3. As a result of the shift, this container will be relocated to bay 37, row 18. At the destination port the container will be unloaded by crane 6. In total 44824 instructions for container placement and handling are generated by the program. The results can be communicated electronically to the port authorities, ship planer and other parties involved in XML

---

[7] http://www.eng.umd.edu/~masoud/dissertation_data

format. A graphical representation of the complete solution for the mega containership is provided in Appendix C.

Table 7.10: Partial solution details for 12,000 TEU containership load plan

Container information	Crane operations			
#10864 [2,4] (3,1,5)	L1	P2		
#10864 [2,4] (3,1,5)	U1	P4		
#10865 [2,4] (3,7,3)	L1	P2	S1	P3
#10865 [2,4] (37,18,2)	U6	P4		
#10866 [2,4] (4,3,12)	L1	P2	S1	P3
#10866 [2,4] (23,4,11)	U4	P4		
#10867 [2,4] (1,6,10)	L1	P2		
#10867 [2,4] (1,6,10)	U1	P4		
#10868 [2,4] (21,19,11)	L3	P2	S4	P3
#10868 [2,4] (13,11,10)	U2	P4		
#10869 [2,4] (22,17,9)	L3	P2	S4	P3
#10869 [2,4] (27,13,8)	U5	P4		
#10870 [2,4] (20,19,11)	L2	P2	S4	P3
#10870 [2,4] (27,14,10)	U5	P4		
#10871 [2,4] (40,6,1)	L4	P2		
#10871 [2,4] (40,6,1)	U6	P4		
#10872 [2,4] (40,13,14)	L4	P2	S6	P3
#10872 [2,4] (29,11,12)	U5	P4		
#10873 [2,4] (3,9,13)	L1	P2	S1	P3
#10873 [2,4] (13,17,11)	U2	P4		
#10874 [2,4] (21,12,5)	L3	P2		
#10874 [2,4] (21,12,5)	U4	P4		
#10875 [2,4] (3,16,7)	L1	P2	S1	P3
#10875 [2,4] (13,14,6)	U2	P4		
#10876 [2,4] (38,10,12)	L4	P2	S6	P3
#10876 [2,4] (27,13,10)	U5	P4		
#10877 [2,4] (21,12,3)	L3	P2		
#10877 [2,4] (21,12,3)	U4	P4		

7.4. Stability constraints and container load planning

In order to show the impact of enforcing stability constraints - specifically vertical stability - on the problem complexity, the example in section 7.3 is used. The constraints concerning the weight of individual containers and the weight of tiers are relaxed in this case while the number of visiting ports, availability of the cranes and list of containers remain the same. Similar to the previous case, the horizontal and longitudinal imbalance threshold is set to 5%. In total 42,387 crane moves are involved in this example which calls for only five container moves mandated by over stowage. Overall utilization rate for cranes at all ports is 89%. Table 7.11 shows the details of crane split and utilization rate. Total turnaround time at all ports is 243.2 hours. The problem was solved on the same computer as described in previous section and the solution was obtained in 13776 seconds (3.8 hours).

A comparison between these two examples shows that relaxing the vertical stability constraints has reduced the solution time from 17.5 hours to 3.8 hours (78% reduction). The total ship turnaround time is also reduced from 291.7 hours to 243.2 hours (17% reduction). This is a result of a sharp decline in number of container shifts from 5974 to 5, however in 16285 occasions heavy containers are sitting on top of lighter containers in the latter example. Increasing the height of stack and tightening the vertical stability constraints dramatically impacts the complexity, running time and structure of the solution. A comparison between tables 7.10 and 7.11 shows how the arrangements of the quay cranes are adjusted according to the problem characteristics.

Table 7.11: Crane split and utilization rates for 12,000 TEU containership without vertical stability constraints

		Crane number					
Port		**1**	**2**	**3**	**4**	**5**	**6**
1	Operation bay range	1-10	11-20	21-30	31-40	N/A	N/A
	No. of containers handled	2400	2400	2397	2400		
	Utilization rate per crane	100%	100%	100%	100%		
	Average utilization rate	100%					
	Time at port (hours)	60					
2	Operation bay range	1-10	11-21	22-30	31-40	N/A	N/A
	No. of containers handled	1500	1676	1620	1688		
	Utilization rate per crane	89%	99%	96%	100%		
	Average utilization rate	96%					
	Time at port (hours)	42.2					
3	Operation bay range	1-6	7-12	13-18	19-24	25-31	32-40
	No. of containers handled	907	1062	1170	1140	1154	1514
	Utilization rate per crane	60%	70%	77%	75%	76%	100%
	Average utilization rate	76%					
	Time at port (hours)	37.9					
4	Operation bay range	1-6	7-12	13-18	19-25	26-31	32-40
	No. of containers handled	907	1038	1092	1245	1350	1727
	Utilization rate per crane	53%	60%	63%	72%	78%	100%
	Average utilization rate	71%					
	Time at port (hours)	43.2					
5	Operation bay range	1-8	9-16	17-24	25-32	33-40	N/A
	No. of containers handled	2400	2400	2400	2400	2400	2400
	Utilization rate per crane	100%	100%	100%	100%	100%	100%
	Average utilization rate	100%					
	Time at port (hours)	60					

To study the impact of partial vertical stability enforcement, we keep the horizontal and latitudinal constraints and only enforce the tier weight imbalance constraints. That is total weight of the containers in each tier must be less than or equal to the total weight of the containers in the tier immediately under it. The results for this case are reported in table 7.12.

Table 7.12: Crane split and utilization rates for 12,000 TEU containership with partial vertical stability constraints

		Crane number					
Port		**1**	**2**	**3**	**4**	**5**	**6**
1	Operation bay range	1-10	11-20	21-30	31-40	N/A	N/A
	No. of containers handled	2400	2400	2400	2397		
	Utilization rate per crane	100%	100%	100%	100%		
	Average utilization rate	100%					
	Time at port (hours)	60					
2	Operation bay range	1-9	10-20	21-30	32-40	N/A	N/A
	No. of containers handled	1590	1580	1627	1717		
	Utilization rate per crane	93%	92%	95%	100%		
	Average utilization rate	95%					
	Time at port (hours)	42.9					
3	Operation bay range	1-6	7-12	13-18	19-24	25-31	32-40
	No. of containers handled	974	1089	1144	1163	1303	1645
	Utilization rate per crane	59%	66%	70%	71%	79%	100%
	Average utilization rate	74%					
	Time at port (hours)	41.1					
4	Operation bay range	1-6	7-12	13-18	19-24	25-31	32-40
	No. of containers handled	981	1130	1264	1292	1871	1372
	Utilization rate per crane	52%	60%	68%	69%	100%	73%
	Average utilization rate	70%					
	Time at port (hours)	46.8					
5	Operation bay range	1-8	9-16	17-24	25-32	33-40	N/A
	No. of containers handled	2400	2400	2400	2400	2400	2400
	Utilization rate per crane	100%	100%	100%	100%	100%	100%
	Average utilization rate	100%					
	Time at port (hours)	60					

The running time for this case is 17765 seconds (4.9 hours). Total ship turnaround time is 250.8 hours and 957 container shifts are required. In 11,304 cells, individual heavy containers are placed on top of lighter containers in a column. Table 7.13 provides a summary of the results for the three cases discussed in this section.

Table 7.13: Comparison of stability constraint enforcement policies on 12,000 TEU containership load planning

Constraints imposed	Imbalance threshold	Running time (hours)	Total turnaround time (hours)	Average crane utilization rate	Total container shifts	Total heavy-on-light cases
Horizontal stability Longitudinal stability Tier weight stability Container weight	5% 5% 5% 5%	17.5	291.7	83%	5974	0
Horizontal stability Longitudinal stability Tier weight stability	5% 5% 5%	4.9	250.8	88%	957	11304
Horizontal stability Longitudinal stability	5% 5%	3.8	243.2	89%	5	16285

7.5. Changes in the demand and container load planning

In all the examples solved so far, it was assumed that the transportation matrix is known and given prior to the ship's departure. We also started from an empty vessel at the first port. However due to the nature of real operations, the demand is subject to uncertainty and may change. The following example presents a case that involves change in the origin-destination matrix after the departure of the vessel. In order to address the issue, we solve the problem again for the forthcoming ports, by starting from an initial existing solution as opposed to having an empty vessel. Since the solution approach is agile enough, by solving the problem iteratively throughout the operations, the stowage pattern can be adjusted based on the changes in the container list. However, we assume that demand fluctuation does not make the problem infeasible, such that the demand does not exceed the capacity of the vessel at any port.

Consider a 2000 TEU containership visiting five ports with the initial transportation matrix shown in Table 7.14. Stability constraints including horizontal and longitudinal equilibrium with 5% threshold are enforced for 4240 containers. Also, weight of each tier must not exceed the weight of its underlying tier with an acceptable 5% tolerance. Containers have different weight with a random distribution and each port has two cranes available. Table 7.15 shows a summary of the results.

Table 7.14: Initial transportation matrix

		Destination			
		2	**3**	**4**	**5**
Origin	**1**	567	51	594	387
	2		477	276	214
	3			285	138
	4				1251

Table 7.15: Summary results for all ports using original demand

	Port 1		**Port 2**		**Port 3**		**Port 4**		**Port 5**		**Total**
Crane	1	2	1	2	1	2	1	2	1	2	
No of containers	800	799	771	789	471	453	1252	1236	991	1009	
Container shifts	0		26		27		72		0		125
Turnaround time	800		789		471		1252		1009		4321

Now we assume that after leaving the first port, demand for shipping containers from port three to port five drops by 100 containers. The new transportation matrix is shown in Table 7.16.

Table 7.16: Modified transportation matrix

		Destination			
		2	**3**	**4**	**5**
Origin	**1**	567	51	594	387
	2		477	276	214
	3			285	38
	4				1251

Having fixed the stowage pattern for the first port, Table 7.16 shows the summary of results after solving the problem for ports two through five.

Table 7.17: Summary results for four ports using modified demand

	Port 1		**Port 2**		**Port 3**		**Port 4**		**Port 5**		**Total**
Crane	1	2	1	2	1	2	1	2	1	2	
No of containers	800	799	766	788	414	409	1250	1235	943	957	
Container shifts	0		20		26		69		0		115
Turnaround time	800		788		414		1250		957		4209

The containers that were removed from the demand were chosen randomly and comparison between tables shows that number of container shifts is decreased by 10 crane moves. Have we had the modified transportation matrix before the operation starts, we could have solved the problem for all ports based on the new demand. Table 7.18 shows the summary of results based on this assumption.

Table 7.18: Summary results for all ports using modified demand

	Port 1		Port 2		Port 3		Port 4		Port 5		Total
Crane	1	2	1	2	1	2	1	2	1	2	
No of containers	800	799	770	784	413	410	1243	1230	946	954	
Container shifts	0		16		26		57		0		99
Turnaround time	800		784		413		1243		954		4194

As tables 7.17 and 7.18 show, both total turnaround time and number of container shifts are slightly higher when solution is adjusted amid the operation. In other words in this example the cost of not having perfect information from the beginning of the planning stage is 15 time units and 16 container moves.

Chapter 8: Summary and future research

Because of the tense competition among ports in recent years, improving the operational efficiency of ports has become an important issue in containership operations. One of the major performance measures is the berthing time at a port. Arrangement of containers both within the container terminal and on the containership play a vital role in determining the berthing time. The berthing time of a containership is mainly composed of the unloading and loading time of containers. Containers are loaded to and unloaded from the containership using quay cranes. The problem of allocating quay cranes to ship's sections is known as crane split. In some cases up to ten quay cranes might be allocated to a ship. Technical requirements determine the range in which each quay crane can operate. The ship's turnaround time is determined by the time that the last allocated carne finishes its job. Since the distribution of containers over the bays affects crane utilization and overall ship berthing time, crane split and stowage problem are interrelated. Given the configuration of cranes at each visiting port, stowage planning must take into account the utilization of quay cranes as well as the reduction of unnecessary shifts simultaneously to minimize the total time at ports over the voyage. Integration of the stowage plan and the crane split results in a more efficient working instruction which increases overall port utilization.

There are many operational regulations in the real world operation. Some of them apply to only certain type of containerships. The designed solution approach should adopt these operation requirements. Stability constraints are of the greatest importance. The trim of the vessel which is the difference in the height of waterline between bow and stern must not exceed a given threshold. The containers on the bow side create a positive tilt while the containers at the stern side create a negative one. Total longitudinal tilt is the summation of these two tilts, the closer it gets to zero the better the trim is. Same calculations are valid for horizontal stability which is the difference between the tilt created by the containers of the left and the ones on the right side of the ship. Although the imbalance between left and right, bow and stern can be adjusted using ballast water to some extent, it will affect both the draft and the performance of the ship and may result in more fuel consumption. Vertical stability rules require that the weight on each tier must be greater or equal than the weight on the tier immediately over it. There also might be some limitations on the total weight of a single column, such as heavy containers must not be put on top of lighter ones. This is the reason that all empty containers are loaded above deck. Finally the total weight of the cargo must comply with the maximum allowed draft which is the depth of vessel below water, therefore at some ports the ship may not be loaded at full capacity.

This dissertation presented an integer programming model and a genetic algorithm which focuses on the containership load planning problem during a voyage. A new compact and efficient encoding based on sorting and assignment policy is introduced. The evaluation procedure of the GA decodes and calculates the value of desired elements of objective function based on the encoded solution. Objective function of the problem is different from those of traditional stowage planning problems in the sense that it tries to minimize the total time spent at all ports by minimizing shifts and maximizing crane utilization simultaneously.

The solution approach is very flexible and can easily embrace real world constraints and parameters such as stability and operational constraints, consideration of various cost of time at ports and technical specifications of quay cranes. Solutions obtained by GA are compared with the ones from the exact solvers for small size problems and in all cases the GA solutions are optimal. However this does not mean that the method guarantees optimality. A non linear lower bound model is presented. Results of other numerical experiments show that a feasible solution can be reached within a reasonable time for practical problems. The numerical experiments affirm that significant savings in overall ship turnaround time at ports can be achieved by considering stowage planning and crane utilization simultaneously. A parallel version of the genetic algorithm is developed and the effect of having multiple processors on running time is investigated.

Crane double cycling is a method of improving efficiency of the cranes at the container ports. In a typical container unloading operation, the crane is sent back to the vessel empty every time that it unloads a container to berth side. In an attempt to reduce the idle cycling time, double cycling solution was introduced in which the crane will load a new container every time it has unloaded a container as such the total berthing time will improve. Goodchild and Daganzo (2007) studied the impacts of crane double cycling on containership turnaround time and results indicate that the practice can reduce the ship dwell time on a single port. Schedule planning for double cycling is proven to be very difficult for real-world cases. However one can argue that in addition to the appropriate crane schedule programming, the stowage planning of the vessel is an important factor for an efficient double cycling operation.

Especially in the presence of multiple cranes, double cycling can be applied in its full potential only if the arrangement and distribution of the containers allows for simultaneous loading and unloading operations. Therefore if the ship planners are informed in advance that double cycling is practiced in the forthcoming ports, they must adjust the storage of the containers to utilize the full potential of the cranes. Due to the flexibility in evaluation procedure, the containership load planning solution proposed in this dissertation can be extended to account for crane double cycling. Incorporating crane double cycling into stowage planning, investigating the impacts of double cycling on the ship turnaround time over her journey as opposed to a single port and studying the solution time for the combined approach is subject of future research.

In order to increase productivity, ports have been pressuring crane manufacturers to increase acceleration, speed and handling capabilities of the cranes. As a result some container ports are equipped with cranes that are capable of lifting multiple containers at the time (mostly double lifting). Similar to the argument made for the double cycling case, it is of special importance to account for such operations at the containership stowage planning stage. Again the proposed solution approach can be modified to take advantage of potential time savings offered by multi-container lifting feature, while designing the stowage planning scheme. Combination of double cycling and multi-crane lifting and the impact on containership stowage planning is another subject for future research.

In this research it was assumed that the information is perfect meaning that list of containers to be transported is known and is made available in advance. However, similar to any other real transportation system, the containership operations are subject to uncertainty. The proposed solution approach addresses the unexpected changes in demand by solving the problem iteratively using the updated transportation matrix. The algorithm is agile enough to update the stowage pattern caused by last minute changes prior to arriving to any future port. Introducing stochasticity of the demand to the model and investigating the benefits of such approach is an area of future research. One can compare the results of stochastic optimization with the results produced by iterative solution of the deterministic model under various scenarios. It was also assumed that the containership has enough capacity to accommodate all the containers throughout the operations. In case that the number of containers exceeds the capacity of the vessel, the algorithm should be able to optimally pick and choose the containers to be transported. Enhancing both the mathematical model and the algorithm to perform container selection is an interesting subject for future studies.

Numerical experiments show that the complexity of this class of problems goes hand in hand with the height of the stack and the enforcement of vertical stability constraints. This makes the containership load planning problem a good candidate for benchmarking commercial solvers. The sample problems in this dissertation were solved using ILOG CPLEX Version 11. A side by side comparison between performance of CPLEX and other solvers (e.g. XPRESS, GAMS and LINDO) based on generating problems with different degrees of complexity from the proposed mathematical model is a subject of future investigation.

Finally as it was discussed the containership planning process follows a multi-tier hierarchical decision making model which consists of ship routing, berth planning, quay crane scheduling and unload/load sequencing. The decision made at a higher tier imposes constraints to the solution at lower tier. Potential benefits may be introduced to the containership planning process by integrating the decisions at different tiers. Joint optimization of operations between the yard operations and stowage planning is an example. The integration of load/unload planning with containership ship routing problem, and combining berth allocation with containership load planning are promising topics for future research.

Bibliography

- Alander, J.T., (1992) On optimal population size of genetic algorithms, CompEuro '92, Computer Systems and Software Engineering Proceedings, pp. 65-70.
- Ambrosino D., Sciomachen A., Tanfani E.(2004) Stowing a containership: the master bay plan problem, Transportation Research Part A 38 pp 81–99.
- Aslidis A. H. (1989) Combinatorial algorithms for stacking problems, Doctoral Dissertation, Massachusetts Institute of Technology.
- Aslidis A. H. (1990) Minimization of overstowage in containership operations. Operational Research, 90:451–471.
- Avriel M, Penn M (1993) Exact and approximate solutions of the containership stowage problem. Computers and Industrial Engineering 25(1–4): 271–274.
- Avriel M, Penn M, Shpirer N (2000) Containership stowage problem: complexity and connection to the coloring of circle graphs. Discrete Applied Mathematics 103(1–3):271–279.
- Avriel M, Penn M, Shpirer N, Witteboon S (1998) Stowage planning for containerships to reduce the number of shifts. Annals of Operations Research 76: 55–71.
- Barrass C. B. and Derrett D. R. (2000), Ship Stability for Masters and Mates, Butterworth Heinemann, ISBN 0750641010.

- Pielage B.J., Conceptual Design of Automated Freight Transport Systems, Methodology and Practice, PhD Thesis, The Netherlands TRAIL Research School, 2005
- Bischoff E.E., Ratcliff M.S.W. (1995), Loading Multiple Pallets, Journal of Operational Research Society 46:1322-1336.
- Bish E K (2003) A multiple-crane-constrained scheduling problem in a container terminal. European Journal of Operational Research 144: 83–107
- Bomba Michael S., Harrison Robert and Walton C. Michael, (2001), Planning for Megacontainerships: Statewide Transportation Planning Approach, Transportation Research Record No. 1777, p. 129.137, 2001
- Broy Manfred (2002), Software Pioneers: Contributions to Software Engineering, Springer Verlag, ISBN: 3540430814.
- Chen C.S, Lee S.M., Shen Q.S. (1995), An analytical model for container loading problem, European Journal of Operational Research No 80 pp.68-76.
- Cheng Runwei, Gen Mitsuo (2000), Genetic Algorithms and Engineering Optimization, Wiley-IEEE, ISBN 0471315311.
- Coffman Jr. E. G., Garey M.R., Johnson, D.S. (1996) Approximation algorithms for NP-hard problems, PWS Publishing Co. , ISBN:0-534-94968-1, Pages: 46 - 93 .
- Collins J.J., Eaton M. (1997), A Global Representation Scheme for Genetic Algorithms, Proceedings of International Conference on Computational Intelligence, Dortmund, Germany, pp. 1-15, Springer LNCS 1226.

- Cudahy B. J. (2006), Box Boats: How Container Ships Changed the World, Fordham University Press
- Daganzo C. F. (1989). The crane scheduling probem. Transportation Research B: Methodology, 23:159–175.
- De Jong. K. (1975) An Analysis of the behavior of a class of genetic adaptive systems. PhD. Thesis, University of Michigan.
- Delgado Alberto, Jensen R. and Schulte C. (2009), Generating Optimal Stowage Plans for Container Vessel Bays, Principles and Practice of Constraint Programming - CP 2009, Vol. 5732:6-20.
- Dorigo Marco, Thomas Stützle (2004), Ant Colony Optimization, MIT Press, ISBN: 0-262-04219-3
- Dubrovsky O., Levitin G., and Penn M. (2002), "A genetic algorithm with a compact solution encoding for the containership stowage problem", Journal of Heuristics 8, pp. 585–599
- Dyckhoff H., (1990) A typology of cutting and packing problems. European Journal of Operational Research, 44:145–159.
- Eiben A.E. and Smith J.E. (2003) Introduction to Evolutionary Computing, Springer, ISBN 3-540-40184-9.
- Fogel D. B. and Atmar, J. W. (1990) Comparing genetic operators with gaussian mutations in simulated evolutionary processes using linear systems. Biological Cybernetics 63, pp. 111-114.
- Giemsch P. (2004) Stowage Planning seen as a 4D-Packing Problem, 1st ESICUP Meeting, Lutherstadt Wittenberg, Germany, March 18-20, 2004

- Giemsch P., Jellinghaus A (2003). Organization models for the containership stowage problem. Paper presented at the annual international conference of the German Operations Research Society (OR 2003), Sep 3–5, Heidelberg.
- Glover F. (1986). Future paths for Integer Programming and Links to Artificial Intelligence, *Computers and Operations Research*, Vol. 5, pp:533-549.
- Glover F., Laguna M. (1997). Tabu Search. Kluwer Academic Publishers, Boston, MA.
- Goodchild A.V. and C.F. Daganzo (2007), "Crane double cycling in container ports: Planning methods and evaluation", Transportation Research Part B, 41:8, pp. 875-891.
- Goodchild, A.V., C.F. Daganzo (2005). Crane Double Cycling in Container Ports: Effect on Ship Dwell Time, Research Report, Institute of Transportation Studies, University of California , Berkeley.
- Goldberg, D.E. (1989). Genetic Algorithms in search, optimization and machine learning, Addison-Wesley.
- Guan, Y., W. Q. Xiao, R. K. Cheung, and C. L. Li (2002). A Multiprocessor Task Scheduling Model for Berth Allocation: Heuristic and Worst Case Analysis. Operations Research Letters, 30, 343–350.
- Haghani A, Kaisar E I (2001) A model for designing container loading plans for containerships. Working paper, University of Maryland, presented at Transportation Research Board 2001 Annual Meeting.

- Holland, J. (1975). Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor.
- Imai Akio, Kazuya Sasaki, Etsuko Nishimura, Stratos Papadimitriou. (2006) Multi-objective simultaneous stowage and load planning for a containership, with container rehandle in yard stacks, European Journal of Operational Research, 171: 373-389.
- Akio Imai, Etsuko Nishimura, Masahiro Hattori, Stratos Papadimitriou (2007), Berth allocation at indented berths for mega-containerships, European Journal of Operational Research, Volume 179, Issue 2, 1 June 2007, Pages 579-593, ISSN 0377-2217.
- Iris F.A. Vis, Rene de Koster.(2003) Transshipment of containers at a container terminal:An overview, European Journal of Operational Research 147, pp:1–16
- Jensen R. Tommy, Toft Bjarne (1995), Graph coloring problems, Wiley-Interscience, New York,ISBN 0-471-02865-7.
- Kim, K.H., Park, Y.M., (2004). A crane scheduling method for port container terminals. European Journal of Operation Research 156, 752-768.
- Levinson Marc (2006), The Box, How the Shipping Container Made the World Smaller and the World Economy Bigger. Princeton University Press. ISBN 0-691-12324-1.
- Lewis Ted G., H. El-Rewini (1992), Introduction to Parallel Computing, Prentice-Hall, Inc. pp 31-32, 38-39.

- Lim, A., Rodrigues, B., Xiao, F., Zhu, Y., (2004). Crane scheduling with spatial constraints. Naval Research Logistics 51, 386-406.
- Lloyd's register, (2003), Ultra large containership (ULCS) fact sheet, Lloyd's register publications, www.lr.org/Publications
- Lodi Andrea, Martello Silvano, Monaci Michele, (2002), Two-dimensional packing problems: A survey, European Journal of Operational Research 141:241–252
- Martello S., Pisinger D., Vigo D. (2000) The Three-Dimensional Bin Packing Problem, Operations Research 48:256-267
- Meersmans, P.J.M., Dekker, R. (2001), Operations research supports container handling, Technical Report EI 2001-22, Erasmus University Rotterdam, Econometric Institute
- Moccia L., J.F. Cordeau, M. Gaudioso, and G. Laporte. (2006), A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. Naval Research Logistics, 53:45–59.
- Papadimitriou Christos H. , Steiglitz Kenneth (1998), Combinatorial Optimization: Algorithms and Complexity, ISBN 0486402584, Dover Publications
- Park Y-M, Kim K H (2003) A scheduling method for berth and quay cranes. OR Spectrum 25: 1–23
- Peterkofsky R I, Daganzo C F (1990), A Branch and Bound Solution Method for the Crane Scheduling Problem, Transportation Research-B, 24B: pp. 159-172

- Pisinger David, Hans Kellerer, Ulrich Pferschy (2004), Knapsack Problems, Springer, ISBN 3540402861.
- Rawlins J. E. Gregory (1991), Foundations of genetic algorithms, Morgan Kaufmann, ISBN 1558601708.
- Rigby GR, Taylor AH, Hallegraeff GM, Mills P (1995), Progress in research and management of ship's ballast water to minimise the transfer of toxic dinoflagellates. In: Lassus P, Arzul G, Erard E, Gentien P, Marcaillou C (eds), Harmful marine algal blooms, Lavoisier/Intercept, Paris, pp. 821-824.
- Rodrigue Jean-Paul, Comtois C. and Slack B. (2009), Geography of Transport Systems, ISBN 978-0415483247.
- Scheithauser G. (1991) A three-dimensional bin packing algorithm, Journal of Information Processing and Cybernetics 27:263-271
- Scheithauser G. (1999) LP-based bounds for the container and multi-container loading problem, International Transactions in Operational Research 6 (2), 199–213.
- Shields J J. (1984) Container stowage: a computer aided preplanning system. Marine Technology 21: 370–383.
- Schaffer J. D., Caruana R. A., Eshelman L. J. and Das R. (1989) A study of control parameters affecting online performance of genetic algorithms for function optimization. Proceedings of the Third International Conference on Genetic Algorithms, 51-60. La Jolla, CA: Morgan Kaufmann.

- Schonfeld P. , O. Sharafeldien (1985), Optimal Berth and Crane Combinations in Containerports, Journal of Waterway, Port, Coastal and Ocean Engineering, Vol. 111, No. 6, November/December, pp. 1060-1072
- Steenkan Dirk, Voss Stefan and Stahlbock Robert (2004), Container terminal operation and operations research, a classification and literature review, OR Spectrum, 26: 3-49
- Toussaint Marc, Compact genetic codes as a search strategy of evolutionary processes, 2005, In A. H. Wright, M. D. Vose, K. A. D. Jong, and L. M. Schmitt, editors, Foundations of Genetic Algorithms, 8th International Workshop, FOGA 2005, Aizu-Wakamatsu City, Japan, January 5-9, 2005, Revised Selected Papers, volume 3469 of Lecture Notes in Computer Science, pages 75–94. Springer, ISBN:0302-9743.
- Todd D.S and P. Sen (1997), "A Multiple Criteria Genetic Algorithm for Container Ship Loading", Proceedings of the Seventh International Conference on Genetic Algorithms.
- UNCTAD, Review of Maritime Transportation 2010, United Nations Publication, UNCTAD/RMT/2010, ISBN 978-92-1-112810-9.
- Wijnolst N., Scholtens M., Waals F., 1999, Malacca-Max:The Ultimate Container Carrier, Delft Marine Technology Series 1383-7249 14
- Wilson I.D., Roach P.A (1999), Principles of combinatorial optimization applied to containership stowage planning, Journal of Heuristics, 5:403-418.

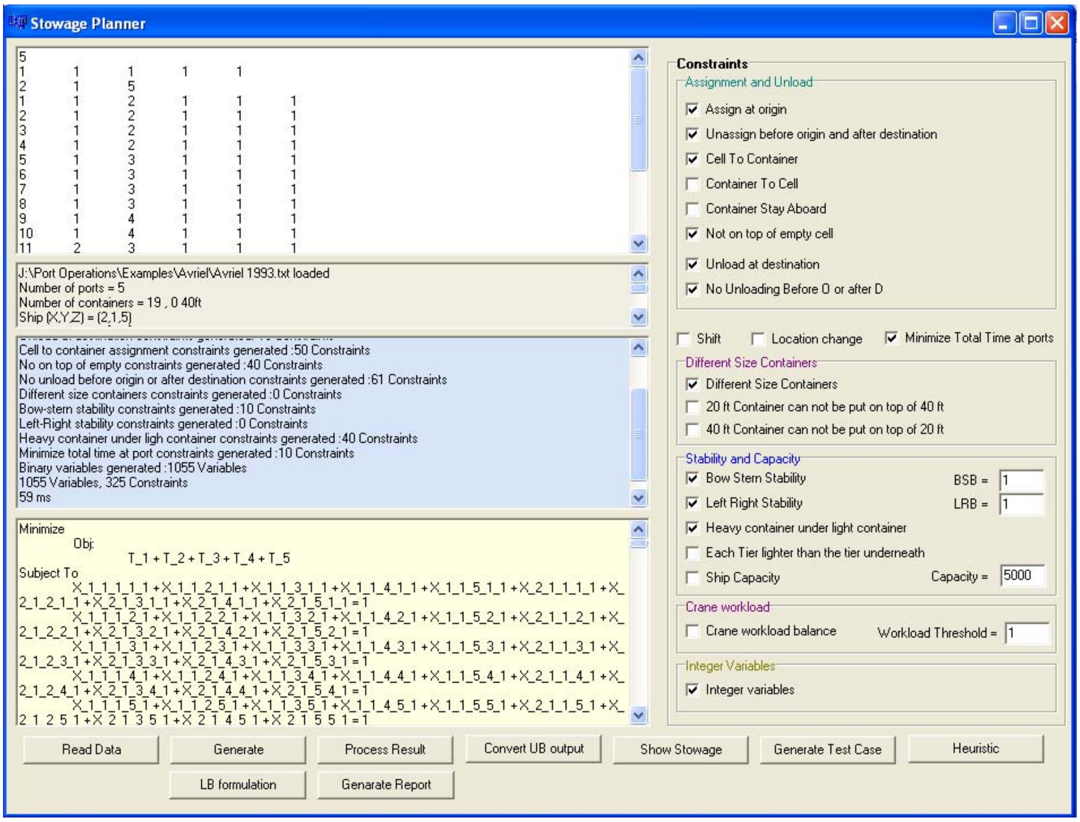
- Wilson I.D., Roach P.A (2000), Container stowage planning: a methodology for generating computerised solutions, Journal of the Operational Research Society, Vol. 51, 11:1248-1255.
- Wilson, D.J., Roach, P.A., and Ware A.J., (2001), “Container Stowage Pre-Planning: Using Search to Generate Solution a Case Study,” Knowledge-Based Systems 14, 3-4, pp. 137-145.
- Winter Thomas (1999), Online and Real-Time Dispatching Problems, PhD Dissertation, Vom Fachbereich f¨ur Mathematik und Informatik, der Technischen Universit¨at Braunschweig.
- Yang H. C. (2004), The impact of bigger vessels on shipping & ports, Shipping, Logistics and Ports Research Center , Korean Maritime Institute.
- Zhang Haipeng, Kap Hwan Kim (2009), Maximizing the number of dual-cycle operations of quay cranes in container terminals, Computers & Industrial Engineering, Volume 56, Issue 3, Intelligent Manufacturing and Logistics, April 2009, Pages 979-992, ISSN 0360-8352.

Appendix A: Algorithm Implementation Interface

To implement the solution discussed in this dissertation a computer program is written in C++ which has the following features:

- Generating random test problems in accordance with the procedure described in 6.1.
- Generating input file for CPLEX solver based on the mathematical model described in 3.6.
- Solving the problem based on the proposed algorithm in 5.2.
- Visualizing the solution provided by the mathematical solver and the genetic algorithm.
- Generating detailed unloading, shifting and loading plan at each port based on the solution.

Figure A.1 shows the main interface of the program where assumptions and parameters for the objective function and constraints can be set by the user. In this figure an expanded form of the mathematical formulation for a sample problem is produced which can be sent to the CPLEX optimization engine. Statistics for the number of constrains and variables are provided. Figure A.2 shows the user interface for generating random test cases for containership load planning. Sample problems are produced based on the specified parameters and layout of the vessel. The number of visiting ports and different characteristics of the containers including weight, size and type can be adjusted. Figure A.3 shows the interface for the genetic algorithm solver where different parameters for solving the problem can be set. After solving the problem summary of the solution attributes is reported and the details of unload, shift and loading plan for containers as well as crane operations at each port is reported in a file. The program produces a convergence diagram for the algorithm which shows the average, best and worst fitness value of each iteration as shown in figure A.4. Even small containership load problems involve a large number of variables. This makes keeping track of location of individual containers in the vessel at different ports very difficult for the user. A visualization tool is developed that automatically displays the stowage plan of the vessel at different bays and columns of the ship is a color coded fashion. The colors show the origin and destination port of the container. This tool also provides a graphic indicator for the shifting and unloading containers. Figures A.5 and A.6 show a snapshot of this tool.



```
5
1       1       1       1       1
2       1       5
1       1       2       1       1       1
2       1       2       1       1       1
3       1       2       1       1       1
4       1       2       1       1       1
5       1       3       1       1       1
6       1       3       1       1       1
7       1       3       1       1       1
8       1       3       1       1       1
9       1       4       1       1       1
10      1       4       1       1       1
11      2       3       1       1       1
```

```
J:\Port Operations\Examples\Avriel\Avriel 1993.txt loaded
Number of ports = 5
Number of containers = 19 , 0 40ft
Ship (X,Y,Z) = (2,1,5)
```

```
Cell to container assignment constraints generated :50 Constraints
No on top of empty constraints generated :40 Constraints
No unload before origin or after destination constraints generated :61 Constraints
Different size containers constraints generated :0 Constraints
Bow-stern stability constraints generated :10 Constraints
Left-Right stability constraints generated :0 Constraints
Heavy container under ligh container constraints generated :40 Constraints
Minimize total time at port constraints generated :10 Constraints
Binary variables generated :1055 Variables
1055 Variables, 325 Constraints
59 ms
```

```
Minimize
        Obj:
                T_1 + T_2 + T_3 + T_4 + T_5
Subject To
        X_1_1_1_1_1 +X_1_1_2_1_1 +X_1_1_3_1_1 +X_1_1_4_1_1 +X_1_1_5_1_1 +X_2_1_1_1_1 +X_
2_1_2_1_1 +X_2_1_3_1_1 +X_2_1_4_1_1 +X_2_1_5_1_1 =1
        X_1_1_1_2_1 +X_1_1_2_2_1 +X_1_1_3_2_1 +X_1_1_4_2_1 +X_1_1_5_2_1 +X_2_1_1_2_1 +X_
2_1_2_2_1 +X_2_1_3_2_1 +X_2_1_4_2_1 +X_2_1_5_2_1 =1
        X_1_1_1_3_1 +X_1_1_2_3_1 +X_1_1_3_3_1 +X_1_1_4_3_1 +X_1_1_5_3_1 +X_2_1_1_3_1 +X_
2_1_2_3_1 +X_2_1_3_3_1 +X_2_1_4_3_1 +X_2_1_5_3_1 =1
        X_1_1_1_4_1 +X_1_1_2_4_1 +X_1_1_3_4_1 +X_1_1_4_4_1 +X_1_1_5_4_1 +X_2_1_1_4_1 +X_
2_1_2_4_1 +X_2_1_3_4_1 +X_2_1_4_4_1 +X_2_1_5_4_1 =1
        X_1_1_1_5_1 +X_1_1_2_5_1 +X_1_1_3_5_1 +X_1_1_4_5_1 +X_1_1_5_5_1 +X_2_1_1_5_1 +X_
2 1 2 5 1 +X 2 1 3 5 1 +X 2 1 4 5 1 +X 2 1 5 5 1 =1
```

Constraints

Assignment and Unload
☑ Assign at origin
☑ Unassign before origin and after destination
☑ Cell To Container
☐ Container To Cell
☐ Container Stay Aboard
☑ Not on top of empty cell
☑ Unload at destination
☑ No Unloading Before O or after D

☐ Shift ☐ Location change ☑ Minimize Total Time at ports

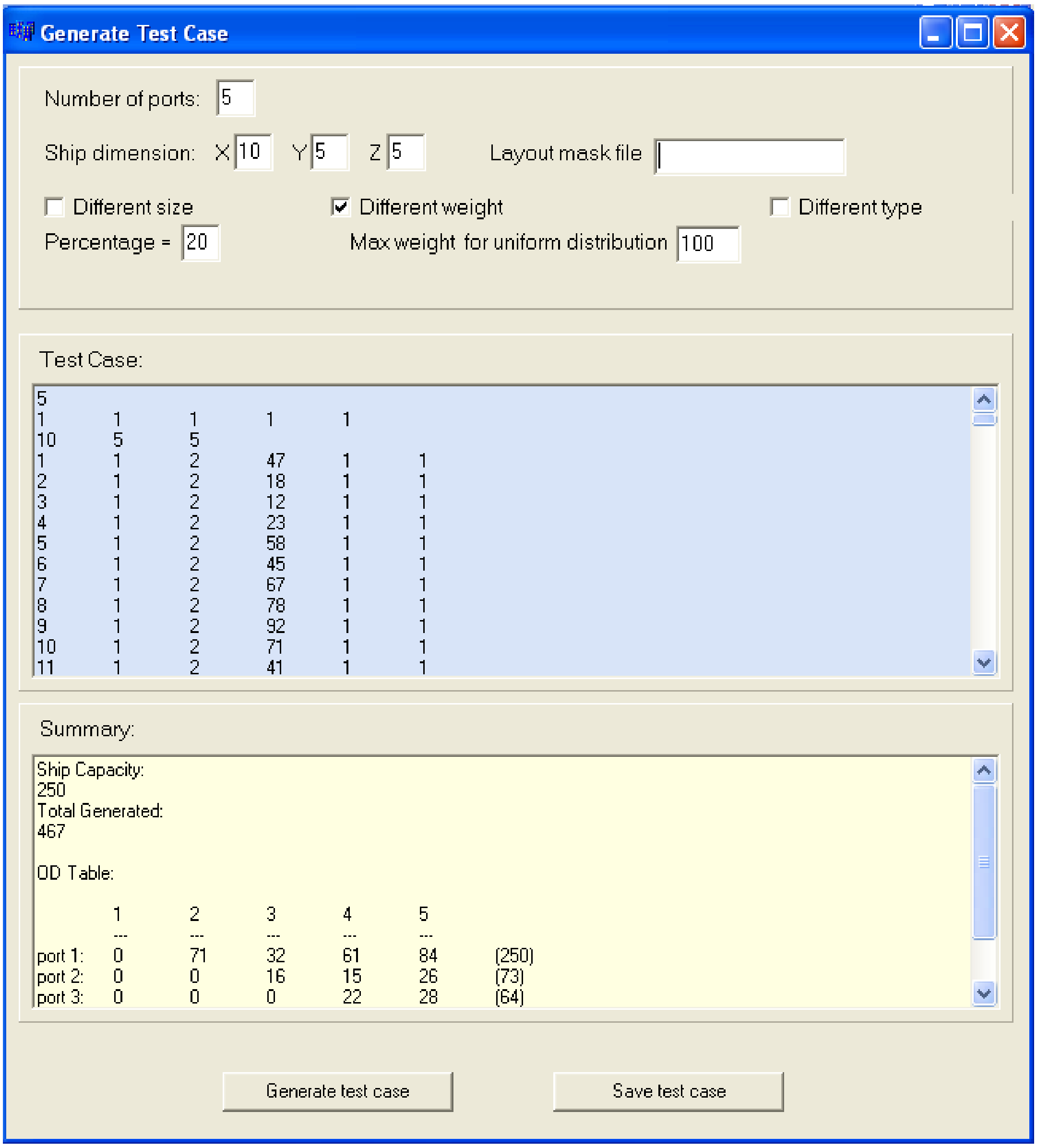
Different Size Containers
☑ Different Size Containers
☐ 20 ft Container can not be put on top of 40 ft
☐ 40 ft Container can not be put on top of 20 ft

Stability and Capacity
☑ Bow Stern Stability BSB = 1
☑ Left Right Stability LRB = 1
☑ Heavy container under light container
☐ Each Tier lighter than the tier underneath
☐ Ship Capacity Capacity = 5000

Crane workload
☐ Crane workload balance Workload Threshold = 1

Integer Variables
☑ Integer variables

Figure A.1. The user interface and formulation output for a sample problem



Number of ports: 5

Ship dimension: X 10 Y 5 Z 5

Layout mask file

☐ Different size

Percentage = 20

☑ Different weight

Max weight for uniform distribution 100

☐ Different type

Test Case:

```
5
1       1       1       1       1
10      5       5
1       1       2       47      1       1
2       1       2       18      1       1
3       1       2       12      1       1
4       1       2       23      1       1
5       1       2       58      1       1
6       1       2       45      1       1
7       1       2       67      1       1
8       1       2       78      1       1
9       1       2       92      1       1
10      1       2       71      1       1
11      1       2       41      1       1
```

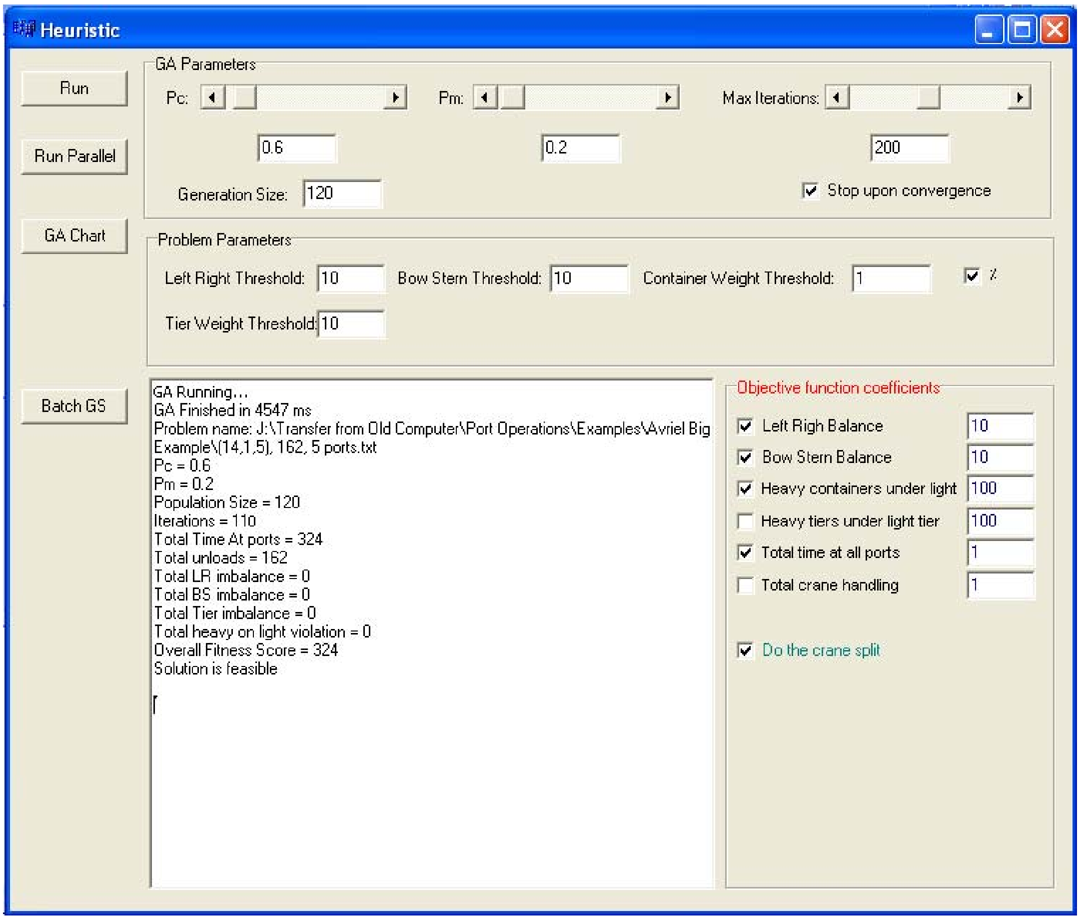
Summary:

```
Ship Capacity:
250
Total Generated:
467

OD Table:

        1       2       3       4       5
        ---     ---     ---     ---     ---
port 1: 0       71      32      61      84      (250)
port 2: 0       0       16      15      26      (73)
port 3: 0       0       0       22      28      (64)
```

Figure A.2. Generating a test case



Pc: 0.6

Pm: 0.2

Max Iterations: 200

Generation Size: 120

☑ Stop upon convergence

Left Right Threshold: 10

Bow Stern Threshold: 10

Container Weight Threshold: 1 ☑ %

Tier Weight Threshold: 10

```
GA Running...
GA Finished in 4547 ms
Problem name: J:\Transfer from Old Computer\Port Operations\Examples\Avriel Big
Example\(14,1,5), 162, 5 ports.txt
Pc = 0.6
Pm = 0.2
Population Size = 120
Iterations = 110
Total Time At ports = 324
Total unloads = 162
Total LR imbalance = 0
Total BS imbalance = 0
Total Tier imbalance = 0
Total heavy on light violation = 0
Overall Fitness Score = 324
Solution is feasible
```

☑ Left Righ Balance 10

☑ Bow Stern Balance 10

☑ Heavy containers under light 100

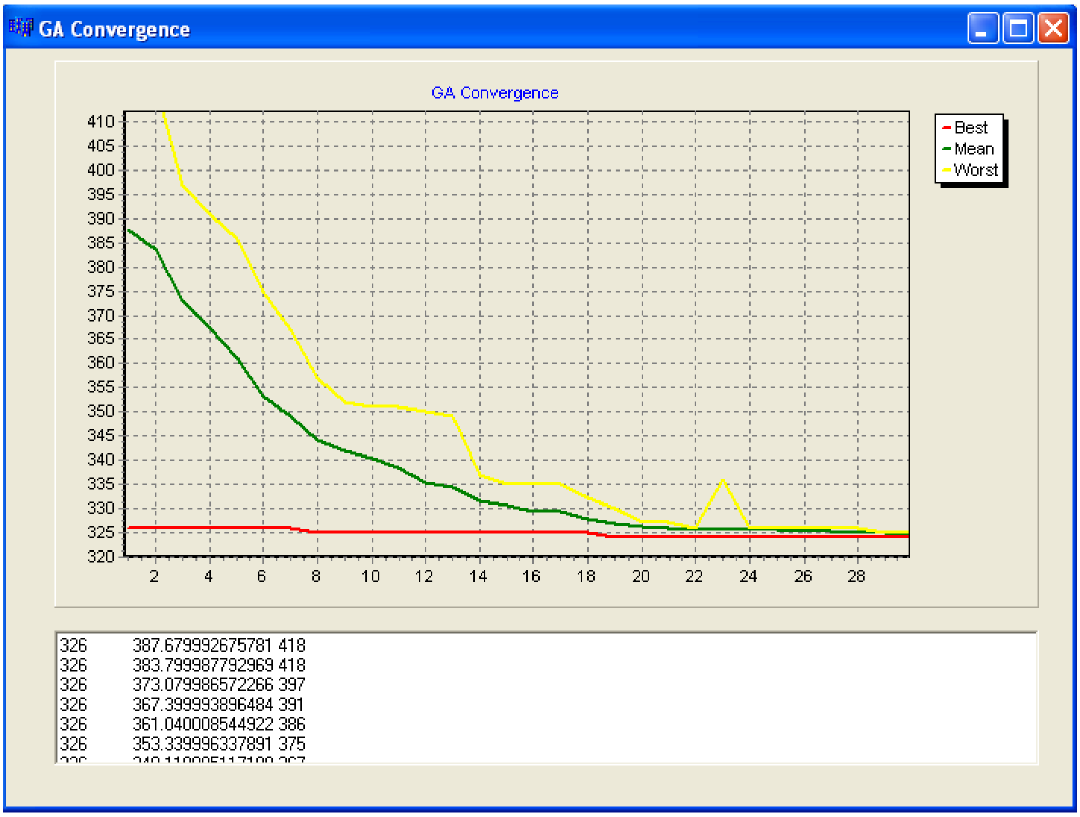
☐ Heavy tiers under light tier 100

☑ Total time at all ports 1

☐ Total crane handling 1

☑ Do the crane split

Figure A.3. Solver interface



```
326       387.679992675781 418
326       383.799987792969 418
326       373.079986572266 397
326       367.399993896484 391
326       361.040008544922 386
326       353.339996337891 375
```

Figure A.4. Convergence diagram for a sample problem

Display this column: X -1 Y -1

☑ Container Labels ☐ Cranes Info

			#273	#180	#281	#251	#126	#118	#238		#303	#453	#409	#283	#375	#380	#173	#407	#322	#520	#383	#208
		#109	#85	#259	#6	#128	#173	#295	#39		#402	#383	#112	#166	#187	#151	#152	#132	#354	#474	#472	#477
		#287	#9	#88	#176	#235	#224	#36	#117		#104	#259	#364	#136	#129	#106	#335	#314	#316	#561	#456	#581
		#11	#121	#183	#279	#51	#69	#133	#223		#226	#208	#301	#285	#254	#224	#306	#369	#177		#445	#562

			#226	#184	#301	#312	#127	#165	#214		#287	#456	#169	#167	#393	#381	#438	#195	#359	#379	#495	#210
		#116	#90	#282	#24	#99	#181	#240	#82		#403	#426	#111	#163	#424	#427	#193	#329	#108	#478	#437	#184
		#303	#96	#136	#187	#307	#221	#22	#160		#138	#275	#121	#253	#347	#336	#326	#256	#258	#570	#559	#293
		#13	#95	#283	#246	#80	#77	#141	#280		#273	#206	#281	#446	#454	#248	#219	#153	#368	#242	#292	#455

		#16	#276	#194	#265	#253	#131	#191	#228		#452	#457	#370	#418	#411	#442	#440	#204	#98	#178	#459	#492
		#138	#63	#48	#25	#106	#200	#266	#83		#412	#340	#346	#183	#176	#428	#415	#118	#270	#479	#436	#194
		#27	#100	#166	#147	#252	#215	#32	#189		#320	#293	#100	#312	#124	#99	#131	#264	#189	#576	#453	#583
		#5	#137	#244	#212	#52	#78	#123	#236		#231	#429	#297	#447	#315	#255	#266	#371	#185	#261	#294	#565

		#46	#249	#230	#310	#41	#134	#149	#264		#156	#431	#210	#186	#171	#366	#192	#410	#123	#196	#536	#370
		#156	#42	#20	#8	#97	#300	#219	#28		#420	#105	#348	#404	#159	#405	#200	#110	#313	#482	#431	#433
		#31	#142	#211	#170	#309	#311	#61	#146		#323	#233	#96	#251	#309	#360	#127	#444	#373	#577	#563	#586
		#23	#164	#272	#250	#53	#1	#150	#296		#257	#164	#277	#302	#239	#247	#443	#158	#197	#274	#298	#566

		#74	#289	#268	#317	#4	#103	#154	#256		#196	#436	#119	#388	#441	#384	#421	#425	#141	#156	#275	#169
		#175	#18	#57	#87	#125	#227	#306	#7		#175	#130	#351	#406	#356	#128	#181	#115	#296	#485	#480	#164
		#40	#111	#186	#171	#262	#263	#70	#197		#324	#291	#120	#234	#252	#267	#449	#228	#160	#175	#276	#535
		#26	#206	#213	#284	#67	#66	#157	#313		#298	#433	#225	#284	#262	#271	#149	#157	#146	#452	#257	#568

		#84	#290	#232	#37	#12	#188	#168	#314		#178	#437	#330	#390	#170	#399	#174	#168	#133	#397	#259	#502
		#178	#72	#30	#47	#203	#269	#288	#33		#341	#145	#352	#202	#349	#125	#435	#238	#236	#489	#417	#429
		#44	#112	#167	#172	#239	#245	#62	#185		#328	#232	#325	#243	#307	#245	#450	#214	#209	#420	#249	#522
		#105	#208	#217	#302	#15	#81	#158	#270		#294	#194	#241	#250	#376	#269	#191	#430	#117	#287	#231	#232

		#91	#292	#291	#38	#89	#174	#204	#229		#379	#408	#102	#391	#416	#385	#382	#154	#101	#395	#549	#409
		#196	#34	#3	#143	#151	#271	#216	#73		#353	#362	#142	#378	#361	#97	#126	#260	#280	#491	#484	#206
		#10	#114	#179	#159	#315	#267	#101	#177		#331	#455	#327	#222	#235	#263	#240	#148	#155	#412	#289	#470
		#140	#210	#241	#285	#71	#135	#153	#258		#292	#184	#217	#212	#432	#227	#165	#400	#139	#303	#273	#572

Figure A.5. Visual tier by tier presentation of the solution for a sample problem

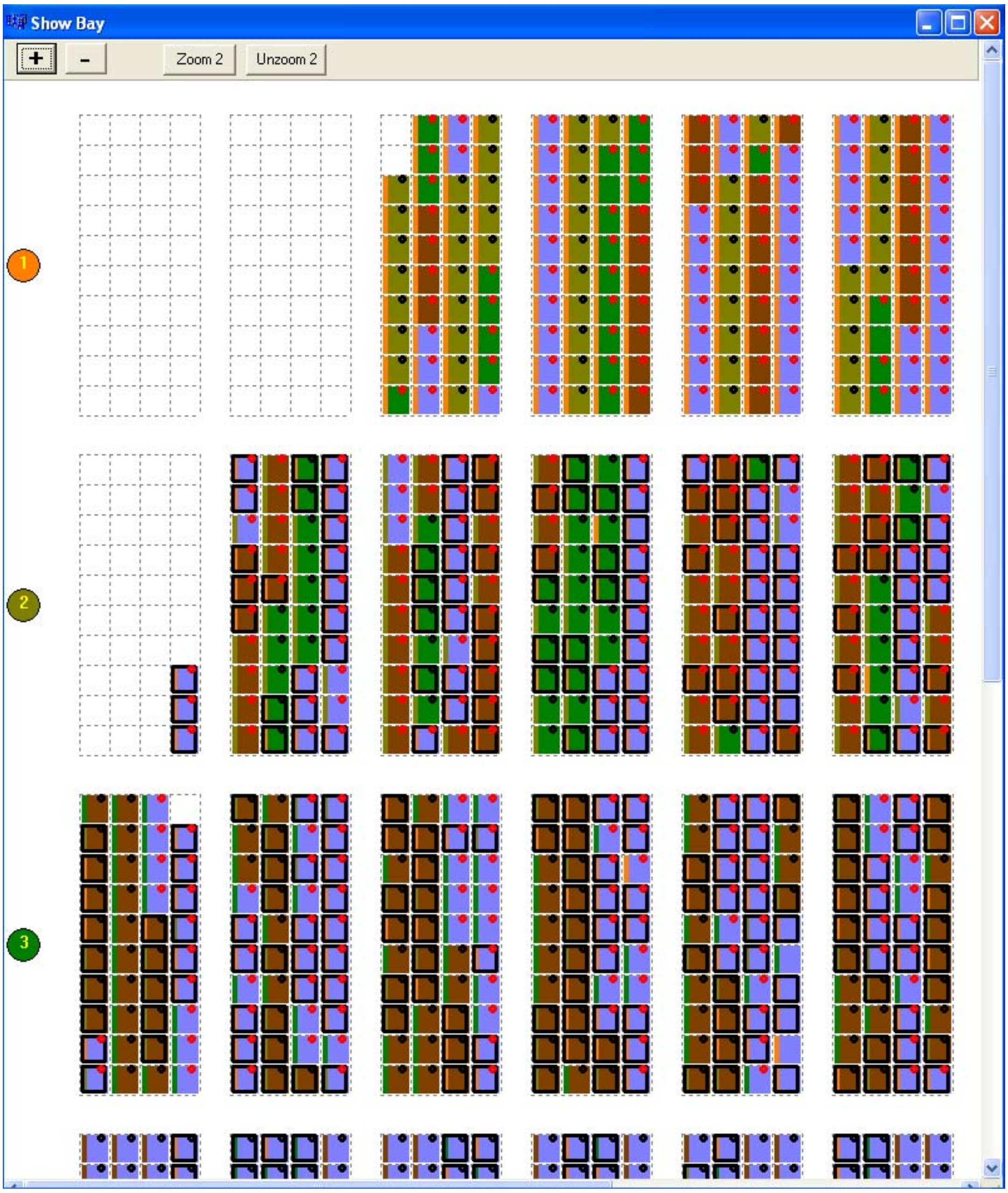


Figure A.6. Visual bay by bay presentation of the solution for a sample problem

Appendix B: Solution details for a sample problem

The input and output details of the scenario 3 of the sample problem solved in section 4.3.1 are as following:

Program input:

Container number	Origin port	Destination port	Weight	Size (number TEU)	Type
1	1	2	1	1	1
2	1	2	1	1	1
3	1	2	1	1	1
4	1	2	1	1	1
5	1	3	1	1	1
6	1	3	1	1	1
7	1	3	1	1	1
8	1	3	1	1	1
9	1	4	1	1	1
10	1	4	1	1	1
11	2	3	4	1	1
12	2	3	4	1	1
13	2	5	4	1	1
14	3	5	1	1	1
15	3	5	1	1	1
16	3	5	1	1	1
17	3	5	1	1	1
18	3	5	1	1	1
19	4	5	1	1	1
20	1	2	1	1	1
21	1	2	1	1	1
22	1	2	1	1	1
23	1	2	1	1	1
24	1	3	1	1	1
25	1	3	1	1	1
26	1	3	1	1	1
27	1	3	1	1	1
28	1	4	1	1	1
29	1	4	1	1	1
30	2	3	4	1	1
31	2	3	4	1	1
32	2	5	4	1	1
33	3	5	1	1	1
34	3	5	1	1	1
35	3	5	1	1	1
36	3	5	1	1	1
37	3	5	1	1	1
38	4	5	1	1	1

Program output:

- Each row starts with the container identification number followed by #
- [origin port, destination port]
- L=Load, U=Unload, S=Shift, number after L,U,S indicates the crane number
- P=Port, followed by the port number

```
#1 [1,2] (4,1,4)    L2      P1      U2      P2
#2 [1,2] (2,1,4)    L1      P1      U1      P2
#3 [1,2] (2,1,2)    L1      P1      U1      P2
#4 [1,2] (3,1,5)    L2      P1      U2      P2
#5 [1,3] (3,1,3)    L2      P1
#5 [1,3] (3,1,3)    U2      P3
#6 [1,3] (1,1,3)    L1      P1
#6 [1,3] (1,1,3)    U1      P3
#7 [1,3] (1,1,5)    L1      P1
#7 [1,3] (1,1,5)    U1      P3
#8 [1,3] (3,1,2)    L2      P1
#8 [1,3] (3,1,2)    U2      P3
#9 [1,4] (2,1,1)    L1      P1
#9 [1,4] (2,1,1)
#9 [1,4] (2,1,1)    U1      P4
#10 [1,4] (3,1,1)   L2      P1
#10 [1,4] (3,1,1)   S1      P3
#10 [1,4] (2,1,3)   U1      P4
#11 [2,3] (3,1,5)   L2      P2      U2      P3
#12 [2,3] (2,1,3)   L1      P2      U1      P3
```

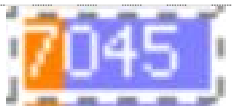
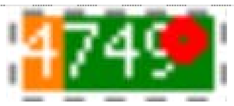
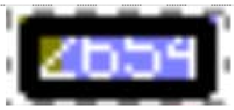
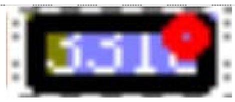
#13 [2,5] (2,1,2)	L1	P2		
#13 [2,5] (2,1,2)	S1	P4		
#13 [2,5] (4,1,2)	U2	P5		
#14 [3,5] (3,1,3)	L2	P3		
#14 [3,5] (3,1,3)	U2	P5		
#15 [3,5] (3,1,4)	L2	P3		
#15 [3,5] (3,1,4)	U2	P5		
#16 [3,5] (3,1,1)	L2	P3		
#16 [3,5] (3,1,1)	U2	P5		
#17 [3,5] (3,1,2)	L2	P3		
#17 [3,5] (3,1,2)	U2	P5		
#18 [3,5] (1,1,5)	L1	P3		
#18 [3,5] (1,1,5)	U1	P5		
#19 [4,5] (2,1,2)	L1	P4	U1	P5
#20 [1,2] (4,1,5)	L2	P1	U2	P2
#21 [1,2] (4,1,3)	L2	P1	U2	P2
#22 [1,2] (2,1,3)	L1	P1	U1	P2
#23 [1,2] (2,1,5)	L1	P1	U1	P2
#24 [1,3] (1,1,2)	L1	P1		
#24 [1,3] (1,1,2)	U1	P3		
#25 [1,3] (1,1,4)	L1	P1		
#25 [1,3] (1,1,4)	U1	P3		
#26 [1,3] (3,1,4)	L2	P1		
#26 [1,3] (3,1,4)	U2	P3		
#27 [1,3] (1,1,1)	L1	P1		
#27 [1,3] (1,1,1)	U1	P3		
#28 [1,4] (4,1,1)	L2	P1		

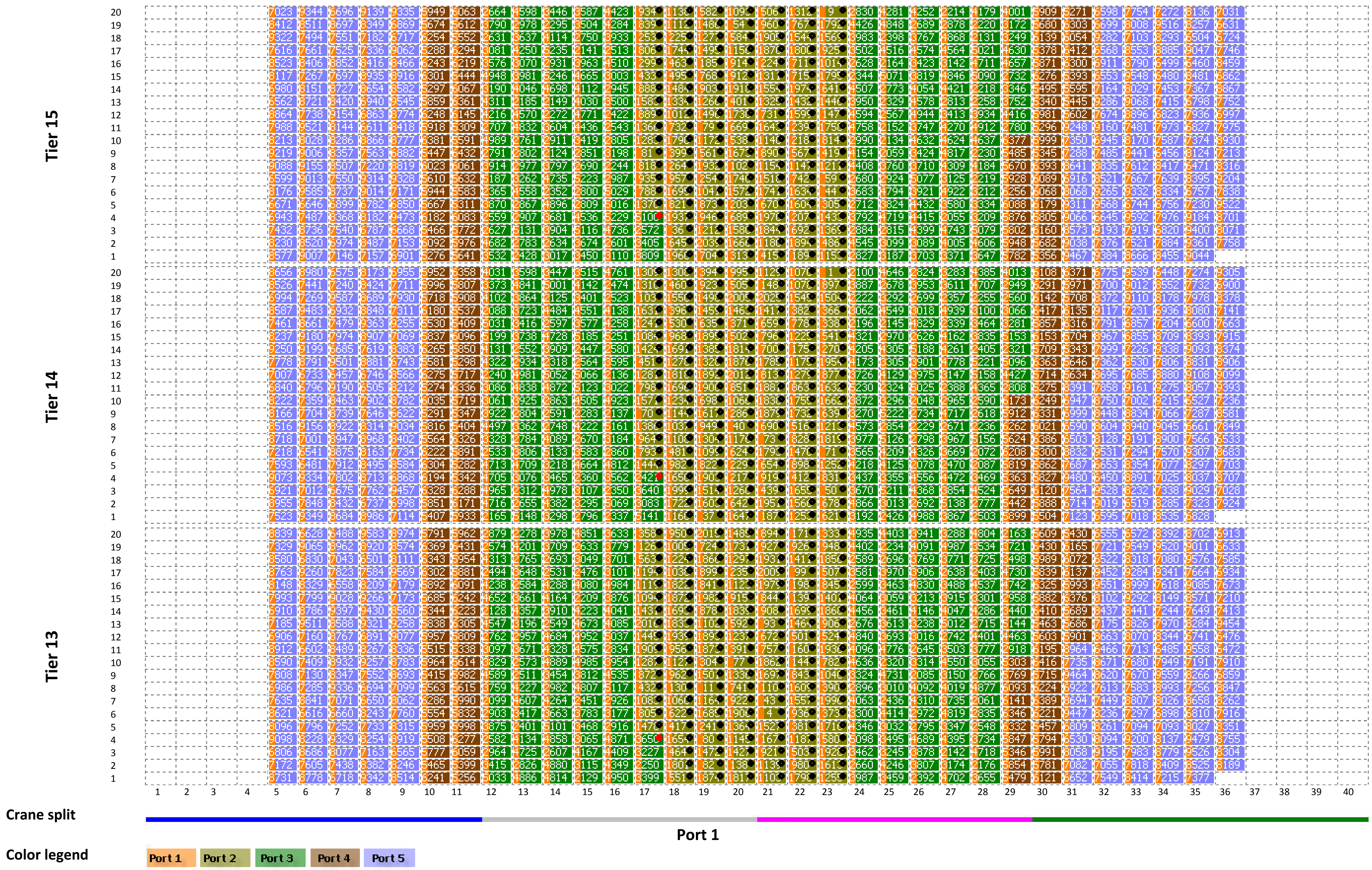
#28 [1,4] (4,1,1)				
#28 [1,4] (4,1,1)	U2	P4		
#29 [1,4] (4,1,2)	L2	P1		
#29 [1,4] (4,1,2)				
#29 [1,4] (4,1,2)	U2	P4		
#30 [2,3] (4,1,4)	L2	P2	U2	P3
#31 [2,3] (2,1,4)	L1	P2	U1	P3
#32 [2,5] (4,1,3)	L2	P2		
#32 [2,5] (4,1,3)	S1	P4		
#32 [2,5] (2,1,1)	U1	P5		
#33 [3,5] (1,1,4)	L1	P3		
#33 [3,5] (1,1,4)	U1	P5		
#34 [3,5] (1,1,2)	L1	P3		
#34 [3,5] (1,1,2)	U1	P5		
#35 [3,5] (1,1,3)	L1	P3		
#35 [3,5] (1,1,3)	U1	P5		
#36 [3,5] (1,1,1)	L1	P3		
#36 [3,5] (1,1,1)	U1	P5		
#37 [3,5] (3,1,5)	L2	P3		
#37 [3,5] (3,1,5)	U2	P5		
#38 [4,5] (4,1,1)	L2	P4	U2	P5

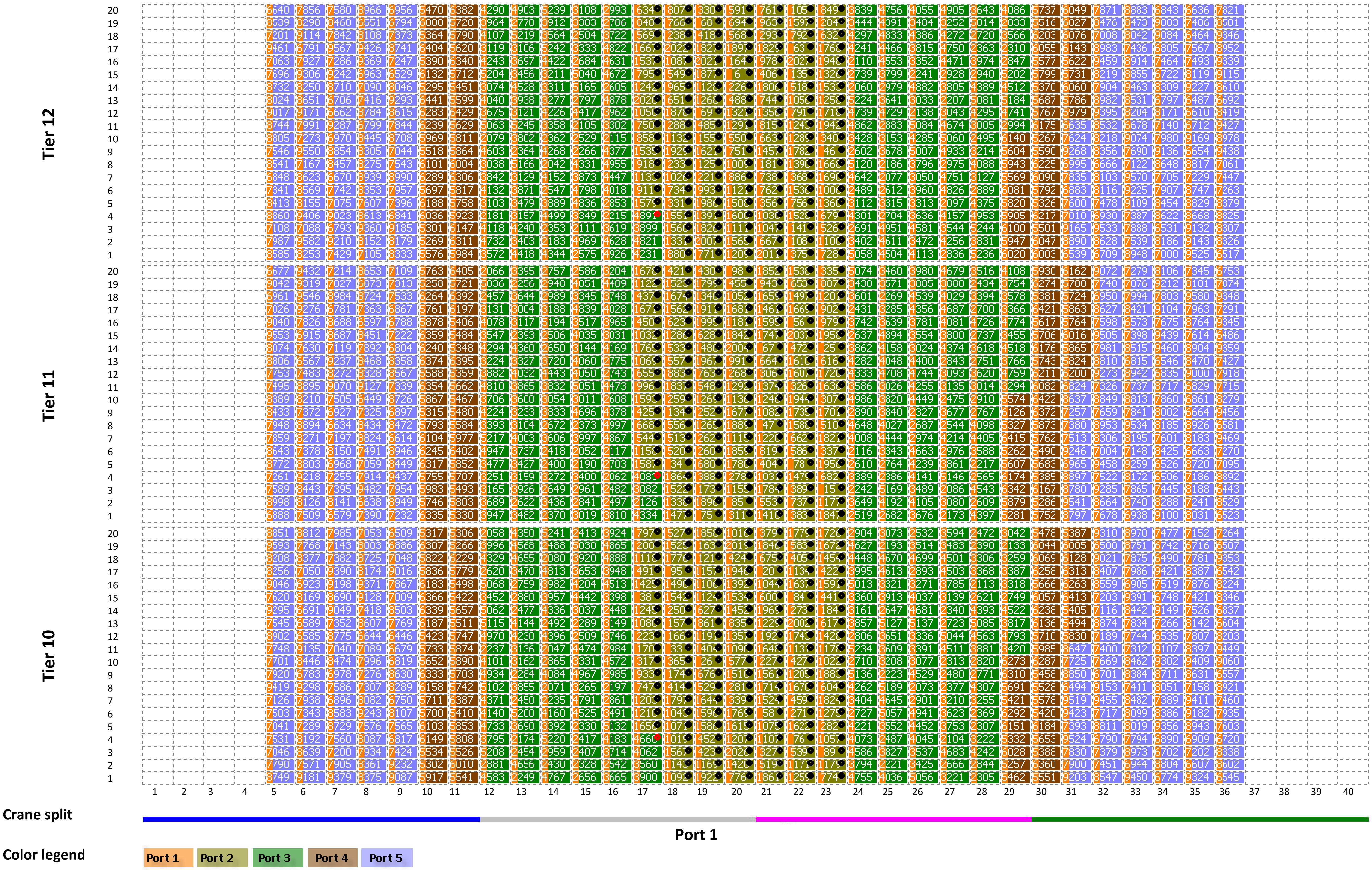
Appendix C: Visual solution details for a mega containership

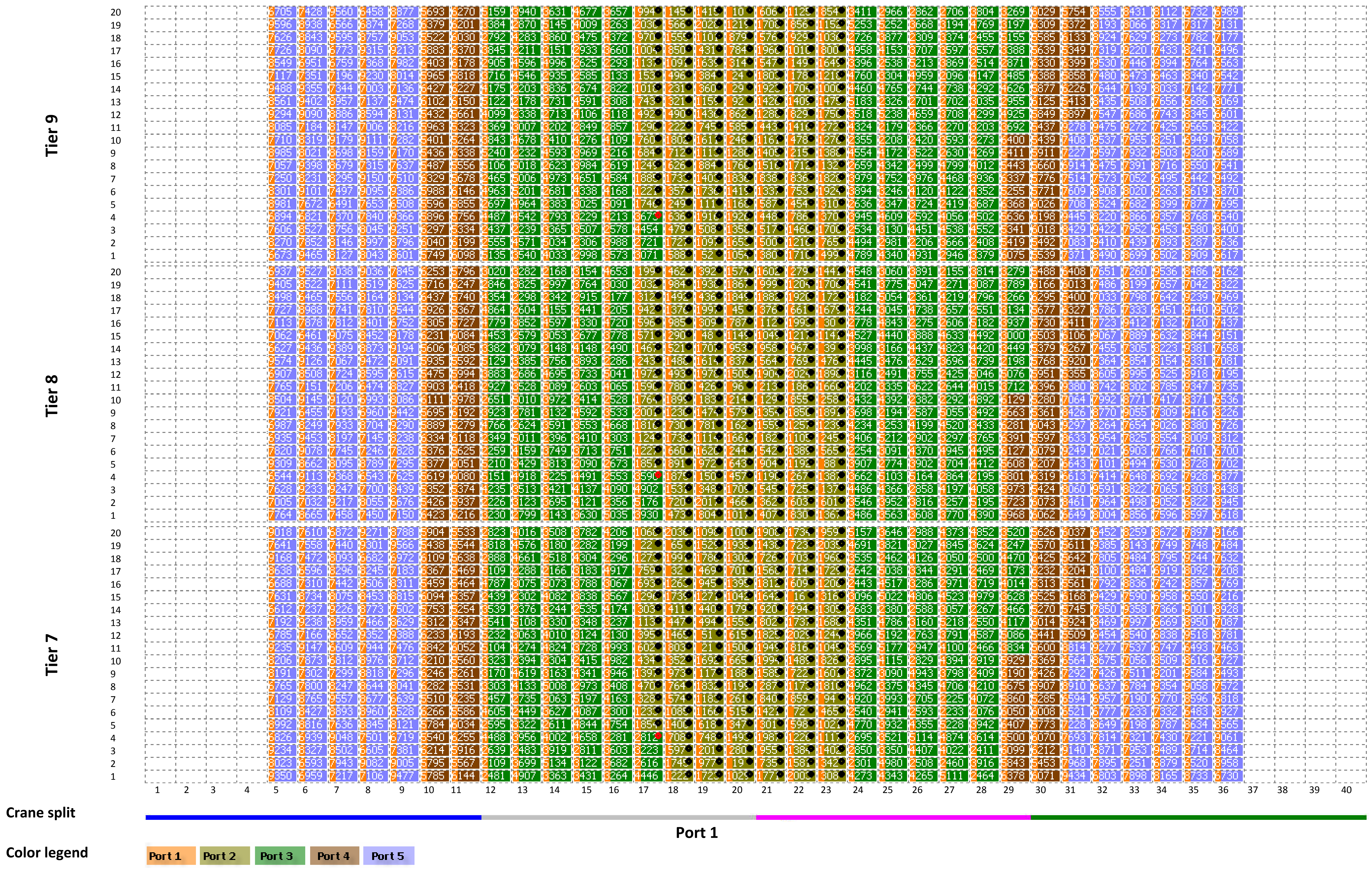
This section provides a visual representation of the container stowage planning and crane split solved for the mega containership example is section 7.3. A tier-by-tier view of the vessel's layout is used to show the results. Horizontal members from 1-40 and vertical numbers from 1-20 show the bays and row indexes respectively. Each box represents a container which consists of a narrow strip to the left and a wider strip to the right. The narrow strip has the color of the origin port and the wide strip matches the color of the destination port. The container identification number appears on the box. The colors assigned to each port are displayed in the legend section. Results are only displayed for ports one to four since the containership is empty at the last port. The crane split results in terms of the range of bays that each crane operates on is shown at the bottom of each layout. Table C.1 is a guide for interpretation of the graphic layout display.

Table C.1: Description of graphic symbols for stowage and crane split display

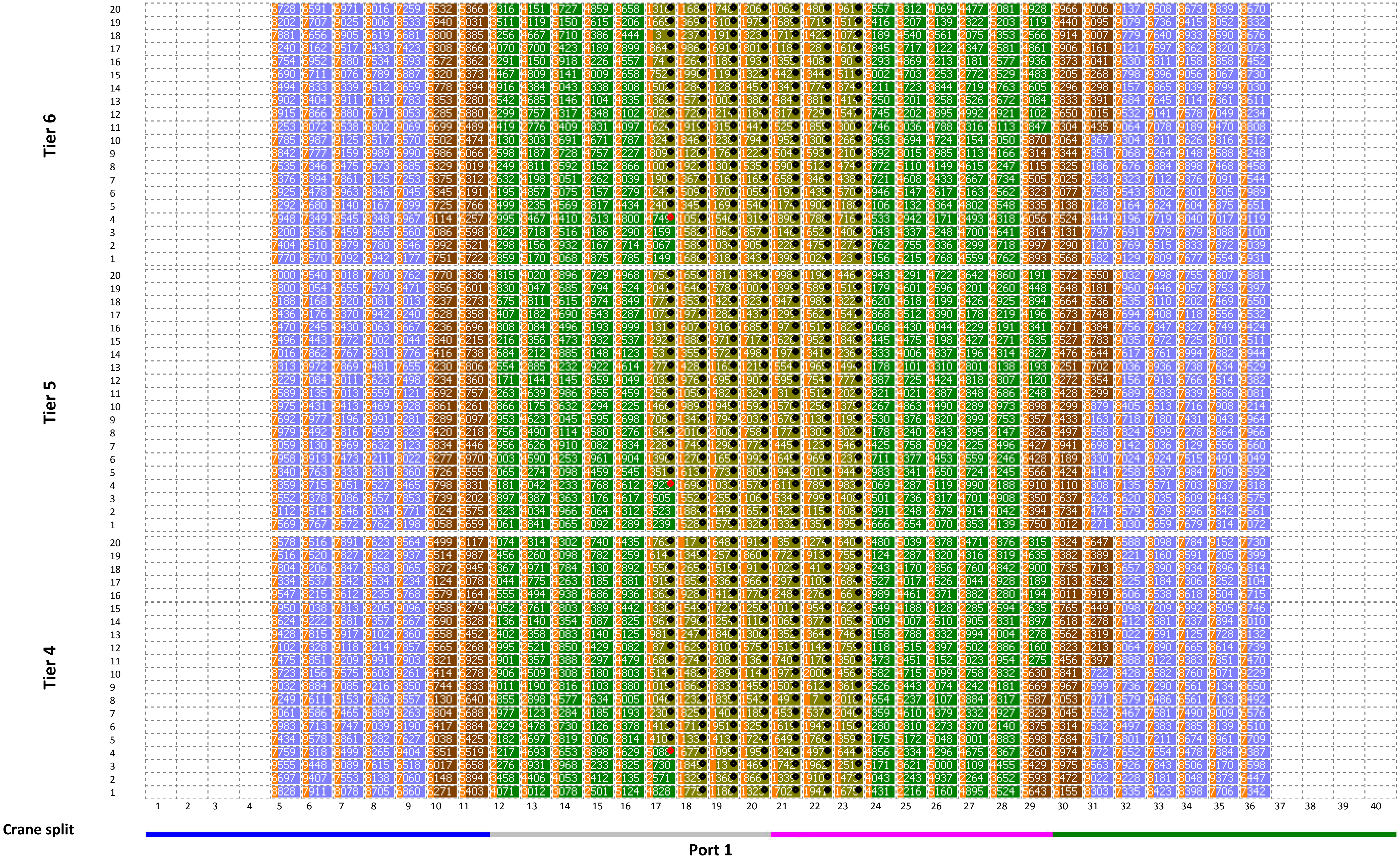
Symbol	Meaning	Sample
White box	The cell is empty	
Colored box without frame and with no dots	The container stays at the corresponding cell through next port	
Black dot on the top right corner	The container will be unloaded at the next port	
Red dot on the top right corner	The container will be shifted at the next port	
Solid black frame around the box	The container is shifted at the current port	
Solid black frame around the box and a black dot at the top right corner	The container has been shifted at the current port and will be unloaded at the next port	
Solid black frame around the box and a red dot at the top right corner	The container has been shifted at the current port and will be undergo another shift at next port	


Tier 15
Tier 14
Tier 13
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Crane split
Port 1
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5

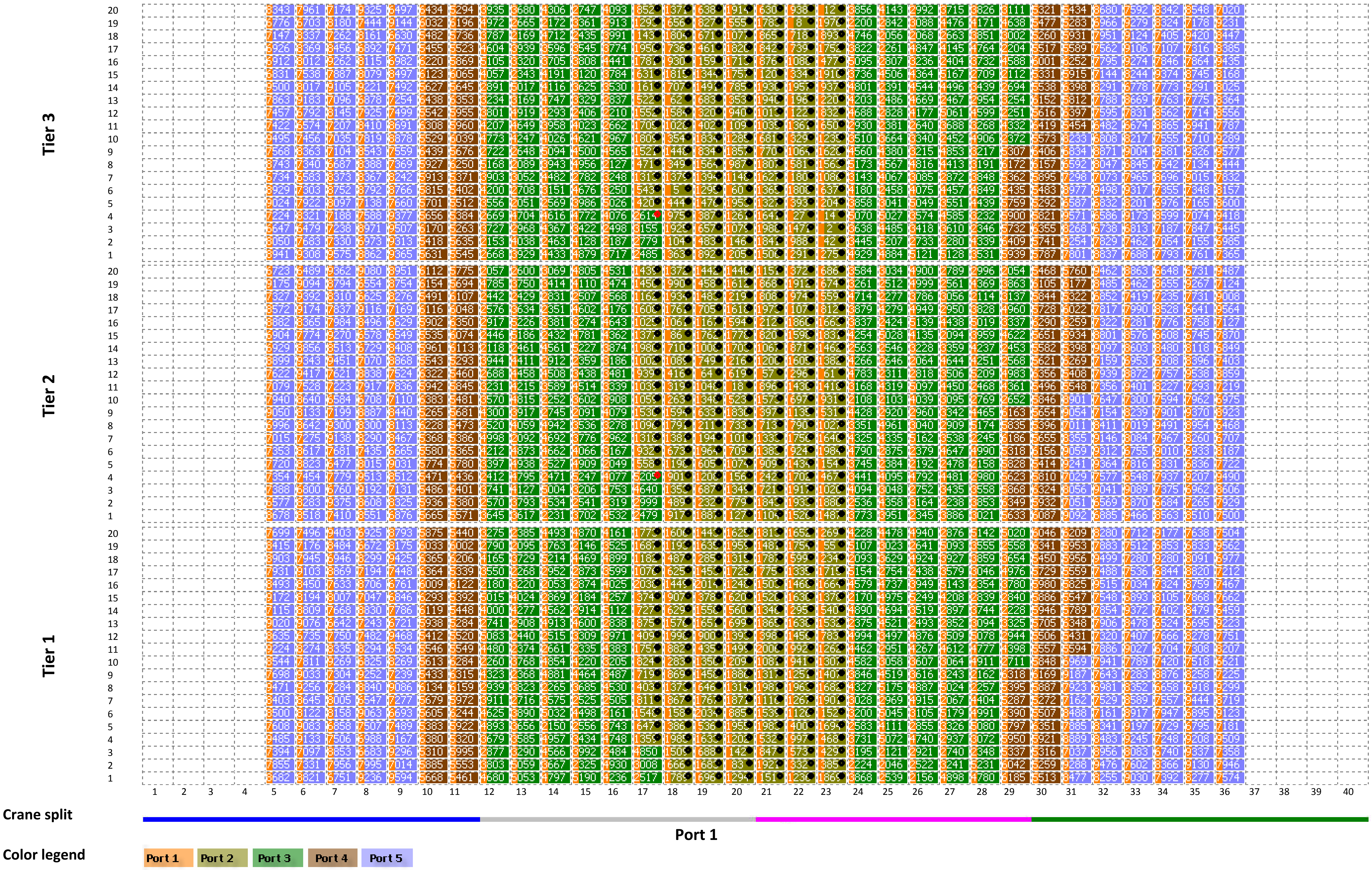

Tier 12
Tier 11
Tier 10
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Crane split
Port 1
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5

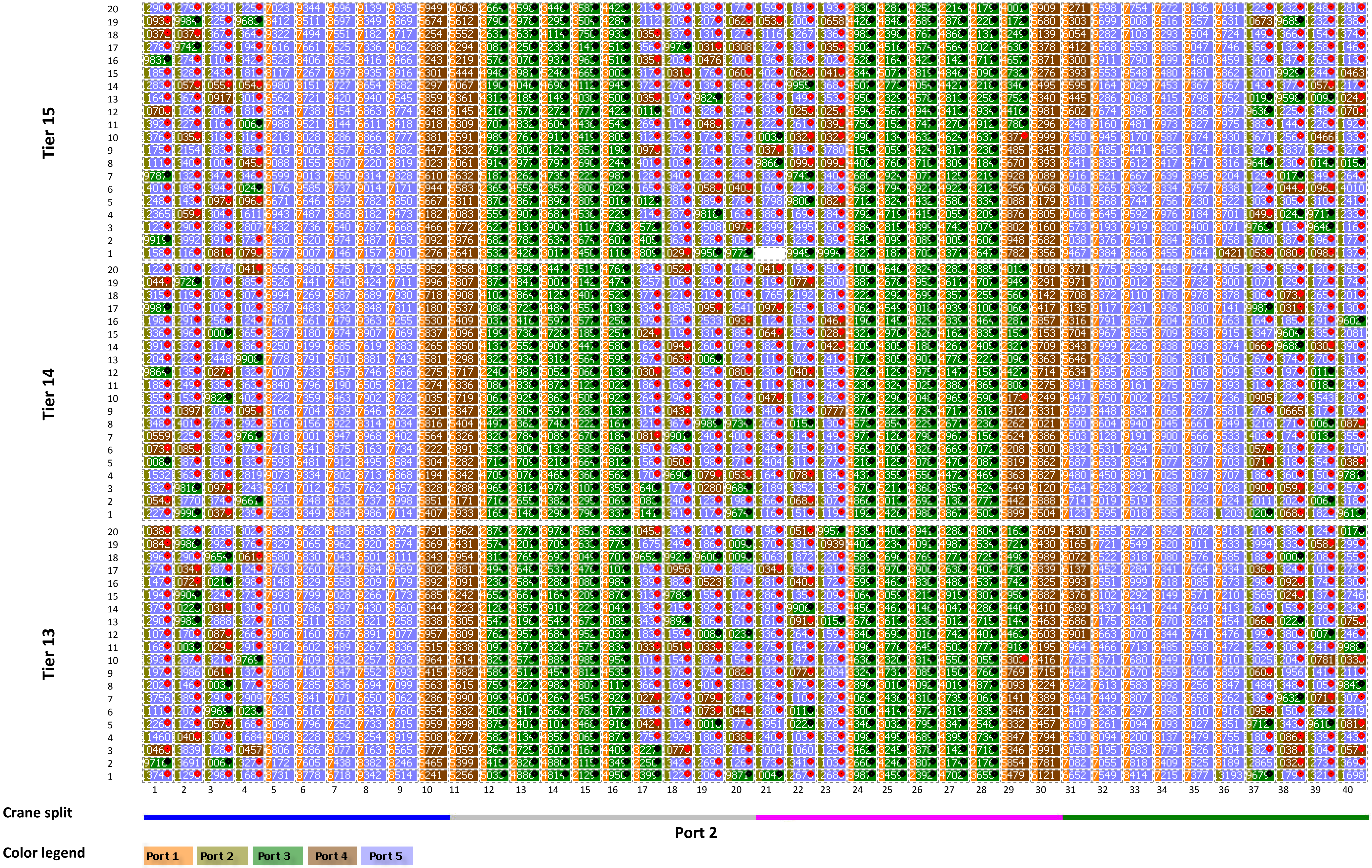


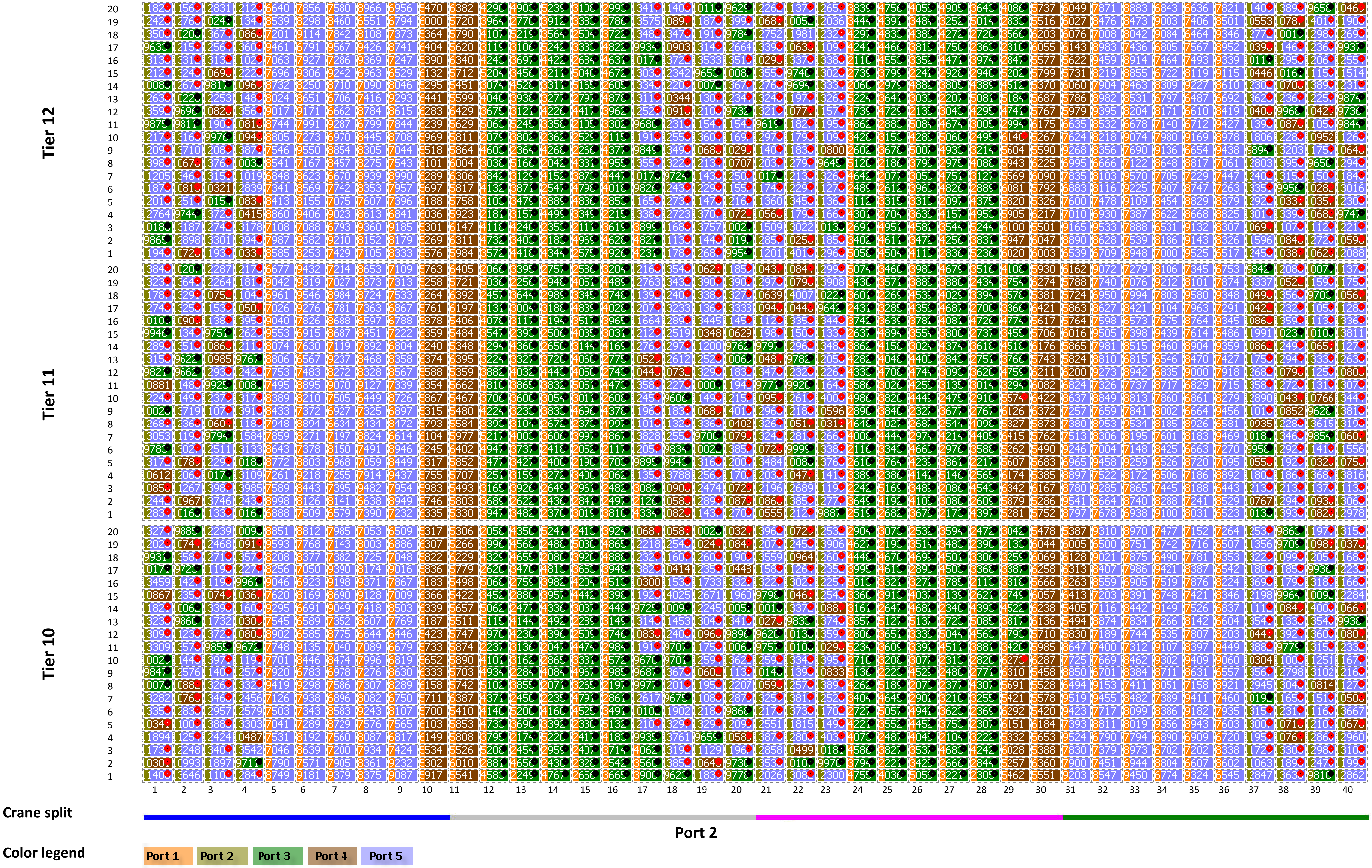
Tier 9
Tier 8
Tier 7
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Crane split
Port 1
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5

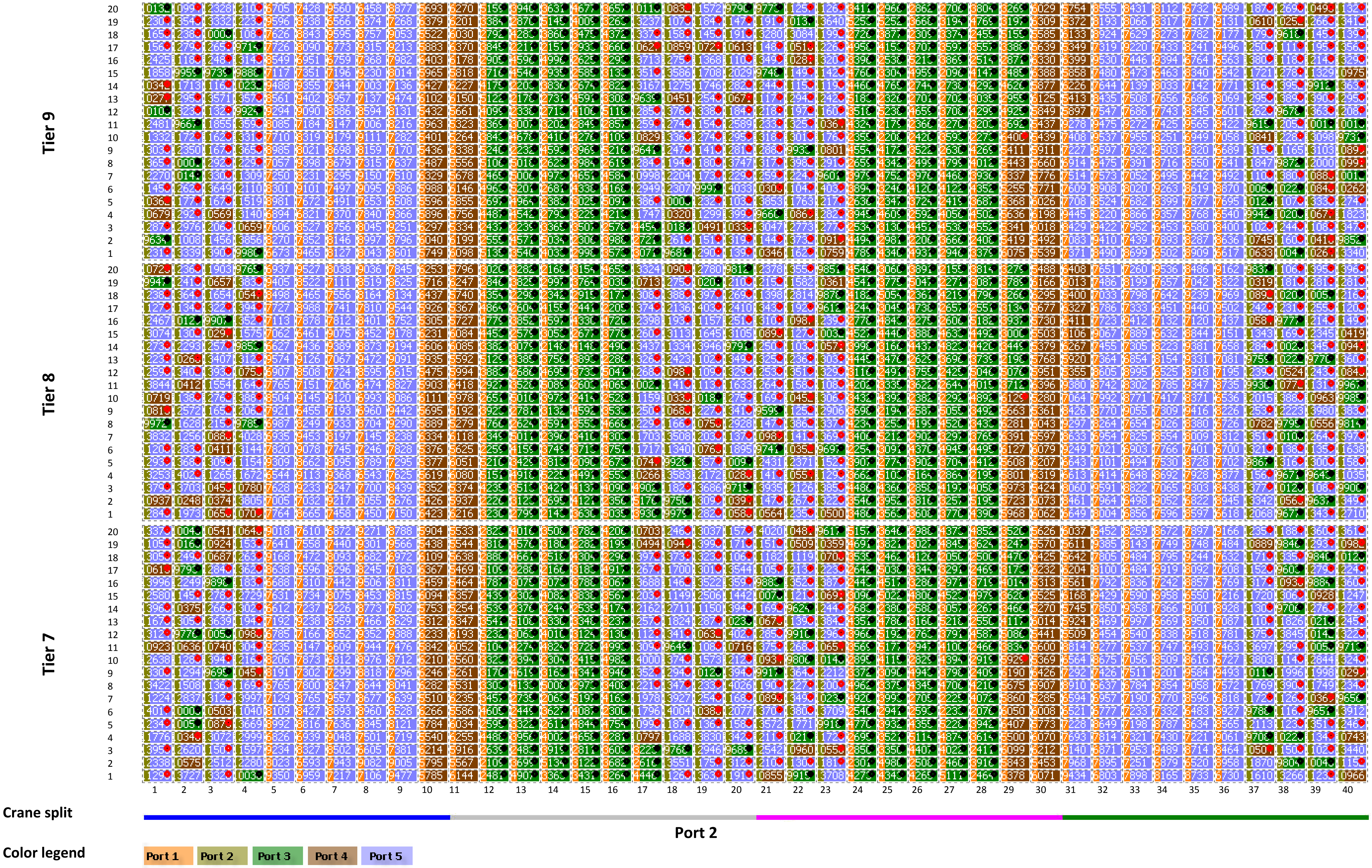


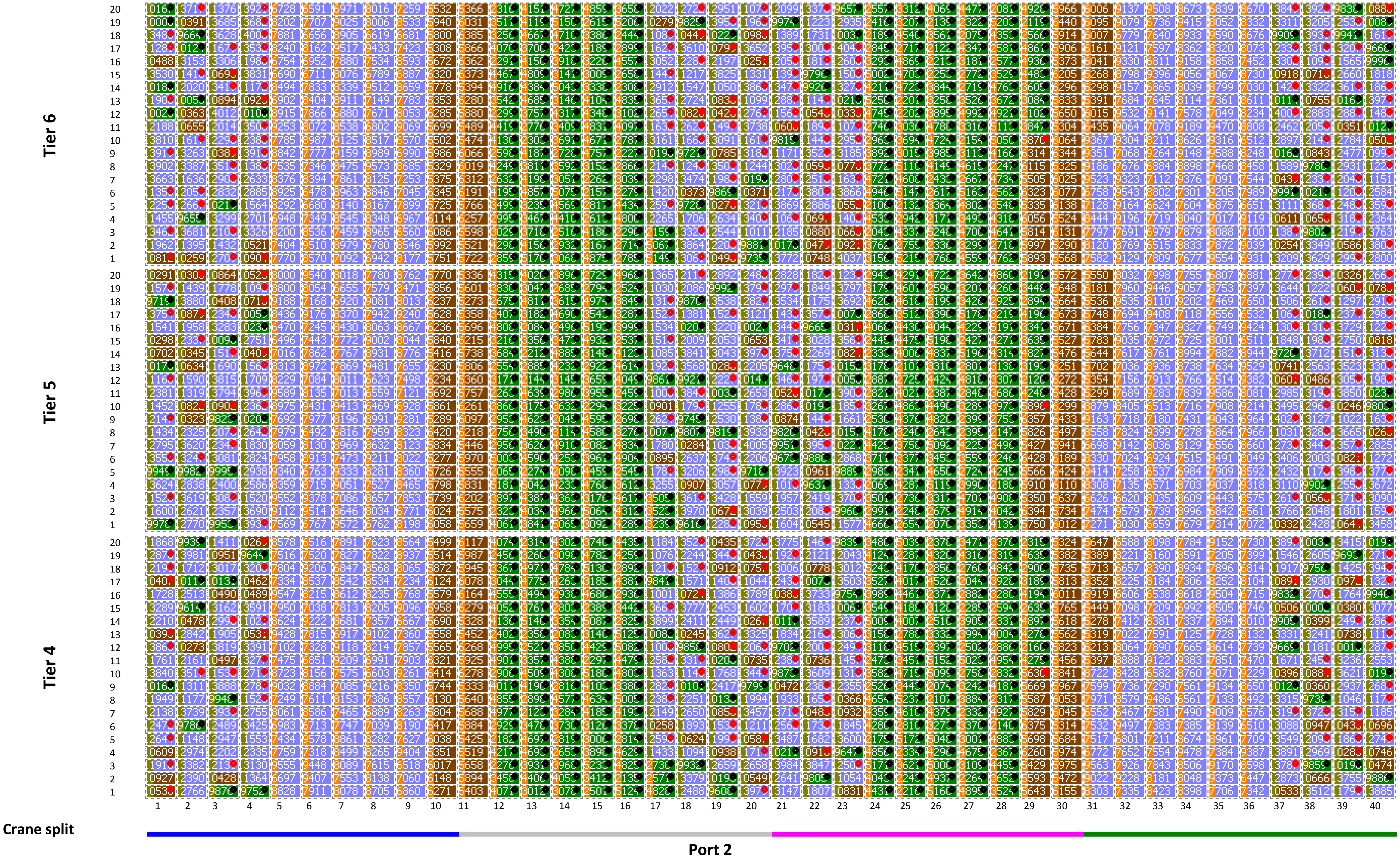
Color legend Port 1 Port 2 Port 3 Port 4 Port 5

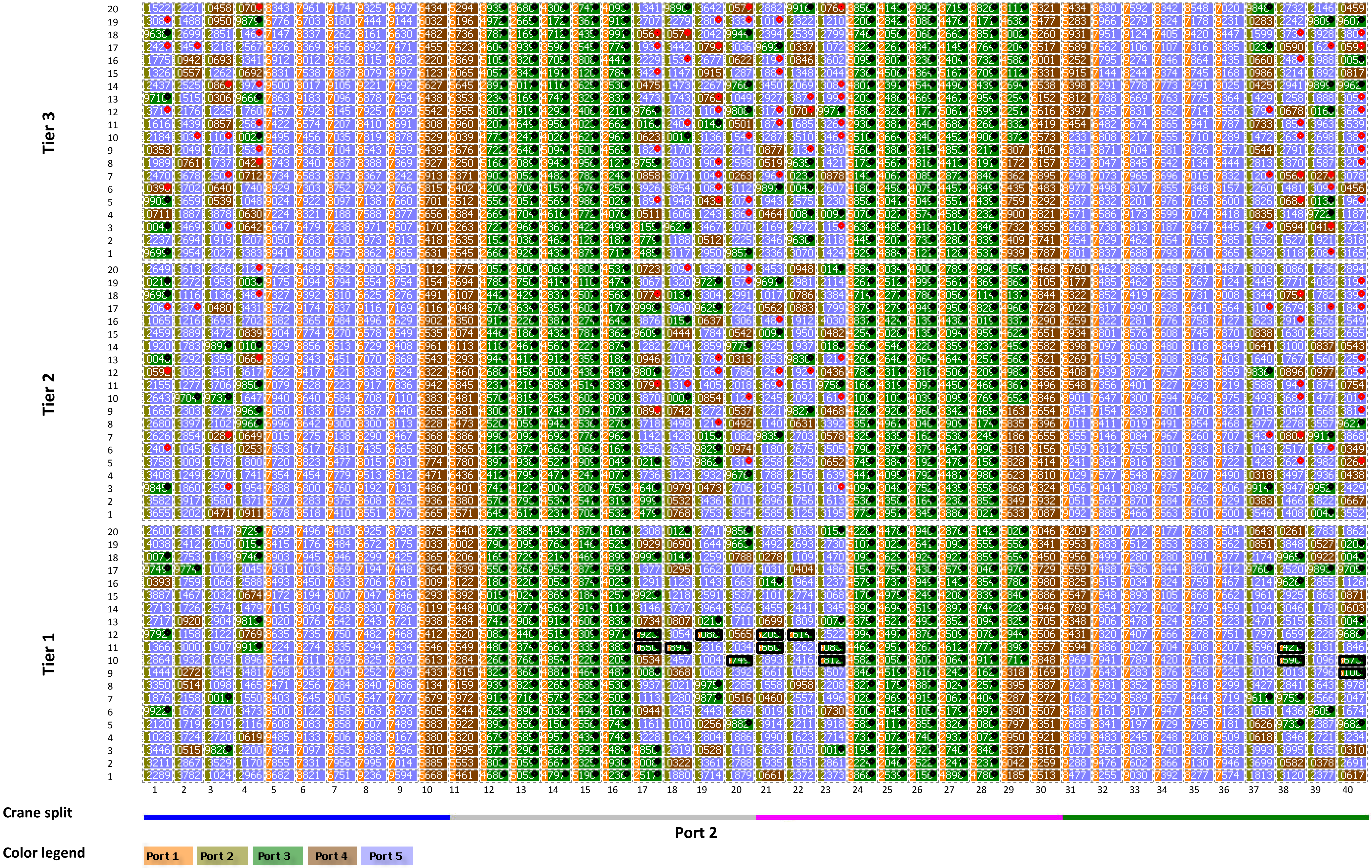

Tier 3
Tier 2
Tier 1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
Crane split
Port 1
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5

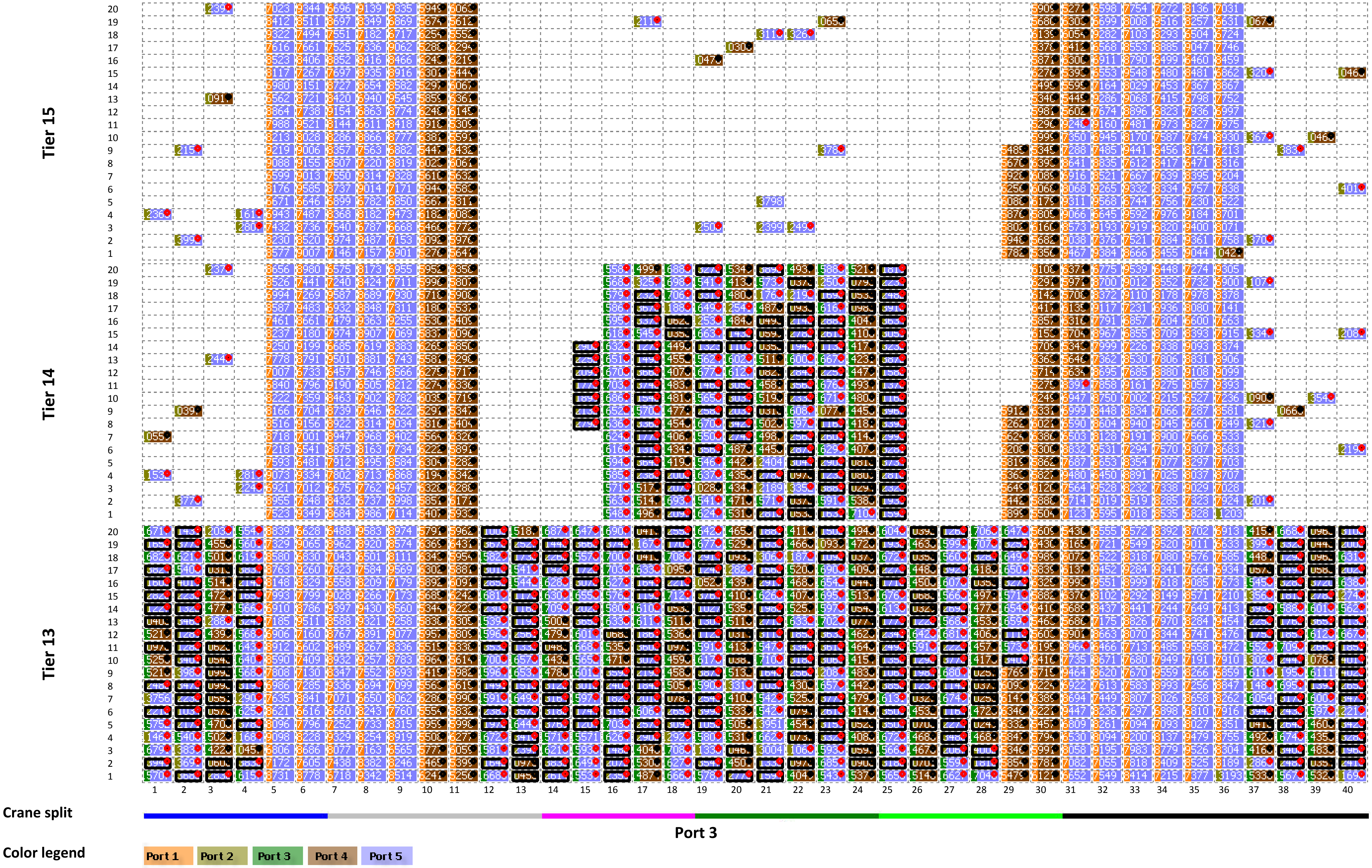

Tier 15
Tier 14
Tier 13
Crane split
Port 2
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5

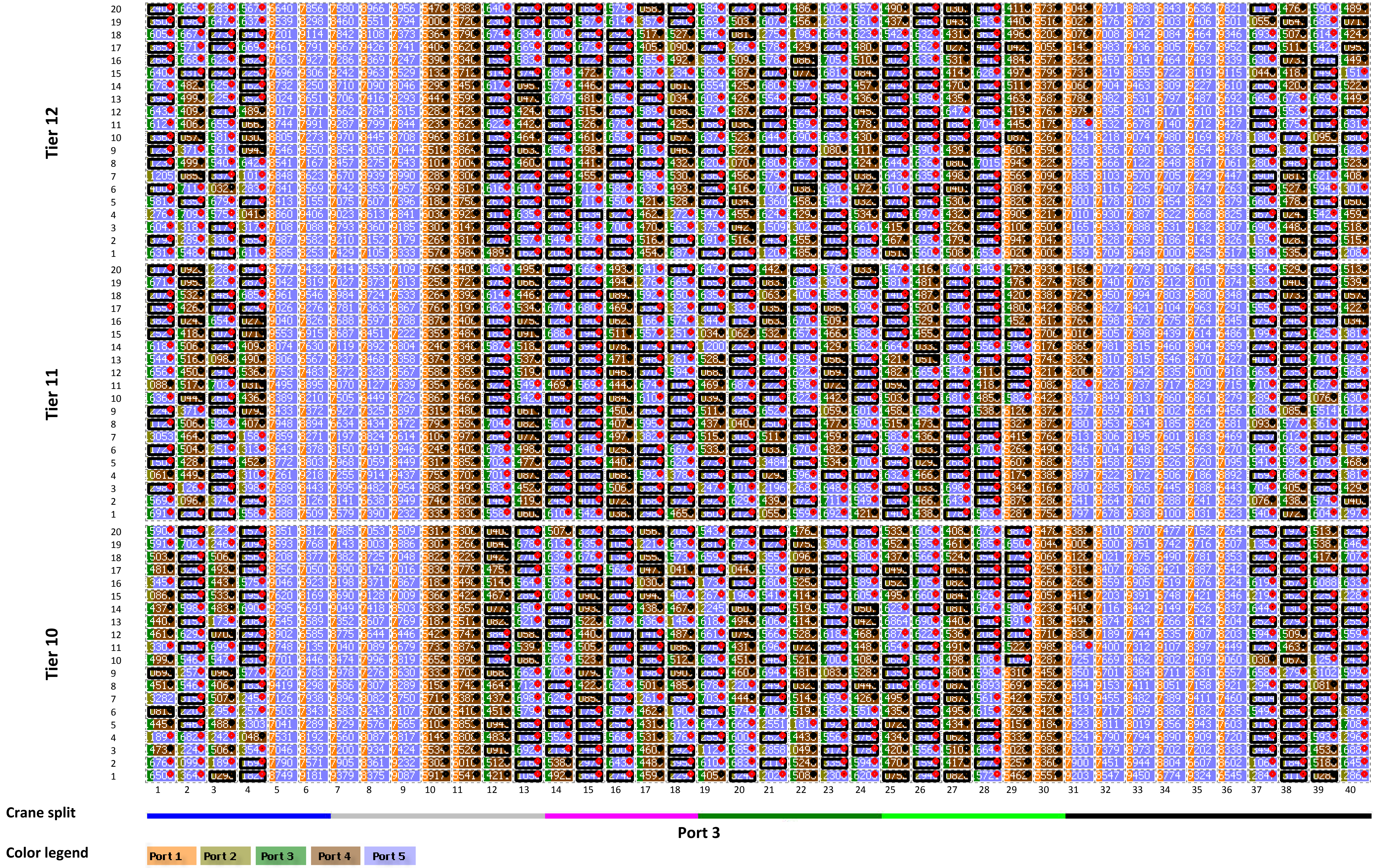

Tier 12
Tier 11
Tier 10
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Crane split
Port 2
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5

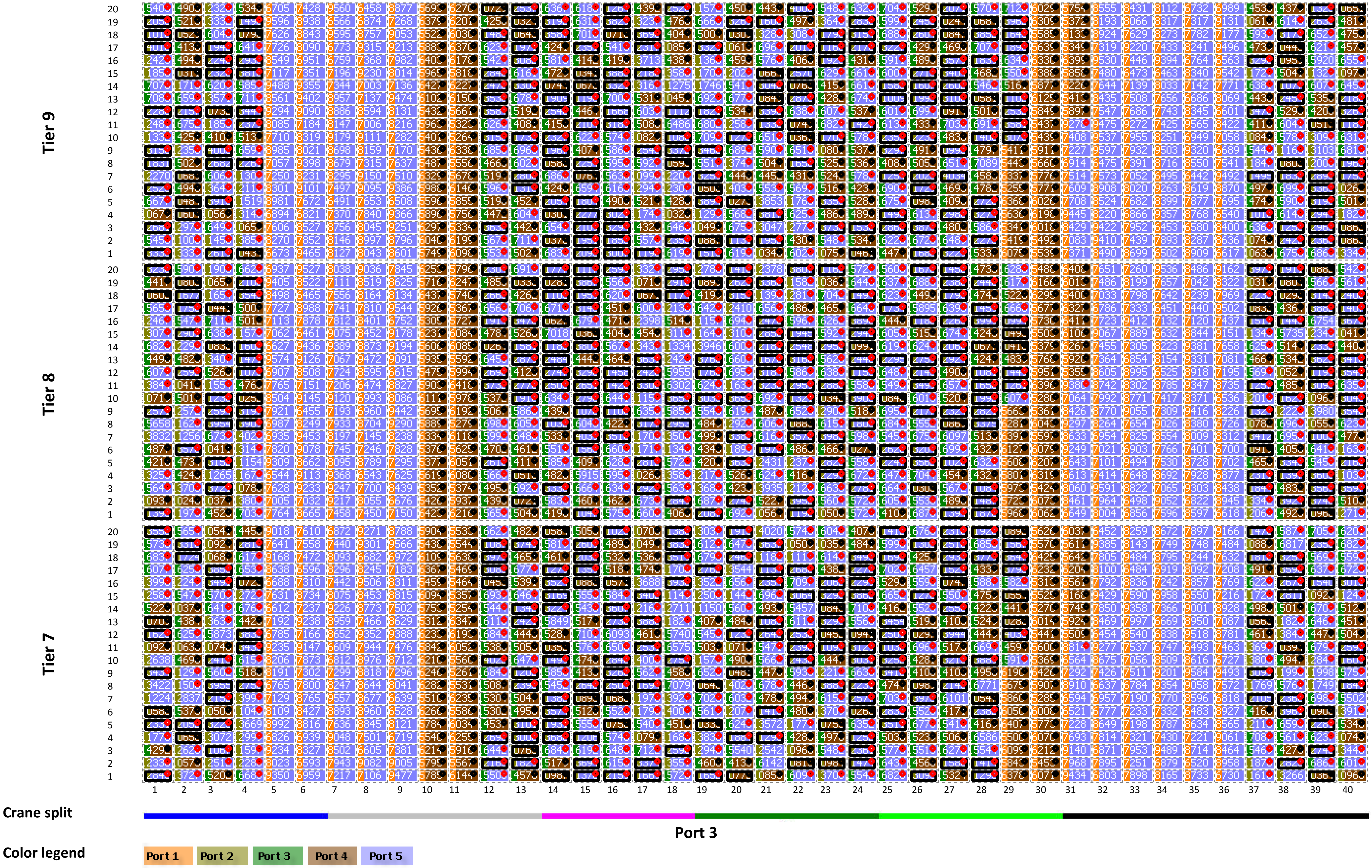

Tier 9
Tier 8
Tier 7
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
Crane split
Port 2
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5

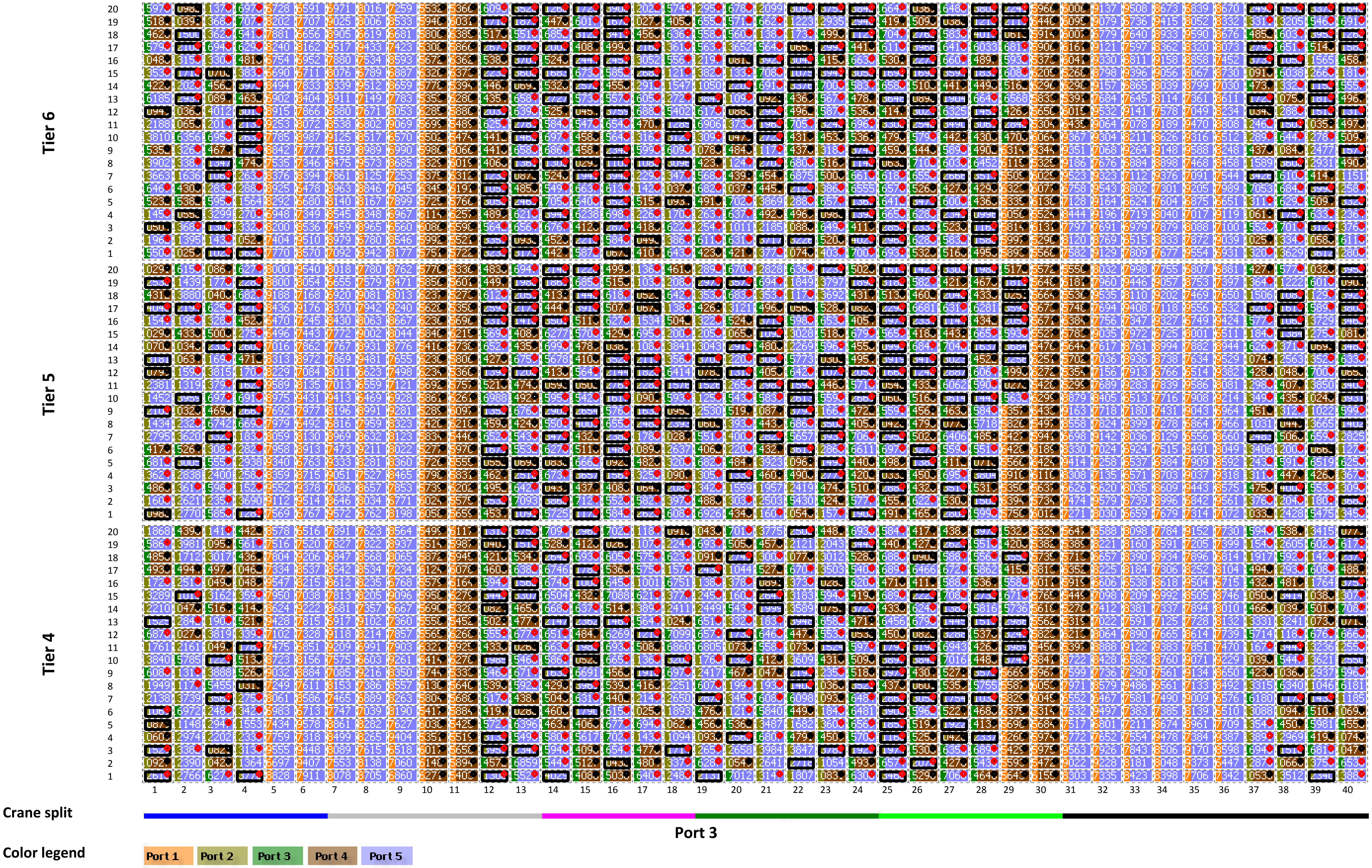

Tier 6
Tier 5
Tier 4
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
Crane split
Port 2

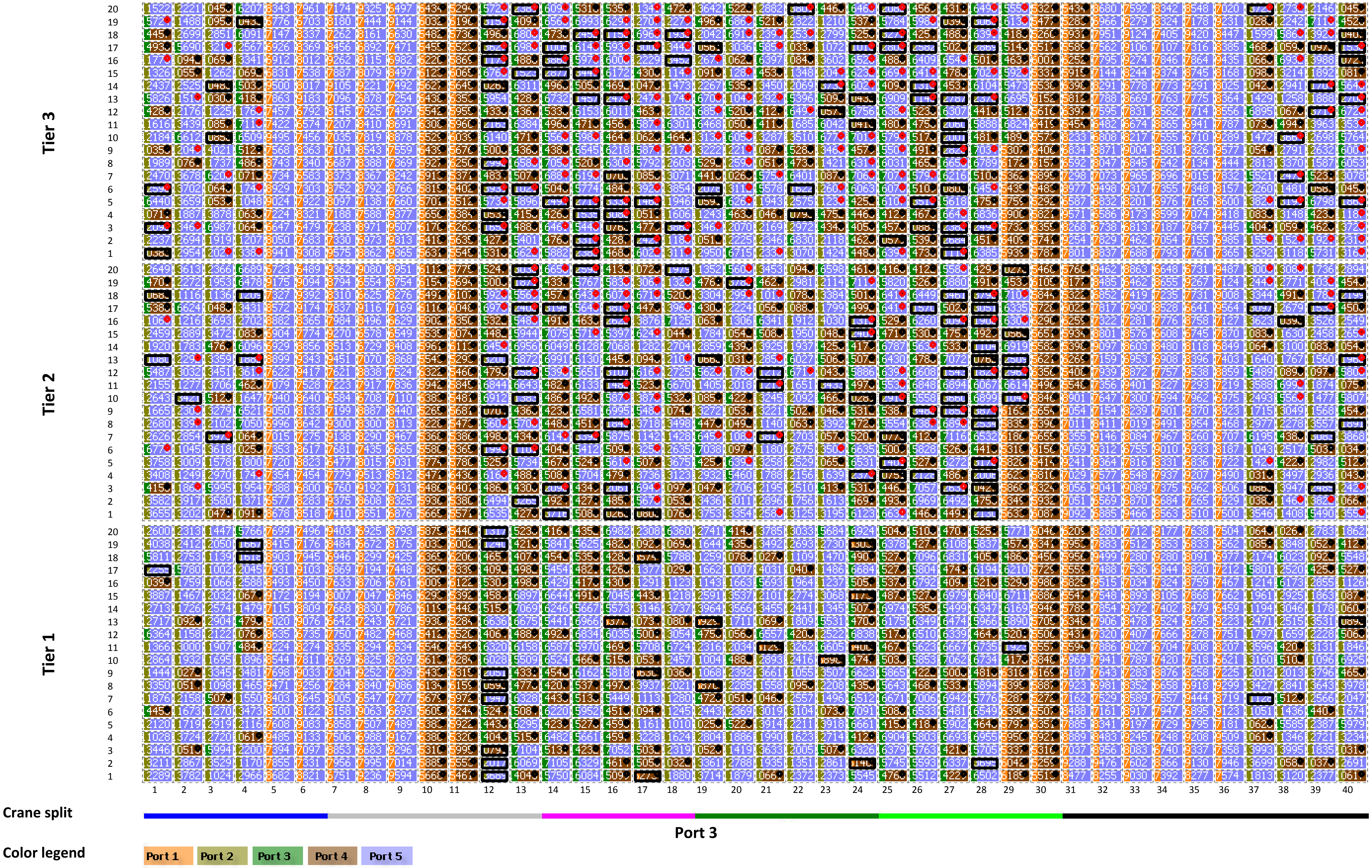

Tier 3
Tier 2
Tier 1
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
Crane split
Port 2
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5

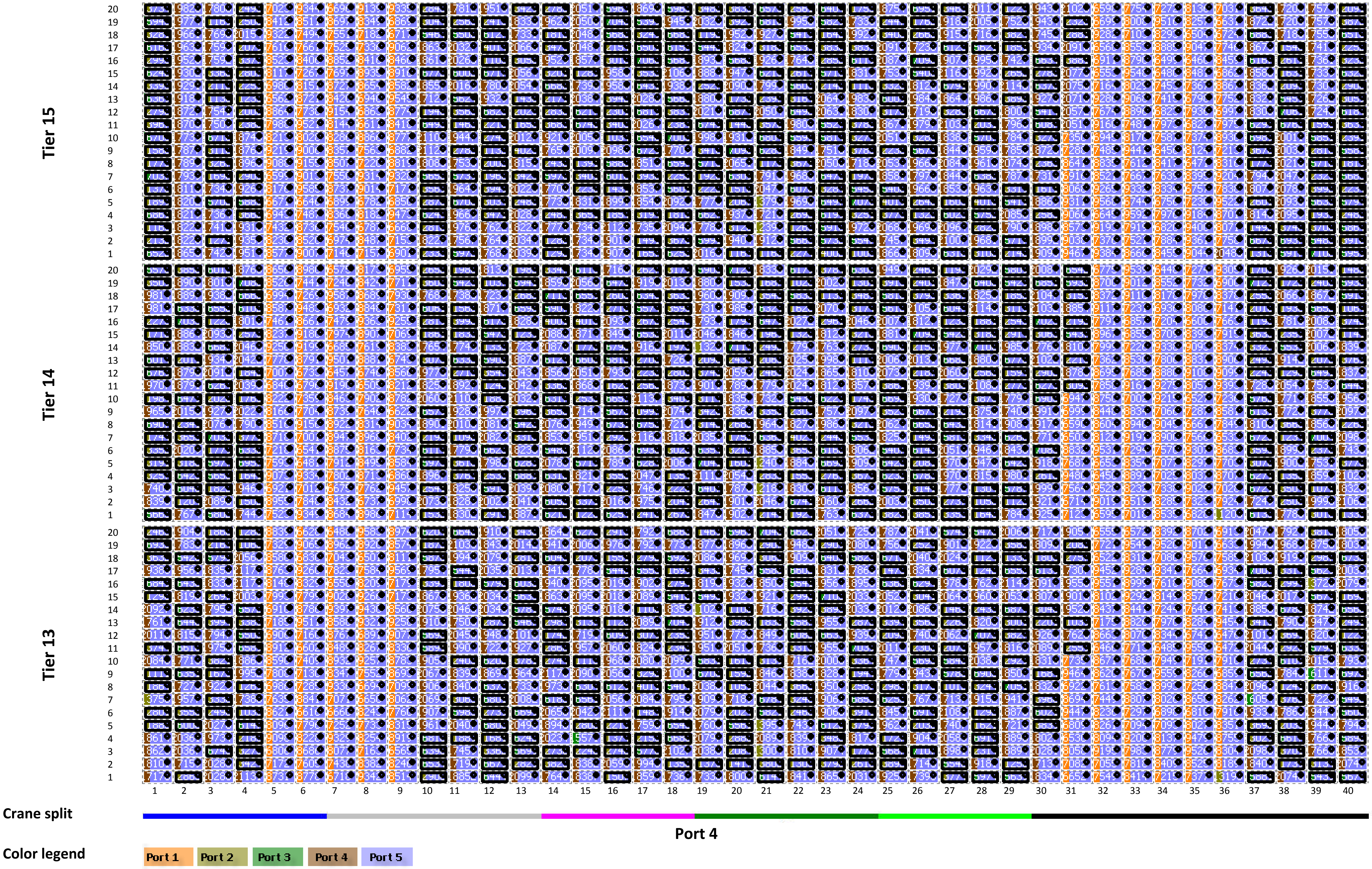

Tier 15
Tier 14
Tier 13
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
Crane split
Port 3
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5

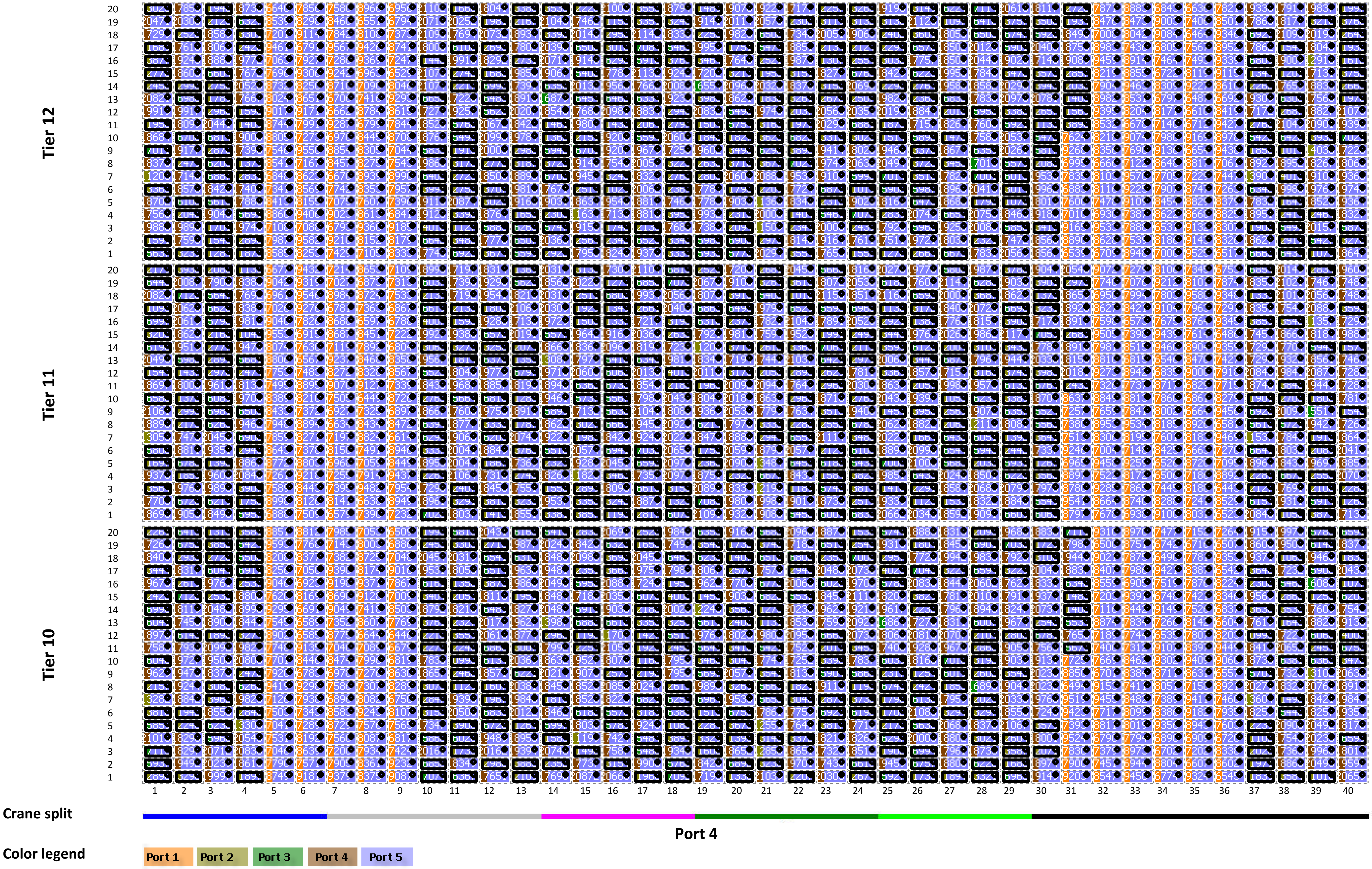

Tier 12
Tier 11
Tier 10
Crane split
Port 3
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5

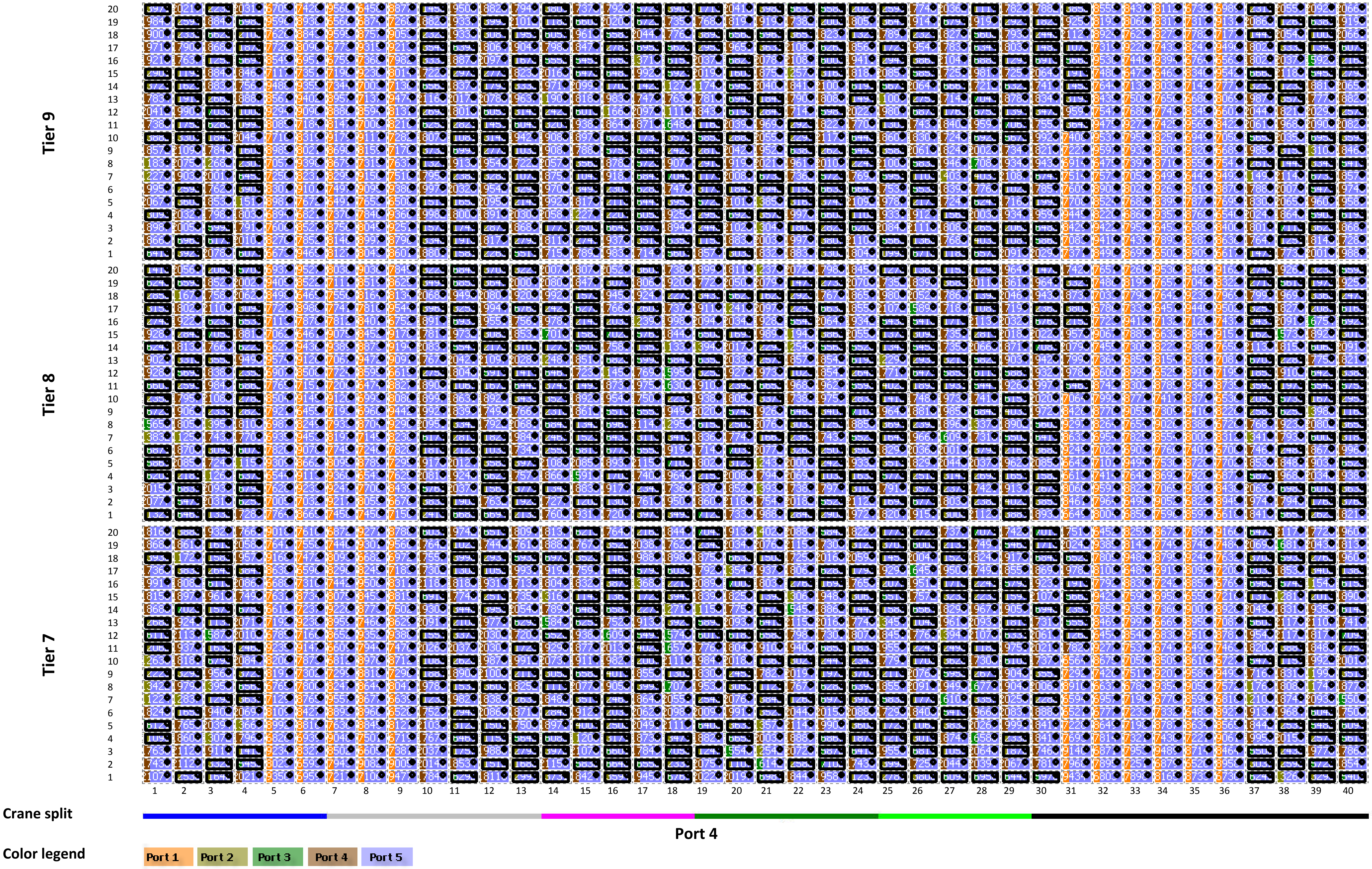

Tier 9
Tier 8
Tier 7
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
Crane split
Port 3
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5

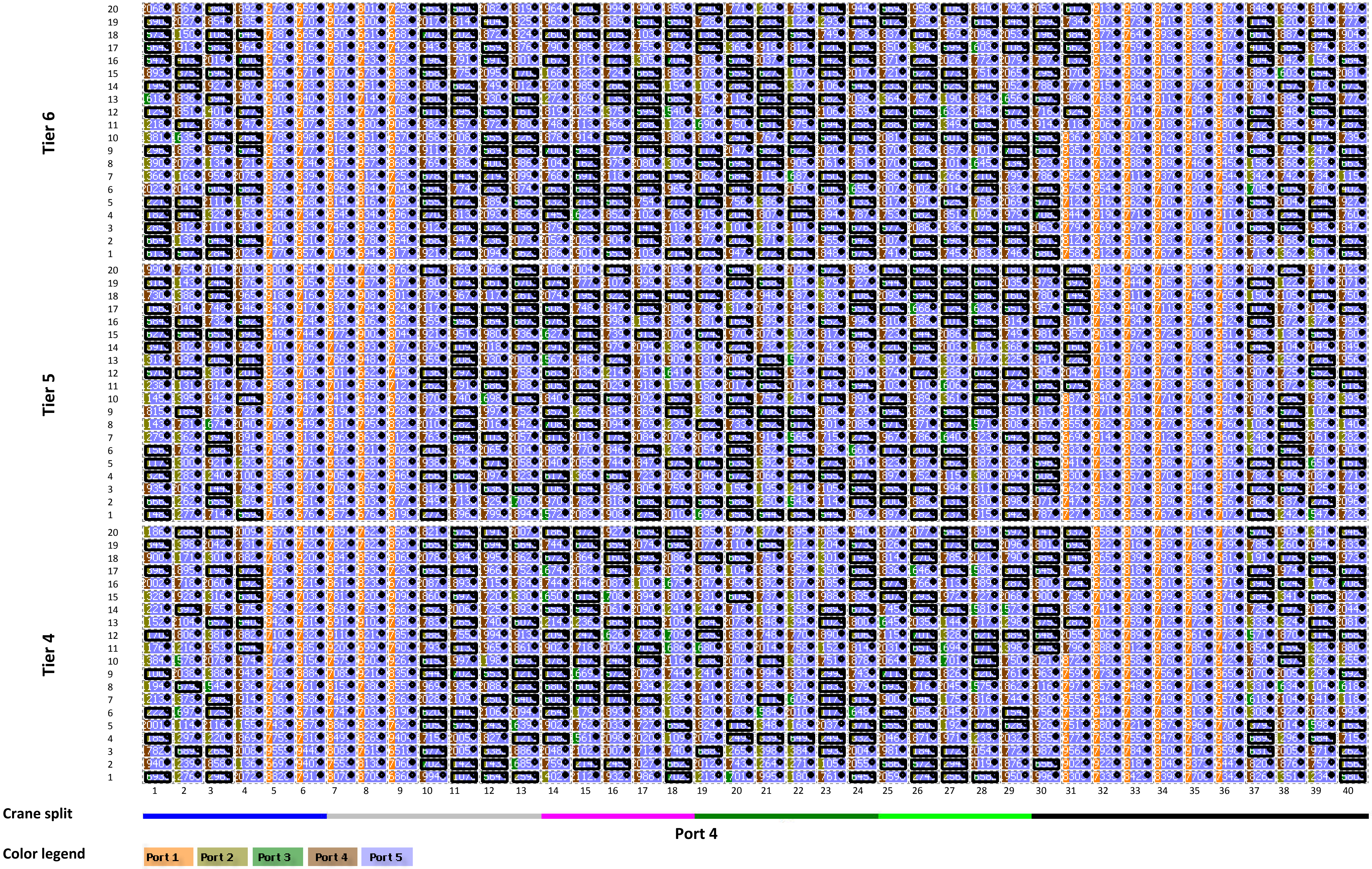

Tier 6
Tier 5
Tier 4
Crane split
Port 3
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5

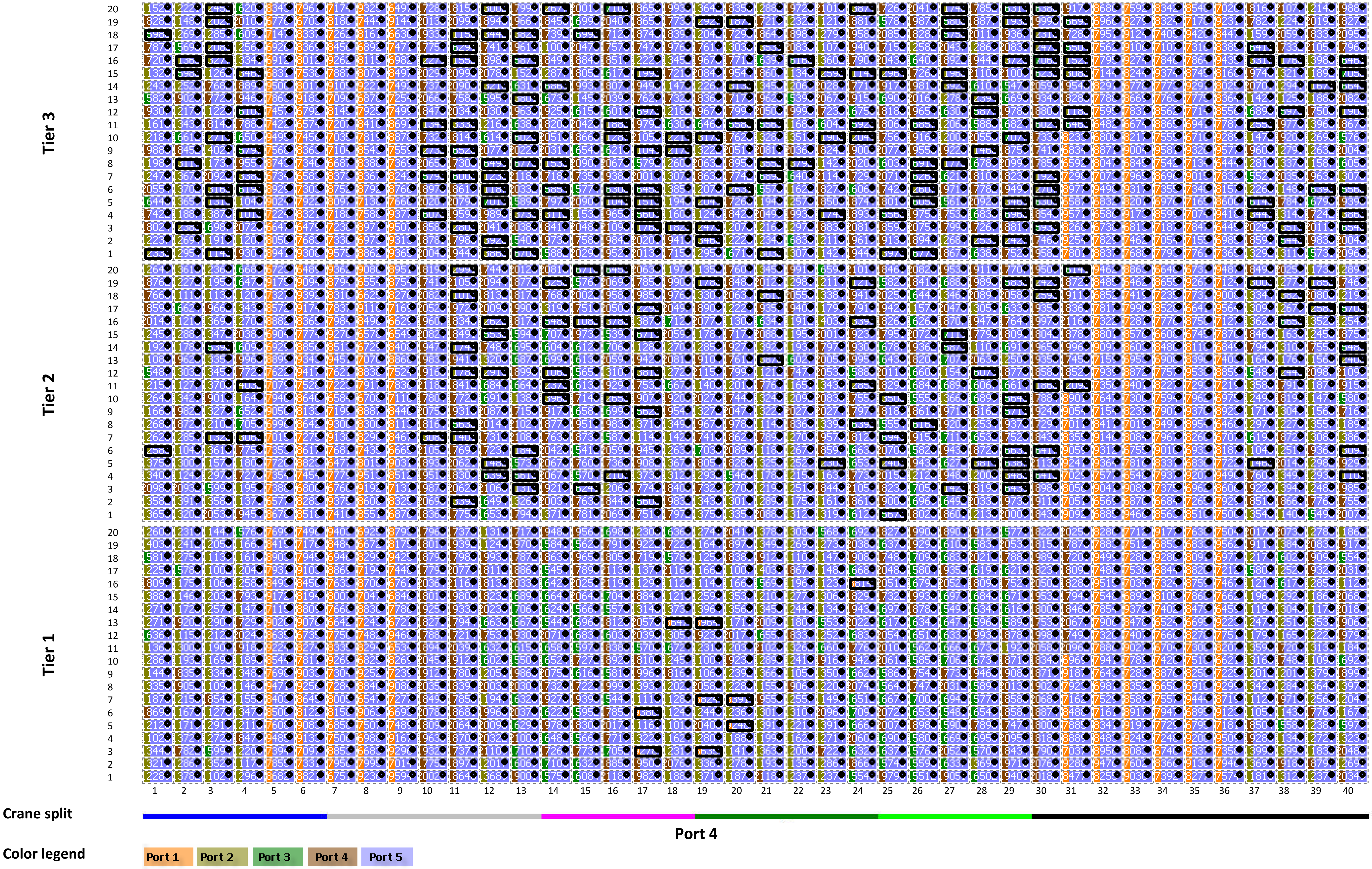

Tier 3
Tier 2
Tier 1
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
Crane split
Port 3
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5


Tier 15
Tier 14
Tier 13
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Crane split
Port 4
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5


Tier 12
Tier 11
Tier 10
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Crane split
Port 4
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5


Tier 9
Tier 8
Tier 7
Crane split
Port 4
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5


Tier 6
Tier 5
Tier 4
Crane split
Port 4
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5


Tier 3
Tier 2
Tier 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Crane split
Port 4
Color legend
Port 1
Port 2
Port 3
Port 4
Port 5