Università Degli Studi di Bergamo

School of Doctoral Studies

Doctoral Degree in **TECHNOLOGY INNOVATION AND MANAGEMENT**

XXXIII Cycle

PHD THESIS

Models and methods for the Slab Stack Shuffling problem

MANUEL CAVOLA

Director:

Prof Giuseppe Bruno

Univeristà Degli Studi Di Napoli Federico II

Academic year 2019/2020

Contents

List of Tables

List of Figures

Introduction

The shipbuilding industry represents a crucial industrial sector for many national economies either for strategical perspectives or for the direct impacts direct in terms of produced income and induced employment. It involves a vast and intricate network of links and exchanges within the sector itself and with other industries. Its supply chain has a global extension and it deals with the production of the raw materials (mainly steel slabs and profiles, paints and pipes), of the essential equipment (motors, electrical and electronic systems), of auxiliary components (furniture, accessories, special devices) which are purchased from numerous and specialised suppliers, worldwide located. Considering the (pre-Covid pandemic) growth of the demand, the increasing competition, the need to reduce the "time to market" within a production context typically inspired by an Engineer-To-Order philosophy, the management of the entire supply chain particularly complex and challenging. Among the many issues, one of the specific aspects concerns the co-existence along the various production steps, of very "traditional" labour intensive processes represented by the hull building with innovative productions related, for instance, to the control and telecommunication systems. However, if the "traditional" approach still represents, despite the attempts to innovate practices and procedures, the cultural mainstream that involves the organization of the information systems, whose setting is mainly based on not innovative hardware and software systems.

This work has been developed in this actual context, thanks to the collaboration to relevant enterprises of the sector engaged in processes of re-engineering of the production and logistics procedures to be translated in the definition of an innovative Decision Support Systems (DSS) whose design and realization have absorbed a relevant part of the entire work.

Within the DSS, actually implemented and introduced in the ordinary processes management, opportunities to embed optimization approaches to perform the most critical logistic processes have been considered.

The first example of introduction of ad-hoc procedures oriented to a process optimization is represented by the inbound handling of large steel slabs that constitutes the most crucial logistic operation, which is specific and original of this sector.

In practise, vessel hulls are typically built in sections ("blocks"), whose production is entrusted to specialised suppliers that cut, bend, shape and weld steel slabs and profiles in order to obtain a wide variety of subassemblies. These subassemblies are then delivered to the shipyard where the assembly process of the whole hull is performed. During these phases, the coordination between steel mill, subassembly centres and shipyards is fundamental.

The entire process starts with the delivery by the steel mill of steel slabs, from which, through complex cutting operations, all the components needed to produce subassemblies are provided. These slabs are steel plates that can weigh up to more than 13 tons and whose extension can reach 46 m^2. Due to their physical characteristics, steel slabs are stored in stacks. The storage and retrieval of the steel slabs from the stacks represent one of the main issues to overcome to avoid bottlenecks in the production process. In the literature, the problem has been defined as the Slab Stack Shuffling (SSS) problem. However, due to the specificity of the problem and of the involved industrial sector, the state of the art is not particularly rich. Then in this work a general framework of the problem is proposed in order to illustrate the various aspects and determinants affecting its management. Therefore, the proposal of a model and of an algorithm able to tackle some variants of the problem.

In practice, the thesis can be divided into two parts. The first part (Chapters 1 and 2) is devoted to the description of the main characteristics of the shipbuilding sector and of the typical production and logistics processing occurring in the context.

In particular, Chapter 1 is devoted to the description of peculiarities of the shipbuilding sector and market, highlighting the existing (pre-Covid pandemic) trends. Then a focus on the supply chain of the sector is proposed.

In Chapter 2, the attention is paid to the illustration of the industrial case study whose cooperation has been fundamental for the development of the work. In particular, after the description of the inputs (steel slabs and profiles), and of the outputs (components and subassemblies), the typical production and logistic problems are analysed and the issues related to the slab storage and retrieval operations underlined.

Indeed, being the slabs stacked one on the other, when a slab is not on the top of its stack, to retrieve it, a shift of the elements above the target one is necessary. Each shift is called "shuffle", and the less is the number of shuffles, the faster is the retrieval process. In literature, the problem that aims at choosing appropriate slabs for an order or a group of orders to minimise shuffles during the retrieval process is known as the Slab Stack Shuffling (SSS) Problem. This problem has always been contextualised as part of the slab creation process and occurs between continuous casting and the hot rolling process. However, its study is fundamental to develop the best possible solutions to the case study.

The second part of the work, (Chapters 3 and 4) illustrate the state of the art on the SSS problem and the proposal of new models and heuristics, related to some variants of the problem.

In particular, in Chapter 3, we start analysing the Stacking problems, a class of problems arising in the container handling with some similarities with the SSS problems. Then, a literature review of the main works specifically devoted to the SSS problem is conducted. Finally, as result of this analysis, we propose an original framework able to include all the elements that can characterise the SSS problem.

In the last chapter, a first mathematical formulation is introduced able to include some of the variants indicated in the proposed framework. Then, one of the proposed model is solved on a set of randomly generated instances – whose procedure of generation is in turn presented – through the use of a commercially available solver (CPLEX). The significant computing times associated to the solution instances suggest the proposal of appropriate heuristics to solve the problem that are consequently tested. Finally, a brief description of the implemented DSS that should include the optimization procedures to be implemented, as further developments of the present work, is reported.

1 An introduction to the shipbuilding sector: trends, actors, and logistics

Summary

This chapter provides an overview of the shipbuilding sector. Specifically, we start outlining the main characteristics of the sector, particularly emphasizing the segment dealing with new ships production, namely the *Shipbuilding Industry*. Afterwards, we present an analysis of the shipbuilding market in terms of produced items, competition aspects, entry/exit barriers, and industry trends before the spread of the Covid-19 emergency.

Finally, we focus on the description of the shipbuilding supply chain, detailing its most prominent logistics phases.

1.1 The shipbuilding sector

The shipbuilding industry is a dynamic and competitive sector that has traditionally been of crucial importance within the economies of many countries - such as Great Britain, France, Germany, Italy, USA, Japan, Korea and China - in terms of both direct and induced employment (Ferrari, 2012). It has an intensive network of links and exchanges with other industries in the manufacturing and service sector, particularly the mechanical and metallurgical industries, and those concerned with the provision of high value-added and high-qualification services. This sector is also characterized by labour-intensive processes with a high degree of complexity, thus requiring significant skill levels.

In general, shipbuilding is defined as the set of activities relating to the construction of ships and pleasure crafts; however, the manufacturing processes encompass very heterogeneous practices that need to be specifically characterized and analysed.

In the broad context of shipbuilding, three main segments can be identified: (i) *the Shipbuilding Industry*, dealing with the production of new ships; (ii) the *Ship Repair Industry*, mainly oriented at maintenance activities; and (iii) the *Conversion Industry*, aimed at the conversion of existing ships. Despite their similarities, each of these industries faces its own demand and presents different trends. This thesis explicitly focuses on the *Shipbuilding Industry*.

In this segment, production processes mostly concern three types of outputs: *high-tech ships*, *standard ships*, and *cruise ships*. *High-tech ships* include chemical vessels (i.e., oil tankers carrying chemicals in bulk), gas tankers transporting liquefied natural gas, and Roll-on/roll-off ships (RoRo). The latter are designed to carry wheeled cargo in combination with containers (RoCon) or passengers (RoPax, e.g., ferries). Figure 1.1 provides some examples of the vessels mentioned above.

On the other hand, *standard ships* are (unwheeled) cargo ships not devoted to passengers transportation, whose design and construction is not relatively complex, comprising container ships, bulkers, liquid tankers, and general cargo vessels (see Figure 1.2).

Finally, *cruise ships* are those used for the transport of passengers, for which it is possible to stay overnight on board. Their dimensions can vary in a vast range, i.e., from a few tons up to 150-200 thousand tons Gross Tonnage (GT, i.e., the measurement of the ship's internal volumes). Cruise ships can reach a length of almost 400 m, heights of 70 m and can accommodate up to 5-6 thousand passengers in addition to the crew. They are characterized by the high quality of the equipment, interior finishes and furnishings that render them actual "floating hotels" (Figure 1.3). These elements help to understand the complexity of this kind of ship compared to the former ones. Think, for instance, that standard ships may require up to 550 000 parts for a complex vessel, while 900 000 parts are necessary for cruise ships (Gourdon et al., 2019).

We should note that nowadays, differently from a few decades ago, the construction of a ship is only partly carried out by shipbuilders. Indeed, it is increasingly common for shipbuilders to buy significant production parts from external manufacturers and assemble them internally. Therefore, we can consider the shipbuilding industry an "*assembly industry*" that relies heavily on work-in-progress inputs. As Gourdon et al. (2019) underline, shipbuilders' value-added lies, on average, only between 20% and 30% of the final production value.

For all these outputs, construction activities are carried out in specialized factories called *shipyards*, where production/assembly is performed in a fixed-station. In other words, all the resources and materials are conveyed to the fixed-station, and the ship to be built remains fixed, i.e., it does not move within the shipyard. Note that this the only possible way, given the considerable size of the product (Pareschi, 2007), and renders the overall logistic production process particularly difficult to manage.

Therefore, the shipbuilding industry, prominently that of the high-tech and cruise ships, is characterized by high complexity, mainly due to the peculiarity of combining the most varied skills and actors in a closely interconnected system. The latter, in turn, raises the challenge to manage a considerable amount of information coming from multiple and diversified sources.

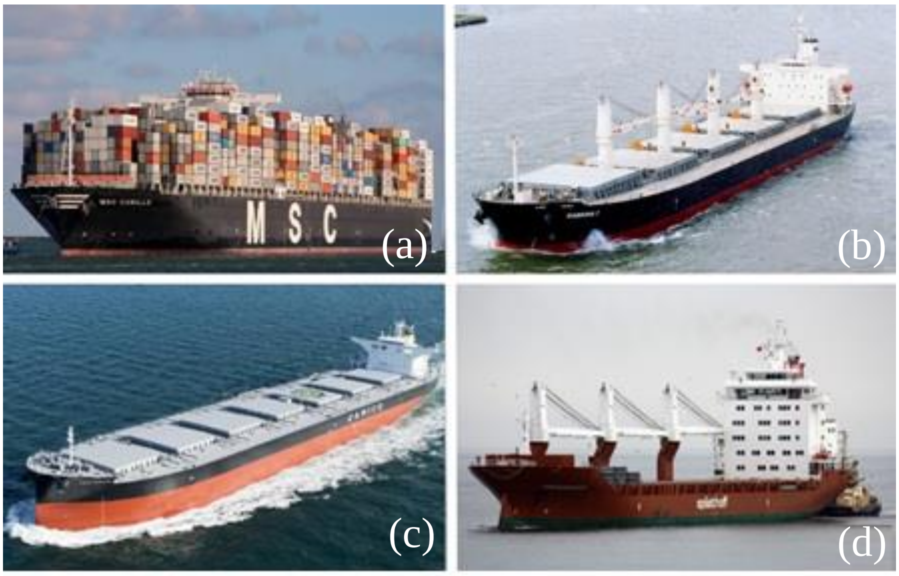


Figure 1.1 – High Tech ships: (a) Chemical vessel; (b) Gas tanker; (c) RoRo ship; (d) Ferry

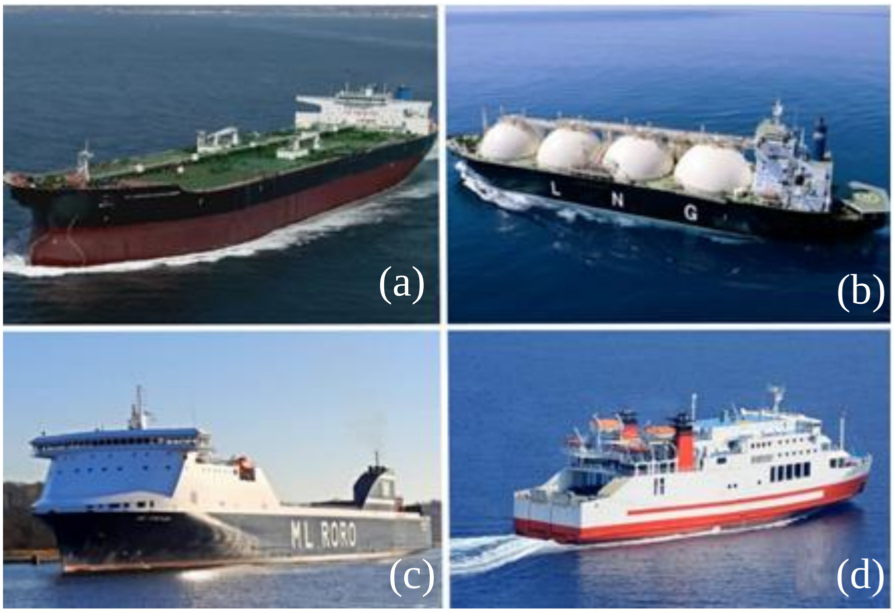


Figure 1.2 – Standard ships: (a) Container ship; (b) Bulker ship; (c) Tanker; (d) General cargo ship

Figure 1.3 – A cruise ship

Additionally, according to the classification proposed by Wortmann (1983), it can be considered as an Engineer-To-Order (ETO) sector since the customer's order "goes back" to the design/engineering phases to trigger the production process. Indeed, the level of customization, especially in high-tech and cruise ships, is very high. Although some may belong to the same series, they usually have distinctive elements that determine a broad diversification of the final products, thus requiring very flexible supply chains.

In general, it is well-known (and relatively intuitive) that the demand in this sector typically depends on the mobility demand for transport, that is, the need to move goods or offer services. In turn, the demand for transport depends on the demand for final goods and services, and, hence, ultimately on general economic performance. Therefore, the consequence of an economic crisis may tighten the demand for transport itself and consequently for new commercial ships.

In practice, these are the effects brought by the outbreak of the recent Covid-19 pandemic, which determined a profound crisis in the international shipping sector, leading to a significant reduction in the growth prospects for the year 2020 (Nautilus, 2020). In fact, it caused a collapse in demand and a sudden interruption of the production activities, reflected in the postponement of new ships orders and of the corresponding expected revenues (LBJ, 2020).

Finally, a further characteristic of the shipbuilding sector is that the supply adapts to the demand with a significant delay, given the long throughput times (three years, on average) elapsing from the moment the order for a new ship is received and the moment of the final delivery. Notably, despite the great recession in 2009, global ship completions were historically high in 2010 and 2011.

In other words, although Covid-19 exacerbated the difficulties of shipbuilding companies to meet an already highly volatile demand, its actual impacts on the market can only be assessed in the next future. As a result, business levels remained relatively stable. Moreover, according to the data provided by Fincantieri S.p.A., a world-leading company, especially in the cruise ship field, production volumes should return to pre-Covid levels in the second half of 2021, and revenues in the second half of 2020 should be broadly in line with those in the corresponding period of 2019 (La Stampa, 30 October 2020). This evidence testifies that the problem under investigation in the present thesis keeps being still of both practical and economic relevance in the shipbuilding industry, even in these uncertain times.

1.2 Recent trends in the shipbuilding market

This section presents an analysis of recent trends in the shipbuilding market. We feel the need to highlight the data hereafter discussed refer to the period before the outbreak of the Covid-19 pandemic. As we already underlined, these projections may be partially reshaped in the next future due to the pandemic's unknown economic impact.

In general, the shipbuilding sector has been characterized by remarkable growth in the last years; in 2018, the volume of new cruise ship orders reached 22 billion dollars, i.e., 35% of the total (ICE, 2018). This increase has led to a growth in European shipyards compared to Asian ones, which are more focused on standard ships production.

However, oil tankers and bulkers remain the most produced ships since they more easily allow achieving economies of scale and minimizing unitary transportation costs.

The general increase in the business volumes has also been driven by the need to produce more fuel-efficient ships in compliance with the latest environmental regulations. Indeed, fuel represents a relevant quota of the overall operating costs. Finally, growth is also expected in the construction of RoRo ferries, offshore market ships and gas transport ships due to the increasing energy needs (BRS GROUP, 2019).

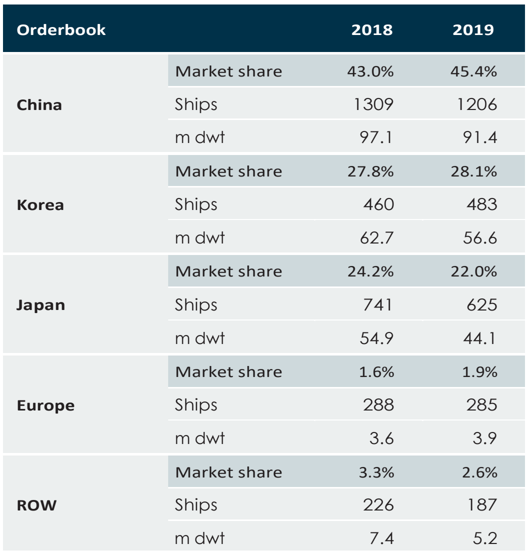
Thus, the market trend shows a global recovery in the shipbuilding sector compared to the 2016 crisis. This market recovery also positively impacted new construction prices that had fallen by around 25% in 2016 since 2009 (DSF, 2016) and increased up to 10% in 2018 (BRS GROUP, 2019). Nevertheless, the 2017 delivery level was still higher than the new orders' level. This evidence indicates an enduring situation of imbalance between supply and demand. In 2016, this oversupply caused a sharp decline of new ships' orders, leading the world's leading shipbuilders to reduce the number of active shipyards to cut costs (OECD, 2017). However, it is interesting to point out that Europe shows an opposite trend to the rest of the world market. In particular, the Italian shipbuilding industry (as we also detail in the next section) minimized these negative impacts, given its intense focus on high value-added market segments characterized by severe entry barriers.

Notably, another recent phenomenon in the shipbuilding industry concerns the frequent resort to outsourcing of non-core activities. Unlike a few decades ago, where shipbuilding companies were heavily vertically integrated as they carried out most of the ships' production processes, one can see a gradual shift to an "*assembly shipyard*" paradigm. Accordingly, many activities, especially low-tech ones, are allocated to external partners. In contrast, core ones are still performed "in-house" as they can be a source of competitive advantage (Mello et al., 2010). More often than not, this has also been accompanied by merging strategies, through which bigger companies acquired smaller ones, thus creating large industrial groups (horizontal integration) to better tackle competition in an increasingly dynamic and global context.

1.3 A closer look at the shipbuilding supply side

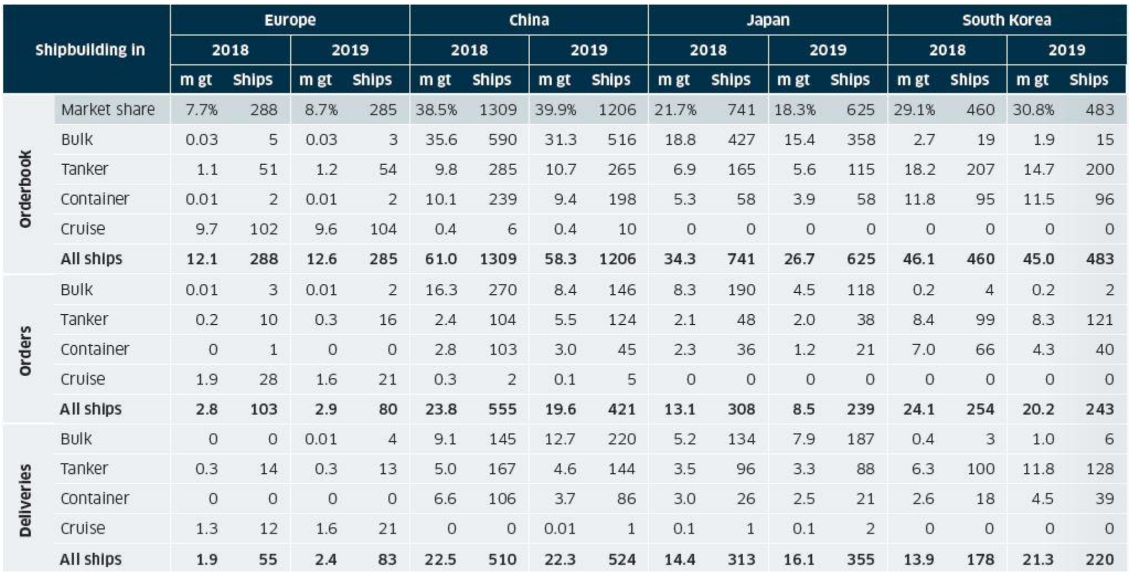
A sort of duopoly on the supply side characterizes the shipbuilding market, given the presence of the Asian shipyards, on the one hand, and the European ones, on the other. The latter are gathered in an umbrella organization named CESA (Community of European Shipyards Associations). According to the United Nations Conference on Trade and Development, the three largest economies involved in shipbuilding (i.e., China, Korea and Japan) contributed to 90% of the global completions of ships in Gross Tonnage (GT) tons in 2018 (UNCTAD, 2019). In particular, the Chinese shipbuilding industry stands out with a considerable expansion of its production capacity mainly driven by the need to have a fleet that would help support the significant import and export flows.

The market shares held by the different players vary according to the considered segments. For example, cruise ship production is mainly concentrated in Europe (particularly in Germany, Italy, France, and Finland), while standard ships are largely manufactured in Asian countries, especially Japan and Korea. As concerns the Chinese industry, although it has historically focused on producing standard ships, given the low labour costs and economies of scale resulting from the significant production volumes achieved in recent years, it is shifting its production to cruise ships. Lower labour costs are also the primary motivation behind the strengthening of Eastern Europe industries, i.e., Croatia and Romania, and new players' entry in the market, like India and Vietnam in Asia. To provide the reader with some recent quantitative data on the shipbuilding industry, Table 1.1 shows the order book, expressed in millions of gross capacity tonne (i.e., deadweight tonnage - dwt), referring to years 2018 and 2019. It emerges that European shipyards, having lower production volumes, have less capacity to exploit economies of scale than Asian shipyards. This fact depends on the type of production. Indeed, by looking at Table 1.2, one can note that, despite a very slight decrease in orders in 2019, it is quite evident that the European yards are still predominant in the field of cruise ships.



Orderbook		2018	2019
China	Market share	43.0%	45.4%
	Ships	1309	1206
	m dwt	97.1	91.4
Korea	Market share	27.8%	28.1%
	Ships	460	483
	m dwt	62.7	56.6
Japan	Market share	24.2%	22.0%
	Ships	741	625
	m dwt	54.9	44.1
Europe	Market share	1.6%	1.9%
	Ships	288	285
	m dwt	3.6	3.9
ROW	Market share	3.3%	2.6%
	Ships	226	187
	m dwt	7.4	5.2

Table 1.1 - Distribution of shipbuilding market shares (source: BRS GROUP, 2020)



Shipbuilding in		Europe				China				Japan				South Korea			
		2018		2019		2018		2019		2018		2019		2018		2019	
		m gt	Ships	m gt	Ships	m gt	Ships	m gt	Ships	m gt	Ships	m gt	Ships	m gt	Ships	m gt	Ships
Orderbook	Market share	7.7%	288	8.7%	285	38.5%	1309	39.9%	1206	21.7%	741	18.3%	625	29.1%	460	30.8%	483
	Bulk	0.03	5	0.03	3	35.6	590	31.3	516	18.8	427	15.4	358	2.7	19	1.9	15
	Tanker	1.1	51	1.2	54	9.8	285	10.7	265	6.9	165	5.6	115	18.2	207	14.7	200
	Container	0.01	2	0.01	2	10.1	239	9.4	198	5.3	58	3.9	58	11.8	95	11.5	96
	Cruise	9.7	102	9.6	104	0.4	6	0.4	10	0	0	0	0	0	0	0	0
	All ships	**12.1**	**288**	**12.6**	**285**	**61.0**	**1309**	**58.3**	**1206**	**34.3**	**741**	**26.7**	**625**	**46.1**	**460**	**45.0**	**483**
Orders	Bulk	0.01	3	0.01	2	16.3	270	8.4	146	8.3	190	4.5	118	0.2	4	0.2	2
	Tanker	0.2	10	0.3	16	2.4	104	5.5	124	2.1	48	2.0	38	8.4	99	8.3	121
	Container	0	1	0	0	2.8	103	3.0	45	2.3	36	1.2	21	7.0	66	4.3	40
	Cruise	1.9	28	1.6	21	0.3	2	0.1	5	0	0	0	0	0	0	0	0
	All ships	**2.8**	**103**	**2.9**	**80**	**23.8**	**555**	**19.6**	**421**	**13.1**	**308**	**8.5**	**239**	**24.1**	**254**	**20.2**	**243**
Deliveries	Bulk	0	0	0.01	4	9.1	145	12.7	220	5.2	134	7.9	187	0.4	3	1.0	6
	Tanker	0.3	14	0.3	13	5.0	167	4.6	144	3.5	96	3.3	88	6.3	100	11.8	128
	Container	0	0	0	0	6.6	106	3.7	86	3.0	26	2.5	21	2.6	18	4.5	39
	Cruise	1.3	12	1.6	21	0	0	0.01	1	0.1	1	0.1	2	0	0	0	0
	All ships	**1.9**	**55**	**2.4**	**83**	**22.5**	**510**	**22.3**	**524**	**14.4**	**313**	**16.1**	**355**	**13.9**	**178**	**21.3**	**220**

Table 1.2 - Orders for new commercial ships (source: BRS GROUP, 2020)

Focusing solely on the European market, Figure 1.4 shows the distribution of new ships orders (in Gross Tonnage) and the corresponding number of ships (in green) by country in 2019 (BRS GROUP, 2020). We can note that Italy leads the ranking (although its number of ships is lower or at least comparable to other countries), mainly thanks to its 'flagship' shipbuilding group *Fincantieri*. This latter is the largest cruise shipbuilder in the world, with a total of 37 large units to be delivered before 2027. It recently finalized the take-over of its Norwegian affiliate group *Vard* and started a negotiation to become the majority shareholder of the France leading shipyard, *Chantiers de l'Atlantique*. However, this acquisition failed because of the wide-diffused concern (expressed by various major cruise shipowners) of a potential monopolistic position for Fincantieri (BRS GROUP, 2020).

Moving our attention to the Rest of the World (RoW), we note that the order book for shipyards (see Table 1.3) continues to crumble from 7.4 million tons deadweight in 2018 to 5.2 million tons deadweight in 2019. RoW market share dropped from 3.3% to 2.6%. Deliveries also decreased, from 4.6 million tons deadweight in 2018 to 3.5 million deadweights in 2019. Consequently, the ratio between the current order book and yearly output shrank to 1.5 in 2019 against 1.9 in 2018. The collapse of the Philippines leading shipyard Hanjin Subic in 2019 had a significant impact on RoW performance. Indeed, it was a great contributor, representing about 30% of RoW new orders in 2017 (although it did not secure any new order in 2018). CSBC (Taiwan), another key-player, did not manage to secure orders in 2019 despite accounting for 50% of RoW new orders in 2018. At the end of 2019, 13 RoW shipyards secured new orders, with 89% of them managed by just two shipyards: the Philippine group Tsuneishi Cebu and the Vietnamese Hyundai Vinashin (BRS GROUP, 2020).

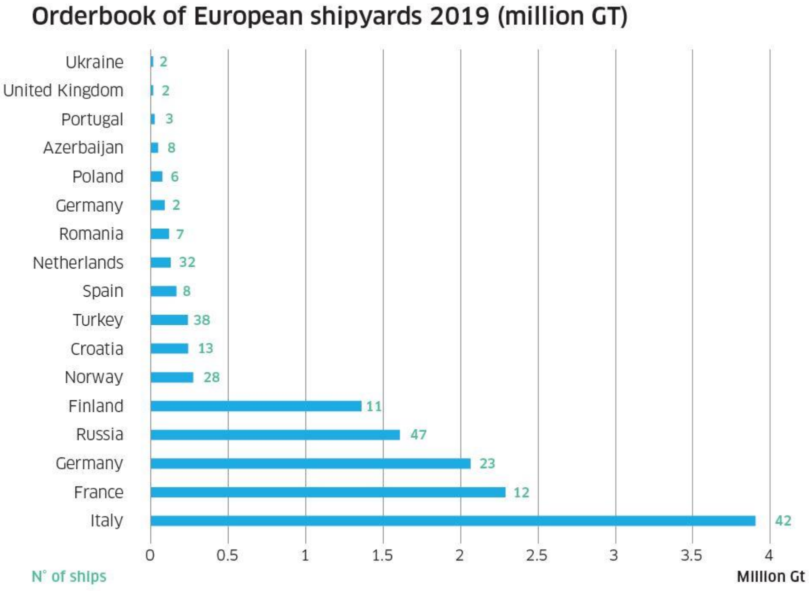
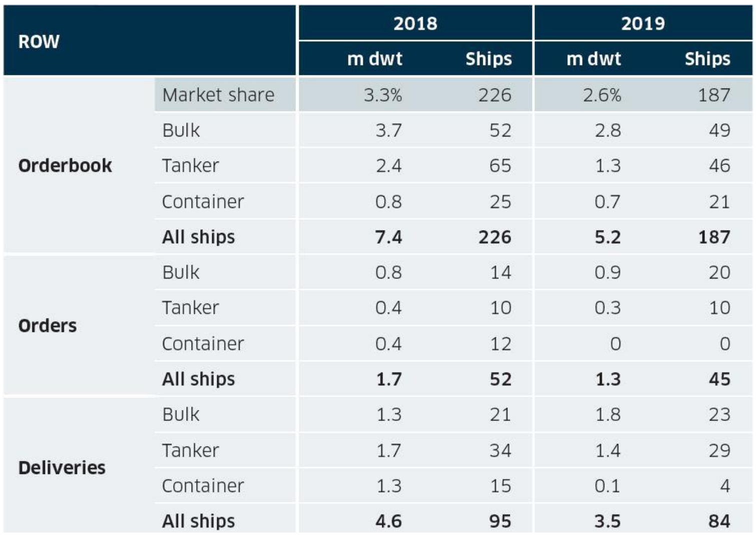


Figure 1.4 - Allocation of orders between European countries by GT and number of ships (source: BRS GROUP, 2020)



ROW		2018		2019	
		m dwt	Ships	m dwt	Ships
Orderbook	Market share	3.3%	226	2.6%	187
	Bulk	3.7	52	2.8	49
	Tanker	2.4	65	1.3	46
	Container	0.8	25	0.7	21
	All ships	**7.4**	**226**	**5.2**	**187**
Orders	Bulk	0.8	14	0.9	20
	Tanker	0.4	10	0.3	10
	Container	0.4	12	0	0
	All ships	**1.7**	**52**	**1.3**	**45**
Deliveries	Bulk	1.3	21	1.8	23
	Tanker	1.7	34	1.4	29
	Container	1.3	15	0.1	4
	All ships	**4.6**	**95**	**3.5**	**84**

Table 1.3 – Rest of world's Orders for new commercial ships (source: BRS GROUP, 2020)

We should also underline that it is customary in the shipbuilding sector to have very few major-players (typically one per country) controlling large portions of the national market shares. The presence of a limited number of players can be explained by significant entry/exit barriers in the market.

Entry barriers are typically high, due to the need to have the specific infrastructure, consolidated and complex engineering, design and high-level management skills related to many suppliers sub-contractors involved in the shipbuilding process. High costs are incurring for equipment, buildings, and transport necessary for the initial production and labour costs. Indeed, shipbuilding requires a very diverse and skilled workforce, having much of the activities still carried out by hand.

Besides, it is necessary to find suppliers with certifications issued by dedicated bodies, called classification societies (RINA, Lloyd Register, Bureau Veritas, ABS, among others), both for the design, development and building of structural components for naval use. Finally, the quality of the produced output and companies' reputations are also important within the shipbuilding industry. Consequently, the threat posed by new entrants to the market is low; this is even more relevant in high-tech and cruise ship segments requiring outstanding technological and organizational expertise, experience, and a highly qualified workforce.

Concerning exit barriers, the considerable capital investments for installations and infrastructures make it challenging to reorient any economic activity linked to shipbuilding or repairing. Note that the resale value for shipyards is practically low or null. Moreover, political restrictions, due to the strategic importance of the sector, may impose further barriers. In a time of economic crisis, national governments typically support financial plans to avoid the closure of construction sites to keep them active and, thus, curbing the natural tendency for unprofitable businesses to exit the market.

1.4 The shipbuilding supply chain

As we discussed in the previous sections, the shipbuilding sector's supply chain has been significantly changing in the last decades. Indeed, while most of the whole production processes took place in-house and in a single country, the recent developments in information and communication technologies have significantly contributed to creating and expanding such production networks, promoting the creation of Global Value Chains (GVCs). The increasing efficiency in information sharing, communication and freight transport has enabled companies to collaborate over large distances along the supply chain. GVCs have also shifted the shipbuilding industry towards an interconnected production approach that renders the efficient and effective management of an increasingly extensive and global network of suppliers one of the main challenges in this sector.

Furthermore, due to the high number and complexity of the processes and the need for coordination between construction sites and suppliers, a strong efficiency in the site itself is essential. Many studies have shown how the improvement of coordination between construction sites, suppliers and customers is becoming increasingly crucial (Fleischer et al.,1999, Chryssolouris et al., 2004; Celik et al., 2009; Guneri et al., 2009).

According to Vlachakis et al. (2016), identifying core competencies and outsourcing rationalization, identifying clear supplier roles and responsibilities, and creating long-term alliances are fundamental to achieve effective management.

The value chain of the naval sector involves many organizations, institutions and companies. In addition to the final customer (i.e., the *shipping company*), construction sites and suppliers, it is possible to highlight many other players, such as research centres, insurance companies, banks, temporary agencies, classification societies and official authorities (Held, 2010). Figure 1.5 depicts the whole variety of the actors involved in shipbuilding, thus denoting the network's inherent complexity. More specifically, *research centres* are concerned with studying new technologies and process innovation; *insurance companies* offer insurance cover to construction sites during and after the construction phases, while *banks* provide the necessary financial support. *Temporary agencies* are in charge of selecting qualified workers, whereas *classification societies* are responsible for verifying and certifying that the vessel's construction process complies with the required standards. Finally, *official authorities* are the bodies responsible for granting the necessary concessions for carrying out production activities.

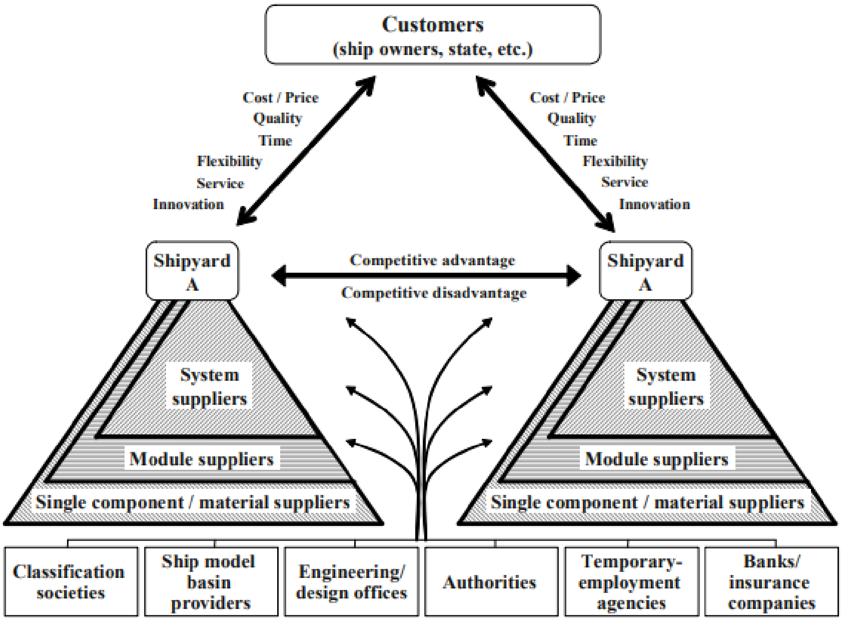


Figure 1.5 – Actors involved in the shipbuilding industry (source: Held, 2010)

For the sake of completeness, we should also mention that, due to the variety of items needed, ranging from relatively simple and standardized parts to more complex systems, many suppliers are involved in the shipbuilding supply chain. We can distinguish between those that provide materials, components and systems and those that offer engineering and design services. Besides, a further subdivision into *specialized* and *generic suppliers* can be made.

Specialized suppliers are those concerned with the supply of propellers, rudders, and navigation systems. The others produce general-purpose products and typically supply various industrial sectors. Overall, it can be said that the majority of these suppliers are small or medium-sized firms. This element certainly influences these suppliers' role (particularly the specialized ones), as it tends to exacerbate their subordination and degree of dependence on their customer, namely the shipbuilding company.

1.5 The main processes in the shipbuilding supply chain

This section discusses the main processes characterizing the shipbuilding supply chain.

As we highlighted in the previous sections, shipyards must have adequate systems and robust management/organizational competencies to share information, develop production plans, control materials, and achieve high-quality standards for the components to be used. Each shipyard can organize its production differently, depending on the type of ship to be built and the decisions concerning outsourced activities.

Despite the diversity of the output and the product's complexity, shipbuilding production processes are generally similar. A (possibly) unified framework is depicted in Figure 1.6, which displays the three main phases we can distinguish within the shipbuilding supply chain, i.e., *pre-production*, *production*, and *post-production* phases. The pre-production phase includes the steps of design and project management. The production phase comprises the hull construction and equipment/systems purchasing, then assembling and system integration. Finally, post-production concerns the so-called *In-Service Support (ISS)*, mainly involving customer support, repair and maintenance activities. As a ship reaches the end of its service life, which for commercial vessels is estimated in about 25 years, they are disassembled (namely, "*ship breaking*"), and recycling/disposal occurs. We next describe these phases in detail.

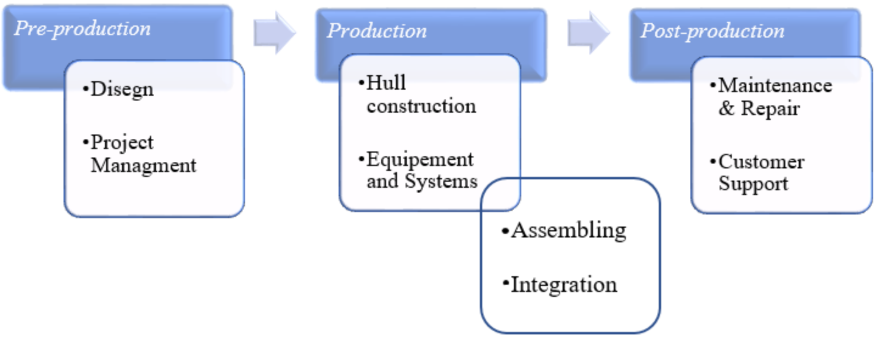


Figure 1.6 – The shipbuilding supply chain

- *Pre-production*

This phase focuses on the product's design and involves various stakeholders. The design process begins with the definition of the requirements by the shipowner. Based on these indications, the shipbuilder, supported by its architects, can define the ship's parameters and characteristics, such as technical features (e.g., engines) and the hull's layout. A thorough engineering analysis then follows, dealing with the study of the ship's structure, noise and vibration, weight and stability. In the pre-production phase, contracts with various suppliers are also set up. The necessary materials (mainly steel slabs and profiles, paints and pipes) are purchased, together with essential (motors, electrical and electronic systems) and auxiliary components (furniture, special equipment and devices).

- *Production*

This phase concerns the processes and activities carried out for the realization of the final product. The main elements and systems involved in this phase are the hull, the standard and ship-specific systems (Brun and Frederick, 2017). Hulls are built in sections called blocks, whose primary raw material is steel. Steel slabs and profiles are cut, straightened, shaped, and welded together, usually by specialized suppliers, to fabricate the hull's subassembly.

Standard systems include ship operation equipment, basic accommodations, electrical systems/plant and electronic navigation and communication systems, auxiliary systems and environmental pollution control.

Ship-specific systems depend on the vessel's purpose. For example, in large commercial carriers, the propulsion system is the most important because it aims to move the ship as quickly and efficiently as possible for long distances. On the contrary, accommodations (e.g. furniture) are more critical in cruise ships and passenger's vessels.

The main activities in assembly and integration can be summarized as follows:

- Hull blocking and assembly: hull subassemblies are coated with primer and other special marine coatings, welded together to form large units, and welded into position to form the ship. Once assembled, the ship is ready for launch and outfitting.
- Outfitting: after the launch, the ship is berthed for completion. The main pieces of machinery (piping systems, deck gear, lifeboats, accommodation equipment, insulation, rigging and deck coverings) are installed in this step.
- Systems integration: systems integrators install the ship-specific systems and ensure subsystems' cross-functionality.

- *Post-production*

Post-production services involve all the *In-Service Support* (*ISS*), i.e., mainly maintenance and repairing activities, and technical training. ISS are generally planned in predefined time windows and are required by the aforementioned "classification societies" (see Section 1.4) to evaluate the ships' dynamic condition. The shipowner is responsible for implementing the ISS, which is typically performed by the original shipbuilder or specialized service providers. Technical training is needed to instruct and update personnel on the systems' operational functionalities and maintenance.

Each of the above-described processes has a wide range of activities to be accurately coordinated, which have stimulated the increasing interests of research scholars in studying various logistics problems arising in the shipbuilding supply chain. Frequently investigated topics concern, for instance, storage operations (Fechter et al., 2015), sheets and profiles' cutting (Haessler et al., 1979), and (sub)assembly welding process (Cho et al., 1998; Iwankowicz, 2016; Derakhshan et al., 2018). This work will focus on the hull's production phase, particularly on steel management and cutting and on the subassemblies' composition process. We will detail these aspects in the next chapters, where we will describe them in the context of an Italian manufacturing steel company working in the cruise ship sector.

1.6 Conclusions

This chapter introduced several aspects related to the shipbuilding sector. In particular, we highlighted the main characteristics of the so-called shipbuilding industry, i.e., the specific segment dealing with the process of building new ships. We emphasized the recent trends in the market, the leading players and actors involved and provided a more in-depth description of its most crucial supply chain processes. Finally, we also briefly outlined various research streams that arose analysing the complex shipbuilding industry's logistics.

The latter will be further investigated in the next chapter, where we will describe the case study of an Italian manufacturing steel company working in the cruise ship sector. Specifically, we will narrow our focus around the cutting and subassemblies composition processes, which will be at the core of the developments we discuss in the present thesis.

2 A case study in the shipbuilding industry: Palescandolo S.p.A.

Summary

This chapter presents the case study of a manufacturing company (Palescandolo S.p.A.) operating in the production of components and subassemblies for cruise ships. After introducing its history and mission, we outline its core processes, such as slabs and profile handling, components and subassemblies production. We particularly focus on describing logistic operations and their role within the shipbuilding sector's supply chain. Finally, we identify and analyze in-depth the critical issues (i.e., "hot-spots") characterizing two of its main processes (storage and retrieval).

2.1 A brief introduction to Palescandolo S.p.A.

Palescandolo S.p.A. (PLS) is a company owned by the Palescandolo family, operating in the steel sector since 1939. Over the years, the company has been specializing in the production of slabs, profiles and tubular. The group has rapidly become a leader in Italy in the production and marketing of welded beams for shipbuilding. Since 2018, it has actively managed a naval cutting/assembly centre, which offers its services mainly in the cruise shipbuilding market. This new centre represented the first logistics service provider in the shipbuilding sector, supporting two Italian shipyards' activities. In the following, we will refer to these shipyards as Shipyard1 (SY1) and Shipard2 (SY2) for reasons of a confidentiality agreement. The latter belongs to a big customer, who assured outsourcing a significative quota of its logistics and production processes needs by stipulating some long-term contracts with PLS.

In the first stages of its life, PLS mainly focused on slabs and profiles handling macro-process, thus working as a temporary buffer for the shipyards. In practice, it stocked slabs and profiles and then delivered them according to handling orders weekly released by each shipyard. The efficiency achieved in performing these operations led PLS to extend its activities to manufacturing beams' and subassemblies' components, which are fundamental parts of the ship's hull.

Consequently, PLS enlarged its centre, whose current plant layout is depicted in Figure 2.1. The centre comprises seven spans: in particular, spans A and B are the slab cutting area; span C is dedicated to the subassembly production; span D is devoted to the profiles cutting, while spans E, F and G are assigned to the slabs and profile storage. Figure 2.2. displays the whole PLS production process, which we detail in the next sections.

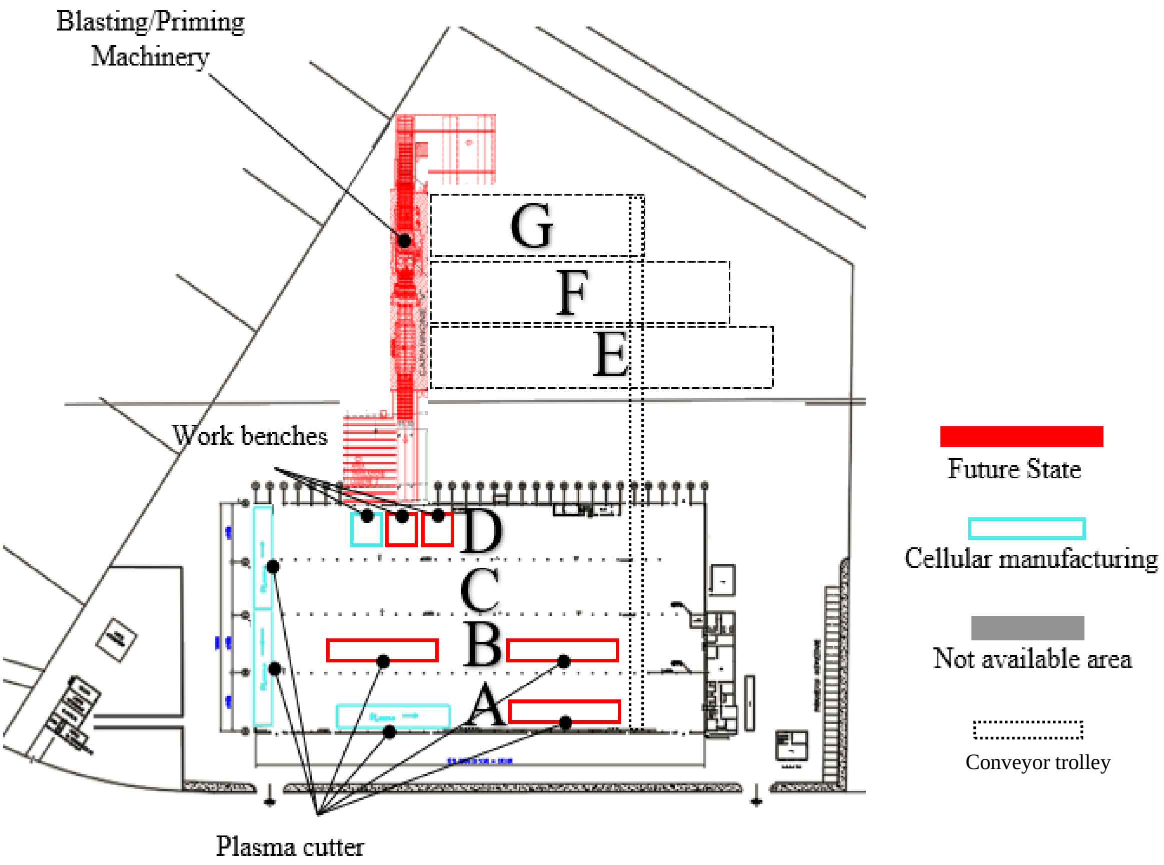


Figure 2.1 – The current PLS' centre layout

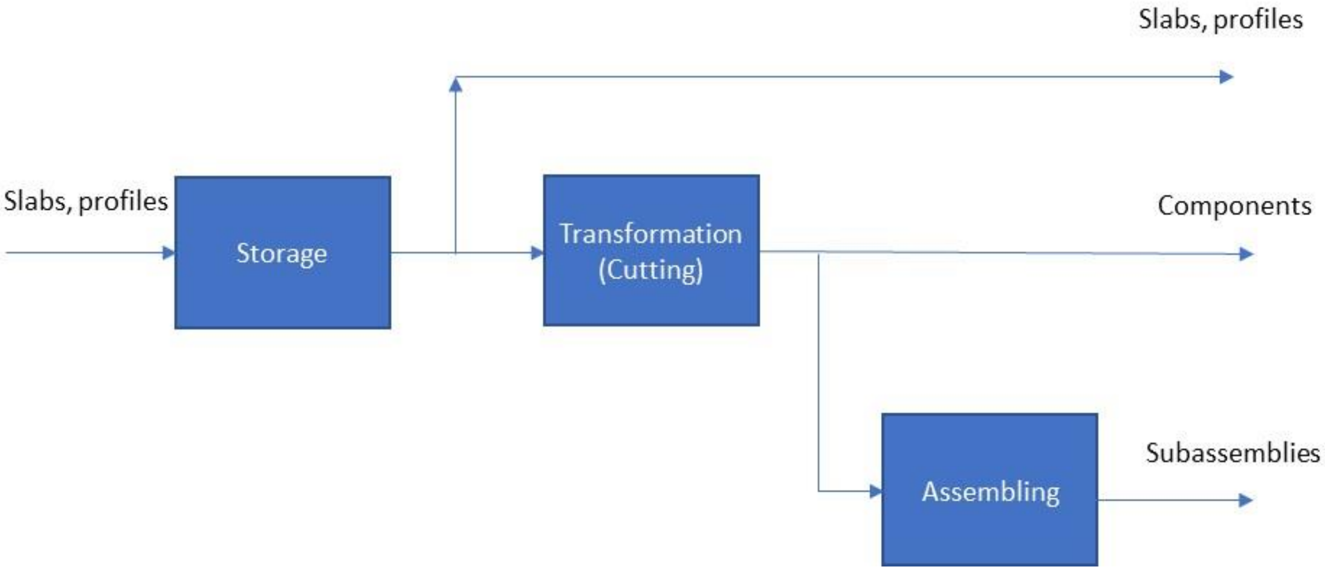


Figure 2.2 – The PLS production process

2.2 The inputs: slabs and profiles

As we anticipated above, slabs and profiles are the primary production process's inputs, supplied to PLS by steel mills based on orders released by the shipbuilder.

A slab is a plate - generally made of mild steel (i.e., with low carbon content) - of rectangular section characterized by a set of attributes, such as length, width, thickness, steel grade. In the "shipbuilding jargon", these four attributes uniquely define an *item*; in other words, all the slabs that belong to the same item present the same values of these attributes. Besides, it is also important to mention that slabs quality must be certified by temporary certification, released by classification societies (see Section 1.4).

The attributes of the slabs can vary in dependence on the specific cruise ship to be built. Table 2.1 indicates typical possible ranges (in mm) for length, width, thickness and weight (in tons), while Figure 2.3 shows an example of a steel slab. From that picture, it is possible to notice the enormous dimensions of these slabs, which dramatically affect the logistic operations, as we will discuss next.

Profiles are elements made of mild steel, with length as the dominant dimension; in addition to the slab attributes (length, width, thickness and steel degree), each item is also characterized by a specific shape. Typical ranges for the profiles' attributes and examples of their shapes are provided in Table 2.2 and Figure 2.4, respectively.

Attribute	Min	Max
Length (mm)	4000	18000
Width (mm)	1500	3000
Thickness (mm)	4	70
Weight (tons)	0,5	13

Table 2.1 – Typical ranges of slabs attributes

Attribute	Min	Max
Length (mm)	6000	18000
Width (mm)	15	400
Thickness (mm)	4	30
Weight (tons)	0,007	3

Table 2.2 – Typical ranges of profiles attributes

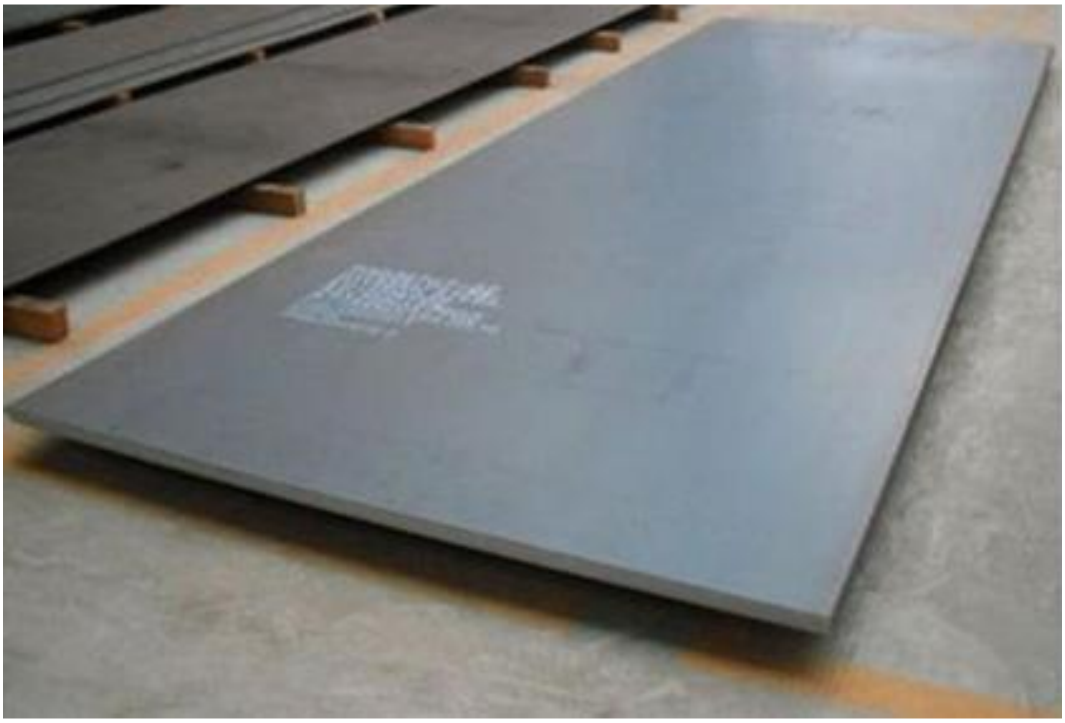


Figure 2.3 – A steel slab

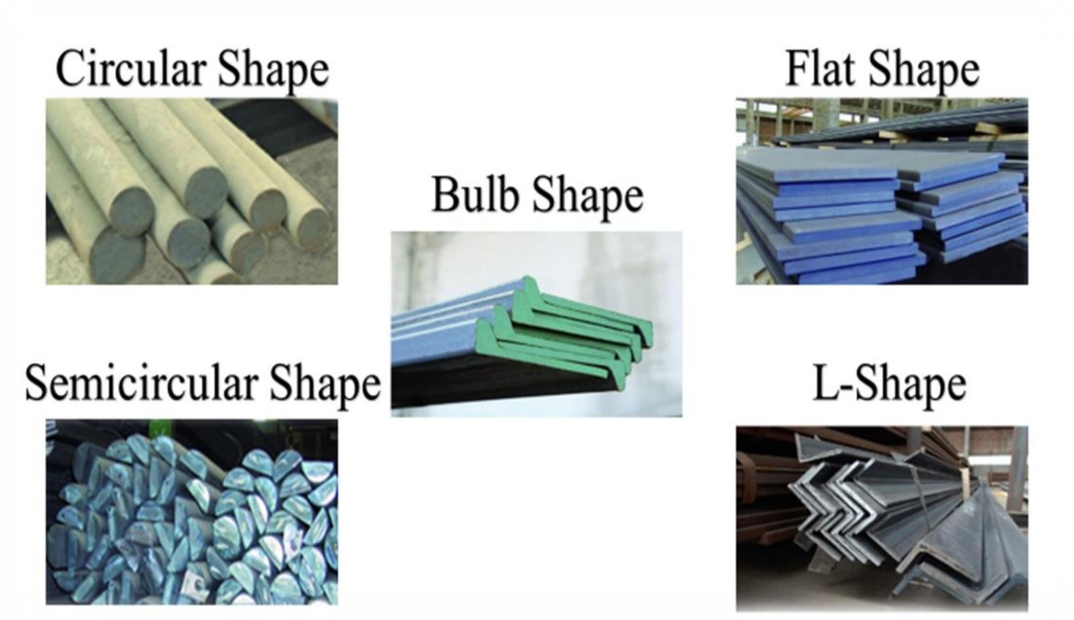


Figure 2.4 – Examples of profiles shapes

2.3 The outputs: slabs and profiles, components, subassemblies

The output of the production process can belong to four different families of items: slabs, profiles, components and subassemblies.

Slabs and profiles that are already produced by steel mills (or by other suppliers) and must be only delivered to shipyards are usually stored at the PLS centre to reduce trips and stocks accumulation at the shipyards. In this case, they do not undergo any processes, and they are stored and handled when delivered.

Components are parts derived from slabs and profiles as a result of cutting operations and eventual additional processes. Usually, from a slab, components of various dimensions and shapes are generated. However, each component is characterized by a given thickness and steel grade. It means that when a specific component has to be produced, it is necessary to use a slab belonging to a specific item. Components are generated according to pre-defined "*cutting schemes*" that the shipbuilder identifies during the design phase to minimize scraps (Figure 2.5). These can be further processed in order to reduce their length and modify their ends.

Subassemblies are items produced through an appropriate assembly of components and represent parts of the ships' hull installed at the shipyards. Their dimensions are generally huge, and, consequently, the related assembly operations are quite challenging (Figure 2.6).

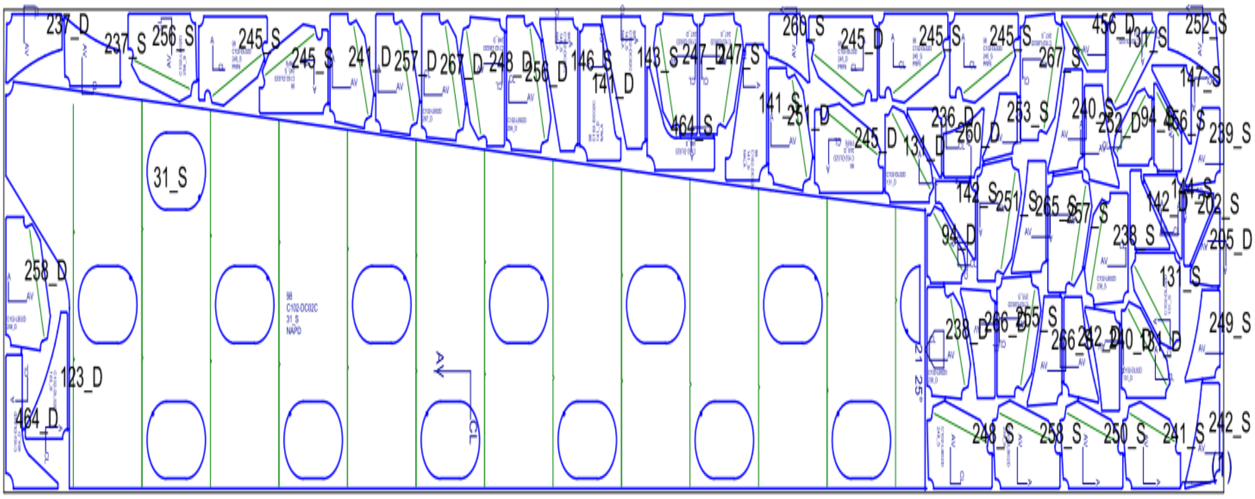


Figure 2.5 – An example of a slab cutting scheme

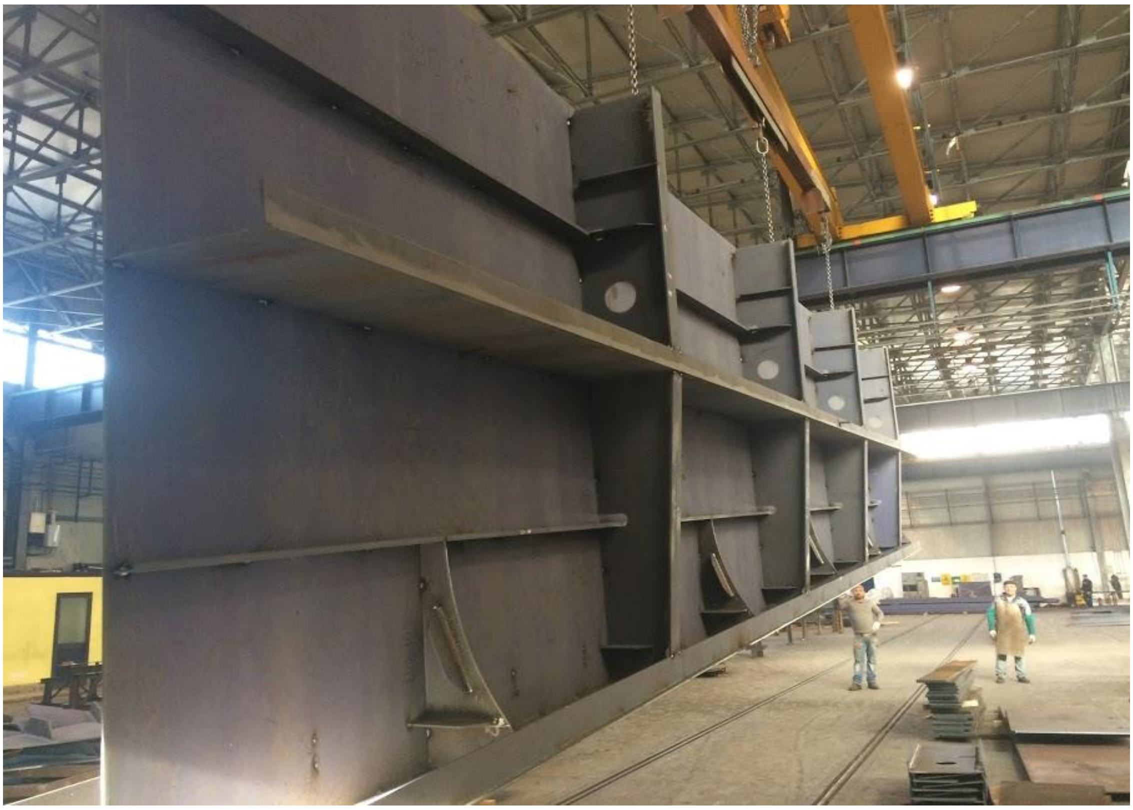


Figure 2.6 – An example of a subassembly

2.4 The cutting process

Cutting is undoubtedly the most relevant operation within the transformation process. Cutting may eventually follow sand-blasting and priming operations. Sand-blasting is a mechanical procedure used to remove oxides, salts, rust, and other materials deposited on the metal surface. It is usually performed before painting or applying protective materials through sand jets oriented on the metal surface to clean it by scraping. Priming is instead a technique that involves applying a thin layer of protective paint (i.e., the primer), which preserves from the oxidizing action of wet air and other aggressive agents.

Two different kinds of cutting procedures are performed, in dependence on the typology of the input material. The cutting of slabs is operated by an automatic machine (plasma cutter) that can be numerically programmed to produce items of given geometries (Figure 2.7). On the other hand, the cutting of the profiles is mainly performed through a small semi-automatic torch as profiles may be reduced in length, holding their own original geometries. This operation is more labour intensive than that performed by the plasma cutter (Figure 2.8).

The obtained components may require additional processing such as shaping, chamfer, press, flange, raking and lowering. Then, some might be not further assembled at PLS and delivered to shipyards or external assembly centres. Figure 2.9 depicts the whole process; we label its main steps by a progressive number (i.e., from 1 to 8) and provide the corresponding description in Table 2.3.

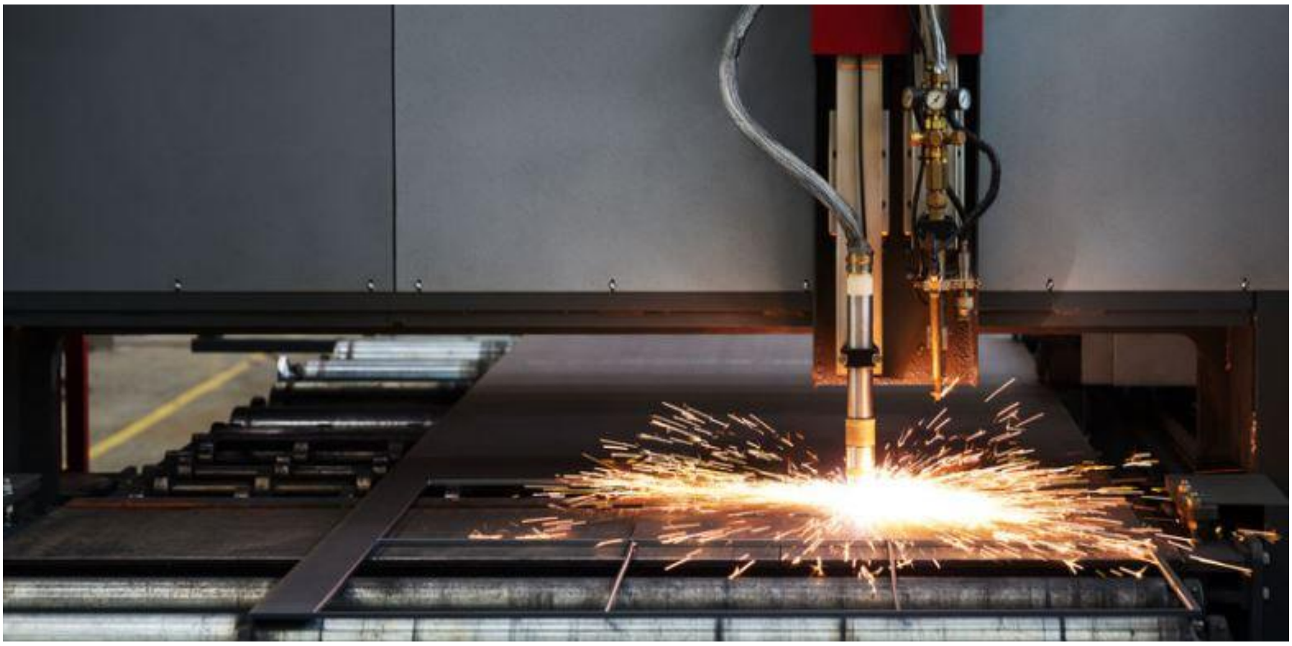


Figure 2.7 – Plasma cutter

Figure 2.8 – The cutting process through a manual torch

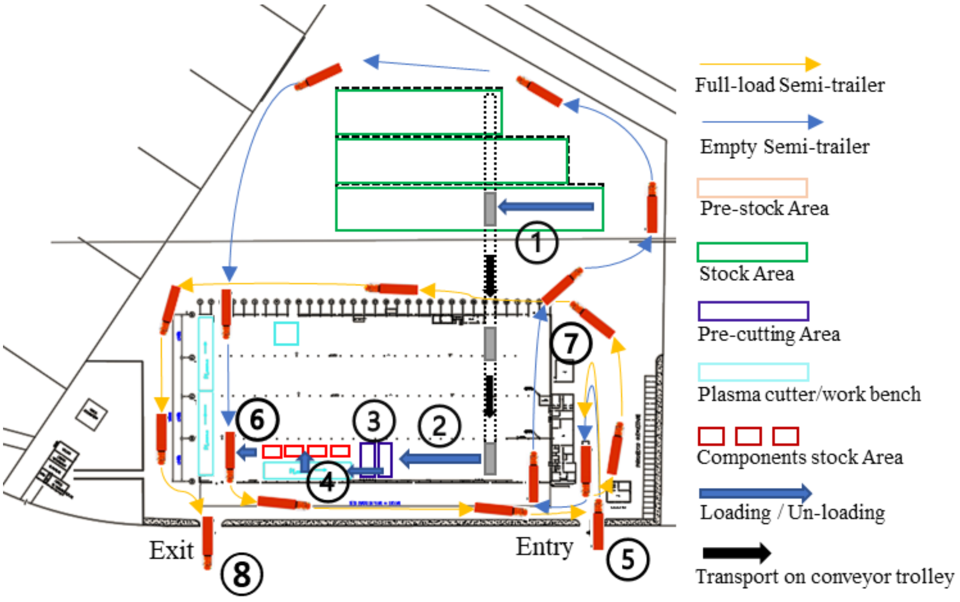


Figure 2.9 - The PLS cutting process

Step	Description
1	An operator picks up the selected slabs and profiles and load them on the conveyor trolley to transport them to the indoor plant.
2	From the conveyor trolley, the slabs and profiles are put in the pre-cutting area.
3	From the pre-cutting area, slabs and profiles are loaded on plasma cutters and workbenches, respectively.
4	After cutting, the produced components are stocked on pallets or new stacks, depending on their dimensions, and located in the components stock area.
5	Each empty semi-trailer is weighed to measure its tara weight once entered into the centre.
6	Semi-trailers reach the components stocking area for the delivery, where components are loaded.
7	Full-loaded semi-trailers are weighted again. Then, the delivery documents are filled in.
8	Semi-trailers leave the centre towards the assigned shipyard.

Table 2.3 – Description of the main steps of the PLS cutting process (depicted in Figure 2.9)

2.5 The assembling process

As we already underlined, part of the components produced in the cutting phase (and the subsequent operations) are used to produce subassemblies. Essentially, assembling consists of welding slabs' and profiles' components according to a given *scheme*, i.e., a predefined design of the subassembly (Figure 2.10). This phase is very labour-intensive due to the uniqueness and peculiarities of the final products. We should also highlight that their enormous dimensions render this operation particularly difficult to manage (Figure 2.11). Once produced, subassemblies are delivered to shipyards to finalize the ship hull production.

In line with what we did in the previous section, Figure 2.12 depicts the process, whose main steps are detailed in Table 2.4.

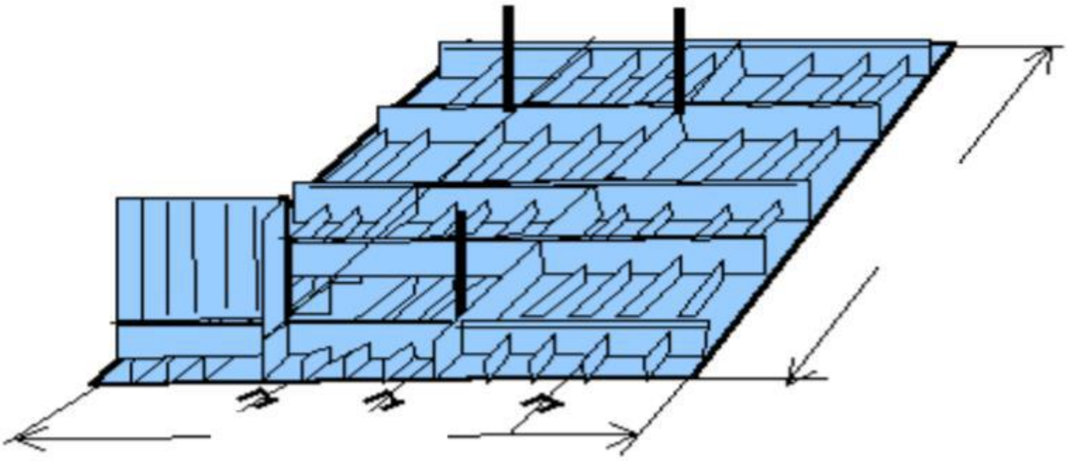


Figure 2.10 – Scheme of a subassembly

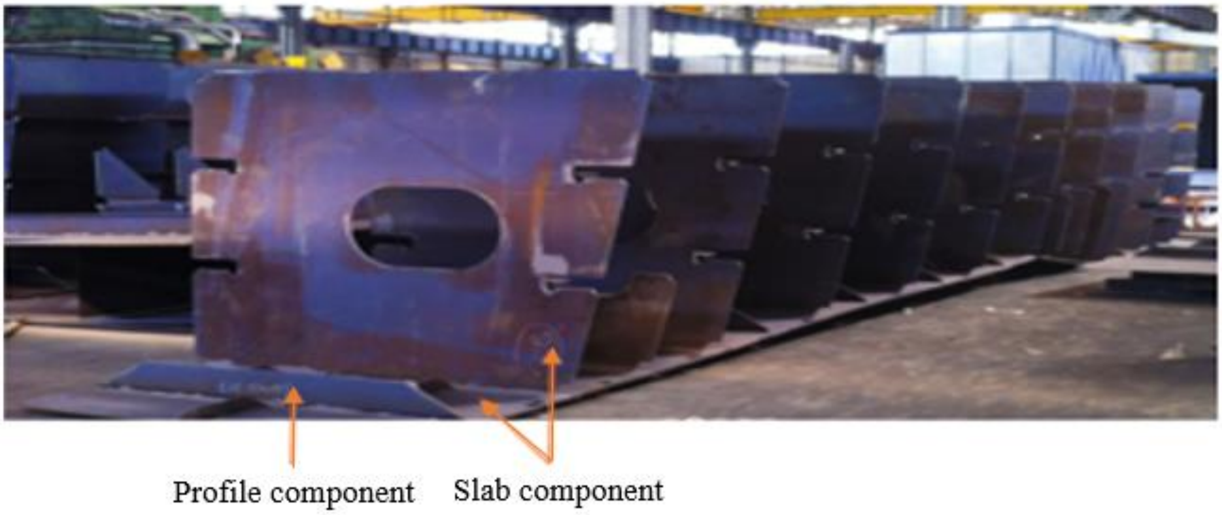


Figure 2.11 – An example of a subassembly

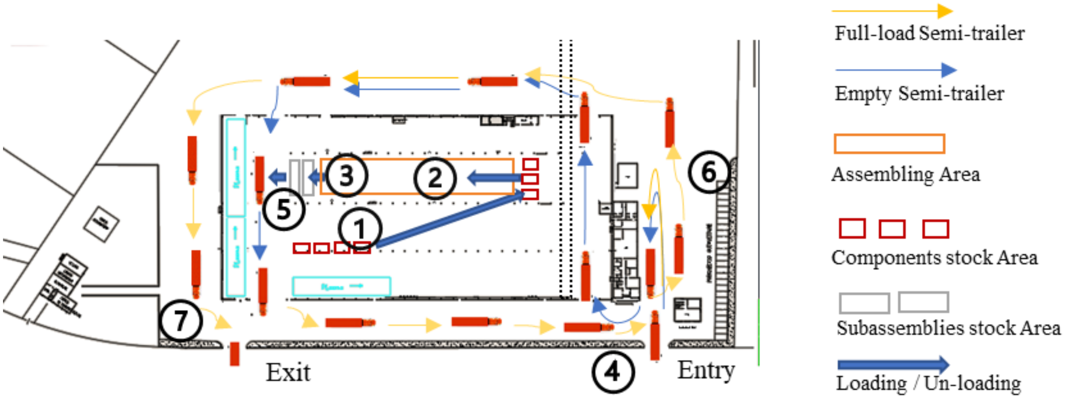


Figure 2.12 - The PLS assembling process

Step	Description
1	Components are moved to the new stock area in the subassemblies span.
2	All the additional operations on components, included welding, are performed in the subassemblies span.
3	Once produced, subassemblies are moved to a dedicated stock area located on the exit side.
4	Each empty semi-trailer is weighed to measure its tara weight once entered into the centre.
5	Semi-trailers reach the subassemblies stocking area for the delivery, where subassemblies are loaded.
6	Full-loaded semi-trailers are weighted again. Then, the delivery documents are filled in.
7	Semi-trailers leave the centre towards the assigned shipyard.

Table 2.4 – Description of the main steps of the PLS assembling process (depicted in Figure 2.12)

2.6 The slabs storage and retrieval processes

The characteristics of the considered materials (i.e., slabs, profiles, components, subassemblies) render the logistic operations very sector-specific and crucial. In particular, their dimensions require a peculiar organization of the storage areas and laborious procedures supported by specific industrial tools. As a result, long times are needed to perform even basic handling operations.

In the following, we detail two main logistic processes, i.e., slabs and profiles storage and retrieval.

➢ *The storage process*

After a formal check-in, the inbound slabs and profiles, delivered by steel mills or suppliers through semi-trailers, are stored in dedicated areas divided into spans and, in turn, into pitches (Figure 2.13).

There, slabs grouped in “*stacks*” with a maximum height of about two meters. We can distinguish between *dedicated* or *random* stacks, whether they host one or more types of slabs, respectively. Considering that the thickness can vary from four to 70 mm, each stage can contain up to 300 slabs; an example of a stack is shown in Figure 2.14.

Slabs are moved by industrial magnetic cranes (Figure 2.15) capable, considering their weight, of lifting and relocating slabs along the span.

On the other hand, profiles are delivered in "*packages*", coming from the same steel casting, and are stored in specific "stalls" (Figure 2.16) designed according to their sizes and shapes. Profiles are moved using lifting beams (Figure 2.17).

The storage process is detailed in Figure 2.18 and Table 2.5.

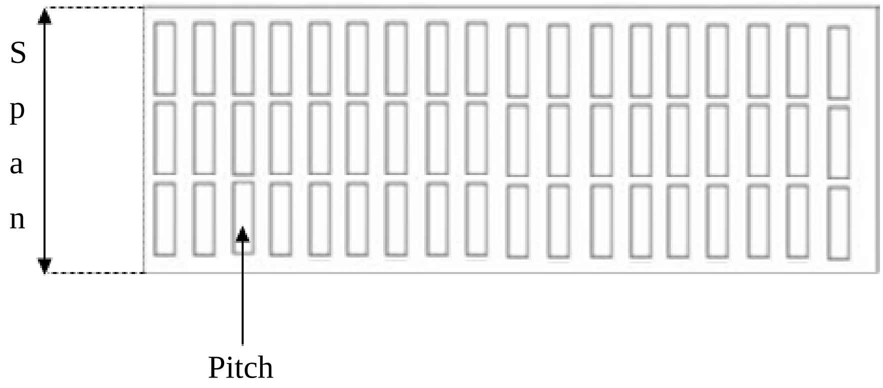


Figure 2.13 – Layout of a span for slabs storage

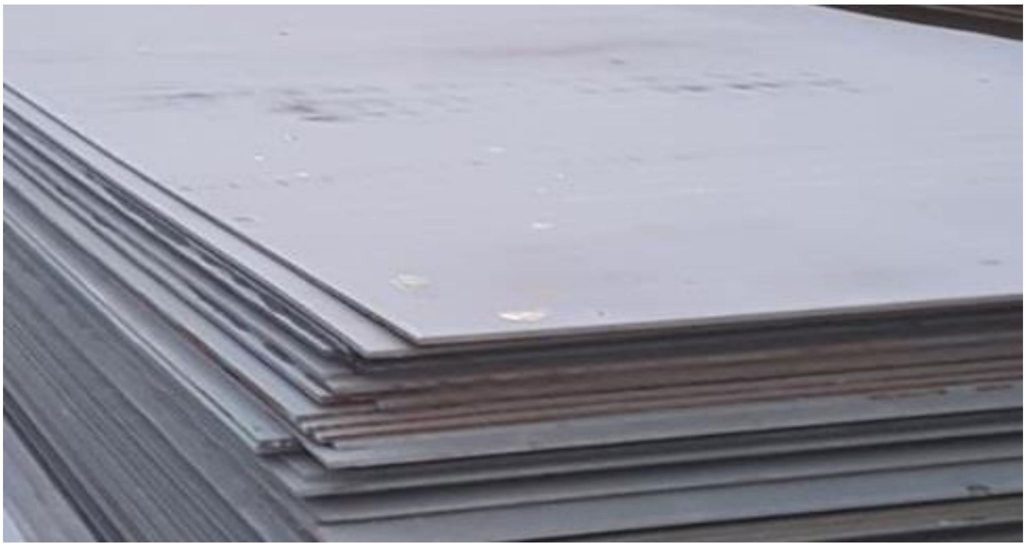


Figure 2.14 – An example of a slabs stack

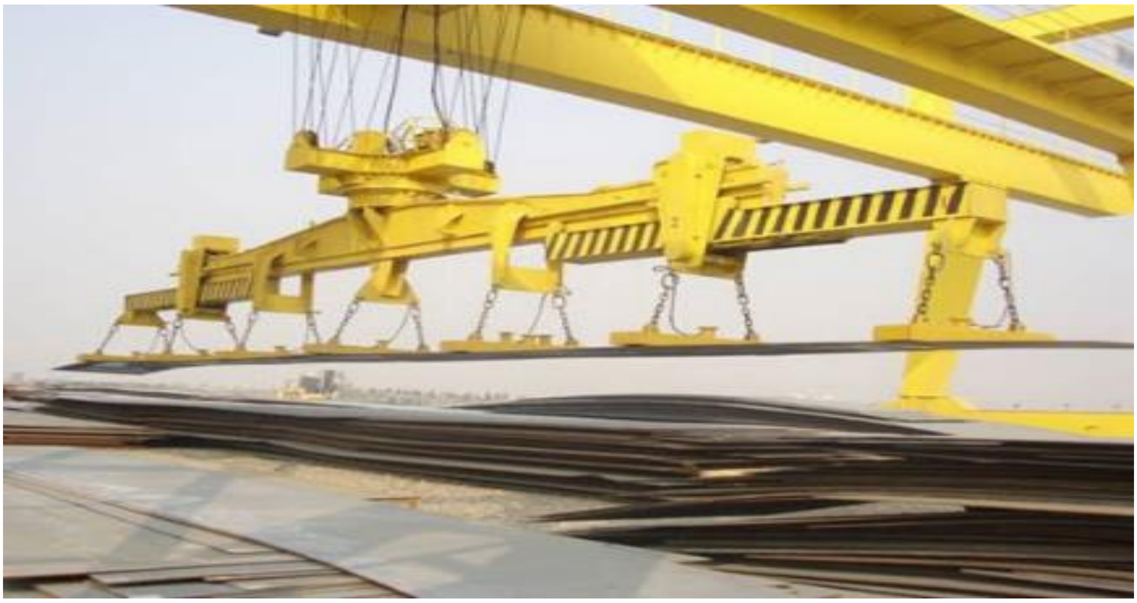


Figure 2.15 – Industrial magnetic crane

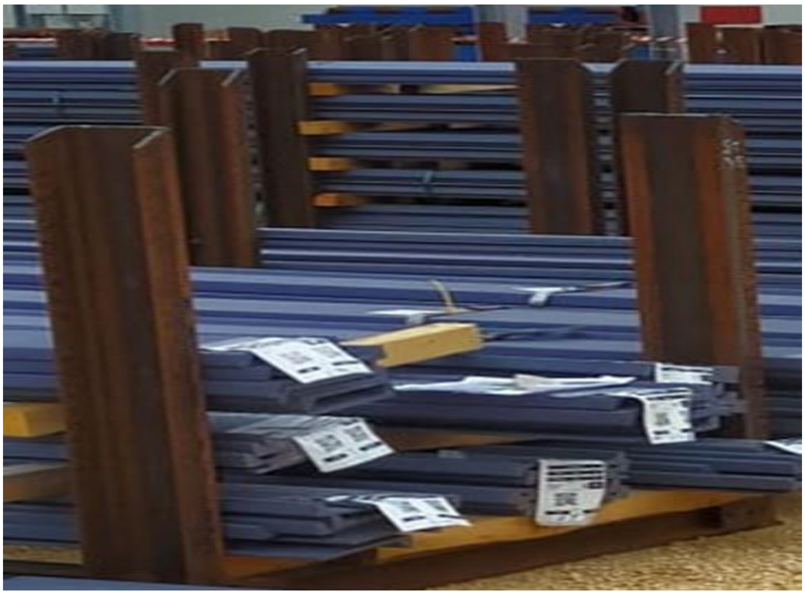


Figure 2.16 – “Stalls” of profiles

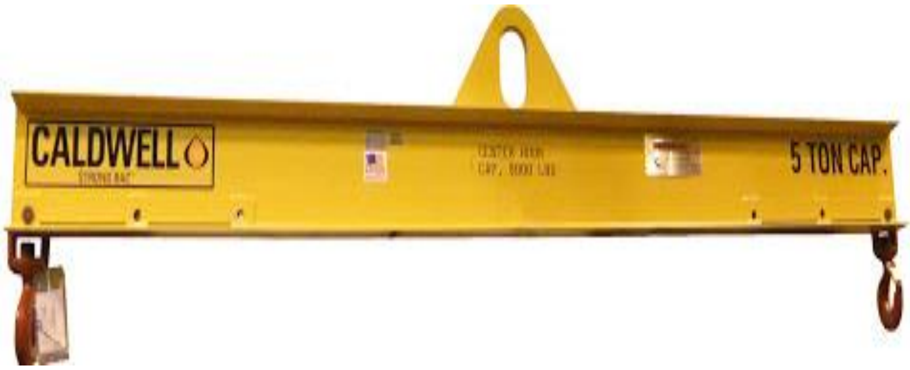


Figure 2.17 - Lifting beam

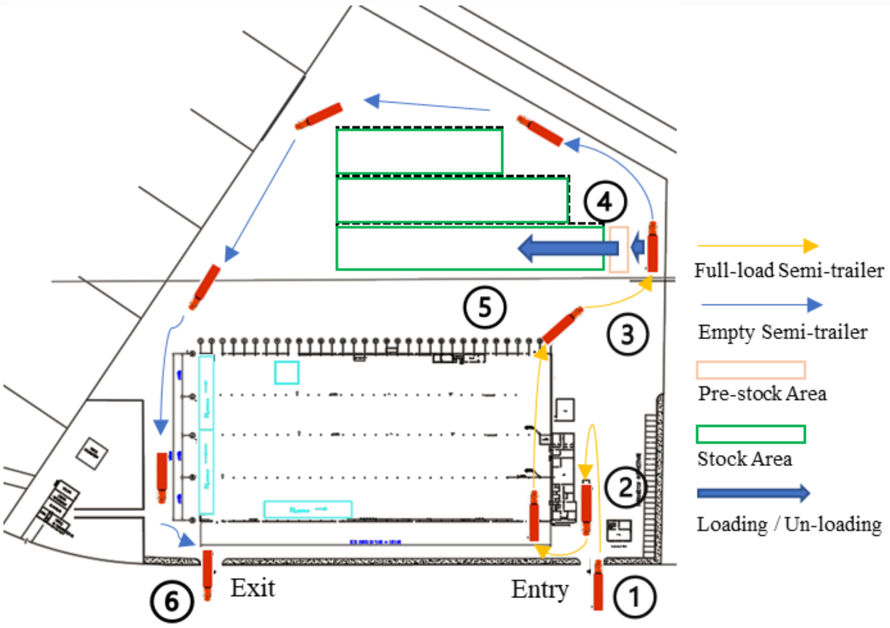


Figure 2.18 - The PLS assembling process

Step	Description
1	Semi-trailers enter the centre.
2	Semi-trailers are weighted, and a PLS operator collects the delivery note.
3	Once registered, a copy of the delivery note is given to another operator who will handle the material's stock. Meanwhile, semi-trailers reach the pre-stock area near the stock area, on the entry side.
4	Slabs and profiles are unloaded and moved by an industrial magnetic crane or a lifting beam, respectively. Both the machines move in a parallel direction to the spans. In the meantime, a PLS operator checks the information reported on the delivery note. In case of a negative check, the operator communicates the event, and a complaint procedure to the supplier is started.
5	In case of a positive check, materials are labelled and definitively stocked in the stock area.
6	The empty semi-trailers leave the plant.

Table 2.5 – Description of the main steps of the PLS storage process (depicted in Figure 2.18)

➢ *The retrieval process*

Based on production orders, materials have to be moved towards different destinations, such as the cutting, assembly, or exit areas. To this end, as a given output to be produced is associated with a specific slab item, a retrieval process requires a first step consisting of the selection, among the available stacks and the suitable slabs, of a slab belonging to a given item. Once identified the slab with a given position in a stack (*target slab*), in order to pick it up, all the slabs positioned over it should be shifted and reallocated, temporarily or permanently, on the top of other stacks (Figure 2.19). Each of these operations is denoted as a "*shuffle*". Figure 2.20 shows the retrieval sequence of a target slab both if the moved slabs are repositioned in the original stack or if their reallocation to another stack is permanent.

These activities are very time-consuming and, therefore, to save time and reduce costs, it is essential to minimize their times (and cranes' shift). A not optimal selection of the slabs to be taken can cause significant delays and an unjustified increase in handling costs, generating further delays in the subsequent logistic operations. For further details on the process, readers can refer to Figure 2.21 and Table 2.6.

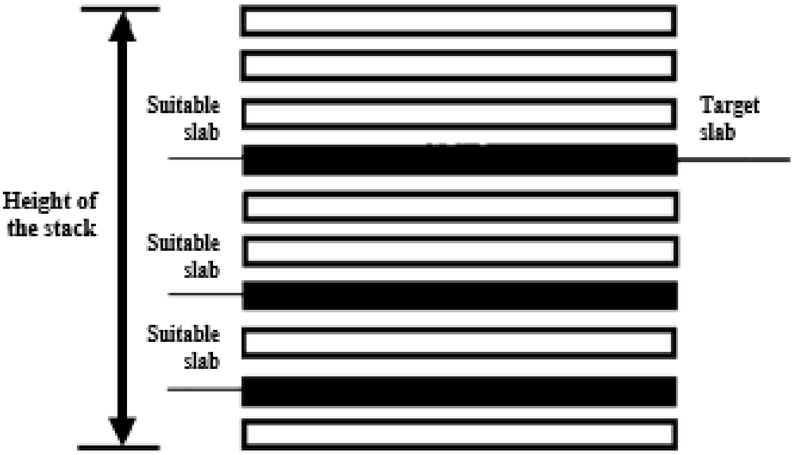


Figure 2.19 – Target slab selection

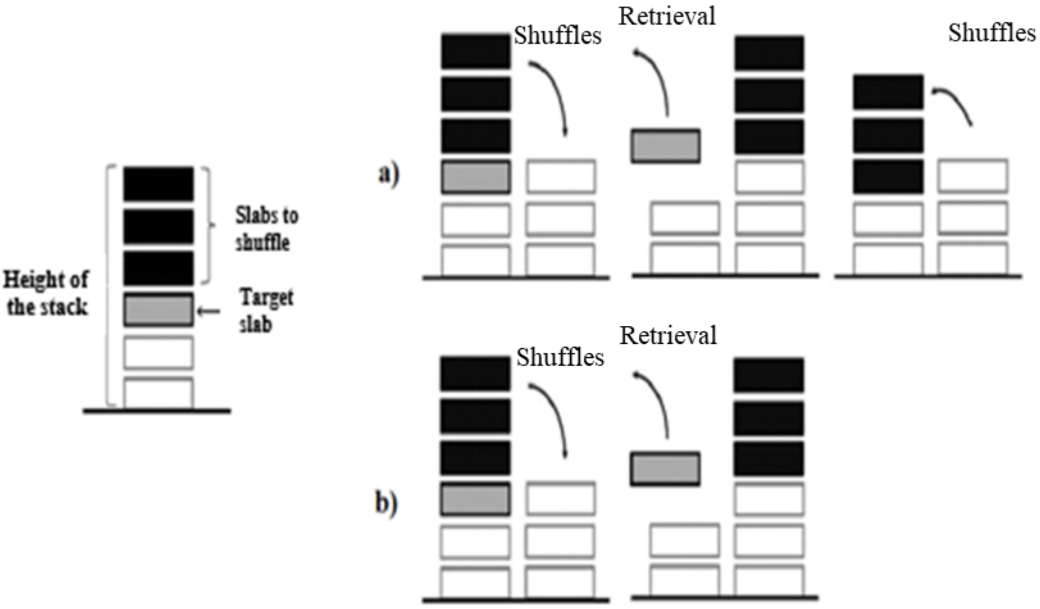


Figure 2.20 – Example of a retrieval sequence with (a) or without (b) slabs repositioning

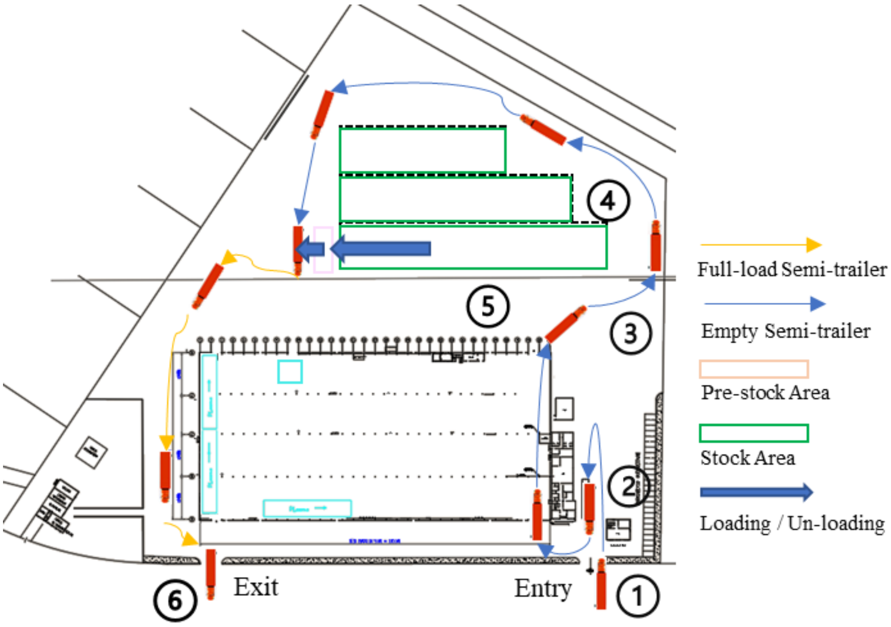


Figure 2.21 - The PLS retrieval process

Step	Description
1	An operator picks up slabs and profiles according to the received handling order and moves them to a pre-delivery area near the exit side.
2	Each empty semi-trailer is weighed to measure its tara weight once entered into the centre.
3	Semi-trailers reach the pre-delivery area for the delivery, where slabs or profiles are loaded.
4	Full-loaded semi-trailers are weighted again. Then, the delivery documents are filled in.
5	Semi-trailers leave the centre towards the assigned shipyard.

Table 2.6 – Description of the main steps of the PLS retrieval process (depicted in Figure 2.21)

2.7 Production planning in PLS: a quantitative analysis

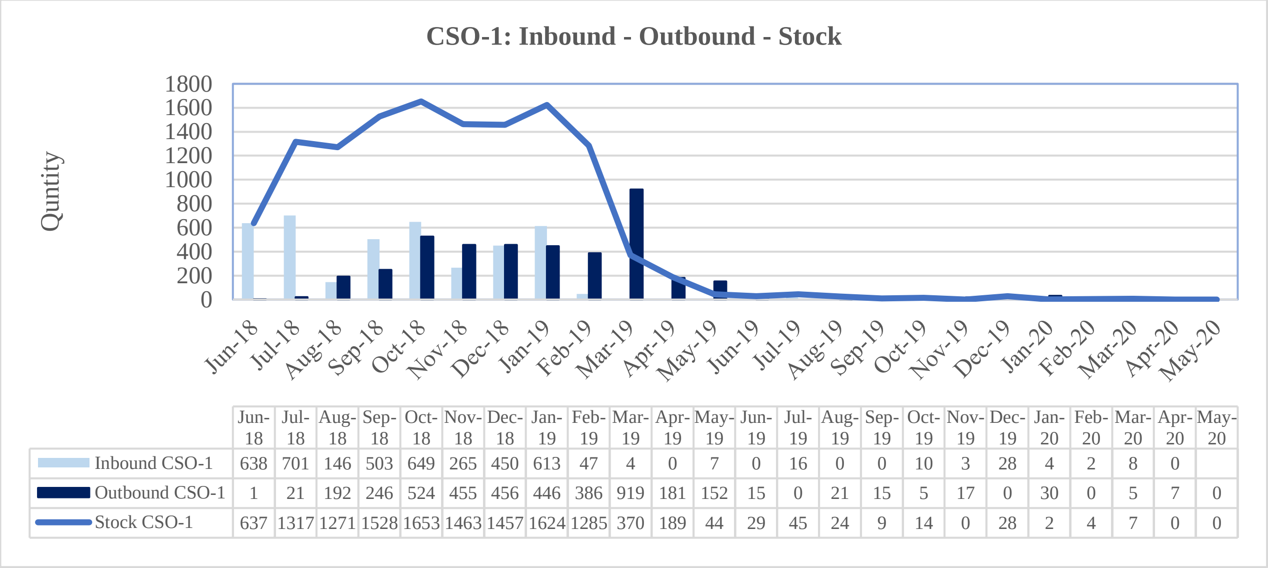
Generally, the collaboration between shipbuilders and cutting/assembly suppliers is regulated by contracts concerning specific ship orders. Specifically, production planning is driven by the so-called Cruise Ship Orders (CSOs), i.e., the Bill Of Materials related to ships production. Usually, PLS processes CSOs related to different ships each year (three, for instance, during 2018).

A single CSO regulates the arrival dates of input materials (slabs and profiles) and the delivery dates to shipyards of components and subassemblies. Suppose we assume the arrival dates as release times and delivery dates as due dates. In that case, the logistic processes should be adequately scheduled to optimize some performance indicators, satisfying logistic and technological constraints.

In the following, we present a quantitative analysis of typical CSOs to provide the reader with an idea of the production planning complexity in this industry, given the variety of the outputs and production volumes.

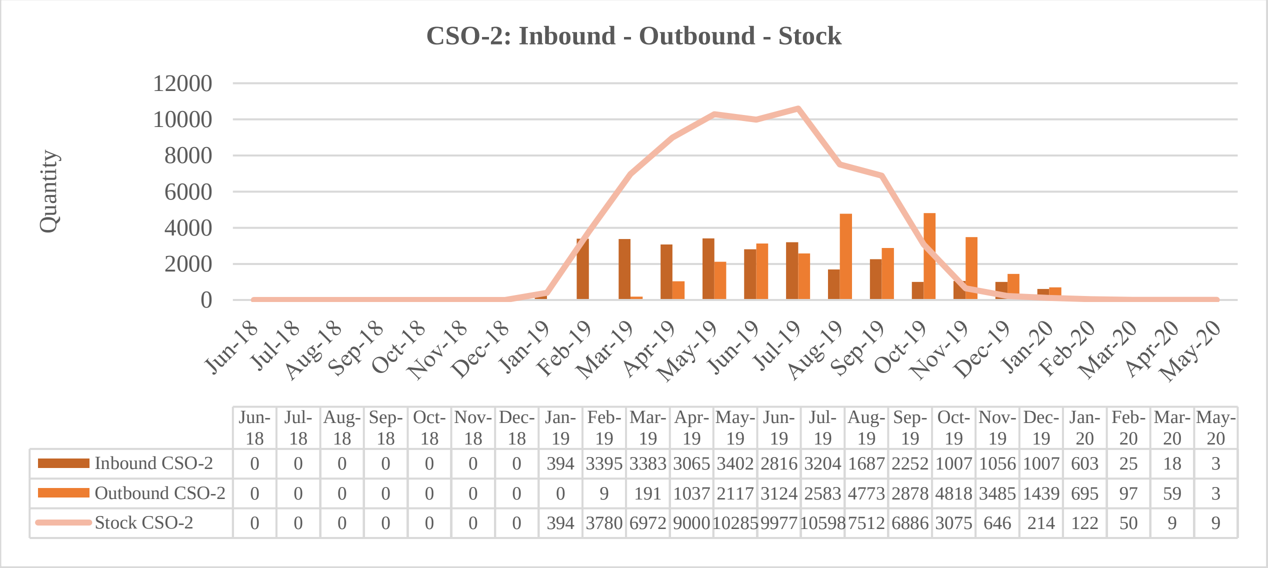
To this end, we analyze three CSOs, from June 2018 until May 2020, representative of the production planned for two whole years.

Figures 2.22, 2.23, and 2.24 display the demand for inbound and outbound slabs as per the three CSOs. We can note that, due to the time lag between the inbound and outbound slabs, especially during the first phase of a CSO management process, a significant peak in the number of stored slabs and profiles occurs. This aspect determines various critical issues. On the one hand, a capacity problem emerges, which is further complicated by the overlapping of the peaks related to different CSOs and the considerable dimension of the material inputs. On the other, this renders the retrieval/storage operations strongly time-consuming. Besides, another critical issue is that a significant portion of slabs and profiles requires cutting operations; this implies the use of expensive resources, e.g., specialized industrial machinery (see Section 2.4) and the involvement of an experienced and dedicated workforce.



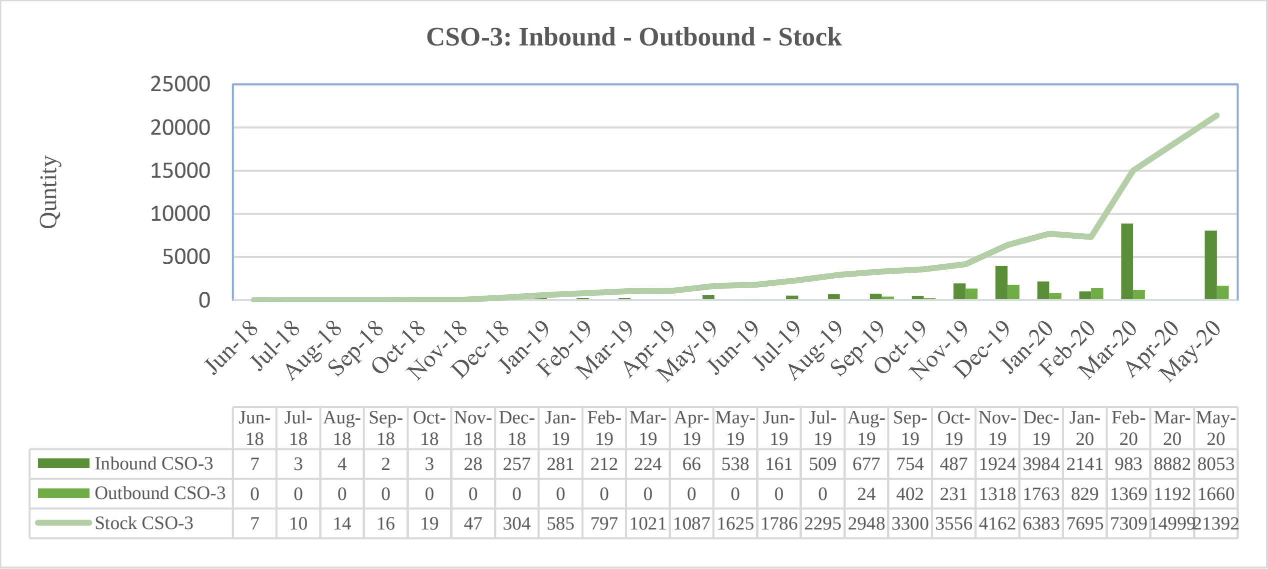
	Jun-18	Jul-18	Aug-18	Sep-18	Oct-18	Nov-18	Dec-18	Jan-19	Feb-19	Mar-19	Apr-19	May-19	Jun-19	Jul-19	Aug-19	Sep-19	Oct-19	Nov-19	Dec-19	Jan-20	Feb-20	Mar-20	Apr-20	May-20
Inbound CSO-1	638	701	146	503	649	265	450	613	47	4	0	7	0	16	0	0	10	3	28	4	2	8	0	
Outbound CSO-1	1	21	192	246	524	455	456	446	386	919	181	152	15	0	21	15	5	17	0	30	0	5	7	0
Stock CSO-1	637	1317	1271	1528	1653	1463	1457	1624	1285	370	189	44	29	45	24	9	14	0	28	2	4	7	0	0

Figure 2.22 - CSO-1: Inbound, Outbound and Stock Quantities



	Jun-18	Jul-18	Aug-18	Sep-18	Oct-18	Nov-18	Dec-18	Jan-19	Feb-19	Mar-19	Apr-19	May-19	Jun-19	Jul-19	Aug-19	Sep-19	Oct-19	Nov-19	Dec-19	Jan-20	Feb-20	Mar-20	Apr-20	May-20
Inbound CSO-2	0	0	0	0	0	0	0	394	3395	3383	3065	3402	2816	3204	1687	2252	1007	1056	1007	603	25	18	3	
Outbound CSO-2	0	0	0	0	0	0	0	0	9	191	1037	2117	3124	2583	4773	2878	4818	3485	1439	695	97	59	3	
Stock CSO-2	0	0	0	0	0	0	0	394	3780	6972	9000	10285	9977	10598	7512	6886	3075	646	214	122	50	9	9	

Figure 2.23 - CSO-2: Inbound, Outbound and Stock Quantities



	Jun-18	Jul-18	Aug-18	Sep-18	Oct-18	Nov-18	Dec-18	Jan-19	Feb-19	Mar-19	Apr-19	May-19	Jun-19	Jul-19	Aug-19	Sep-19	Oct-19	Nov-19	Dec-19	Jan-20	Feb-20	Mar-20	Apr-20	May-20
Inbound CSO-3	7	3	4	2	3	28	257	281	212	224	66	538	161	509	677	754	487	1924	3984	2141	983	8882	8053	
Outbound CSO-3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	402	231	1318	1763	829	1369	1192	1660	
Stock CSO-3	7	10	14	16	19	47	304	585	797	1021	1087	1625	1786	2295	2948	3300	3556	4162	6383	7695	7309	14999	21392	

Figure 2.24 - CSO-3: Inbound, Outbound and Stock Quantities

In Table 2.7, we report the main characteristics of the orders, for slabs, to underline the variety of the input materials. We recall that slabs characterized by the same values of length, width, thickness, and steel grade belong to the same item (see Section 2.4).

	CSO-1	**CSO-2**	**CSO-3**
Number of different items	187	257	345
Number of slabs	4094	6389	10924
Range of length [mm]	10000-15400	4000-16700	4000-15500
Range of width [mm]	2250-3000	1700-3000	1500-3000
Range of thickness [mm]	4-40	4-70	4-40
Percentage of slabs to be cut	45%	88%	98%

Table 2.7 – Characteristics of the slabs in the considered CSOs

It is possible to notice that the number of items and slabs and the percentage of slabs to be cut increase over time (i.e., from CSO-1 to-CSO 3), thus increasing the deriving workload and the corresponding logistic costs.

Figures 2.25, 2.26 and 2.27 show, for each CSO, the distribution of the number of slabs by length, width, and thickness, respectively. In those pictures, slabs are also grouped by item. In practice, if we focus on length (i.e., Figure 2.25), each point represents an item belonging to a specific CSO, comprising a given number of slabs (on the y-axis) of a given length (on the x-axis). Hence, this picture also informs readers of the number of items (corresponding to the number of points) with a given length. The same applies to width and thickness.

To better clarify these aspects, Figures 2.28, 2.29 and 2.30 display, for each CSO, the relative percentages of slabs and items with a given length, width and thickness. For instance, by looking at Figure 2.28 and focusing on CSO-1, we note that about 30% of slabs and items have a length of 11500 mm (the cyan and blue bars, respectively).

Regarding the length, as one can see from Figures 2.25 and 2.28, many items are concentrated in just a few values (e.g., 11500 mm, 12500 mm, and 14600 mm) for all the CSOs. We also observe that these few "classes" of items comprise almost all the slabs required in each CSO. Moreover, we also note that, for some lengths, the number of slabs is very low, despite a non-negligible number of items. For instance, these circumstances occur in the cases of 15400 mm for CSO-1 and CSO-2 or 11500 mm for CSO-3.

Similar considerations can be drawn if we focus on width (see Figures 2.26 and 2.29).

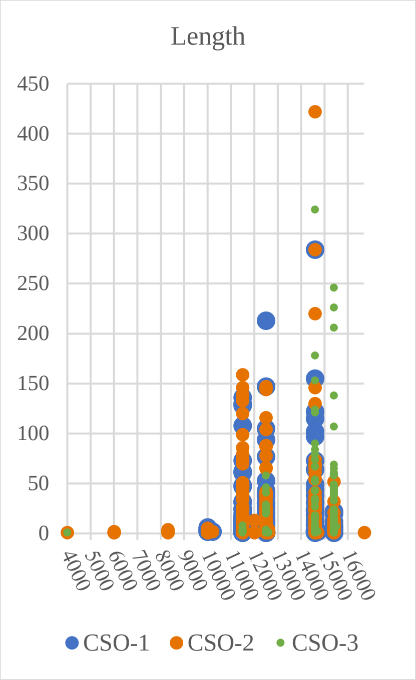


Figure 2.25 – Distribution of the number of slabs (grouped by item) by the length (in mm)

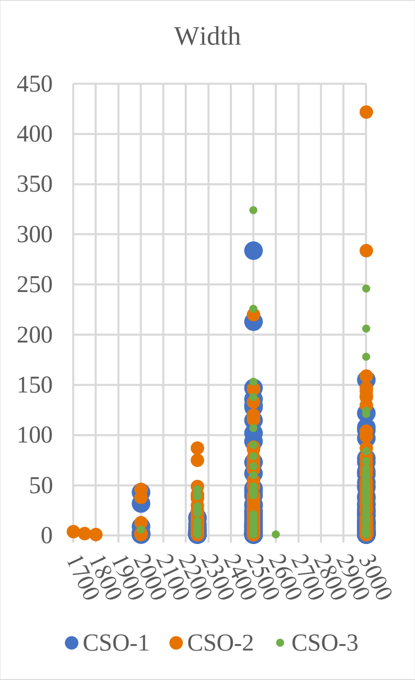


Figure 2.26 - Distribution of the number of slabs (grouped by item) by the width (in mm)

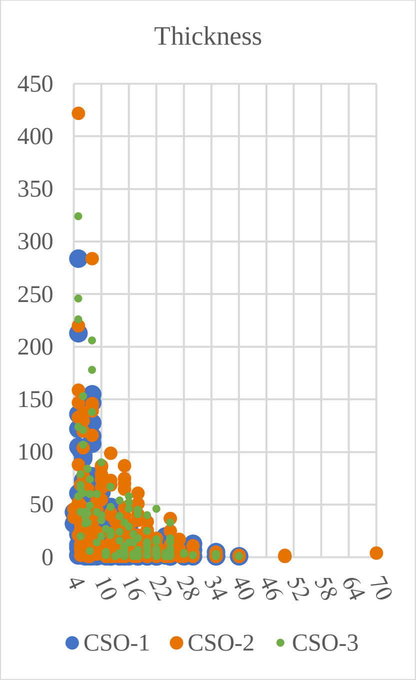


Figure 2.27 - Distribution of the number of slabs (grouped by item) by the thickness (in mm)

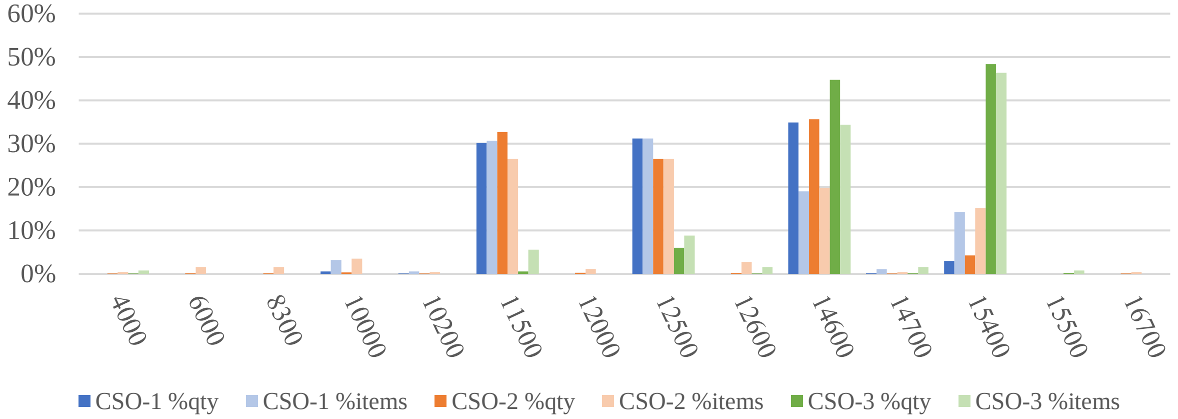


Figure 2.28 – Distribution of the number of items and slabs (in %) by the length (in mm)

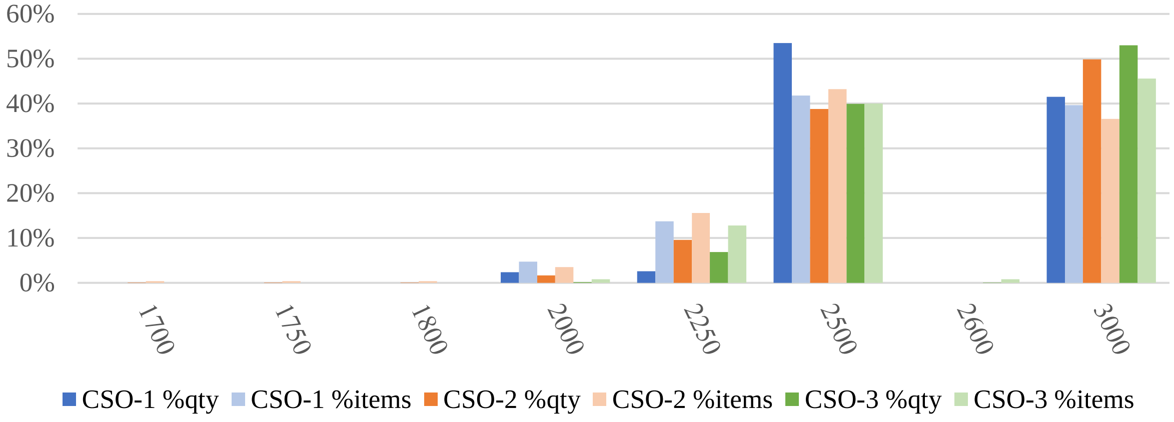


Figure 2.29 - Distribution of the number of items and slabs (in %) by the width (in mm)

If we focus on thickness, the distribution appears significantly different from the other cases. Specifically, we note a higher variability, i.e., the presence of more typologies of items of different thickness. In terms of slabs, we observe that they are mainly concentrated around some particular thickness values. The result is that, in practice, PLS manages many items characterized by low amounts of slabs. As we already emphasized, this aspect also emerged, although less evident, when analysing width and length.

This variety poses severe issues, both for the storage and retrieval processes. For instance, thinner slabs can be accidentally picked-up together with thicker slabs if stacked consecutively, or slabs with higher length may tend to bend if stacked upon shorter ones. Besides, it is also intuitive that larger slabs occupy larger volumes/surfaces. A more in-depth discussion on these problems follows in the next section.

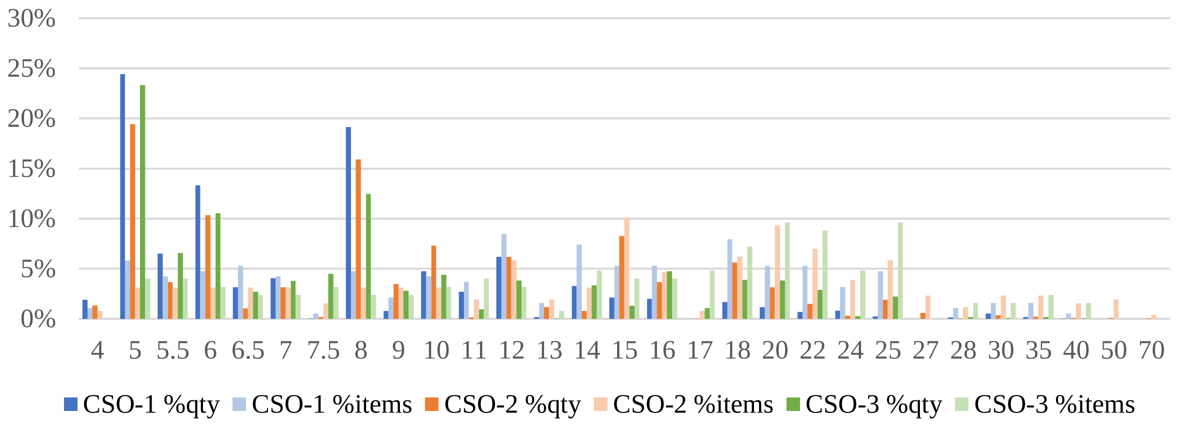


Figure 2.30 - Distribution of the number of items and slabs (in %) by the thickness (in mm)

We now focus our attention on the other input managed by PLS, i.e., profiles. We recall that, differently from slabs, in addition to length, width, thickness, and steel grade, profiles are also characterized by a given shape. Table 2.8 reports the details of the profiles included in the considered CSOs. Note that they are not present in CSO-1. As for slabs, we underline an increase in the number of items and profiles managed and in the number of profiles to be cut (in absolute terms).

	CSO-2	CSO-3
Number of different items	185	252
Number of profiles	20990	30977
Range of length	6000-18000	6000-16000
Range of width	16-320	16-350
Range of thickness	≈0-30	≈0-30
Types of shapes	Flat, L1, Circular, Semi-circular, L2, Bulb	Flat, L1, Circular, Semi-circular, L2, Bulb
Percentage of profiles to be cut	46%	43%

*L1 = “L” shape with equal sides; *L2 = “L” shape with different sides

Table 2.8 – Characteristics of the profiles in the considered CSOs

The analysis summarized next resembles that performed for slabs. Accordingly, Figures 2.31, 2.32 and 2.33 show, for all the CSOs, the distribution of the number of profiles (grouped by item) by length, width, and thickness, while Figures 2.34, 2.35 and 2.36 depict the corresponding percentages. Briefly, we can note that, while in terms of length, there is an evident concentration of items and profiles around some specific values (see Figure 2.34), a more homogenous distribution is found when focusing on width and thickness (although with some local peaks, see Figures 2.35 and 2.36). These results are similar to those gained in the case of slabs.

Therefore, we can ultimately conclude that the management of both types of inputs (i.e., slabs and profiles) raises problems of practical and economic relevance that are worthwhile investigating.

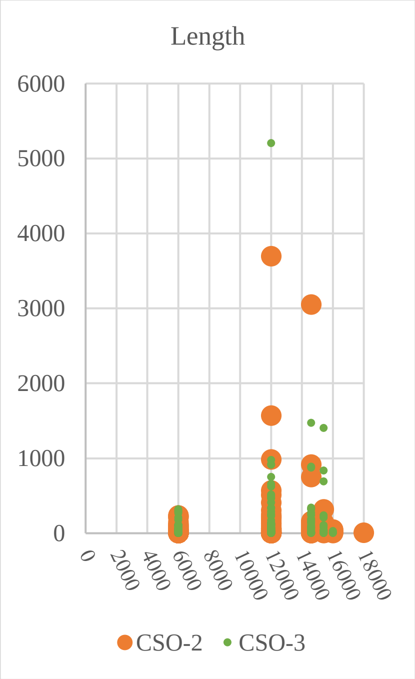


Figure 2.31 - Distribution of the number of profiles (grouped by item) by the length (in mm)

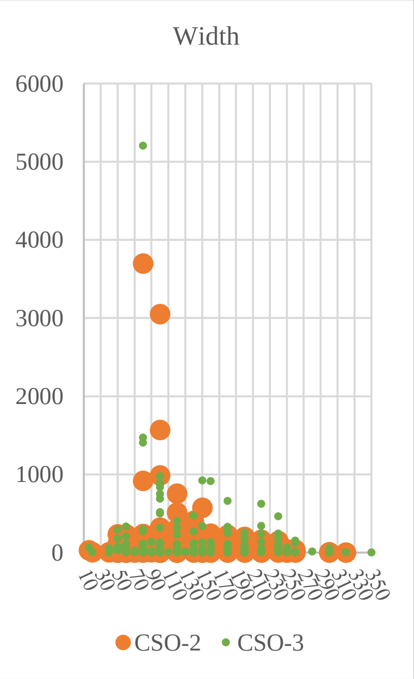


Figure 2.32 - Distribution of the number of profiles (grouped by item) by the width (in mm)

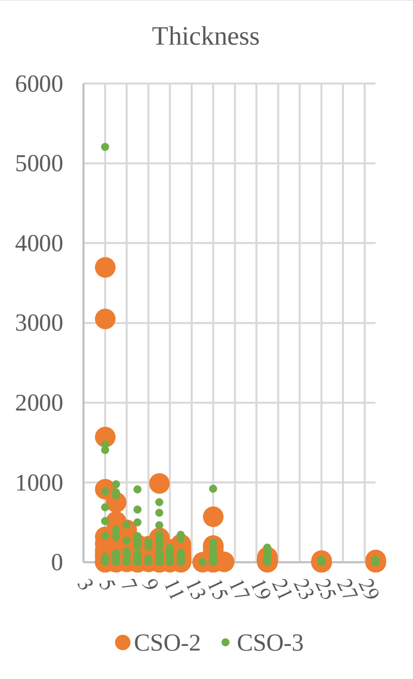


Figure 2.33 - Distribution of the number of profiles (grouped by item) by the thickness (in mm)

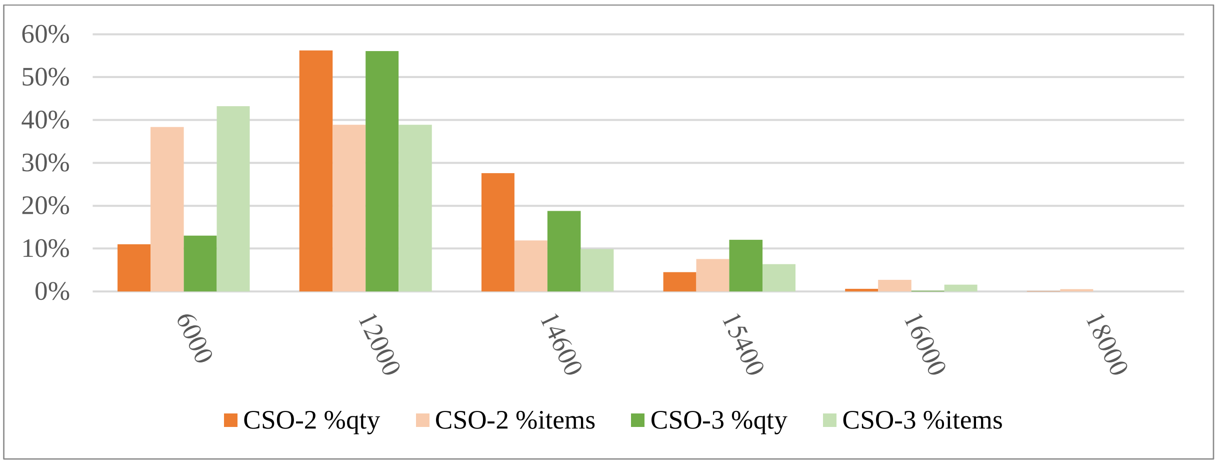


Figure 2.34 – Distribution of the number of items and profiles (in %) by the length (in mm)

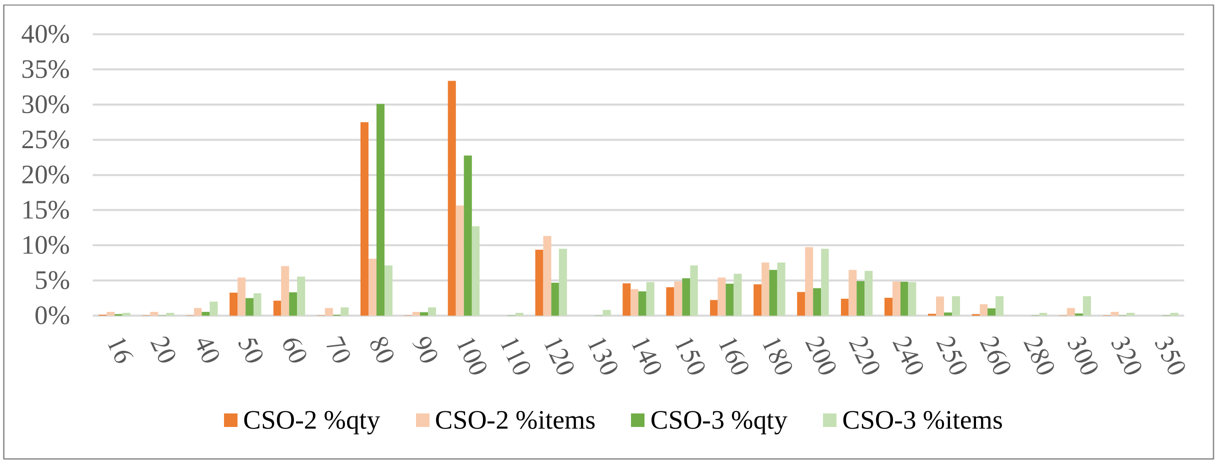


Figure 2.35 - Distribution of the number of items and profiles (in %) by the width (in mm)

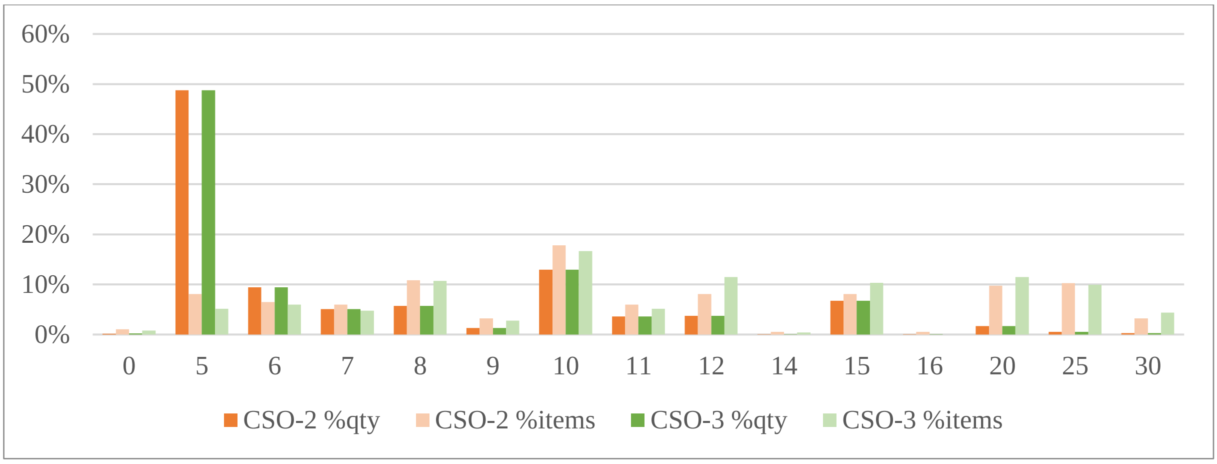


Figure 2.36 - Distribution of the number of items and profiles (in %) by the thickness (in mm)

2.8 Hot-spots identification

After the description and the quantitative analysis of the processes characterizing the PLS supply chain, it emerges that storage and retrieval operations are the most critical ones. These two processes precede all the others and involve managing large amounts of products simultaneously. Indeed, everything starts by retrieving slabs or profiles from their storage areas, whatever their subsequent "flow" might be. Storage areas are the "meeting points" between the continuous production of slabs and profiles coming from the steel mill and the batch production of cutting and assembly centres, driven by the shipyards' orders. Hence, storage areas of cutting/assembling centres are a long-term buffer from which, day by day, small portions of the stored elements are retrieved to “feed” the different processes.

In the following, we try to emphasize the criticalities of these processes, in order to shed light on the "*hot-spots*" of the whole PLS supply chain.

For the sake of clarity, we recall that both slabs and profiles, due to their dimensions, need vast spaces for their storage and huge machinery to be moved.

In particular, slabs can be moved only by industrial magnetic cranes, and the only practical way to store them is to stack them one on top of each other. This method implies various criticalities that can seriously affect the retrieval process and the slabs' quality.

When storing slabs, the first issue to consider is that slabs may tend to bend if they are stacked right upon shorter ones. This phenomenon determines a substantial incompatibility between some items: two slabs differing more than 1,5 m in length or width must not be stacked consecutively.

The second critical issue we mention refers to the slabs' identification procedure. In particular, slabs stocked in the yard present a label reporting all their specific information (e.g., identification code, item, plaque and casting). This label needs to be easily accessible to the operator to identify the slab correctly during the retrieval process. Hence, excessive differences in the slabs' sizes would render impractical this control, since larger slabs will hide the smaller ones.

A third aspect concerns thickness. Generally, significant differences in thickness would make it impossible to pick-up a thick slab stacked on a thinner one because of strong the magnetic interaction between them. Clearly, the magnetic intensity of the crane plays a role here.

Lastly, slabs' quality certifications represent another critical issue. These certifications are issued by classification companies and typically last about six months after the delivery to the centre. This certification, in practice, works like insurance. Indeed, if a slab turns out as defective, all the slabs from the same cast-steel would have to be replaced. If the defective slab has been used during the time window of the quality certification's validity, insurance will cover the costs. Otherwise, they will be borne by the shipbuilder or the company managing the slabs (i.e., PLS, in this case). Therefore, the so-called "*expired slabs*", i.e., slabs whose quality certification is expired, are not used to fulfil any order. Hence, the number of expired slabs may be intended as a proxy of the centre's quality. Similar considerations apply to profiles.

All these issues have significant logistic implications. Indeed, moving the cranes [lifting beams for profiles] along the spans and picking slabs [profiles] from their stacks [stalls] is a time-consuming operation, even in the best possible conditions. Note that for "best possible conditions", we intend the possibility to randomly store slabs [profiles] on any stack [stall] and retrieve the slabs (profiles) in the best position (e.g., on top of a stack).

Therefore, given the "incompatibility" constraints between slabs [profiles] and their expiration dates, it is necessary to intervene in a programmatic and preventive way on the storage and retrieval processes. Clearly, the increasing number of items (and their variability) exacerbates the complexity of these processes (see Tables 2.7 and 2.8).

The variability of these values is not the only difficulty for the adequate storage of slabs or profiles. Indeed, also the frequency and the quantity with which they are retrieved play a role. We next analyze these aspects in more detail.

Focusing on slabs, in Figure 2.37, each item is represented as a dot characterized by two dimensions: the *average picking quantity* (on the y-axis) and the number of *retrieval days* (on the x-axis). Given the three Cruise Ship Orders discussed in Section 2.7, for each item, the number of retrieval days is calculated as the number of days in which at least one slab of that item is picked. The average picking quantity is then obtained, for each item, as the total number of slabs picked during the time-horizon under consideration, divided by the actual picking days.

In that picture, we group items in four quadrants: (i) the left-bottom quadrant (i.e., the pink one) comprises items that are picked in small quantities (less than five) and in a restricted number of days (less than 15); (ii) the right-bottom quadrant (in red), comprises items that are picked rather frequently (more than 15 days) but in low quantities (less than five slabs, on average); (iii) the upper-left quadrat (in yellow) involves items picked not very frequently (less than 15 days) in relatively high quantities (more than five slabs); (iv) finally, the upper-right quadrant (in blue) highlights items that are picked frequently (more than 15 days) in higher quantities (more than five slabs, on average).

Similarly, Figure 2.39 provides the same information for profiles.

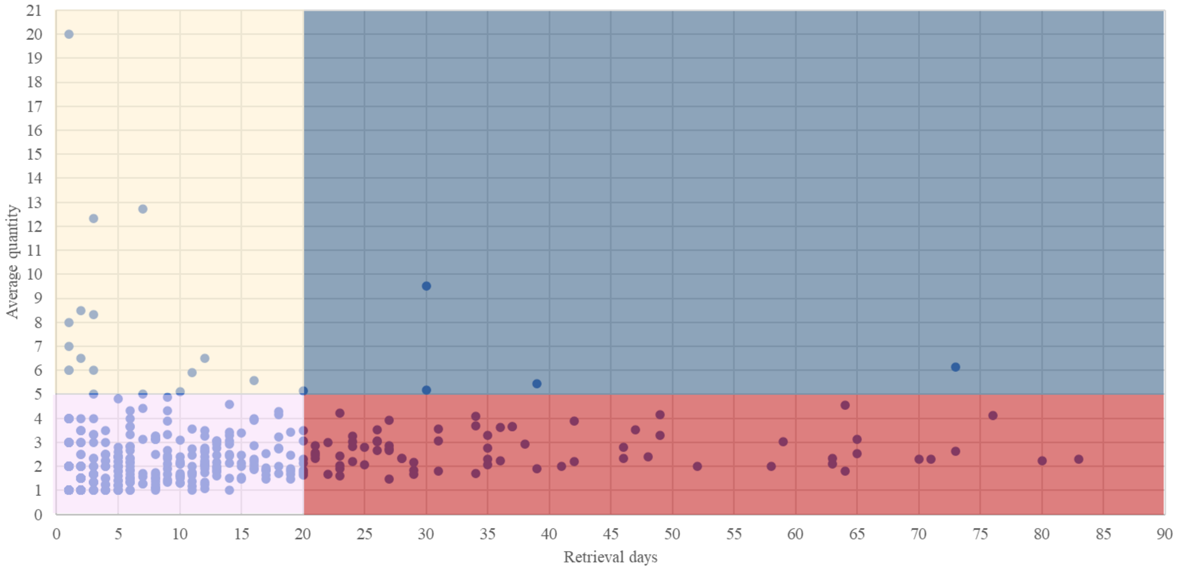


Figure 2.37 – Number of retrieval days and average picking quantities per item (slabs)

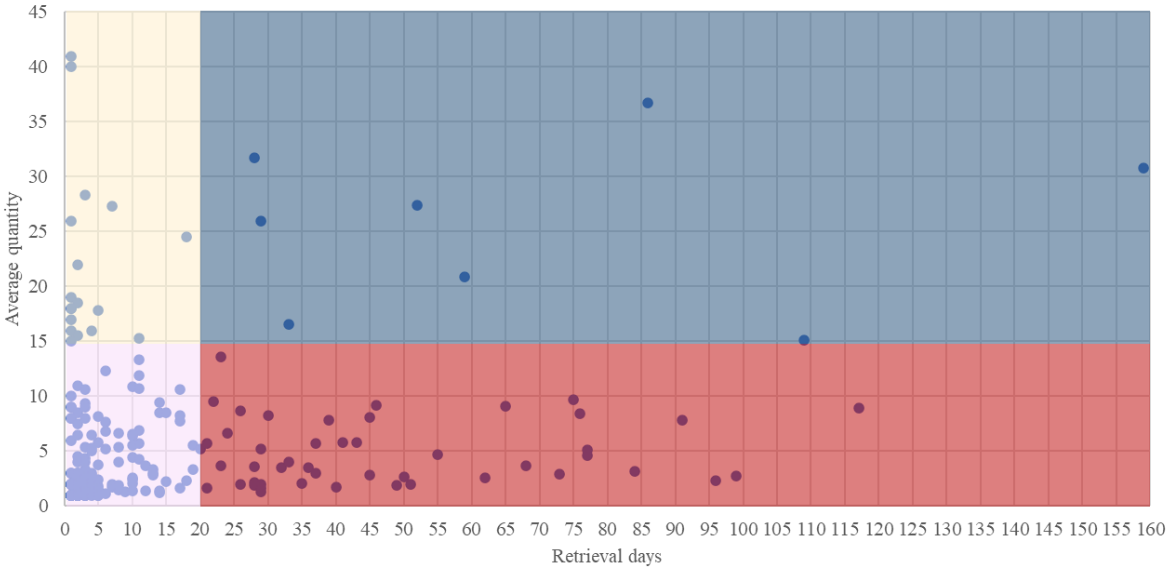


Figure 2.38 - Number of retrieval days and average picking quantities per item (profiles)

The above figures show a very heterogeneous distribution of items by the identified quadrants: it is a matter of fact that, both for slabs and profiles, PLS works with many items that are picked less frequently (and in low quantities). We should also note that the workload related to a minimal number of items (those in the blue quadrants) equals, in practice, the workload implied by the less frequent and scarce ones.

In principle, it would be ideal to use dedicated stacks [stalls] for each item and then and sort slabs [profiles] according to their expiration dates to ease the following retrieval process. However, this is impracticable due to the limited storage capacity. In practice, the main effect of the evident variability above discussed, together with capacity constraints, is the need to mix items in stacks [stalls].

As we already mentioned, the retrieval process is very time-consuming due to the complexity of each retrieval. Therefore, the proper selection of the target slabs [profile] is crucial to reduce retrieval times.

In the literature, the problem that aims at choosing appropriate slabs for an order or a group of orders to minimize shuffles during the retrieval process is known as the Slab Stack Shuffling (SSS) Problem. A thorough analysis of the problem and a comprehensive state-of-the-art will be provided in the next chapter.

2.9 Conclusions

This chapter described the case study of a manufacturing company operating in the shipbuilding industry, namely Palescandolo S.p.A. (PLS). After a brief introduction on its history and mission, we outlined its core processes, such as slabs and profile handling, components and subassemblies production. Our primary focus, however, was on the analysis of its logistics to identify the critical issues (i.e., "hot-spots") characterizing two of its main processes (storage and retrieval).

As we noted, storage capacity, the high variability of the items and their expirations dates are three main problems causing the need to mix items in stacks [stalls], thus rendering the retrieval process very tough and time-consuming. Therefore, the proper selection of target slabs [profiles] to pick-up to fulfil ship orders seeking to minimize the overall retrieval time emerges as a crucial problem in PLS. This problem, known in the literature as the Slab Stack Shuffling (SSS) Problem, will be analysed in details in the chapter. In particular, we will provide a comprehensive overview of the state-of-the-art in the field, propose a novel and unified framework to systematize it and highlight some gaps we aim at filling.

3 Literature review

Summary

The analysis shown in the previous chapters reveals that the slab handling process represents a critical issue in the shipbuilding industry. Indeed, the hull manufacturing process requires the interim storage of slabs at different stages: after their production, at the steel mills; before and after their cutting and intermediate processes; before the final assembly at the shipyards. The huge size of managed items and the need to stacking them in limited storage areas, pose critical challenges, both from a tactical and an operational perspective. Consequently, the proposal of methods and tools to support decisions in this context could be beneficial and could produce significant impacts on the efficiency of the production processes and the whole supply chain.

In the Operations Research and Management Science literature, the stacking problems have been widely studied in the context of port logistics, with reference to containers. Instead, much more limited attention has been devoted to the handling of slabs in the shipbuilding industry.

In this chapter, we first introduce a formal description of the so-called Slab Stuck Shuffling (SSS) problem. Secondly, we propose a general framework, which consists of a set of elements and properties that may characterize the problem. Finally, we review the existing contributions devoted to the SSS problem's investigation. This way, we intend to (i) systematize the sparse literature on the topic; (ii) define different variants of the problem; (iii) identify the existing gaps in the reference literature.

3.1 Stacking problems

The Stacking Problems involve a vast group of problems where the storage area is organized in stacks, and single items are put on top of each other in these stacks **(**Lehnfeld & Knust, 2014). Such peculiar storage method makes the related warehouse management problems different from the classic ones, which consider *bin shelving*, *modular storage drawers*, *pallet racks*, *gravity flow racks*, or *mobile storage racks* (Van den Berg and Zijm, 1999).

Indeed, in this case, all operations to move and get items are executed by cranes located above the stacks, so that direct access is possible only to the topmost item of any stack. This implies that if an item stacked below has to be retrieved, so-called reshuffling (or relocation) is necessary. Since reshuffling operations are usually very time-consuming, they should be avoided as often as possible.

Hence, the objective usually consists of maximizing the efficiency of such operations, in terms of time and costs.

These problems have been widely studied in the context of port logistics, to tackle issues related to containers' handling (Containers Stacking – CS problem). Other variants have been investigated with reference to the management of the various type of steel elements, such as slabs and coils (Steel Stacking – SS problem). Apart from the application context, the item’s stacking leads to different kinds of optimization problems. On the one hand, if incoming items arrive at a storage area, they need to be assigned to positions, which causes *loading problems*. On the other hand, *unloading problems* arise if outgoing items need to be retrieved from the storage area and one has to decide which items will leave the storage in which order and which relocations are performed. *Premarshalling* occurs if items have to be sorted inside the storage area such that all items can be retrieved without any further reshuffle. If incoming items need to be stored while outgoing items need to be retrieved, combined loading/unloading problems appear. In Table 3.1, an overview of the different optimization problems above introduced in the two introduced contexts is provided.

		APPLICATION CONTEXT	
		Containers Stacking Problem (CSP)	**Steel Stacking Problem (SSP)**
OPTIMIZATION PROBLEM	**Loading**	Selecting the best* locations of incoming containers	Selecting the best* locations of incoming slabs/coils
	Unloading	Selecting the best* retrieval sequence for outgoing containers	Selecting the best* outgoing slabs/coils and the best* retrieval sequence for outgoing items
	Pre-marshalling	Reshuffle in the best* way containers in the storage area	Reshuffle in the best* way slabs/coils in the storage area

* according to specific criteria

Table 3.1 Classification of Stacking problems

3.1.1 The Container Stacking problems

In the context of containers management, the stacking problems may arise with reference to different processes:

- the loading and/or unloading of container vessels;
- the assignment of storage locations to incoming containers in a terminal;
- the reassignment of positions to already stacked containers within the terminal (pre-marshalling versions);
- the retrieving of containers from a terminal.

An example of a container terminal is reported in Figure 3.1.

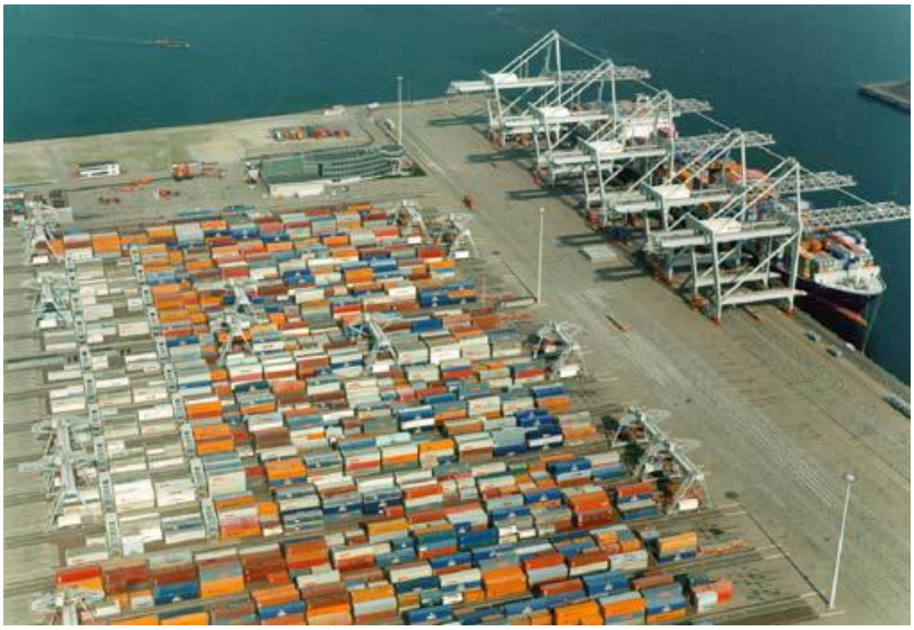


Figure 3.1 – Container terminal

The CS loading problem aims at selecting the best locations for the containers, once these reach a pre-stocking area. Locations are chosen according to many different criteria, among which: minimise the handling effort of cranes and ensuring a vessel's stability (Kim et al., 2000); minimise the vessel space occupied by the containers that need to reach different destinations (Wilson & Roach, 2000) or the number of shuffles (Avriel & Penn, 1993).

In the premarshalling versions, containers are already stacked and re-shuffled to minimise future handlings for the retrievals, exploiting the eventual remaining space in the terminal (Kim & Bae, 1998; Meisel & Wichmann, 2010).

In the unloading problems, containers need to be retrieved from a stocking area to satisfy delivery requests, in order to minimise an objective function often related to the number of shuffles (Malucelli et al., 2008).

3.1.2 The Steel Stacking problems

The reference items of the Steel Sacking problems can be classified into two main categories: (i) *flat rectangular items*, such as slabs and plates, and (ii) *round items*, such as hot rolling and cold rolling coils. In Figure 3.2, different layouts are shown, depending on whether the yard is devoted to the stacking of slabs or coils. It is possible to notice that while the slabs are vertically stacked, the coils form inclined stacks, in which each coil is positioned between two coils of the lower level.

As in the case of CS problems, different optimization problems can be defined to deal with the *loading*, *unloading* and *pre-marshalling of the stacked elements.*

In the loading problems, the slabs arrive in a pre-storage area in a certain order, and their future retrieval sequence is already known. Hence, the problem consists of defining a storage plan (i.e., assigning the optimal positions in the stacks), in order to minimise the total number of shuffles that will be necessary for the future retrievals (Ko, 2007; Ko et al., 2007; Kim et al., 2011).

In Figure 3.3, a set of slabs is represented, with the indication of their arrival time *x* and their leaving time *y*. In Figure 3.3(a), slabs are stacked in the pre-storage area according to their arrival time. In contrast, Figures 3.3 (b) and (c) show two different possible assignments of the slabs in the storage area, that will respectively generate 7 and 4 shuffles during the future retrieval process. Hence the last configuration represents a better storage plan to minimise the total number of shuffles.

In the Coil Shuffling problem, the coils are stacked in the so-called "inclined stack" (Figure 3.4) and the objective is to minimise the number of shuffles necessary for the retrievals (Tang et al.,2012).

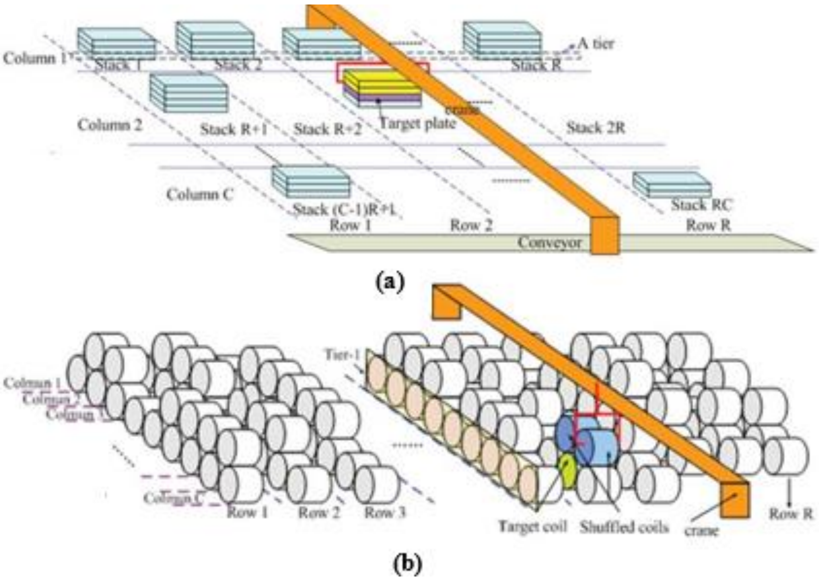


Figure 3.2 - (a) Layout of a slab yard (b) Layout of a coil yard (Tang et al.,2012)

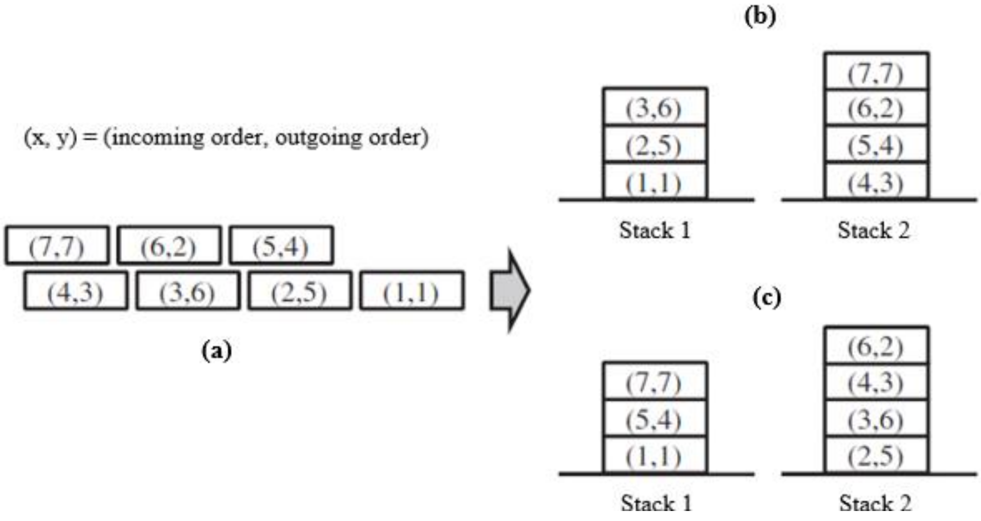


Figure 3.3 – An example of slab stacking problem: (a) arrival slabs, (b) stacking causing 7 shuffles and (c) stacking causing 4 shuffles (Kim et al, 2011)

Figure 3.4 – Coils inclined stacks

3.2 The Slab Stack Shuffling problem

3.2.1 Problem introduction

The Slab Stack Shuffling (SSS) problem belongs to the category of Stacking problems and, specifically, to the sub-class of Steel Stacking *unloading* problems.

To introduce this problem, let consider a storage area containing a set of slabs, with different characteristics, in terms of size (i.e., length, width, and thickness), steel degree, etc. An *item* identifies univocally a typology of slabs, i.e., a combination of given characteristics.

Such slabs are stored in a yard, divided into spans and pitches, as shown in Figure 3.5. Each pitch is devoted to hosting a stack. Different layouts can be selected to organize the slabs in stacks: for example, the slabs belonging to the same item typology can be stacked all together or in different stacks, mixed with other slab typologies. In Figure 3.6, we show a first case (Figure 3.6(a)), in which the slabs are organized in dedicated stacks, and a second case (Figure 3.6(b)), in which they are randomly mixed.

An *order* lists the items and the related quantities, requested by a customer or by a production/assembly workstation, in the case of a distribution centre or a production warehouse, respectively. The *order picking* is the process of retrieving the slabs from the yard to satisfy the requests contained in the order. As each stack can be accessed only from its topmost slab, it is easy to understand that, if a slab below has to be retrieved, it is needed to shift all the slabs above the *target slab* to satisfy the order. With the term, *shuffle* is intended the temporary or permanent shift of the single element stacked above the t*arget slab*.

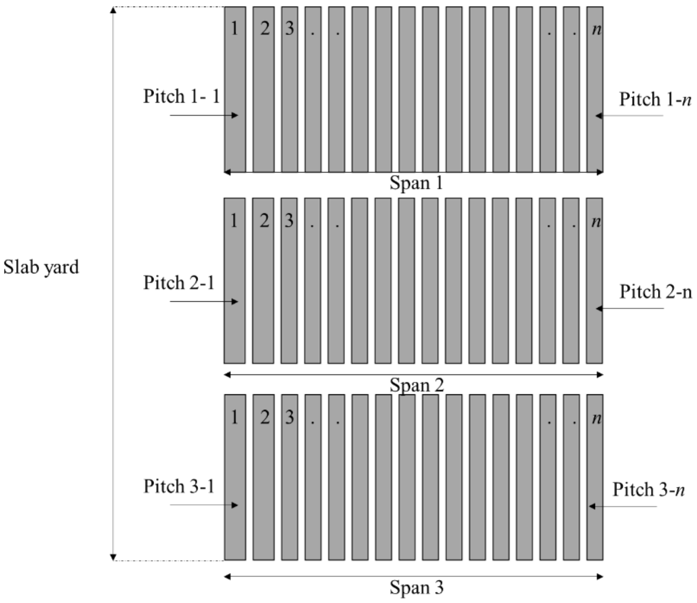


Figure 3.5 – An example of slab yard: spans and pitches

	item B	item C	
item A	item B	item C	
item A	item B	item C	item D
item A	item B	item C	item D
item A	item B	item C	item D
item A	item B	item C	item D
item A	item B	item C	item D
item A	item B	item C	item D

(a)

	item B	item A	
item A	item A	item C	
item C	item B	item D	item D
item A	item B	item A	item C
item D	item A	item C	item C
item B	item B	item D	item D
item C	item D	item B	item D
item C	item A	item B	item A

(b)

Figure 3.6 – Dedicated Stacks (a), Random Stacks (b)

These operations are very time-consuming and may significantly impact the efficiency of the whole process. Once a target slab is identified, the crane, necessary to retrieve the slabs, must be moved along the span and placed precisely on the stack where the target slab is located (Figure 3.7). Once the crane approaches the stack, an operator adjusts the retractable magnets according to the slabs' length and activates them, by regulating their intensity according to the weight to lift and his own experience (Figure 3.8). Then, the top-most slab is lifted and shifted, temporary or permanently, on another stack (Figure 3.9). This operation repeats as many times as the number of slabs above the target one. Once reached, this latter is lifted and transported to the delivery point, usually located at the far end of the span.

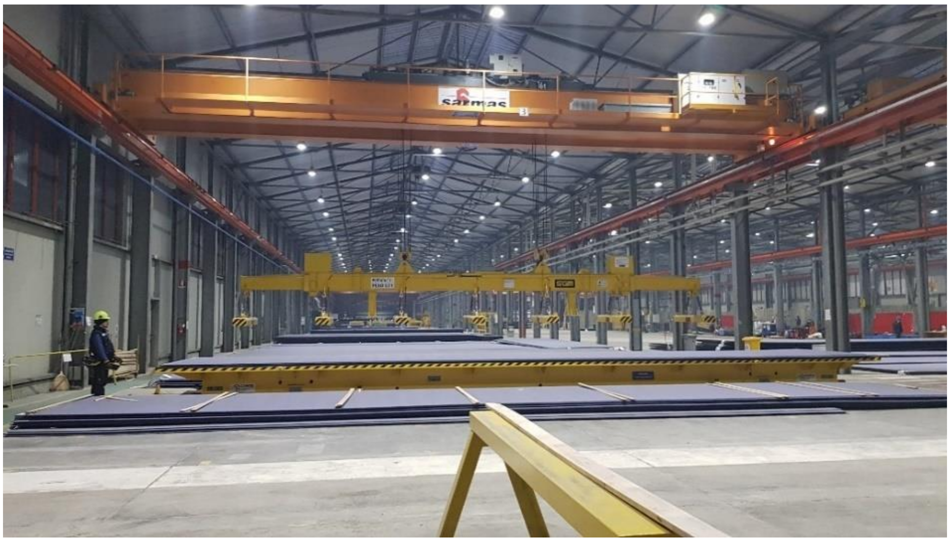


Figure 3.7 – Crane approaching the stack

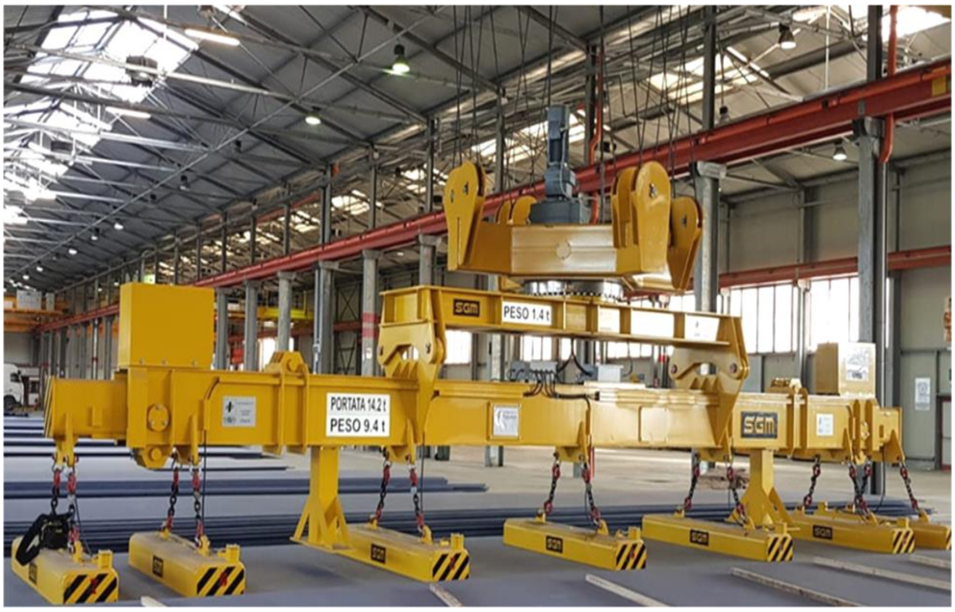


Figure 3.8 – Magnets positioning and activation

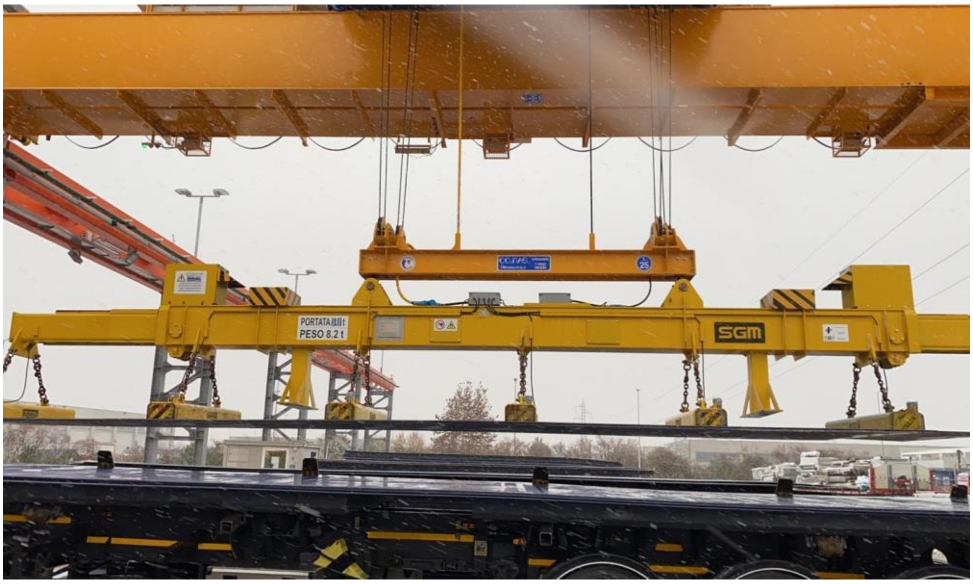


Figure 3.9 – Slab lifting

All these operations are very time-consuming and strongly influenced by the adopted stacking method. Indeed, if the slabs of different items are stored in dedicated stacks, the picking times tend to be minimized; but, at the same time, such a layout requires a high number of pitches and a vast storage area. On the contrary, mixing slabs of different item typologies in the same stacks would allow minimizing the occupied storage area but, at the same time, it implies higher retrieval times.

In this case, an unwarranted choice of the slabs to retrieve may produce significant delays in the subsequent process; hence, the retrieval process needs to be optimized.

The Slab Stack Shuffling (SSS) problem is defined in literature as the problem of choosing appropriate slabs to retrieve from a stack to satisfy an order or a group of orders to minimise shuffles during the retrieval process.

3.2.2 The SSS problem and the other Stacking problems

The SSS problem presents many similarities and differences with the other Stacking problems above introduced.

First of all, it is necessary to highlight that it belongs to the class of the unloading problems; hence, it assumes that the stored products have been stacked, with no indication on the future picking orders. This assumption represents the main difference with the loading problems, where the positioning of the incoming products is optimized based on the a priori knowledge of the future retrieval orders.

Also, within the same class of unloading problems, significant differences emerge between the SSS and the other problems. In the case of CS problems, the first difference regards the number of elements in the same stack. In the container problems, generally, almost four tiers are reached (Kim et al., 2000), while in the SSS context, the number of slabs stored in the same stack can exceed the hundreds (see Figure 3.10). Moreover, the characteristics of the retrieval orders are completely different. In the unloading CS problems, an order is composed of a list of specific containers to be picked-up; hence, only the optimal sequence in which they have to be retrieved has to be defined. Instead, in the SSS problems, the order lists the items typologies to be retrieved with the associated quantities (Lehnfeld & Knust, 2014). Thus, for each retrieval, it needs also to select the best slabs among those belonging to the same required item.

In the case of SS, the coil unloading problems present peculiar characteristics due to the managed elements. Indeed, the coils can be stored only in inclined stacks, made up of two levels (Figure 3.11). Specifically, each element on the upper level has to be placed among two coils of the lower level (triangular correlation). Due to such a storage method, in a coil shuffling problem, the retrieval of a target coil requires a maximum of two shuffles. Indeed, the worst case is represented by the retrieval of a coil positioned at the lower level; for example, the one depicted in green in Figure 3.11. In this case, two shuffles are required to remove the coils above it; the one in the same stack and the other positioned in the adjacent one. This implies that the mathematical models that deal with the coil shuffling problem and their solution methods are not applicable to the case of SSS.

Due to its peculiar characteristics, the contributions devoted to the SSS problem represent a separate body of literature and need to be deeply analysed to provide a state of the art. This analysis could help to understand which variants of the problem have been explored and which gaps exist in the reference literature.

Figure 3.10 – Example of a Container Stack (a) and a Slabs Stack (b)

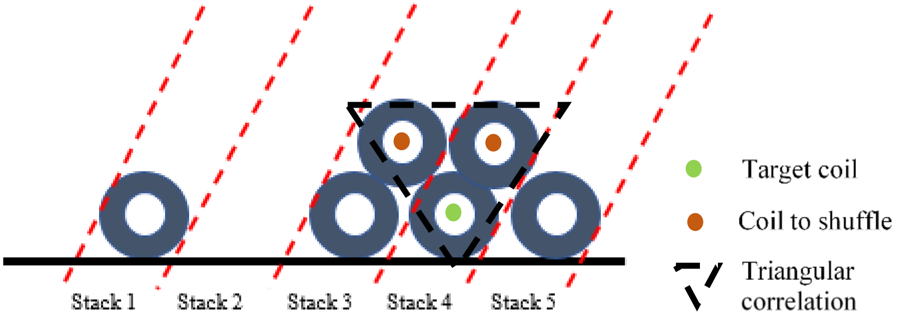


Figure 3.11 – Coil Stack

3.3 The framework

3.3.1 General notation

To better describe the properties and the characteristics of the process above introduced and to formalize the SSS problem, it is useful to introduce a formal notation.

Notation:

Basic Elements	
$J = \{1, \dots, n\}$	set of slabs, indexed by j;
$I = \{1, \dots, m\}$	set of items, indexed by i;
$F = \{1, \dots, p\}$	set of stacks, indexed by f;
$H = \{1, \dots, a\}$	set of spans, indexed by h;
A_j	span where the stack j is located;
$J_i \subset J$	subset of slabs belonging to item i ($\bigcup_{i \in I} J_i = J$; $J_i \cap J_k = \emptyset, \forall i, k \in I$);
Layout Characteristics	
$I_f \subseteq I$	subset of items that can be hosted in a given stack f;
$F_i \subseteq F$	subset of items that may host item i;
p_j	stack in which the slab j is located;
p_0	fictitious stack, indicating the point where target slabs have to be delivered;
D_j	initial number of slabs positioned above the slab j;
Order Characteristics	
q_i	number of requested slabs of item i;
$\bar{q} = \sum_{i \in I} q_i$	number of retrievals;
$\bar{J} = \bigcup_{i:\, q_i > 0} J_i \subset J$	subset of candidate slabs to fulfil the order;
Time parameters	
t_0	time to lift up and lower down a slab;
t_p	unit time to transfer the single target slab to the delivery point;
$t_{ff'}$	unit time to transfer a slab from the top of stack f to the top of the stack f';
Further parameters	
d_j	deadline of the slab j;
l_i	deadline for the fulfilment of the request of item i in the order;

3.3.2 Framework definition

The Slab Stack Shuffling (SSS) problems may present different characteristics, according to the elements and the assumption considered in the problem setting, the application contexts, etc.

In order to support the study of the literature and to identify the variants of the problems that have been neglected so far, in this section we propose a theoretical framework, that systematize all the possible elements, that may characterize the problem (Imenda, 2014).

The proposed framework is based on the following elements:

1. order typologies;
2. shuffle definition;
3. layout characteristics;
4. objective function;
5. deadline constraints.

In the following, the single elements are deeply discussed and analysed

1. *Order typologies*

In general, an order is a list of items whose slabs need to be picked up to be sent to the next production process, i.e. the hot rolling, the slab cutting, the delivery to a new plant or to the shipyard, etc.

The orders may be classified into:

- *Item* or *Family orders*;
- *Single* or *Multiple orders;*
- *Sorted or Not-sorted orders.*

Item or Family orders

The order lists a set of items to be retrieved with the associated quantities. If hard constraints related to the characteristics of the requested item exist, we talk about *item order*. In this case, only the slabs belonging to the requested item may contribute to the order fulfilment. Instead, we refer to *order family* if a set of different items may satisfy the technical requirements of a given order (i.e., width, steel-grade, weight). In this case, all those items meeting such requirements are included in the same family, and all the associated slabs may fulfil the order.

Single or Multiple orders

This classification refers to the possibility for an item to be present in the order more than one time. In the single orders, each item can be present only one time in the list ($q_i \leq 1, \forall\, i \in I$), while in the multiple order, it may be present more than one time ($\exists\, i \in I$: $q_i \geq 2$). In both cases, the feasible condition for the problem is represented by the presence of candidate slabs at least equal to the slabs requested in the order ($|J_i| \geq\ q_i, \forall\, i \in I$).

Sorted or Not Sorted orders

The third classification distinguishes between the S*orted* and *Not sorted orders.* In the first case, the slabs need to be retrieved according to a pre-defined sequence. For example, the sorted list {1,4,3,1} indicated that it is needed to retrieve first the item 1, than the items 4 and 3, and finally a slab of the item 1 again. This could reflect constraints related to the transportation (track loading and unloading) or the schedule of subsequent processes. In the case of *not-sorted* orders, it is possible the slabs of any requested item at any moment of the retrieval process. In this case, the order is represented by a not sorted list, in which each item is characterized by the requested quantity.

Of course, in the case of sorted orders the SSS problem is much more constrained and solution space more limited.

2. *Shuffling method*

The definition of the shuffling method is fundamental to describe the SSS problem. It is worth recalling that the term shuffle refers to the temporary or permanent shift of the single elements stacked above the *target slab*.

The first aspect to be defined relates to the position where the slabs may be shifted, once the target slab is retrieved. If they are constrained to be put onto the initial stack, the shuffling process follows a *repositioning mechanism. Alternatively, i*f they can be relocated onto a different stack, we talk about shuffles *without repositioning.*

In the first case, we need to introduce a further sub-classification, based on the moment in which the repositioning occurs. The shifted slabs can be repositioned after every single retrieval (not-consecutive repositioning) or when all the retrievals from the same stack are completed (*consecutive repositioning).*

Instead, in the second case, a different sub-classification needs to be introduced, based on the criteria used to select the destination stack. A first option is represented by the stacks that have no target slabs and, hence, will not be visited during the retrieval process. In this case, the shifted slabs play the role of barrier only one time during the whole retrieval process (*one-time barrier*). Alternatively, shifted slabs may be moved toward any stack. In this case, they might play more than one time the role of barriers and be shuffled again (*multiple-times barrier*).

3. Layout characteristics

As introduced above, the slabs can be organized in stacks according to different criteria. For example, the slabs belonging to the same item typology can be stacked all together or in different stacks, mixed with other slab typologies. In order to better describe the layout adopted in the storage area, we first distinguish among *dedicated* and *random* stacks. In the first case, slabs of a given item can be located only in correspondence of specific stacks and such stacks can host only slabs of that item. In the second case, slabs of a given item can be located in different stacks, mixed with other typologies.

In order to identify the single layouts, we introduce the following symbol:

$\boldsymbol{m|n}$ where m indicates the number of stacks in which the slabs of the same item are stored, and n the number of items typologies contained in the same stack.

The **layout $\mathbf{1|1}$** refers to a situation in which the slabs of each given item are hosted in a single stack and this stack host only the slabs associated to that item (Figure 3.12 (a)). In this case, the following conditions hold:

$$|F_i| = 1 \quad \forall\, i \in I$$

$$|I_f| = 1 \quad \forall\, f \in F$$

The **layout $\boldsymbol{m|1}$** refers to the condition in which slabs of a given item are hosted in m different stacks, and these stacks host only that item (Figure 3.12 (b)). In this case, the following conditions hold:

$$|F_i| = m \quad \forall\, i \in I$$

$$|I_f| = 1 \quad \forall\, f \in F$$

The **layout $\boldsymbol{1|n}$** refers to the condition in which slabs of each given item are hosted in a single stack, but this stack hosts n different items (Figure 3.12 (c)):

$$|F_i| = 1 \quad \forall\, i \in I$$

$$|I_f| = n \quad \forall f \in F$$

The **layout m$\boldsymbol{|n}$** refers to the condition in which slabs of each given item may be hosted in $\boldsymbol{m}$ different stacks, each of which may host slabs of n different items (Figure 3.12 (d)):

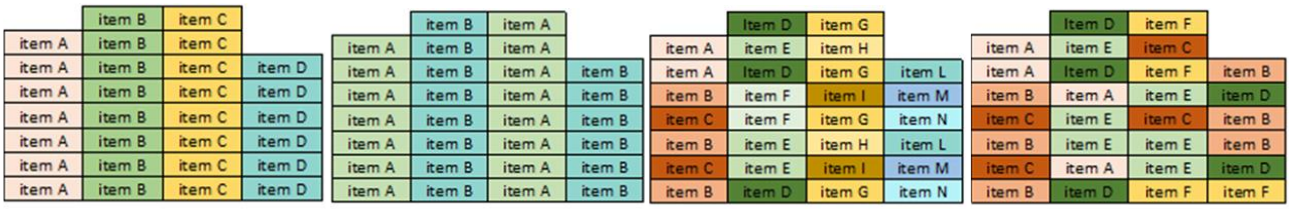
$$|F_i| = m \quad \forall\, i \in I$$

$$|I_f| = n \quad \forall f \in F$$

The *random* case refers to the condition in which any slab could be hosted by any stacks and any stacks could host slabs of any item:

$$F_i = F \,\forall\, i \in I$$

$$I_f = I \,\forall\, f \in F$$



(a) Layout 1|1 (b) Layout 2|1 (c) Layout 1|3 (d) Layout 2|3

Figure 3.12 – Example of dedicated layout

4. *Deadline constraints*

As shown in the previous chapters, the slabs may present a temporary certification quality. These certifications, released by classification companies, have a fixed duration; hence, they cannot be used beyond such deadlines. In order to take into account such aspects, constraints related to the impossibility of using slabs with expired certifications could be integrated in the problem (*slab deadline*). Moreover, the deadline may also refer to the picking operations and it may indicates a term within which the retrieval to satisfy the request of a given item has to be completed (*order deadline*).

5. *Objective Function*

In the SSS, the main elements that could be considered in the objective function are:

- *the number of shuffles;*
- *the retrieval time;*
- *the expired slabs;*
- *the spans' workload balance;*

Number of shuffles

In each different version of the shuffle, the *number of shuffles* often plays a central role, being the most time-consuming operation in the whole retrieval process. The SSS models have been developed around the different ways to consider the shuffles, according to the different cases of study proposed in the literature. This term, often, is the only term that is considered to evaluate the process, and it is always present in each proposed model, even with other objectives pursued at the same time.

Retrieval time

The retrieval time is usually composed of three terms:

- the lifting time, representing the time required to lifting up and lowering down a slab. It can be considered as the product between the number of shuffles and a parameter $\boldsymbol{t}^*$, representing the unit time needed for each operation.
- the shuffle time, representing the time needed to move the slabs between different stacks. It depends on the stack from which the slabs depart (origin) and the one to which they are moved (destination). For each couple of stacks $\boldsymbol{f}$and $\boldsymbol{f'}$, such time parameters $\boldsymbol{t_{ff'}}$ can be considered proportional to the distance between the two stacks and the average crane speed.
- the delivery time, representing the time to move any target slab from the its departing stack $\boldsymbol{f}$to the delivery point.

Expired slabs

If the slabs are characterized by given deadlines, the decision maker could also aim to reduce the total number of expired slabs at the end of the retrieval process, as a proxy of the quality of the warehouse.

Spans' workload balance

In some contexts, it could be necessary to balance the workload assigned to each span and, hence, to each crane. Usually, this balance is obtained by regulating the relative number of slabs retrieved from each span.

The elements of the framework above introduce are summarized in Table 3.2.

<table>
<tr><td rowspan="3"><b>Order typologies</b></td><td colspan="4">Item order</td><td colspan="2">Family order</td></tr>
<tr><td colspan="4">Single order</td><td colspan="2">Multiple order</td></tr>
<tr><td colspan="4">Sorted order</td><td colspan="2">Not-sorted</td></tr>
<tr><td rowspan="2"><b>Shuffling method</b></td><td colspan="4">With Repositioning</td><td colspan="2">Without repositioning</td></tr>
<tr><td colspan="2"><i>Consecutive</i></td><td colspan="2"><i>Not-consecutive</i></td><td><i>One-time barrier</i></td><td><i>Multiple barrier</i></td></tr>
<tr><td rowspan="2"><b>Layout characteristics</b></td><td colspan="4">Dedicated</td><td colspan="2">Random</td></tr>
<tr><td>$1|1$</td><td>$1|n$</td><td>$m|1$</td><td>$m|n$</td><td colspan="2"></td></tr>
<tr><td rowspan="2"><b>Deadline constraints</b></td><td colspan="6">Slab deadline</td></tr>
<tr><td colspan="6">Order deadline</td></tr>
<tr><td rowspan="4"><b>Objective function</b></td><td colspan="6">Number of shuffles (to be minimized)</td></tr>
<tr><td colspan="6">Retrieval time (to be minimized)</td></tr>
<tr><td colspan="6">Expired slabs (to be minimized)</td></tr>
<tr><td colspan="6">Span workload (to be balanced)</td></tr>
</table>

Table 3.2 – The framework

3.4 The state of the art

Most of the contributions devoted to the SSS problem refer to the steel industry. In this context, a slab yard functions as a storage buffer between the continuous casting and the steel rolling.. Slabs need to be picked up from the slab yard according to a so-called *rolling schedule*, that represents the retrieval order.

The seminal paper on the SSS problem is by Tang et al. (2001), that has then inspired the works by Tang et al. (2002) and Singh et al. (2004). All these works analysed the SSS problem by referring to the hot rolling mill at the Baoshan steel plant in Shanghai.

In order to formulate a mathematical model, the authors adopted the same assumptions:

- slabs must be retrieved from the yard one by one following the sequence expressed in the rolling order (sorted order);
- for each item of the work order, there is a single type of slab that meets its requirements (item order);
- the set of slabs that meets the $\boldsymbol{i}$-th request of the order is different from the one that meets the requirements for the $\boldsymbol{k}$-th request (for any $\boldsymbol{i} \neq \boldsymbol{k}$). Therefore, the possibility that the same item is required twice or more in a work order is excluded (single order);
- if the target slab is not on the top of a stack, the slabs above it must be temporarily moved to allow the target slab to be retrieved and then repositioned in their original order (repositioning mechanism);

- a shuffle is intended as the shift and consequent repositioning of a given slab.

In order to formulate the model, we may refer to the above notation and to the following groups of decision variables:

x_{ij} binary variable equal to 1 if and only if slab $j \in J$ is retrieved to satisfy the request of item $i \in I$;

S_{ij} positive integer variable, representing the number of shuffles necessary to retrieve the slab j for the request *i;*

Hence, the model can be formulated as follows:

$$\sum_{i=1}^{m} \sum_{j \in J_i} S_{ij} x_{ij} \qquad Min! \qquad (1)$$

subject to

$$S_{ij} = D_j - \sum_{k=1}^{i-1} \sum_{r \in \{h | p_h = p_j\}} \min\left(1, \max\left(D_j - D_k, 0\right)\right) x_{sr} \qquad \forall\, j \in J_i, i \in [1, m] \qquad (2)$$

$$\sum_{j \in J_i} x_{ij} = 1 \qquad \forall\, i \in I \qquad (3)$$

$$x_{ij} = 0/1 \qquad \forall\, j \in J, i \in I \qquad (4)$$

The objective function (1) aims at minimizing the sum of the shuffles needed to retrieve the slabs $j \in J$ for the requests of the items $i \in I$ included in the order. The constraint (2) evaluates the shuffles to retrieve the slab $j \in J$ in order to assign it to the i-th item, as the difference between the initial number of slabs above *j* (D_j) and the slabs initially positioned above *j* that have been retrieved to satisfy any item *k* before the item *i* ($\sum_{k=1}^{i-1} \sum_{r \in \{h | p_h = p_j\}} \min\left(1, \max\left(D_j - D_k, 0\right)\right) x_{sr}$). The group of constraint (3) ensure that each slab *j* can be assigned at most to one item. Finally, the constraints (4) define the binary nature of the decision variable x_{ij}.

The developed model is in a non-linear model; for this reason, the three works proposed an heuristics approach to solve it. Even if the assumptions behind the models are the same, the proposed solution approaches are different. Tang et al. (2001) proposed a two-step heuristic algorithm, in which an initial solution is generated and, then, improved through a local search. Tang et al. (2002) proposed a modified genetic algorithm, in which the population and genetic operators are designed ad hoc to solve the problem. In particular, a change was made in the crossover operation, and a local search operation was added in some iterations. A comparison of the two approaches, applied to the same instances of the problem, showed that the modified genetic algorithm always produces better solutions. Singh et al. (2004) developed an improved parallel genetic algorithm to overcome the problem of a premature convergence of the conventional genetic algorithm. By solving the same test instances, it emerged that the parallel genetic algorithm produces an improvement of the solutions, almost equal to 6%.

Fernandes et al. (2012) studied the SSS problem, by modifying the assumptions made by Tang et al. (2001), Tang et al. (2002), Singh et al. (2004). Indeed, the authors allowed the same item to be requested several times in each order (*multiple orders*).

With reference to the above notation, the model by Ferandes et al. (2012) can be formulated as follows:

$$R - T \qquad Min! \tag{5}$$

subject to

$$\sum_{i\in I} x_{ij} \leq 1 \qquad \forall\, j \in J_i \tag{6}$$

$$x_{ij} = 0/1 \qquad \forall\, j \in J, i \in I \tag{7}$$

where the two terms of the objective function can be expressed as reported below:

$$R = \sum_{i=1}^{m} \sum_{j\in J_i} D_j\, x_{ij} \tag{8}$$

$$T = \sum_{i=2}^{m} \sum_{j\in J_i} \sum_{k=1}^{i-1} \sum_{r\in\{s:|D_j\geq D_s \text{ and } p_j = p_s} x_{ij}x_{kr} \qquad \forall\, i \in I \tag{9}$$

The objective function (5) aims at minimizing the number of shuffles during the retrieval process, given by the difference between the dependent variables R and T, defined by the relation (8) and (9), respectively. Specifically, the equation (8) represents the sum of slabs initially above each slab j chosen to satisfy the request of item i. On the contrary the equation (9) represents the sum of the slabs initially stacked above each retrieved slab j, but previously retrieved to satisfy an item k < i. The group of constraints (6) ensures that a slab j can be unused or retrieved to satisfy one and only one item i. In the end the constraints (7) define the binary nature of the variable x_{ij}.

The non-linear model (5-9) is than linearized considering the following group of decision variables:

w_{ijkm}	binary variable equal to one if and only if the slab j and $m \in J$ are selected for the retrieval of item i and $k \in I$, respectively;

and adding the following groups of constraints:

$$w_{ijkm} \leq x_{ij} \qquad \forall\, j, m \in J\ \forall\, i, k \in I \tag{10}$$

$$w_{ijkm} \leq x_{km} \qquad \forall\, j, m \in J\ \forall\, i, k \in I \tag{11}$$

$$w_{ijkm} \leq x_{ij} + x_{km} - 1 \qquad \forall\, j, m \in J\ \forall\, i, k \in I \tag{12}$$

These constraints ensure that the variable w_{ijkm} is equal to 1 only when both the variables x_{ij} and x_{km} are equal to 1.

The authors compared the model results with a constructive heuristic, showing that as the problem's complexity increases, the model's results are always better in terms of the number of slabs shuffled and performed in good computing time.

Later, Tang and Ren (2010) reformulated the SSS problem considering several new features, that are summarized in the following:

- the concept of order family is introduced, indicating a set of slabs belonging to items with similar characteristics, suitable to meet the same requests of an order;
- the types of slabs required in succession have similar characteristics in terms of thickness, width and steel grade to minimise the number of required rollers' changes;
- the slabs above the target ones have not to be repositioned in the initial positions. They should be placed as close as possible to their original stack to reduce the distance covered by the magnetic crane. The authors assume that there are always enough places to put the shuffled slabs in the adjacent stacks;
- once moved to a nearby stack, a shuffled slab will always remain on top so that it can be retrieved in subsequent periods without further shifts;
- multiple spans are considered with dedicate cranes;
- the objective function is defined as the total retrieval time and it is given by the sum of three contributes: the shuffling time, the lifting time and the time to deliver the target slabs from their original stacks to a fictitious stack, representing the delivery point of each span.
- a deadline is introduced to indicate that an item has to be retrieved within a given time to satisfy the order. This condition takes into account the limited capacity of the cranes and tends to balance the workload among them. Indeed, a solution that minimizes the shuffles could produce an unbalanced workload, concentrated in a single span. Deadline constraints avoid that such congestion produces significant delays in the retrieval process.

In order to introduce the model, the following additional decision variables have to be introduced:

y_i positive integer variable, representing the span of the slab j assigned to item i;

T_{ij} positive integer variable, representing the retrieval time to assign slab j to the i-th item of the sequence;

S'_{ij} positive integer variable, representing the number of shuffles necessary to assign the slab j to the item i, requested in the order.

Considering the above notation, the model can be formulated as follows:

$$\sum_{i=1}^{m}\sum_{j\in J_i} T_{ij}x_{ij} \qquad Min! \tag{13}$$

s.t.

$$\sum_{j\in J_i} x_{ij} = 1 \qquad \forall\, i \in I \tag{14}$$

$$\sum_{i\in I} x_{ij} \leq 1 \qquad \forall\, j \in J_i \tag{15}$$

$$S'_{ij} = \max\left(D_j - \max\left(x_{kr}(S'_{kr}+1) \middle| r \in \{s\colon |p_j = p_s\}, k \in [1,i]; 0\right)\right. \qquad \forall\, j \in J_i, i \in [1,m] \tag{16}$$

$$y_i = \sum_{j\in J_i} A_j x_{ij} \qquad \forall\, i \in I \qquad (17)$$

$$\sum_{k=1}^{i} \sum_{j\in J_i,\, A_j=y_i} T_{kr}x_{kr} \le l_i \qquad \forall\, i \in I \qquad (18)$$

$$x_{ij} = 0/1 \qquad \forall\, j \in J, i \in I \qquad (19)$$

where:

$$T_{ij} = 2t_p\left(p_j - p_0\right) + t_s S'_{ij} + t_0 \qquad (20)$$

As mentioned, the objective function (13) aims at minimizing the whole retrieval time, given by the sum of three contributes (20). The first regards the above introduced *delivery time* , i.e. the time to move the crane from the delivery point p_0 to the stack of the slab *j* and back to the delivery point $(2t_p(p_j - p_0))$; the second term represents a simplified version of the *shuffle time*, in which the stack of origin (*f*) and of destination (f') are not considered and a fixed time t_s is considered for each shuffle; finally, the third term t_0 considers the *lifting time*, i.e. the time required to lifting up and lowering down a slab, only related to the retrieved slab. The groups of constraint (14) and (15) ensure that each slab j can be assigned at most to one item i and that a slab j can be retrieved to satisfy one and only one item *i* respectively. Constraints (16) evaluate the shuffle to retrieve a slab *j* in order to satisfy the i-th item of the order. According to the proposed equation, this value is null if any slab positioned above the slab *j* has been retrieved to satisfy the request of any item *k* before *i*. Otherwise, it is equal to the difference between the number of slabs initially above the slab *j* (D_j) and those that have been already retrieved to satisfy the request of any item k before i (max $(x_{kr}(S'_{kr} + 1)|r \in \{s: |p_j = p_s\})$. Constraints (17) evaluate, for each item i, the span from which the target slab is retrieved (y_i). This evaluation is performed to allow the group of constraints (18) to impose that each item's delivery deadline l_i is respected. Indeed, the first term of these constraints $(\sum_{k=1}^{i} \sum_{j\in J_i,\, A_j=y_i} T_{kr}x_{kr})$ correspond to the retrieval time spent in the span $y_i$to satisfy all the items before *i*.

The proposed model is non-linear and very complex to solve; hence, the authors proposed a heuristics algorithm based on segmented dynamic programming. This algorithm consists of dividing the original problem into several consecutive segments to form a series of subproblems that can be solved with a dynamic programming approach. The union of the solutions of the individual segments returns a solution to the starting problem. Since the partitioning strategy may cause the overall optimal to be lost, two improvement strategies are proposed. The application to real-scale instances has shown that the heuristic is very effective and efficient. It reduces the overall workload of a crane by about 11% on average.

A last contribute to mention is Cheng and Tang (2010). The authors studied an original version of the SSS:

- the concept of work order with the concept of order sequence. Each order sequence has an associated weight, and it is necessary to choose a slab from the yard that meet its technological requirements (thickness, steel grade and weight) to fulfil each order;
- target slabs belonging to the same stack are always retrieved in a descending order to minimise the shifts;
- the concept of slab family is replaced by the concept of order family, intended as a set of slabs belonging to a larger group of items that meets the technological requirements (thickness, degree of steel and weight) of the same order. Since generally, the elements of an order family are more numerous than those of a slab family, there is a wider choice of slabs to select to satisfy an order, leading to a greater chance of reaching solutions with fewer shuffles;
- after each retrieval, the shuffled slabs do not have to be repositioned in the previous position; the authors assume that there are enough stacks to allow the shuffled slabs to be placed where there are no other potential target slabs. Therefore, each slab is moved only once during the entire retrieval process (no repositioning mechanism, single barrier slabs).
- the yard is split into two sub-yards and the workload balance between them is optimized.

In order to introduce the model, further parameters need to be introduced:

W_i	total weight of the *i*th order
w_j	weight of the slab *j;*

The model is formulated as follow:

$$k_1 \sum_{i=1}^{m} \sum_{j=1}^{n} S_{ij} x_{ij} + k_2 \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} S_{ij} x_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} w_j} + k_3 \frac{|N_1 -}{N_1 + N} \tag{21}$$

$$+ k_4 \sum_{i=1}^{m} \left(W_i - \sum_{j=1}^{n} x_{ij} w_j \right) \quad Min!$$

subject to

$$S_{ij} = D_j - \max \left(x_{kr}(D_r + 1) | r \in \left\{ s : \left| p_j = p_s, D_s < D_j \right. \right\}, \right. \quad \forall j \in J_i, i \in I \tag{22}$$

$$N_h = \sum_{i=1}^{m} \sum_{j \in \{r | A_r = h\}}^{a} S_{ij} x_{ij} \quad \forall h \in H = \{1,2\} \tag{23}$$

$$\sum_{j \in J_i} x_{ij} = 1 \quad \forall i \in I \tag{24}$$

$$\sum_{i \in I} x_{ij} w_j \leq W_i \quad \forall i \in I, j \in J \tag{25}$$

$$x_{ij} = 0 \quad \forall j \in \{r \cap J_i = \emptyset, i \in I\} \tag{26}$$

$$x_{ij} = 0/1 \quad \forall j \in J, i \in I \tag{27}$$

The objective function (21) aims at the minimizing the sum of four terms, weighted according to the parameters (k_1, k_2, k_3, k_4) . The first represents the sum of the shuffles necessary to retrieve each slab *j* assigned to the *i*th order ($k_1 \sum_{i=1}^{m} \sum_{j=1}^{n} S_{ij} x_{ij}$). The second one denotes the average shuffles per unit weight, and it is given by the ratio between the sum of the shuffles necessary to retrieve the slabs to satisfy any order ($\sum_{i=1}^{m} \sum_{j=1}^{n} S_{ij} x_{ij}$) and the sum of the weights of all the retrieved slabs ($\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} w_j$). The third term ($k_3 \frac{|N_1 - N_2|}{N_1 + N_2 + 1}$) measures the differences of shuffles between two spans, being N_h the number of shuffles occurred in each span h. Finally, the last term ($k_4 \sum_{i=1}^{m}(W_i - \sum_{j=1}^{n} x_{ij} w_j$)) considers the differences between the total weight of each order and the total weight of the slab assigned to it. In this model the number of shuffles necessary to retrieve a slab *j* for the order *i* is evaluated by the equation (22). According to this latter, the number of shuffles to retrieve a slab *j* to satisfy the order *i* is given by the difference between the initial number of slabs above j and the initial number of slabs (plus 1) of the lowest slab k retrieved that initially was positioned above the slab *j*. Constraints (24) ensure that to each order *i* is assigned one and only one slab *j*. On the contrary the group of constraints (25) ensures that the slab *j* retrieved for each order *i* won't exceed the weight associated to this order. The constraints (26) control that a slab that do not verifies the technological requirements of the order i, are not assigned to this latter. In the end the constraints (27) regulate the binary nature of the variables.

Due to the complexity of the model, the authors proposed a scatter search algorithm to solve the problem, and the results show that the shifts decrease by 36.9% compared to the manual program.

As shown, even if the context of development of all the analysed works is the same, many different models adapt with their specific goal to address and characteristics have been proposed in literature. Hence, it is necessary to highlight the elements that distinguish the various problems through the definition of a framework that systemises all the possible characteristics of the problem to bring out the literature gaps and give a contribute to the extension of the SSS problem's field.

3.5 Classification of existing contributions and research gaps

The Slab Stack Shuffling (SSS) aims at choosing appropriate slabs to retrieve from a set of stacks, to satisfy an order or a set of orders and to optimize given objectives.

The analysis of the literature revealed that different variants of the SSS problem have been investigated but, also, that some of the possible declinations have been neglected. To give a clear picture of what relevant studies have been found and identify which elements have been neglected, we refer to the framework introduced in section 3.3. We recall that it is based on five different characteristics: order typologies, shuffling method, layout characteristics; objective function; deadline constraints.

In Table 3.3, we classify the papers analysed in the previous section, by referring to the introduced framework. Such classification provides interesting indications for scholars that intend to investigate the topic. It is possible to notice that all the analysed papers consider random stacks. This is probably due to the fact that they focus on the case of steel production plants, where slabs, from continuous casting, need to be stored, waiting to undergo the next rolling phase. At this level of the supply chain, the number of items is not particularly high. Hence there is no need to create dedicated stacks. Dedicated layouts make much more sense within yards that serve cutting/assembly centres, where a wider range of slabs typologies are available. The fact that the SSS problem with dedicated stacks has never been addressed in the literature is the first gap we intend to fill.

Another interesting feature concerns the shuffling method. When the repositioning method is adopted, only the not-consecutive approach has been considered. In this case, any shuffled slab is repositioned on the previous stack after each retrieval. In the real case, it is much more realistic to consider that the shuffled slabs are repositioned only after that all the retrievals from the same stack are completed. This assumption certainly adds further complexity to the problem but is very important because of two reasons. First, it could allow obtaining better solutions and, then, it better reflects realistic management of this kind of operations.

As concerns the shuffling method without repositioning, the only contributions assume to position the shuffled slabs on stacks that do not contain target slabs (one-time barrier). However, this assumption seems to be quite restrictive, as enough space has to be available in the yard.

As concerns the order typology, most of the extant studies consider sorted orders. Once again, this feature is much frequent because of the application context analysed in the considered papers. In the steel mill, the retrieval sequence is mandatory because the production is not by batches as in the cutting/assembly centres. It means that the elements of an order are not linked together. Hence, each request cannot be overtaken by any other as in a cutting/assembly centre. In this latter, the elements of a work order, usually belonging to the same batch, do not present any relative sorting because the outputs that have to be produced should be ready all together.

A further feature on which the literature has not paid attention is the constraints on the slabs' deadline. Unlike in the steel industry, in the assembly phase, there is the problem of associating slabs with a certification of quality and relative usability, that has a certain duration and therefore a deadline.

			Tang et al. (2001), Tang et al. (2002), Singh et al. (2004).	Tang and Ren (2010)	Cheng and Tang (2010)	Ferandes et al. (2012)
Order		Item order	X	X		X
		Family order			X	
		Single order	X	X		X
		Multiple order			X	
		Sorted order	X	X		X
		Not-sorted			X	
Shuffling method	**With Repositioning**	*Consecutive*				
		Not-consecutive	X			X
	Without repositioning	*One-time barrier*		X	X	
		Multiple barrier				
Layout characteristics	**Dedicated**	$1\vert 1$				
		$1\vert n$				
		$m\vert 1$				
		$m\vert n$				
	Random	*Random*	X	X	X	X
Deadline constraints	Slab deadline					
	Order deadline			X		
Objective function	Number of shuffles (to be minimized)		X		X	X
	Retrieval time (to be minimized)			X		
	Expired slabs (to be minimized)					
	Span workload (to be balanced)				X	

Table 3.3 – Papers classification

3.6 Conclusions

In this chapter, we introduced the Stacking problems and, in particular, the Slab Stack Shuffling (SSS) problem, which consists of selecting the appropriate slabs to retrieve from a set of stacks, to satisfy an order or a group of orders and minimize the shuffles during the retrieval process.

The main differences between this problem and the other Stacking problems are first analysed. Then, a literature review is conducted to deepen the models and methods proposed to tackle the problem. Moreover, a theoretical framework has been proposed, with the aim of classifying the existing contributions, and highlighting the main research gaps in the literature. The framework is based on five main characteristics: order typologies, shuffling method, layout characteristics, objective functions, deadline constraints. By positioning the analysed paper within the proposed framework, we were able to identify new versions of the problem that could be investigated. In the next chapter, some new models and relative heuristic approaches – able to fill a portion of the identified gaps– will be discussed.

4 Models and heuristics for the Slab Stack Shuffling problem

Summary

In the previous chapter, we released a general framework able to describe various variants of the SSS problem. In this chapter we describe and solve the version of the problem defined as *dedicated item-stack relation* in which slabs and profiles are characterized by a deadline affecting priorities about the retrieval process.

In particular, we illustrate a mathematical model able to describe some variants of the problem. Then we show a tailored heuristic proposed to solve the problem whose performances are test on a large set of generated instances. Finally, we briefly describe the structure of a system able to assume the role of a Decision Support System (DSS) and embed the models and methods presented as optimization tools.

4.1 The problem description

We assume the presence of a set of slabs, stored in preassigned stacks. Each slab is characterized by a deadline which represents the time within which the slab has to be retrieved and used in the production process.

We assume a fixed time horizon divided in periods. For each period an order is defined as a set of requests, i.e. pair of information (item, quantity). Each request can be satisfied only selecting slabs belonging to the specific requested item (*item order*) and each item can be requested more times along the time horizon (*multiple order*). In each period requests can be selected and retrieved in any order (*not sorted order*). In general, in order to retrieve a given slab, shuffle operations have to be performed: in practice some slabs have to be removed and repositioned (*shuffle with repositioning*). We also assume that if two or more requests are satisfied by selecting slabs of the same stack, the shuffled slabs are relocated at the end of all the retrievals of the period (*non-consecutive repositioning*).

We also assume that slabs of an item can be allocated to a set *m* given stacks (*1:m*) each of which can host slab belonging to *n* preassigned items (*1:n*).

The objective function includes two terms to be minimized: the number of shuffles and the number of expired slabs, i.e. slabs not retrieved before their own deadlines.

In the following, we illustrate some mathematical models able to describe different variants of the mentioned problem. In particular we introduce a general model for the case **m|n** then we underline how the model can be adapted in order to describe other versions of the problem.

4.1.1 A mathematical model for the SSS problem for the general case m|n (1:m;1:n)

In order to describe the mathematical model, we introduce the following notation:

$J = \{1, \dots, n\}$	set of slabs, indexed by j ;
$I = \{1, \dots, m\}$	set of items, indexed by i ;
$F = \{1, \dots, p\}$	set of stacks, indexed by f;
$T = \{1, \dots, \bar{T}\}$	set of order periods, indexed by t ;
q_i^t	number of slabs of the item i to retrieve in period t ;
$J_i \subset J$	subset of slabs belonging to item i ($\lvert J_i \rvert \geq q_i$) ;
J'	subset of slabs with $d_j \leq \bar{T}$;
p_j	stack of the slab j;
D_j^0	initial position of the slab j (positions are indicated from the top (position 1) to the bottom of the stack)
d_j	deadline of the slab j;

and the following decision variables

x_j^t	Binary variable equal to 1 if slab j is retrieved at period t, 0 otherwise;
S_f^t	Positive integer variable, representing the number of shuffles associated to the stack f needed to satisfy requests at period t;

The model can be then formulated as follows:

$$\sum_{t\in T}\sum_{f\in F} S_f^t + P\sum_{j\in J'}\left(1-\sum_{t=1}^{d_j} x_j^t\right) \qquad Min! \tag{1}$$

Subject to

$$S_f^t \geq D_j^0 x_j^t - \sum_{s=1}^{t}\sum_{k\in J: D_j^0 \geq D_k^0 \text{ and } p_j = p_k} x_k^s \qquad \forall\, j \in J: p_j = f, \forall\, f \in F,\ t \in T \tag{2}$$

$$\sum_{j\in J_i} x_j^t = q_i^t \qquad \forall\, i \in I, t \in T \tag{3}$$

$$\sum_{t\in T} x_j^t \leq 1 \qquad \forall\, j \in J \tag{4}$$

$$x_j^t = 0 \qquad \forall\, j \in J, t = d_j \dots \bar{T} \qquad (5)$$

$$x_j^t = 0/1 \qquad \forall\, j \in J, t \in T \qquad (6)$$

$$S_f^t \geq 0 \qquad \forall\, f \in F, t \in T \qquad (7)$$

The objective function (1) includes two terms to be minimized: the first one represents number of shuffles of each stack *f* in each period of the time horizon ($\sum_{t\in T}\sum_{f\in F} S_f^t$), while the second one is the number of expired slabs, i.e. number of slabs that are not retrieved ($\sum_{t=1}^{d_j} x_j^t = 0$) before their own deadline within the time horizon ($j \in J'$). The second term is weighted by a penalty coefficient (*P*) that represents the relative importance of the second term in comparison with the first one. In practice, when *P* is equal to 0, the objective function focuses only on minimizing shuffles, while when *P* is equal to 1 each shuffle as the same importance of an extra expired slab. When *P* is high enough, e.g. |*T*|x|*J*|, the target function tends to minimize the number of expired slabs because a single extra expired slab would result in an increment of the objective function equivalent to |*T*|x|*J*| shuffles. Clearly, with appropriate *P* calibrations it is possible to obtain trade-off solutions between the two objectives.

Constraints (2) ensure that for each period *t*, the number of shuffles in each stack is equal to the shuffles needed to retrieve the slab in the lowest position. Therefore, this value is equal to the initial position of the slab *j* in the lowest position, retrieved in *t* $\left(D_j^0 x_j^t \right)$, minus the total number of the slabs over the slab *j*, already retrieved till the period *t* $\left(\sum_{s=1}^{t}\sum_{k\in J: D_j^0 \geq D_k^0 \,\&\, p_j = p_k} x_k^s \right)$.

Constrains (3) ensure that, at each period *t*, the number of slabs retrieved for each item *i* $(\sum_{j\in J_i} x_j^t)$ is equal to the amount quantity q_i^t associated to the request of item *i* at period *t* (request satisfaction).

Constraints (4) assures that any slab *j* can be picked at least at one period *t* ($\sum_{t\in T} x_j^t \leq 1$).

Constraints (5) guarantee that a slab *j* is retrieved at period *t* within its own deadline d_j.

Finally, constraints (6) and (7) regulate the nature of the variables.

4.1.2 Adaptation of the model

The model (1÷7) describes the SSS in the case **m|n**, i.e., when a stack can host slabs belonging to *n* preassigned items (*1:n*) each of which can be allocated to a set of *m* given stacks (*1:m*).

In the following we describe how the models for the cases **1|n** (each item is assigned to only one stack and each stack can host *n* items) and **m|1** (each item may be assigned to a predefined set on *m* stacks, but each stack can host only one item) can be derived from the general model **m|n**.

In the case **1|n**, assuming the presence of a set *F* of stacks, the SSS problem can be tackled considering |*F*| separated problems each of them associated to a single stack, *f*. The problem is not trivial for the presence of the deadline constraints that may drive choices, at each period different from the obvious approach LIFO (Last In-First Out) corresponding to the pick-up of the slabs currently positioned at the top of the assigned stack. Consequently, in presence of a single stack, the corresponding model becomes:

$$\sum_{t\in T} S^t + P\sum_{j\in J'}\left(1-\sum_{t=1}^{d_j} x_j^t\right) \qquad Min! \qquad (1a)$$

subject to

$$S^t \geq\ D_j^0 x_j^t - \sum_{s=1}^{t}\sum_{k\in\bar{J}:D_j^0\geq D_k^0} x_k^s \qquad \forall\, j\in\bar{J}, t\ \in T \qquad (2a)$$

$$\sum_{j\in J_i} x_j^t = q_i^t \qquad \forall\, i\in\bar{I}, t\in T \qquad (3a)$$

$$\sum_{t\in T} x_j^t \leq 1 \qquad \forall\, j\in\bar{J} \qquad (4a)$$

$$x_j^t = 0 \qquad \forall\, j\in\ \bar{J}, t = d_j \ldots \bar{T} \qquad (5a)$$

$$x_j^t = 0/1 \qquad \forall\, j\in\ \bar{J}, t\in T \qquad (6a)$$

$$S^t \geq\ 0 \qquad \forall\, t\in T \qquad (7a)$$

In practise the objective function is reformulated such as S^trepresents the shuffles at each period, and the subset J' is intended as the subset of slabs with a deadline lower than $\bar{T}$ belonging to the set of items hosted in the stack *f* under consideration $\bar{I}$ ($\bar{I}$ = I_f and J' ={$j\in\bar{J}$: $d_j \leq\ \bar{T}$ } where $\bar{J}$ = $\bigcup_{i\in\bar{I}}\ J_i$).

In constraints (2a) S_f^t has been replaced by S^t, and *J* by is subset $\bar{J}$.

In constraints (3a) the set of items *I* is replaced by the subset $\bar{I}$; similar substitutions have been realized for constrains (4a), (5a) and (6a) where *J* is replaced by $\bar{J}$. Finally, constraint (7a) is related to S^t.

In the version **m|1** we assume the presence of a set *F* of stacks dedicated to a single item. So, the corresponding problem as many problems as the number of different items. Therefore, considering the single item distributed on *F* stacks, it is possible to formulate the problem in the following way:

$$\sum_{t\in T}\sum_{f\in\bar{F}} S_f^t + P\sum_{j\in J'}\left(1-\sum_{t=1}^{d_j} x_j^t\right) \qquad Min! \qquad (1b)$$

subject to

$$S_f^t \geq\ D_j^0 x_j^t - \sum_{s=1}^{t}\sum_{k\in J:D_j^0\geq D_k^0 \text{ and } p_j=p_k} x_k^s \qquad \forall\, j\in\bar{J}: p_j = f, \forall\, f\in\bar{F},\ t\in T \qquad (2b)$$

$$\sum_{j\in\bar{J}} x_j^t = q^t \qquad \forall t\in T \qquad (3b)$$

$$\sum_{t\in T} x_j^t \leq 1 \qquad \forall\, j \in \bar{J} \qquad (4b)$$

$$x_j^t = 0 \qquad \forall\, j \in \bar{J}, t = d_j \dots \bar{T} \qquad (5b)$$

$$x_j^t = 0/1 \qquad \forall\, j \in \bar{J}, t \in T \qquad (6b)$$

$$S_f^t \geq 0 \qquad \forall\, f \in \bar{F}, t \in T \qquad (7b)$$

The objective function is reformulated such as S_f^t represents the shuffles at each period and in each stack that hosts the item *i* under consideration ($\bar{F}$), and the subset J' is intended as the subset of slabs with a deadline lower than $\bar{T}$ belonging to the item *i* under consideration ($\bar{F} = F_i$ and $J' = j \in \bar{J} : d_j \leq \bar{T}\}$ where $\bar{J} = J_i$.

In constraints (2b) F has been replaced by $\bar{F}$ and *J* has been replaced is subset $\bar{J}$;

Constraints (3c) regard only the only item i under consideration, and similar substitutions have been realized for constrains (4a), (5a) and (6a) where *J* is replaced by $\bar{J}$. Finally, in constraints (7a), related to S_f^t , *F* has been replaced by $\bar{F}$.

4.1.3 The Model 1|1

A particular version of the model is the one related to the case **1|1**, i.e., when each stack can host only one item and each item is assigned to only one stack. This model, as the previous, can be derived from the general model **m|n.**

In the case 1|1, the number of stacks |*F*| is equal to the number of items |*I*|, hence the SSS problem can be tackled both considering |*F*| or |*I*| separated problems each of them associated to a single couple (stack,item),*(f,i)*. Consequently, in presence of a single item hosted in a single stack, the corresponding model becomes:

$$\sum_{t\in T} S^t + P \sum_{j\in J'} (1 - \sum_{t=1}^{d_j} x_j^t) \qquad Min \qquad (1c)$$

Subject to

$$S^t \geq \ D_j^0 x_j^t - \sum_{s=1}^{t} \sum_{k\in \bar{J}:D_j^0 \geq D_k^0} x_k^s \qquad \forall\, j \in \bar{J}\, , t \in T \qquad (2c)$$

$$\sum_{j\in \bar{J}} x_j^t = q^t \qquad t \in T \qquad (3c)$$

$$\sum_{t\in T} x_j^t \leq 1 \qquad \forall\, j \in \bar{J} \qquad (4c)$$

$$x_j^t = 0 \qquad \forall\, j \in \bar{J}\, , t = d_j + 1 \dots |T| \qquad (5c)$$

$$x_j^t = 0/1 \qquad \forall\, j \in \bar{J}\, , t \in T \qquad (6c)$$

$$S^t \geq 0 \qquad t \in T \qquad (7c)$$

In practise the objective function is reformulated such as S^trepresents the shuffles at each period in the only stack *f* that hosts all the slabs of the item i, and, indeed, the subset J' is intended as the subset of slabs with a deadline lower than $\bar{T}$ belonging to the item i hosted in the stack *f* under consideration ($J' = \{j \in \bar{J}: d_j \leq \bar{T}\}$ where $\bar{J} = J_i$). In constraints (2c) S_f^t has been replaced by S^t, and *J* by is subset $\bar{J}$.

Constraints (3c) regards only the only item i under consideration; and similar substitutions have been realized for constrains (4a), (5a) and (6a) where *J* is replaced by $\bar{J}$. Finally, constraint (7a) is related to S^t.

The next subparagraph gives an example of application, relating to the **1|1** case, in order to understand how the solution provided by the model changes as the penalty coefficient P changes, before of showing the instances generation and relative experimentation.

4.1.4 An illustrative example of the case 1|1

Consider a horizon time of 5 periods (T={1,2,3,4,5}) and the presence of only one stack with *n*=20 slabs of the same item; at each slab is associated a deadline whose value is between 1 and 6. Suppose a set of requests for each period $q^1 = 3$; $q^2 = 2$; $q^3 = 3; q^4 = 4; q^5 = 3$. Figure 4.1 illustrates the elements of this example where nuances of increasing intensity are used for different values of deadlines.

Figures 4.1, 4.2 and 4.3 show the optimal solution for the problem, obtained by solving the model (1c)÷(7c) for three different values of the penalty coefficient, *P*, using the software CPLEX. The model script for the CPLEX is listed in Appendix A.

In the first case, with *P=0,* the objective is the minimization of shuffles. The obtained solution satisfies the request with only 4 shuffles while the number of expired slabs that remain in the stack at the end of the time horizon is equal to 7. Information about the number of expired slabs represented can be visually obtained by looking, in the left side of Figure 4.2, at the slabs obscured in each period.

In the second case, the value of the penalty coefficient, *P*, was fixed to |*T*|x|*J*| = 100, to minimize primarily the number of expired slabs in stock at the end of the time horizon and in secondarily the number of shuffles in the whole retrieval process. In this case, the solution provided by the model is shown in Figure 4.3 following the same representation logic described in previous example.

In this case, it is possible to notice that the number of expired slabs present in stock, at the end of the reference time horizon, has significantly decreased, from 7 to 0 expired slabs, while the number of shuffles has increased, from 4 to 47.

Figure 4.4 compares the choices made by the model for the slabs to be retrieve in the event that *P* = 0 and *P* = 100. As can be seen in this figure, for each period *t*, the model selects different target slabs depending on the value of the penalty parameter *P*.

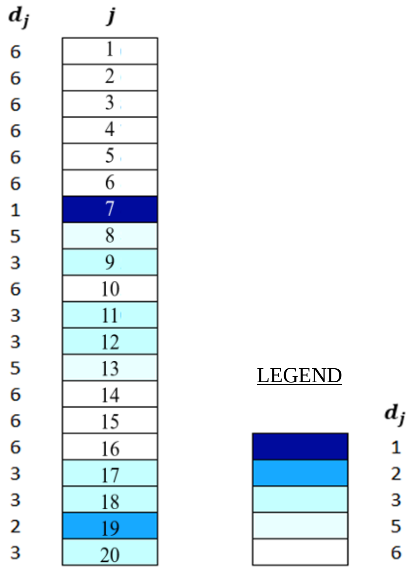


Figure 4.1 – Example of stack representation with deadlines

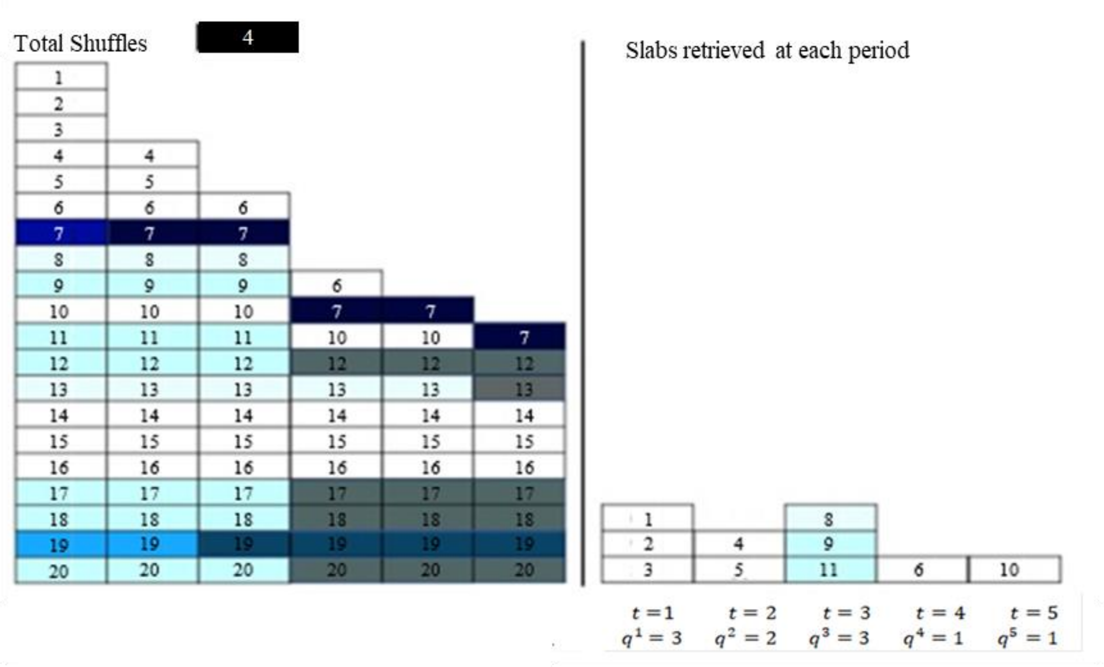


Figure 4.2 - Solution obtained with *P* = 0

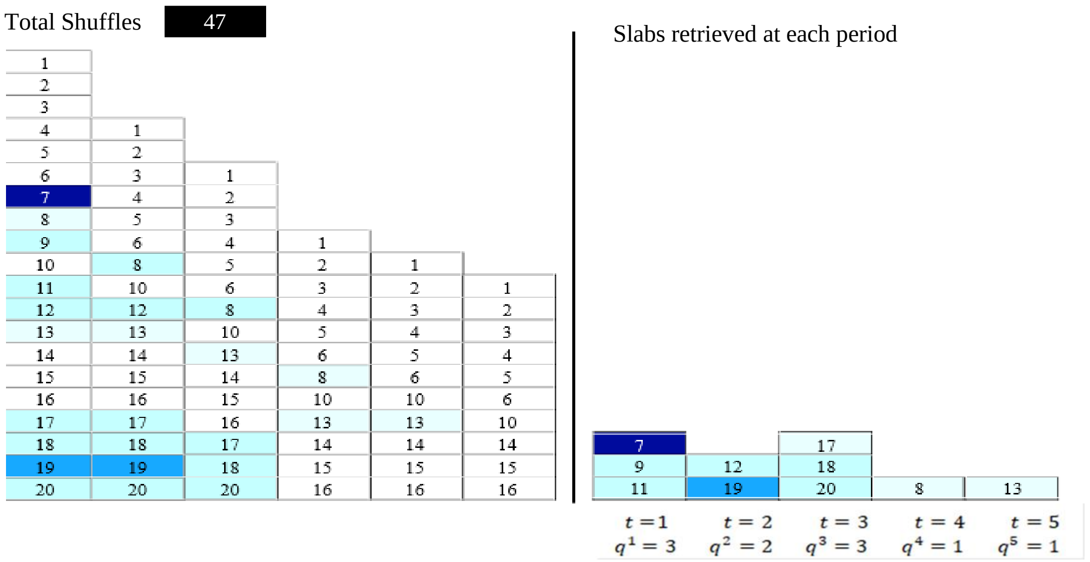


Figure 4.3 - Solution obtained with *P* = 100

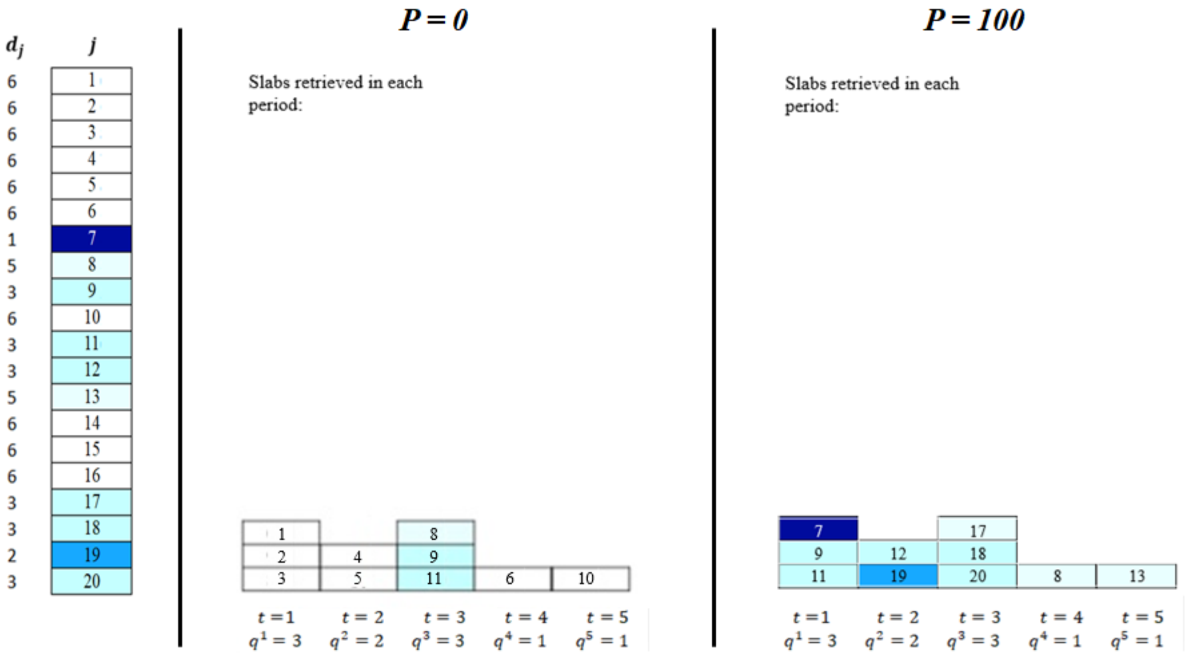


Figure 4.4 – Comparison between solutions with *P* = 0 and *P*= 100

4.2 Computational experiences for the model 1|1

In this section we illustrate the computational experiences related to the model in the version **1|1**. To this aim appropriate test instances have been generated according to the procedure described below.

Instances of the problem have been generated through the creation of a dedicated script in *Matlab* (see Appendix B). Once the input parameters are set, the script automatically provides random instances that can be used to conduct an extensive experimentation of the introduced model. The key input parameter is the number of periods of the time horizon, $|T|$, while the other parameters (number of slabs $|J|$, requests vector $(q^t, t \in T)$, deadlines $(d_j, j \in J)$, and initial slabs' position in the stack $(D_j^0,\ j \in J)$) are defined according to the procedures described below. Instances are generated, fixing $|T| = 5{,}10{,}15{,}20{,}25$.

The number of requests for each period $(q^t\ \forall\ t \in T)$ are randomly generated, according to a uniform integer distribution within the range $[q_{min} = 0,\ q_{max} = 4]$. The number of slabs |*J*| is fixed to ensure that there are always enough slabs in stock to meet total demand $\sum_{t \in T} q^t$ so it is set $|J| = q_{max} * |T|$. Hence, as $|T| = 5, 10, 15, 20$ and 25 |*J*| assumes values equal to 20, 40, 60, 80 and 100.

The slabs' deadlines and positions within the stack are the two key elements in the generation of instances. Indeed, an appropriate selection of these parameters allows for generating non-trivial instances. Trivial instances could be considered cases in which a LIFO (Last-In, First-Out) retrieval rule is able to obtain the optimal solution. In the following, the procedure to assign the initial positions and the deadlines to get feasible and non-trivial instances is described.

Obviously, at each period *t*, in case of a request of q^t slabs, it is necessary the presence in the stack of at least q^t slabs with a deadline higher or equal to *t*. Consequently, iteratively, at each period *t*, the q^t deadlines' values are fixed by randomly choosing a value between *t* and *t*+ 2. About each of the other slabs, a random deadline is assigned in such a way that the 30% of them expires within the reference time horizon, while the residual ones beyond it.

The initial positions of the slabs in the stack are defined in order to reproduce conditions typical of real cases. In practice, in an actual management, slabs with lower deadlines are expected to be located in the lower positions of stacks since they have been allegedly delivered earlier. Therefore, we opt for positioning a consistent percentage of slabs (60%) with lower deadline in the lowest half of the stack. Figure 4.5 shows an example of initial assignment of deadlines, respecting this requirement. Considering that the problem is bi-objective, through the assignment of values to the penalty *P*, it is possible to produce different solutions of the Pareto frontier, i.e. the set of not dominated solutions, in the sense that, for each solution, there are no better solution for both the considered objectives.

As illustrative example, Figure 4.6 shows the solutions obtained on two different instances with $|T|=10$ and $|J|$=40, in the case of $P=0$ and $P = |T| \cdot |J| = 10 \cdot 40 = 400$.

As it can be expected for high penalty values, the total number of shuffles required for the retrieval operations is much higher than in the case where $P = 0$. In particular, considering the objective of the number of shuffles, its *lower bound* (LB_S) and its *upper bound* (UB_S) are the values of shuffles obtained with $P = 0$ and $P = |T|\text{x}|J|$ respectively. On the other hand, *lower bound* (LB_{ES}) and *upper bound* (UB_{ES}) for the number of expired slabs are provided when $P = |T|\text{x}|J|$ and $P = 0$ respectively. It is important to note that the minimum value of slabs expired at the end of the retrievals (LB_{ES}) is not necessarily zero, as it depends on deadlines assigned to the initial positioning of the slabs.

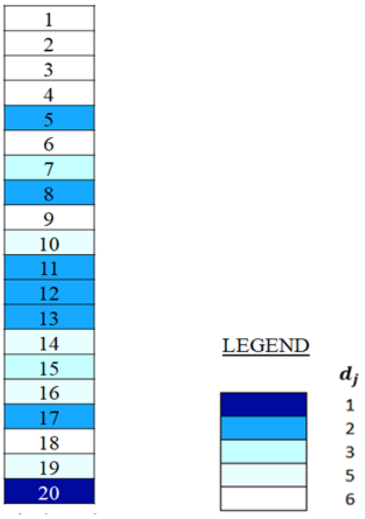


Figure 4.5 – 1.1 Example of non trivial deadline assignament

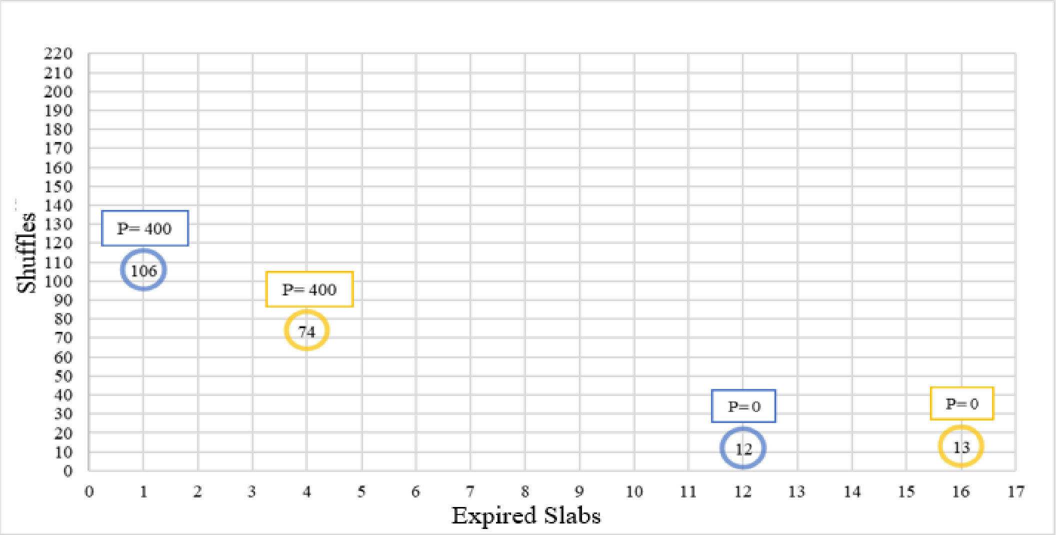


Figure 4.6 – Model solutions for two instances (10, 40) with P=0 and P=400

The tuning of P is crucial to explore the Pareto frontier and drive the decision maker toward regions characterized by a lower number of shuffles or a lower number of expired slabs. This can be shown applying the so-called "ε-constraint method" to represent the Pareto frontier. In practice, starting from the number of expired slabs associated with the solution obtained with $P = 0$, iteratively, we add to the model (1c)-(7c), a further constraint

$$\sum_{j\in J_s}\left(1-\sum_{t\in T:t\le d_j} x_j^t\right)\le Y_k \qquad (8)$$

that fix an upper bound to the total number of slabs that can expire at iteration k. Therefore, the value of upper bound is iteratively updated by setting $Y_{k+1} = Y_k - 1$, until the minimum number of expired slabs (LB_{ES}) is reached.

The obtained Pareto solutions for the two illustrative examples are depicted in Figure 4.7.

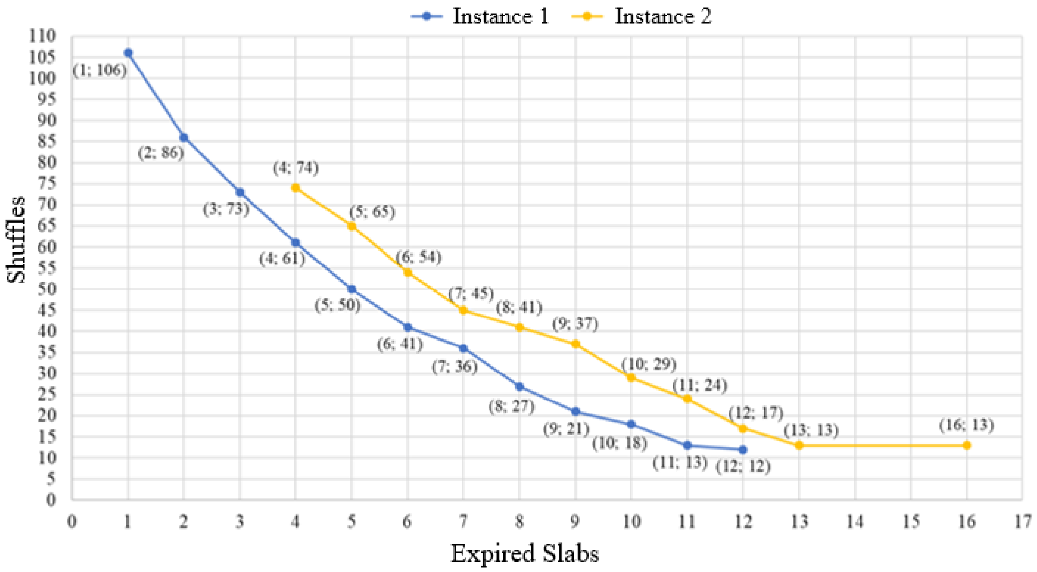


Figure 4.7 – Pareto frontiers for the illustrative examples

4.3 Analysis of the results

In order to test the model, computational experiences have been carried out on random instances assuming the combination of values ($|T|$, $|J|$) = {(5, 20); (10, 40); (15, 60)}. For each combination of values 50 random instances are considered. As previously illustrated, in order to represent the Pareto frontier for a single instance, it is necessary to set the value of the penalty P. To this aim, lower and upper bounds for each objective are evaluated. In particular, for a given instance, the upper bound for the number of shuffles (UB_S) and the lower bounds for the number of expired slabs (LB_{ES}) are determined by solving the model with $P = |T|\text{x}|J|$; on the other hand, the lower bound for the number of shuffles (LB_S) and the upper bounds for the number of expired slabs (UB_{ES}) are calculated setting $P = 0$. Therefore $P_m = \left\lceil \frac{UB_S - LB_S}{UB_{ES} - LB_{ES}} \right\rceil$ represents the slope of the line passing through the extreme points of the Pareto's frontier of coordinates (LB_{ES}, UB_S),

(UB_{ES}, LB_S). In order to get an approximation of the Pareto's frontier, the model has been solved for $P = \{P_m, P_m/2\, , P_m/4\,\}$ using IBM ILOG CPLEX Optimization Studio 12.9 software. Then, a text file that summarizes the results obtained in terms of the total number of shuffles, the number of expired slabs at the end of the reference time horizon and computing time spent resolving the model is obtained as output. Tables indicating the results obtained for each test instance in terms of objective functions are reported in Appendix C.

.

Table 4.1 shows the minimum, the maximum and the average computing times for each combination ($|T|$,$|J|$) = {(5, 20); (10, 40); (15, 60)}, in correspondence of the different adopted penalty values. In evaluating the computing time, a time limit of one hour has been assumed. As it can be noted, computing times present significant variations for the same combinations but generally tend to dramatically increase with the dimension of the instances. In particular with ($|T|$,$|J|$) = (15, 60) and $P = P_m$, 14 instances exceeded the time limit of one hour. In order to show the actual computing times, these instances have been solved to optimality with no computing time limits. Table 4.2, reporting the computing times (in hours) for these "difficult" instances, show how in the case of instance no. 50, the computing time reaches 11,46 hours. Tables indicating the results obtained for each test instance in terms of objective functions are reported in Appendix C. In order to underline the extensive required computing times, the model has also been tested for larger instances with ($|T|$,$|J|$) = {(20, 80); (25, 100)}, only in the case of the extreme points of the Pareto's frontier with P=0 and P=$|T|$x$|J|$. Assuming a time limit of one hour, Table 4.3 shows the number of instances solved to optimality (over 50) and minimum, maximum and average computing times related to these instances. Tables indicating the results obtained for each test instance in terms of objective functions and computing times are reported in Appendix C.

$\lvert T\rvert$	$\lvert J\rvert$	P	*Minimum computing times (s)*	*Maximum computing times (s)*	*Average computing times (s)*
5	20	$\lvert T\rvert x\lvert J\rvert$	0.16	0.86	0.35
5	20	P_m	0.17	1.46	0.42
5	20	$P_m/2$	0.18	1.11	0.35
5	20	$P_m/4$	0.17	0.74	0.32
5	20	0	0.16	0.83	0.30
10	40	$\lvert T\rvert x\lvert J\rvert$	0.26	23.28	2.82
10	40	P_m	3.55	678.19	41.66
10	40	$P_m/2$	2.74	55.88	12.58
10	40	$P_m/4$	1.64	46.02	8.44
10	40	0	0.22	9.95	2.59
15	60	$\lvert T\rvert x\lvert J\rvert$	1.88	225.49	841.81
15	60	P_m	27.98	∞	∞
15	60	$P_m/2$	11.86	1077.70	180.00
15	60	$P_m/4$	8.16	2369.13	194.93
15	60	0	4.08	362.09	39.72

Table 4.1 – Computing times (min, max, and average) for (|*T*|,|*J*|) = {(5, 20); (10, 40); (15, 60)}

Instance number	|*T*|	|*J*|	*Computing times (h)*
4	15	60	1.09
16	15	60	2.19
18	15	60	5.70
19	15	60	1.03
26	15	60	3.88
27	15	60	1.46
31	15	60	10.03
39	15	60	1.54
40	15	60	2.91
41	15	60	5.48
44	15	60	1.05
45	15	60	4.96
47	15	60	6.03
50	15	60	11.46

Table 4,2 – Computing times for "difficult" instances for (|*T*|,|*J*|) = {(15, 60)} and *P*=*P*$_m$

|*T*|	|*J*|	*P*	*Optimal solutions*	*Minimum computing time (s)*	*Maximum computing time (s)*	*Average computing time(s)*
20	80	|*T*|x|*J*|	50/50	4.91	3607.61	832.95
20	80	0	44/50	0.76	∞	636.81
25	100	|*T*|x|*J*|	12/50	5.85	∞	394.86
25	100	0	7/50	25.98	∞	890.45

Table 4,3 - Computing times (min, max, range and average) for (|*T*|,|*J*|) = {(20, 80); (25, 100)}

4.4 A Heuristic approach for the solution of the (1|1) SSS problem

The obtained computing times to optimally solve instances of the (1|1) SSS problem highlights the need of heuristic approaches to reduce computing times and to extend the dimension of solvable instances. Therefore, we illustrate the proposal of an improvement algorithm based on a local search procedure. A local search procedure does not guarantee an optimal solution, but it usually attempts to find a solution that is better than the current one in its neighbourhood. Two solutions are “neighbours”, if one can be obtained through a well-defined modification of the other. At each iteration, a local search procedure performs a search within the neighbourhood and evaluates the various neighbouring solutions. The procedure either accepts or reject the best solution in the neighbourhood, based on a given acceptance-rejection criterion. Usually, the adopted criterion is the objective function. Then the procedure accepts the best solution in the neighbourhood if its objective function value is better than the current one; otherwise, it rejects it and the algorithm stops.

As typical of local search approaches, the proposed procedure is characterized by three steps: the individuation of an initial solution, the local search phase and the stopping criterion. In particular, two different constructive procedures have been defined to provide the initial solution. The first one is able to produce random solutions, while the second one to the minimization of the expired slabs. Since both the heuristics can generate solutions in a very short time, it can be chosen the best provided solution, as initial solution for the successive improvement phase (see Figure 8).

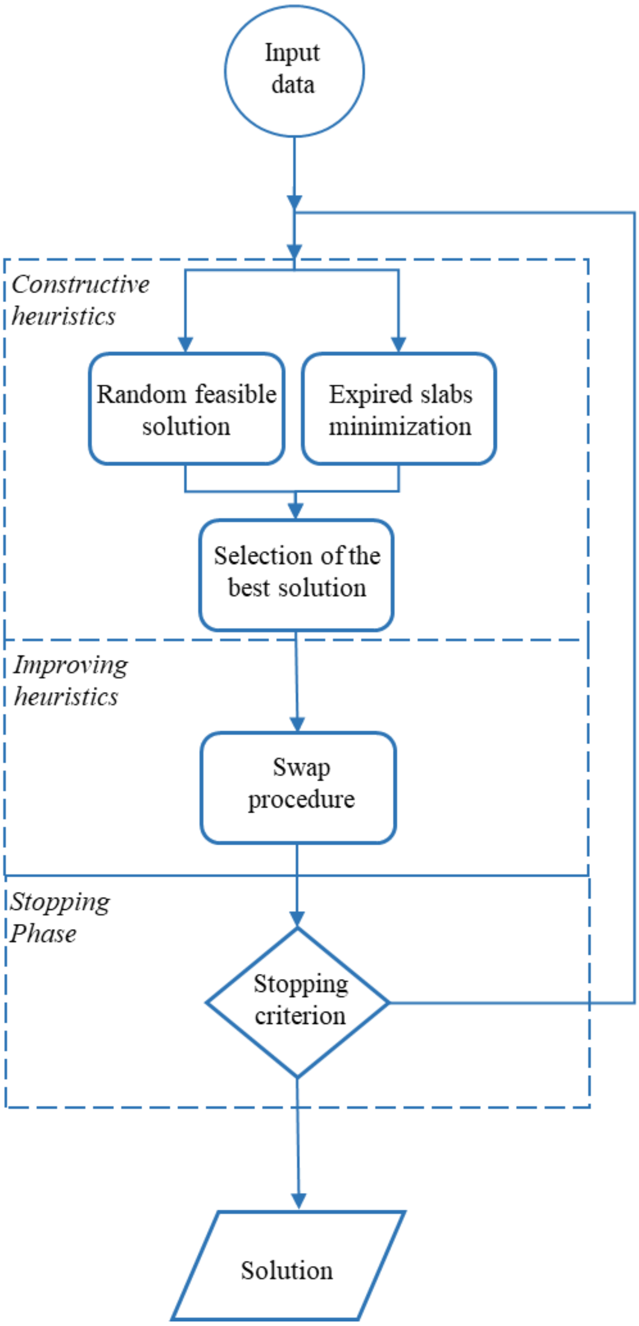


Figure 4,8 – Flowchart of the proposed heuristic

4,4,1 Individuation of an initial solution

In building an initial feasible solution, it is essential to consider the bi-objective nature of the problem. As shown, by tuning the penalty coefficient, it is possible to obtain different solutions in terms of number of shuffles and number of expired slabs. As highlighted in the flowchart of Figure x, two different constructive procedures have been developed. The first one is oriented to the minimization of the number of shuffles, while the second one to the minimization of the expired slabs.

Both the procedures are based on an iterative assignment of the slabs present in the initial stack to the period of retrieving in order to satisfy the amount of slabs requested at each period. In practices at generic iteration k, the algorithm assigns a given slab j to a period t at which it is retrieved. However, they differ according the adopted criterium for the assignment.

In order to describe the heuristics, at iteration k, we say that a slab j with its own deadline d_j can be preliminary assigned to one of the "feasible" period t where a period is considered feasible if (a) it allows to satisfy the expiration constraint ($d_j \geq t$) and (b) the slabs already assigned to that period t till the iteration k does not exceed the request associated to t (q^t). On the other hand, at each period t it is possible to associate the set of "feasible" slabs J_t, i.e the set of slabs with ($d_j \geq t$) still not assigned to any other period. Then the slab j can be definitively assigned to the period t if it is possible to satisfy the requests of the periods successive to t with feasible slabs not still assigned. This can be done with a backward procedure from the last period $\underline{T}$ to the period $t+1$: at generic period $s: t+1 \leq s \leq T$, it is sufficient to verify that the sum between the feasible slabs at s ($|J_s|$) and the eventual difference between the sum of feasible slabs for the successive periods and the sum of the requests for these periods is at least equal to the request at s.

If there are no feasible periods for the slab j at iteration k, it is removed from the stack and it is counted as "expired". To represent this circumstance, we introduce a binary variable e_j equal to 1, if the slab j is expired, 0 otherwise.

We illustrate this aspect through an example reported in Appendix D

➢ *Constructive heuristic to generate random feasible solutions*

In this procedure, at each iteration, a slab is randomly selected from the slack and it is temporary assigned to one of its feasible periods, randomly chosen. This assignment is definitive if it is possible to satisfy the requests of the periods successive to t with feasible slabs not still assigned, excluding the selected slab. If there are no feasible periods for the slab j at iteration k, it is removed from the stack and it is counted as "expired" ($e_j = 1$). The procedure needs a number of iterations equal to the sum between the total number of requested slacks in the horizon time T ($\sum_{t \leq T} q^t$) and the total number of expired slabs ($\sum_{j\in J} e_j$). The procedure is described, using a pseudocode, in Appendix E.

➢ *Constructive heuristic oriented to the minimization of the expired slabs*

This heuristic individuates the optimal solution in terms of minimization of expired slabs. Denote with E_t the set of slabs with deadlines equal to *t* ($E_t = \{j: d_j = t\}$). If the cardinal of this set is not higher than the request for that period ($|E_t| \leq q^t$), the set E_t is definitively assigned to *t*. In case $|E_t| < q^t$, for this period there is a residual request $q^t_{res} = q^t - |E_t|$ to be satisfied. Otherwise, ($|E_t| > q^t$), we randomly extract, from E_t, q^t slabs that are definitively assigned to *t*. The total set of not assigned slabs, are then used to satisfy the residual requests. In particular, starting from *t*=1, we satisfy the residual requests assigning the remaining slabs according to their deadlines. In case there are more slabs with the same deadlines than residual request, the assignment is randomly performed. The procedure is described, using a pseudocode, in Appendix F.

4.4.2 The local search phase

The local search phase is realized through a modification of the current solution performing a pairwise interchange between two slabs k and j (swap (k, j)). However, it is necessary to verify that the *swap*(k, j) is not trivial and feasible. Denote with d_k, d_j and t_k, t_j the deadlines and the assigned periods respectively of the slabs k and j in the current solution, and assume that $t_k \geq t_j$ and, conventionally, that $t_k = 0$ if the slab *k* is not used, in the current solution, to satisfy any request.

A *swap*(k, *j*) is trivial if $t_k = t_j$: this means either that the two slabs have been assigned to the same period, or that the two slabs are not been used to satisfy any request

A not trivial *swap*(k, *j*) is feasible if it leads to a feasible solution of the problem. Indeed, while a swap between k and j does not affect the constraints about the request satisfaction, it can violate the deadlines constraints. It is easy to verify that a not trivial swap(*k*,*j*)) is feasible if $d_j \geq t_k$.

Therefore, the procedure, at the first step, consider all the possible feasible swaps between the slab with highest position, s, in the stack and any other slab below it. If the best solution in the neighbourhood obtained by the swap(*s*,*j**) is better than the current one in terms of objective function ($\sum_{t\in T} S^t + P\sum_{j\in J'}(1-\sum_{t=1}^{d_j} x_j^t)$), the swap(*s*,*j**) is performed, and algorithm continues considering the neighbourhood defined by swap(*s*,*j*), with $s \equiv j^*$; otherwise the procedure considers the swaps involving the slab successive to the slab s, and so on. The procedure, described using a pseudocode in Appendix G, stops when a maximum number of iterations is reached.

4.5 Computational experiences

In order to test the heuristic, computational experiences have been carried out on the same 50 instances randomly generated with (|*T*|,|*J*|) = {(20, 80); (25, 100)} to test the effectiveness of the proposed heuristic, in the case *P*=0 and *P*=|*T*|x|*J*|. For a given instance, the quality of the solution produced by the heuristic is evaluated in terms of percentage error over the solution individuated by CPLEX after one hour of computing times. About the computational times, the comparison is done fixing the stopping criterion equal to 100 iterations; in the case of the solution provided by CPLEX, when the optimal solution is not reached, and the time limit of 3600 seconds is indicated.

Tables 4.4 summarize the obtained results in terms of average, the standard deviation, maximum and the minimum value of the percentage error. The detailed results for each instance are reported in Appendix H.

The performances of the heuristic appear quite interesting. In particular in the case of *P* = |*T*|x|*J*|, even if the algorithm only in few cases reaches the optimal solution, is characterized by very limited percentage errors (0,09% and 0,05%) with very low values of the maximum error. Relatively worst seems to be the results in the case of *P*= 0. Indeed, in this case the average percentage error is of 6,18% and 4,22 % for (|*T*|,|*J*|) = (20, 80) and (|*T*|,|*J*|) = (25, 100) respectively, with a maximum percentage error around the 20% in both the cases. However, this difference is mainly due to the term of expired slabs. If we analyse the differences in terms of shuffles between the solution of the model and the solution provided by the heuristic (Table 4.5) is very low.

Table 4.6 shows the comparison of the performances in terms of minimum, maximum and average computing time that highlights how the computing times of the heuristic are in the order of few minutes.

J	*T*	*P*	Objective Function: Percentage error			
			Average	*Dev.std*	*Minimum*	*Maximum*
80	20	|*T*|x|*J*|	0.09%	0.07%	0.00%	0.30%
100	25	|*T*|x|*J*|	0.05%	0.05%	-0.01%	0.25%
80	20	0	6.18%	6.35%	-4.44%	22.22%
100	25	0	4.22%	9.51%	-17.27%	23.08%

Table 4.4– Average, standard deviation, minimum and maximum value of the percentage error on 50 randomly generated instances

J	*T*	*P*	Number of shufflesr			
			Average	*Dev.std*	*Minimum*	*Maximum*
80	20	0	2.78	3.63	-4	19
100	25	0	2.12	9.02	-24	17

Table 4.5– Average, standard deviation, minimum and maximum value of the percentage error on 50 randomly generated instances

J	*T*	*P*	Computing times: Model solution			Computing times: Heuristic		
			Minimum (sec)	*Maximum (sec)*	*Average (sec)*	*Minimum (sec)*	*Maximum (sec)*	*Average (sec)*
80	20	$|T|$x$|J|$	10.38	3607.61	1322.35	114.99	245.6	190.75
100	25	$|T|$x$|J|$	40.19	30608.23	3034.40	371.66	673.02	522.99
80	20	0	0.76	3605.16	932.37	160.63	564.56	263.72
100	25	0	25.98	3618.01	3223.73	346.63	833.41	562.85

Table 4.6 – Average, Minimum and Maximum value of the heuristic computing time

4.6 Characteristics of an implemented software system

The main objective of the collaboration within the partnership with the firm was the overall re-engineering of the processes with the aim of improving the production and logistic performances. Once the analysis of the processes has been performed, the achievement of this goal has required the digitalization of the procedures. A first attempt performed in this direction has been oriented to the innovation of the information systems organization, still based on an outmoded midrange computer platform referred to generically by the umbrella term AS/400. Then it has defined a project to define a new architecture system ad-hoc designed to efficiently represent and manage the peculiarities of the context, with the objective of creating a complete Enterprise Resource Planning (ERP). The whole architecture has been based on a PostgreSQL relational database, implemented in a complex web-based Management and Decision Support System (DSS).

A brief illustrative overview of the main features and tools embedded in the System is provided in the following.

Starting from the dashboard of the DSS (Figure 4.) it is possible to notice the numerous options and offered tools provided that can be distinguished in *In-bound*, *Out-bound*, *Production* and *Storage* & *Retrieval* Managing menus.

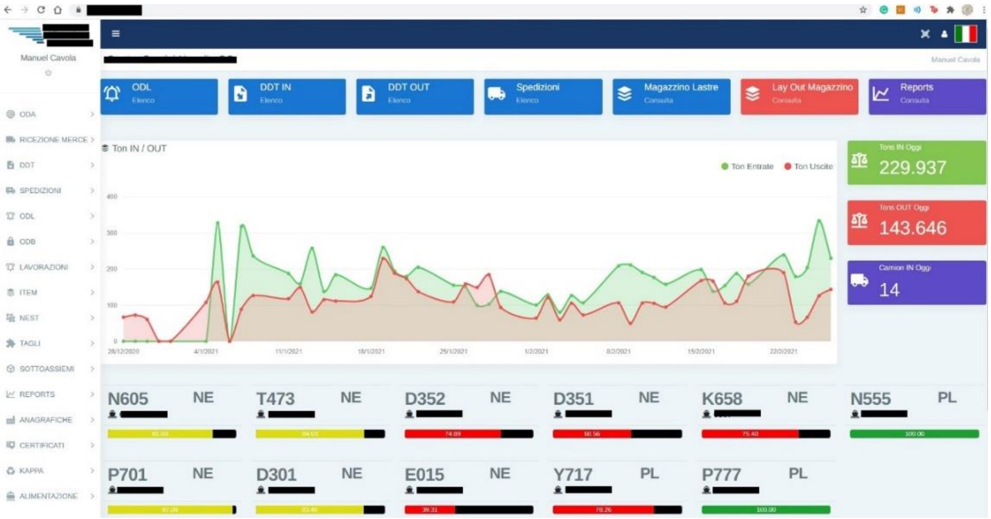


Figure 4.9 – Decision Support System Dashboard

Within the *In-bound* menu, different procedures has been implemented such as Figure 4.10:

- Managing of the delivery notes
- Uploading and downloading of the quality certifications
- Semi-trailer check-in

Concerning the *Out-bound* operations, the main implemented procedures have been (Figure 4.11):

- Semitrailer loading
- Listing of deliveries

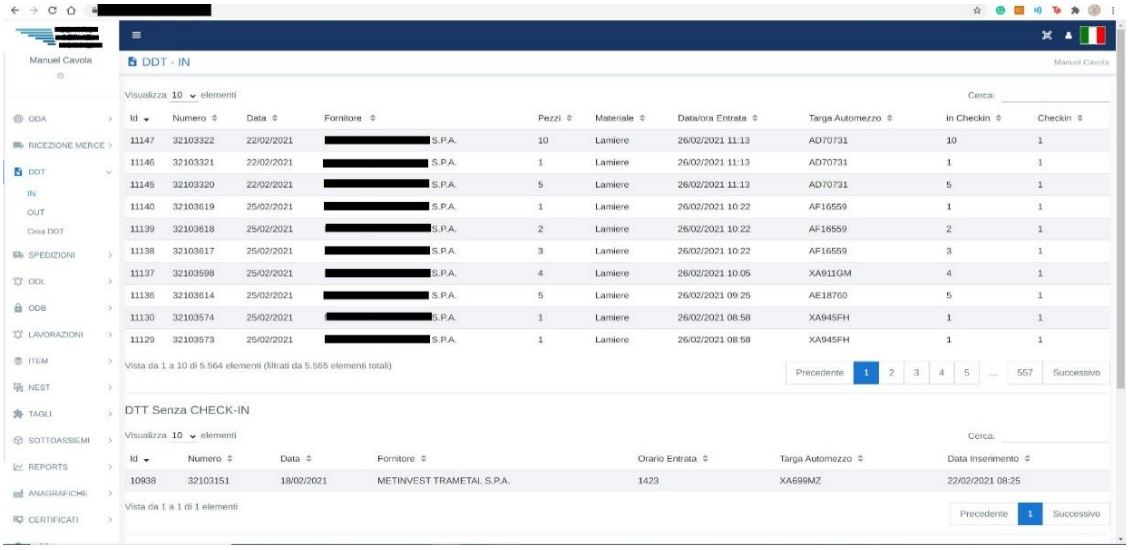
About the *Production* Management procedures, peculiarities procedures have been implemented to support decision for cutting slab optimization in order to reduce processing scraps. A not exhaustive list of implemented procedures are provided in the following. Some of the screenshots related to these procedures are depicted in Figure 4.12.

- Order management
- Cutting scheduling
- Digitalization of the cutting schemes

Finally, handling procedures have been implemented in relation to the *Storage* and *Retrieval* management. As above explained, these two fundamental logistic processes are very peculiar in the context and require appropriate optimization tools that may support the decision makers from the definition of the layout for the slabs and profiles stacking to the retrieval sequencing.

Figure 4.13 show Some of the screenshots related to these procedures:

- Real-time stock level monitoring for each stack and stall
- Slab database management
- Logistic flow description



DDT - IN

Visualizza 10 elementi

Cerca:

Id	Numero	Data	Fornitore	Pezzi	Materiale	Data/ora Entrata	Targa Automezzo	in Checkin	Checkin
11147	32103322	22/02/2021	[redacted] S.P.A.	10	Lamiere	26/02/2021 11:13	AD70731	10	1
11146	32103321	22/02/2021	[redacted] S.P.A.	1	Lamiere	26/02/2021 11:13	AD70731	1	1
11145	32103320	22/02/2021	[redacted] S.P.A.	5	Lamiere	26/02/2021 11:13	AD70731	5	1
11140	32103619	25/02/2021	[redacted] S.P.A.	1	Lamiere	26/02/2021 10:22	AF16559	1	1
11139	32103618	25/02/2021	[redacted] S.P.A.	2	Lamiere	26/02/2021 10:22	AF16559	2	1
11138	32103617	25/02/2021	[redacted] S.P.A.	3	Lamiere	26/02/2021 10:22	AF16559	3	1
11137	32103598	25/02/2021	[redacted] S.P.A.	4	Lamiere	26/02/2021 10:05	XA911GM	4	1
11136	32103614	25/02/2021	[redacted] S.P.A.	5	Lamiere	26/02/2021 09:25	AE18760	5	1
11130	32103574	25/02/2021	[redacted] S.P.A.	1	Lamiere	26/02/2021 08:58	XA945FH	1	1
11129	32103573	25/02/2021	[redacted] S.P.A.	1	Lamiere	26/02/2021 08:58	XA945FH	1	1

Vista da 1 a 10 di 5.564 elementi (filtrati da 5.565 elementi totali)

DTT Senza CHECK-IN

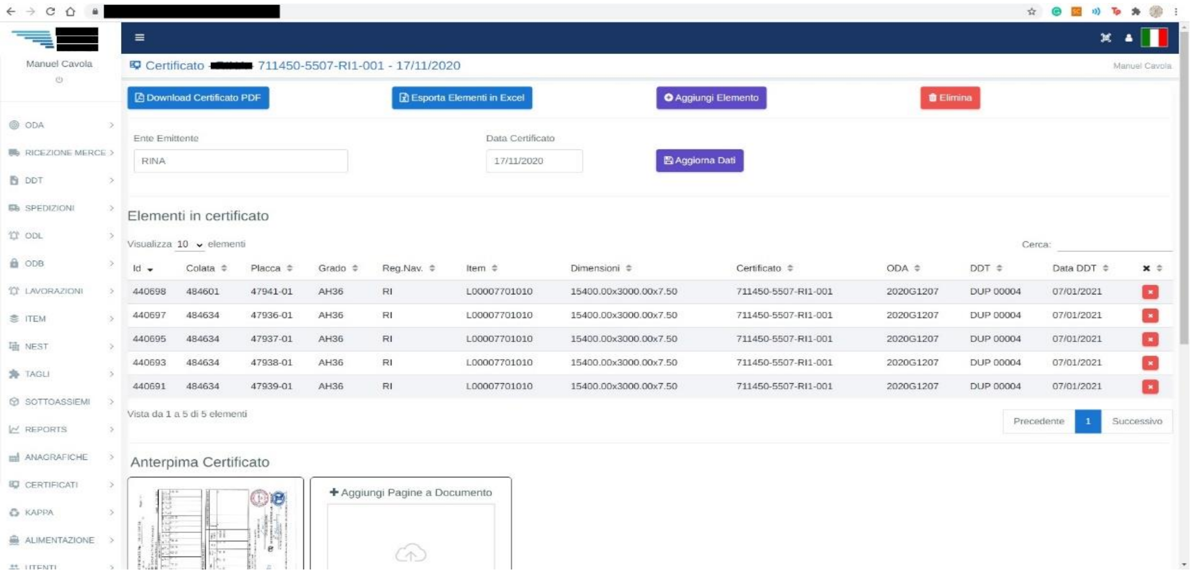
Visualizza 10 elementi

Cerca:

Id	Numero	Data	Fornitore	Orario Entrata	Targa Automezzo	Data Inserimento
10938	32103151	18/02/2021	METINVEST TRAMETAL S.P.A.	1423	XA699MZ	22/02/2021 08:25

Vista da 1 a 1 di 1 elementi

Delivery notes



Certificato [redacted] 711450-5507-RI1-001 - 17/11/2020

Ente Emittente: RINA

Data Certificato: 17/11/2020

Elementi in certificato

Visualizza 10 elementi

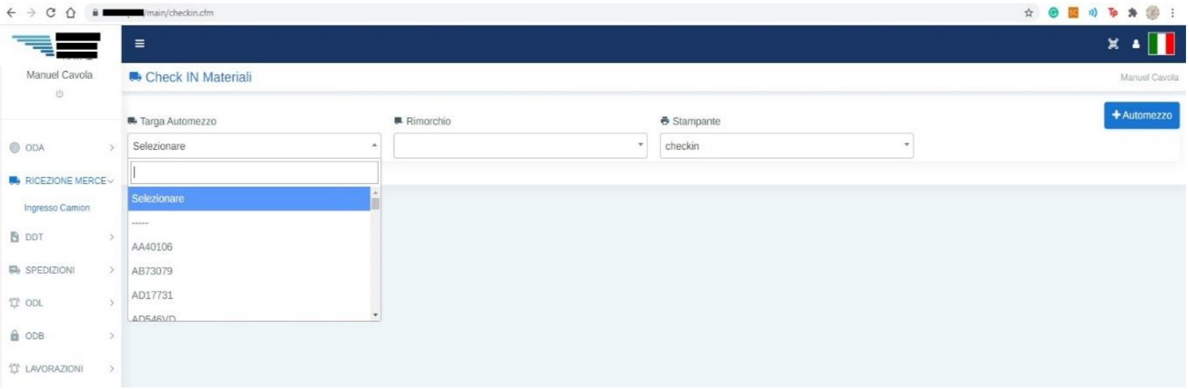
Cerca:

Id	Colata	Placca	Grado	Reg.Nav.	Item	Dimensioni	Certificato	ODA	DDT	Data DDT	✖
440698	484601	47941-01	AH36	RI	L00007701010	15400.00x3000.00x7.50	711450-5507-RI1-001	2020G1207	DUP 00004	07/01/2021	✖
440697	484634	47936-01	AH36	RI	L00007701010	15400.00x3000.00x7.50	711450-5507-RI1-001	2020G1207	DUP 00004	07/01/2021	✖
440695	484634	47937-01	AH36	RI	L00007701010	15400.00x3000.00x7.50	711450-5507-RI1-001	2020G1207	DUP 00004	07/01/2021	✖
440693	484634	47938-01	AH36	RI	L00007701010	15400.00x3000.00x7.50	711450-5507-RI1-001	2020G1207	DUP 00004	07/01/2021	✖
440691	484634	47939-01	AH36	RI	L00007701010	15400.00x3000.00x7.50	711450-5507-RI1-001	2020G1207	DUP 00004	07/01/2021	✖

Vista da 1 a 5 di 5 elementi

Anterpima Certificato

Quality certifications database



Check IN Materiali

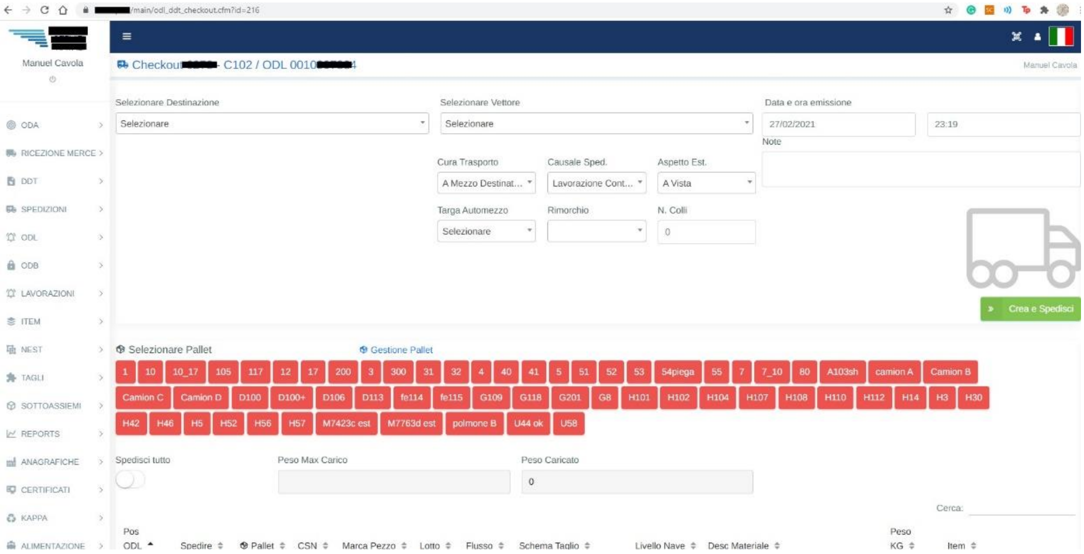
Targa Automezzo: Selezionare

Rimorchio:

Stampante: checkin

Semi-trailers check-in registration

Figure 4.10 – Screenshots of the implemented Inbound procedures



Checkout [redacted] - C102 / ODL 0010 [redacted] 4

Selezionare Destinazione: Selezionare

Selezionare Vettore: Selezionare

Data e ora emissione: 27/02/2021 23:19

Note:

Cura Trasporto: A Mezzo Destinat...

Causale Sped.: Lavorazione Cont...

Aspetto Est.: A Vista

Targa Automezzo: Selezionare

Rimorchio:

N. Colli: 0

Selezionare Pallet

Spedisci tutto

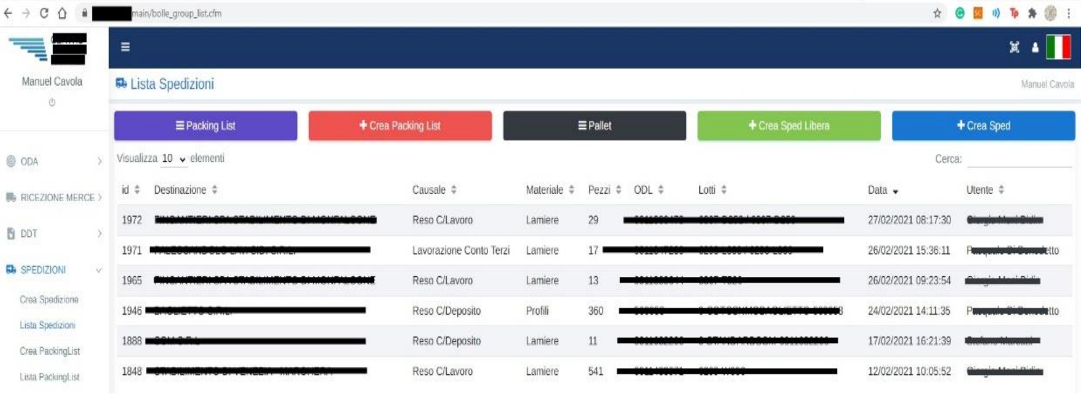
Peso Max Carico:

Peso Caricato: 0

Cerca:

Pos ODL	Spedire	Pallet	CSN	Marca Pezzo	Lotto	Flusso	Schema Taglio	Livello Nave	Desc Materiale	Peso KG	Item

Semitrailer loading



Lista Spedizioni

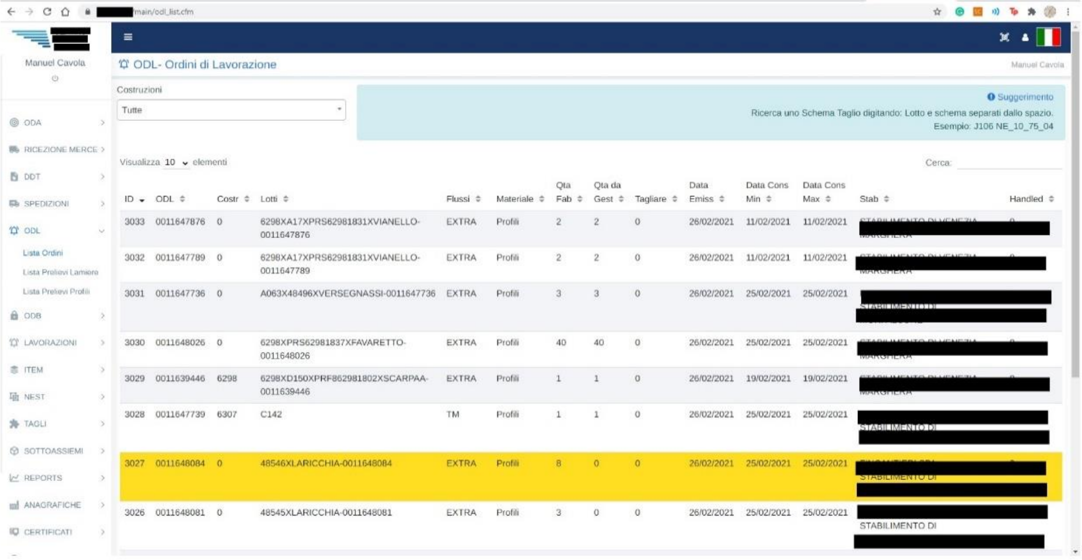
Visualizza 10 elementi

Cerca:

id	Destinazione	Causale	Materiale	Pezzi	ODL	Lotti	Data	Utente
1972	[redacted]	Reso C/Lavoro	Lamiere	29	[redacted]	[redacted]	27/02/2021 08:17:30	[redacted]
1971	[redacted]	Lavorazione Conto Terzi	Lamiere	17	[redacted]	[redacted]	26/02/2021 15:36:11	[redacted]
1965	[redacted]	Reso C/Lavoro	Lamiere	13	[redacted]	[redacted]	26/02/2021 09:23:54	[redacted]
1946	[redacted]	Reso C/Deposito	Profili	360	[redacted]	[redacted]	24/02/2021 14:11:35	[redacted]
1888	[redacted]	Reso C/Deposito	Lamiere	11	[redacted]	[redacted]	17/02/2021 16:21:39	[redacted]
1848	[redacted]	Reso C/Lavoro	Lamiere	541	[redacted]	[redacted]	12/02/2021 10:05:52	[redacted]

Listing of deliveries

Figure 4.11 – Screenshots of the implemented Outbound procedures



ODL- Ordini di Lavorazione

Manuel Cavola

Costruzioni: Tutte

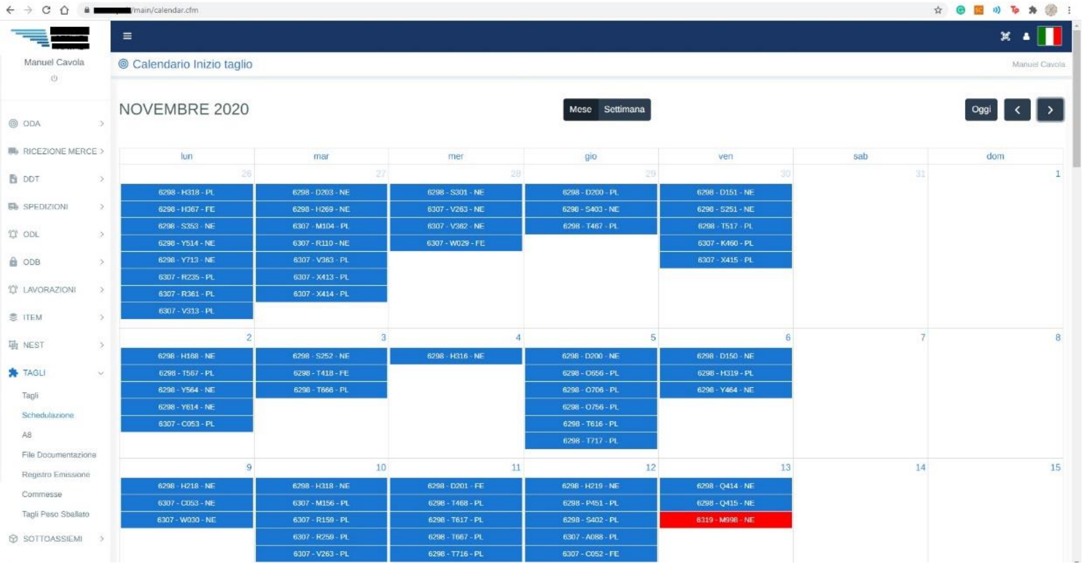
Suggerimento
Ricerca uno Schema Taglio digitando: Lotto e schema separati dallo spazio.
Esempio: J106 NE_10_75_04

Visualizza 10 elementi

Cerca:

ID	ODL	Costr	Lotti	Flussi	Materiale	Qta Fab	Qta da Gest	Tagliare	Data Emiss	Data Cons Min	Data Cons Max	Stab	Handled
3033	0011647876	0	6298XA17XPRS62981831XVIANELLO-0011647876	EXTRA	Profili	2	2	0	26/02/2021	11/02/2021	11/02/2021	[redacted]	[redacted]
3032	0011647789	0	6298XA17XPRS62981831XVIANELLO-0011647789	EXTRA	Profili	2	2	0	26/02/2021	11/02/2021	11/02/2021	[redacted]	[redacted]
3031	0011647736	0	A063X48496XVERSEGNASSI-0011647736	EXTRA	Profili	3	3	0	26/02/2021	25/02/2021	25/02/2021	[redacted]	[redacted]
3030	0011648026	0	6298XPRS62981837XFAVARETTO-0011648026	EXTRA	Profili	40	40	0	26/02/2021	25/02/2021	25/02/2021	[redacted]	[redacted]
3029	0011639446	6298	6298XD150XPRF862981802XSCARPAA-0011639446	EXTRA	Profili	1	1	0	26/02/2021	19/02/2021	19/02/2021	[redacted]	[redacted]
3028	0011647739	6307	C142	TM	Profili	1	1	0	26/02/2021	25/02/2021	25/02/2021	[redacted] STABILIMENTO DI [redacted]	[redacted]
3027	0011648084	0	48546XLARICCHIA-0011648084	EXTRA	Profili	8	0	0	26/02/2021	25/02/2021	25/02/2021	[redacted] STABILIMENTO DI [redacted]	[redacted]
3026	0011648081	0	48545XLARICCHIA-0011648081	EXTRA	Profili	3	0	0	26/02/2021	25/02/2021	25/02/2021	[redacted] STABILIMENTO DI [redacted]	[redacted]

Order management



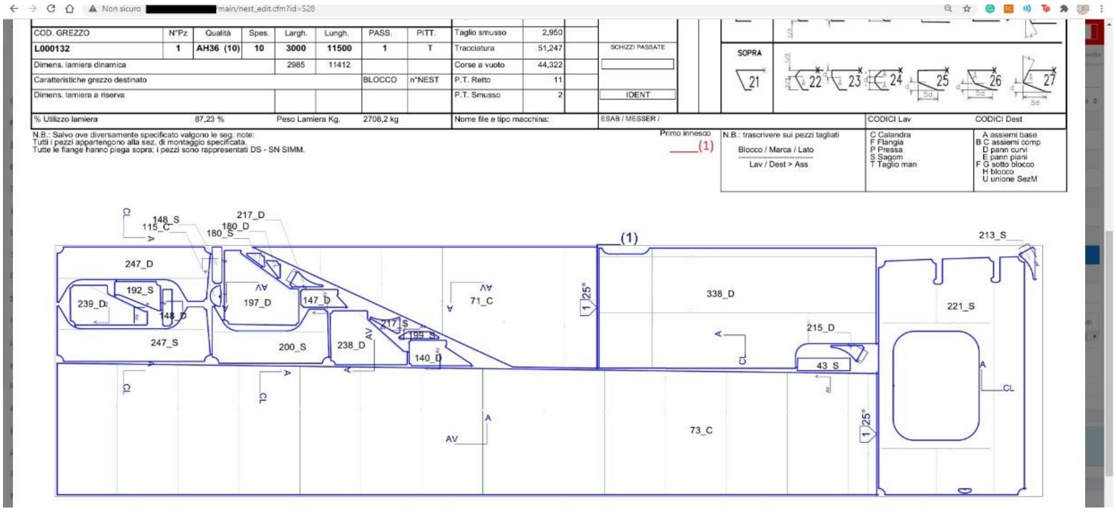
Calendario Inizio taglio

Manuel Cavola

NOVEMBRE 2020

lun	mar	mer	gio	ven	sab	dom
26 6298 - H318 - PL 6298 - H367 - FE 6298 - S353 - NE 6298 - Y514 - NE 6298 - Y713 - NE 6307 - R235 - PL 6307 - R361 - PL 6307 - V313 - PL	27 6298 - D203 - NE 6298 - H269 - NE 6307 - M104 - PL 6307 - R110 - NE 6307 - V363 - PL 6307 - X413 - PL 6307 - X414 - PL	28 6298 - S301 - NE 6307 - V263 - NE 6307 - V382 - NE 6307 - W029 - FE	29 6298 - D200 - PL 6298 - S403 - NE 6298 - T467 - PL	30 6298 - D151 - NE 6298 - S251 - NE 6298 - T517 - PL 6307 - K460 - PL 6307 - X415 - PL	31	1
2 6298 - H168 - NE 6298 - T567 - PL 6298 - Y564 - NE 6298 - Y614 - NE 6307 - C053 - PL	3 6298 - S252 - NE 6298 - T418 - FE 6298 - T666 - PL	4 6298 - H316 - NE	5 6298 - D200 - NE 6298 - O656 - PL 6298 - O706 - PL 6298 - O756 - PL 6298 - T616 - PL 6298 - T717 - PL	6 6298 - D150 - NE 6298 - H319 - PL 6298 - Y464 - NE	7	8
9 6298 - H218 - NE 6307 - C053 - NE 6307 - W030 - NE	10 6298 - H318 - NE 6307 - M156 - PL 6307 - R159 - PL 6307 - R259 - PL 6307 - V263 - PL	11 6298 - D201 - FE 6298 - T468 - PL 6298 - T617 - PL 6298 - T667 - PL 6298 - T716 - PL	12 6298 - H219 - NE 6298 - P451 - PL 6298 - S402 - PL 6307 - A088 - PL 6307 - C052 - FE	13 6298 - Q414 - NE 6298 - Q415 - NE 6319 - M998 - NE	14	15

Cutting scheduling



COD. GREZZO	N°Pz	Qualità	Spes.	Largh.	Lungh.	PASS.	PITT.	Taglio smusso	2,950	
L000132	1	AH36 (10)	10	3000	11500	1	T	Tracciatura	51,247	SCHIZZI PASSATE
Dimens. lamiera dinamica				2985	11412			Corse a vuoto	44,322	
Caratteristiche grezzo destinato						BLOCCO	n°NEST	P.T. Retto	11	
Dimens. lamiera a riserva								P.T. Smusso	2	IDENT
% Utilizzo lamiera	87,23 %			Peso Lamiera Kg.		2708,2 kg		Nome file e tipo macchina:		ESAB / MESSER /

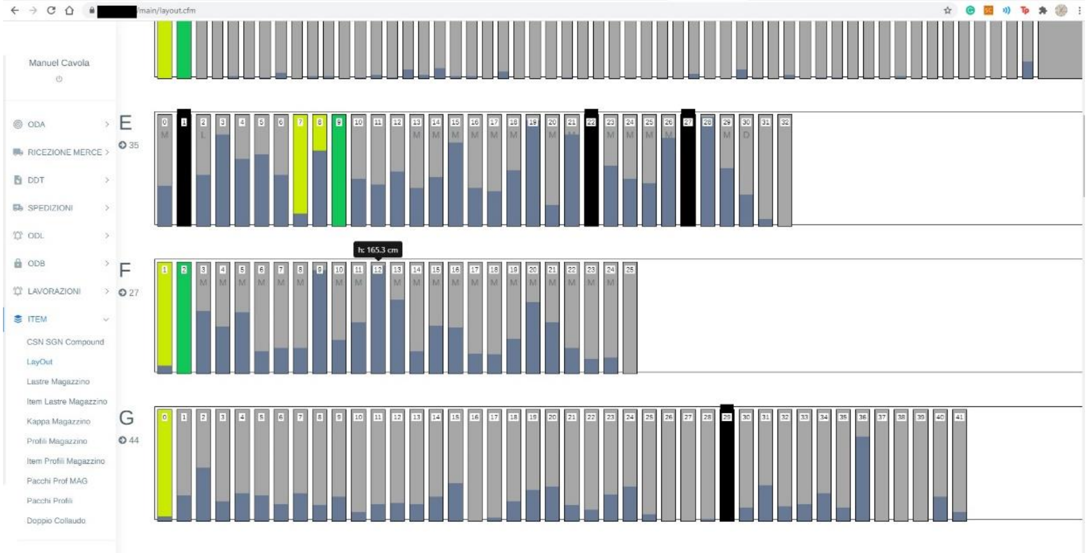
N.B.: Salvo ove diversamente specificato valgono le seg. note:
Tutti i pezzi appartengono alla sez. di montaggio specificata.
Tutte le flange hanno piega sopra; i pezzi sono rappresentati DS - SN SIMM.

Primo innesco
____(1)

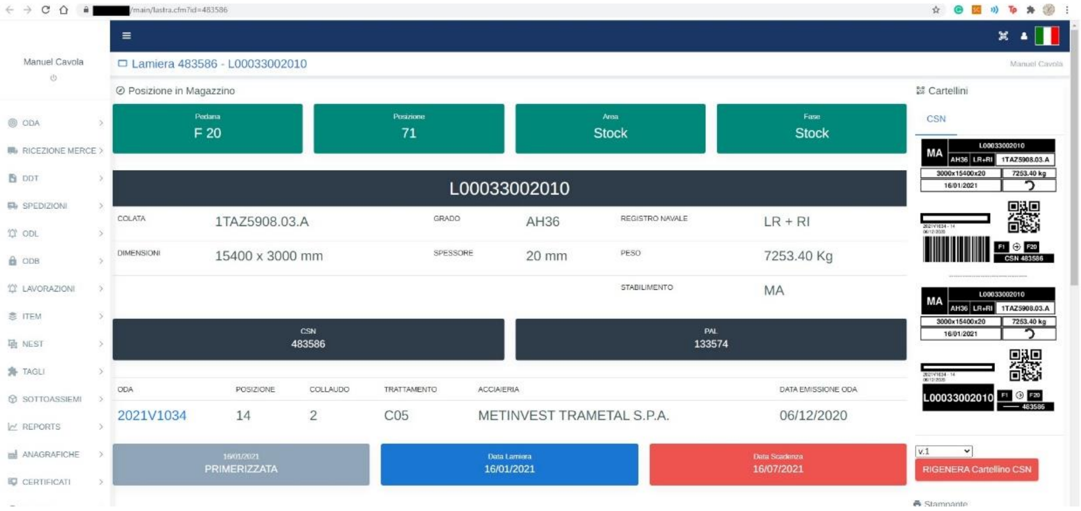
N.B.: trascrivere sui pezzi tagliati	CODICI Lav	CODICI Dest
Blocco / Marca / Lato Lav / Dest > Ass	C Calandra F Flangia P Pressa S Sagom T Taglio man	A assiemi base B C assiemi comp D pann curvi E pann piani F G sotto blocco H blocco U unione SezM

Digitalized cutting scheme

Figure 4.12 – Screenshots of the main implemented Production management procedures



Real-time stock level monitoring for each stack and stall



Lamiera 483586 - L00033002010

Posizione in Magazzino

Pedana	Posizione	Area	Fase
F 20	71	Stock	Stock

L00033002010

COLATA	1TAZ5908.03.A	GRADO	AH36	REGISTRO NAVALE	LR + RI
DIMENSIONI	15400 x 3000 mm	SPESSORE	20 mm	PESO	7253.40 Kg
				STABILIMENTO	MA

CSN 483586

PAL 133574

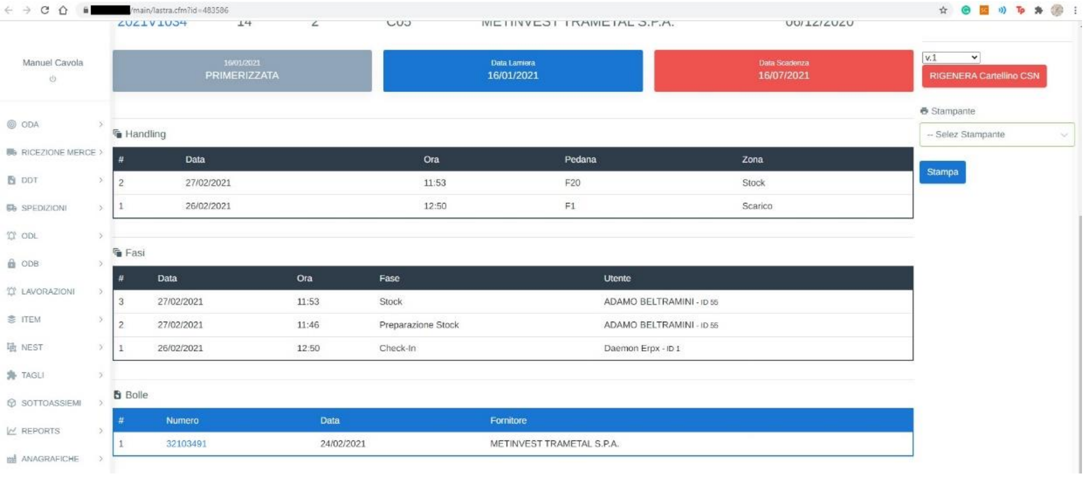
ODA	POSIZIONE	COLLAUDO	TRATTAMENTO	ACCIAIERIA	DATA EMISSIONE ODA
2021V1034	14	2	C05	METINVEST TRAMETAL S.P.A.	06/12/2020

16/01/2021 PRIMERIZZATA

Data Lamiera 16/01/2021

Data Scadenza 16/07/2021

Slab database management



16/01/2021 PRIMERIZZATA

Data Lamiera 16/01/2021

Data Scadenza 16/07/2021

Handling

#	Data	Ora	Pedana	Zona
2	27/02/2021	11:53	F20	Stock
1	26/02/2021	12:50	F1	Scarico

Fasi

#	Data	Ora	Fase	Utente
3	27/02/2021	11:53	Stock	ADAMO BELTRAMINI - ID 55
2	27/02/2021	11:46	Preparazione Stock	ADAMO BELTRAMINI - ID 55
1	26/02/2021	12:50	Check-In	Daemon Erpx - ID 1

Bolle

#	Numero	Data	Fornitore
1	32103491	24/02/2021	METINVEST TRAMETAL S.P.A.

Logistic flow management

Figure 4.13– Screenshots of the main implemented handling management procedures

4.7 Conclusions

In this chapter we have introduced some new mathematical formulations for the Slab Stack Shuffling able to describe some of the variants individuated for the problem. In particular we have focused on the version with the presence of deadline constraints in different cases of item-stack relation. All the models assume the presence of a fixed time horizon divided in periods at which a certain quantity of slabs belonging to specific items is requested. The problems has been formulated as a bi-objective optimization model including, in the objective function, two terms to be minimized: the number of shuffles and the number of expired slabs, i.e. slabs not retrieved before their own deadlines.

As in bi-objective problems, the tuning of the weight (penalty) method can be exploit to generate solutions of the Pareto frontier.

Then the model related to the case in which an item is hosted only in one stack and each stack hosts only one item has been tested by solving the model through CPLEX on instances randomly generated according to an appropriate instance generation procedure. As the solution of the model generally requires, in case of significant dimension of the instances, relevant computing times, we have proposed a heuristic based on a local search procedure.

The comparative performances of the heuristic are quite promising, as it generally obtains good solutions in quite short computing times.

However further efforts are needed to verify and to improve the performance of the proposed algorithm. A first aspect could require the development of more extensive computational experiences with the objective of exploring the possibility of reproducing the Pareto frontier. Then improvements of the heuristics can be performed for instance embedding the local search scheme within a meta-heuristic framework.

Nevertheless, as we have shown how the problem can be characterized by numerous variants in dependence on the combination of the ingredients of the problem, further analysis about the proposal of new formulations should be performed.

General conclusions

This work has been developed in the context of a collaboration with a firm of the shipbuilding sector (described in Chapter 1) interested to introduce innovative practises and procedures, in order to improve the production processes and the operations management. The firm is a first-tier actor in the sector supply chain as it provides subassemblies directly to the shipyard where the ships are assembled and launched.

After an analysis of all the production and logistic processes peculiar of the context (illustrated in Chapter 2) the focus has been devoted to one of the critical logistic main issue in the inbound operation, represented by the handling management of steel slabs, i.e., steel plates that can weigh up to more than 13 tons and whose extension can reach some tens of square meters. Due to their physical characteristics, steel slabs are stored in stacks. Then their storage and retrieval represent one of the main issues to overcome to avoid bottlenecks in the production process.

In the literature, the problem has been defined as the Slab Stack Shuffling (SSS) problem. However, due to the specificity of the problem and of the involved industrial sector, the state of the art is not particularly rich.

A first result of the work is represented by the formulation of a general framework able to describe the variety of the factors and of the conditions that can occur in the practical application. The framework suggests many opportunities to develop future research lines oriented to the proposals of models and methods to solve variants of the problem.

Among the problems individuated within the framework, we have focused on some basic versions of the problem including slabs deadline constraints. The problem has been formulated in terms of bi-objective mathematical programming model in which the minimization of the number of shuffles and the minimization of the number of expired slabs are combined through a penalty coefficient. As the solution of the model with the commercial solver CPLEX, performed on a set of randomly generated instances, has required significant computing times, especially in correspondence of some values of the penalty coefficient, a heuristic based on a local search procedure has been proposed and implemented. The algorithm has been tested on the same instances and a comparison of the performances with those provided by CPLEX has highlighted promising results.

As the objective of the collaboration was the re-engineering of the production and logistic processes through an integrated digitalization of the operations, finally, the general characteristics of a software system specifically designed and implemented has been described. The system, actually introduced and used in substitution of that previously adopted, has been designed in order to host optimization procedures able to solve logistic and operations management.

The obtained results of this work, then, should represent a first important step on the path leading to the definition and of the implementation of a software system able to assume the role of a Decision Support System that may support the production and logistics management.

As highlighted through the proposal of a general framework for the SSS problem, there is a vast field to develop models and methods to solve the various variants of the problem that, once tested and verified, should be embedded within the developed DSS.

Bibliography

Assaf, I., Chen, M., & Katzberg, J. (1997). Steel production schedule generation. *International Journal of Production Research*, *35*(2), 467-477.

Avriel, M., & Penn, M. (1993). Exact and approximate solutions of the container ship stowage problem. *Computers & industrial engineering*, *25*(1-4), 271-274.

Avriel, M., Penn, M., & Shpirer, N. (2000). Container ship stowage problem: complexity and connection to the coloring of circle graphs. *Discrete Applied Mathematics, 103*(1-3), 271-279.

BRS GROUP, (2019). *Shipping and Shipbuilding Markets*, Francia: s.n.

BRS GROUP, (2020). *Shipping and Shipbuilding Markets*, Annual Review

Brun, L., & Frederick, S. (2017). Korea and the Shipbuilding Global Value Chain. *Duke GVC Center*.

Celik, M., Kahraman, C., Cebi, S., & Er, I. D. (2009). Fuzzy axiomatic design-based performance evaluation model for docking facilities in shipbuilding industry: The case of Turkish shipyards. *Expert Systems with Applications*, 36(1), 599-615.

Cheng, X., & Tang, L. (2010, June). A scatter search algorithm for the slab stack shuffling problem. In *International Conference in Swarm Intelligence* (pp. 382-389). Springer, Berlin, Heidelberg.

Cho, K. K., Oh, J. S., Ryu, K. R., & Choi, H. R. (1998). An integrated process planning and scheduling system for block assembly in shipbuilding. *CIRP Annals, 47*(1), 419-422.

Chryssolouris, G., Makris, S., Xanthakis, V., & Mourtzis, D. (2004). Towards the Internet-based supply chain management for the ship repair industry. *International Journal of Computer Integrated Manufacturing, 17*(1), 45-57.

Dekker, R., Voogd, P., & Van Asperen, E. (2007). Advanced methods for container stacking. In *Container terminals and cargo systems* (pp. 131-154). Springer, Berlin, Heidelberg.

Delgado, A., Jensen, R. M., Janstrup, K., Rose, T. H., & Andersen, K. H. (2012). A constraint programming model for fast optimal stowage of container vessel bays. *European Journal of Operational Research, 220*(1), 251-261.

Derakhshan, E. D., Yazdian, N., Craft, B., Smith, S., & Kovacevic, R. (2018). Numerical simulation and experimental validation of residual stress and welding distortion induced by laser-based welding processes of thin structural steel plates in butt joint configuration. *Optics & Laser Technology*, *104*, 170-182.

Dyer, J. H. (1996). How Chrysler created an American keiretsu. *Harvard Business Review*, *74*(4), 42-52.

DSF, 2016. Shipping Market Review: *Danish Ship Finance (DSF).* pp. 106. Available at: https://www.shipfinance.dk/media/1610/shipping-market-review-may-2016.pdf

European Economic Community - *Council Regulation (EEC) No 4064/89* of 21 December 1989 on the control of concentrations between undertakings

Fechter, J., Beham, A., Wagner, S., & Affenzeller, M. (2015, February). Modeling a Lot-Aware Slab Stack Shuffling Problem. In *International Conference on Computer Aided Systems Theory* (pp. 334-341). Springer, Cham.

Fernandes, E. F. A., Freire, L., Passos, A. C., & Street, A. (2012). Solving the non-linear slab stack shuffling problem using linear binary integer programming. *EngOpt*.

Ferrari, C. (2012). "Cantieristica Navale : Caratteristiche e Tendenze Di Un Mercato Globale." Impresa progetto - *Electronic Journal of Management 3*: 1–20.

Fleischer, M., Kohler, R., & Bongiorni, H. B. (1999). *Marine supply chain management. Journal of ship production, 15*(04), 233-252

Fink, A. (2019). Conducting research literature reviews: From the internet to paper. *Sage publications*.

Forster, F., & Bortfeldt, A. (2012). A tree search heuristic for the container retrieval problem. In *Operations research proceedings 2011* (pp. 257-262). Springer, Berlin, Heidelberg.

Gourdon, K., & Steidl, C. (2019). "Global Value Chains and the Shipbuilding Industry." OECD Science, Technology and Industry Working Papers. https://dx.doi.org/10.1787/7e94709a-en.

Guneri, A. F., Cengiz, M., & Seker, S. (2009). A fuzzy ANP approach to shipyard location selection. *Expert systems with applications, 36*(4), 7992-7999.

Haessler, R. W., & Vonderembse, M. A. (1979). A procedure for solving the master slab cutting stock problem in the steel industry. *AIIE Transactions*, *11*(2), 160-165.

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, *130*(3), 449-467.

Hansen, P., & Mladenović, N. (2006). First vs. best improvement: An empirical study. *Discrete Applied Mathematics*, *154*(5), 802-817.

Held, T. (2010). Supplier integration as an improvement driver–an analysis of some recent approaches in the shipbuilding industry. In *Supply Chain Network Management* (pp. 369-384). Gabler.

Hines, P., Rich, N., & Esain, A. (1999). Value stream mapping. *Benchmarking: An International Journal*.

ICE - Agenzia per la promozione all'estero e l'internazionalizzazione delle imprese italiane, Ufficio di Berlino (Agosto 2018). *Nota di mercato*: "l'industria della cantieristica navale in Germania", pp. 3-19.

Imenda, S. (2014). Is there a conceptual difference between theoretical and conceptual frameworks?. *Journal of Social Sciences*, 38(2), 185-195.

Iwankowicz, R. R. (2016). An efficient evolutionary method of assembly sequence planning for shipbuilding industry. *Assembly Automation*.

Kang, J., Ryu, K. R., & Kim, K. H. (2006). Deriving stacking strategies for export containers with uncertain weight information. *Journal of Intelligent Manufacturing*, *17*(4), 399-410.

Kim, K. H., & Bae, J. W. (1998). Re-marshaling export containers in port container terminals. *Computers & Industrial Engineering*, *35*(3-4), 655-658.

Kim, K. H., Park, Y. M., & Ryu, K. R. (2000). Deriving decision rules to locate export containers in container yards. *European Journal of Operational Research*, *124*(1), 89-101.

Kim, B. I., Koo, J., & Sambhajirao, H. P. (2011). A simplified steel plate stacking problem. *International Journal of Production Research*, *49*(17), 5133-5151.

Ko, S., 2007. An efficient stacking policy for the steel plate, *Thesis (MS)*. Pohang University of Science and Technology, Pohang, Korea (in Korean).

Ko, S., et al., (2007). A study on the piling method for the improving efficiency of the steel plate warehouse. *Proceedings of the 2007 Joint Conference of Korean Institute of Industrial Engineers and Korean Operations Research and Management Science Society*, 25–26 May 2007, Jinju, Korea, 490–496 (in Korean).

Lamb, T. (1992). Organization theory and shipbuilding: a brief overview. *Marine Technology and SNAME News*, *29*(02), 71-83.

Lee, Y., & Lee, Y. J. (2010). A heuristic for retrieving containers from a yard. *Computers & Operations Research*, *37*(6), 1139-1147.

Lehnfeld, J., & Knust, S. (2014). Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. *European Journal of Operational Research*, *239*(2), 297-312.

Malucelli, F., Pallottino, S., & Pretolani, D. (2008). The stack loading and unloading problem. *Discrete applied mathematics*, *156*(17), 3248-3266.

Marksberry, P. (2012). Investigating "the way" for Toyota suppliers. *Benchmarking: An International Journal*.

Meisel, F., & Wichmann, M. (2010). Container sequencing for quay cranes with internal reshuffles. *OR spectrum*, *32*(3), 569-591.

Mello, M. H., & Strandhagen, J. O. (2010). Supply chain management in the shipbuilding industry: challenges and perspectives. Proceedings of the Institution of Mechanical Engineers, Part M: *Journal of Engineering for the Maritime Environment, 225*(3), 261-270.

OECD, 2017. Imbalances in the Shipbuilding Industry and Assessment of Policy Responses. Available at: https://www.oecd.org/industry/ind/Imbalances_Shipbuilding_Industry.pdf

Pareschi, A. (2007). Impianti Industriali. Criteri di scelta, progettazione e realizzazione. *Società Editrice Esculapio*.

Platts, K. W., Probert, D. R., & Canez, L. (2002). Make vs. buy decisions: A process incorporating multi-attribute decision-making. *International Journal of Production Economics*, *77*(3), 247-257.

Porter, M. E. (1979). *Harvard Business Review: Strategic Planning*, How Competitive Forces Shape Strategy. Retrieved July 7, 2016.

Ruuska, I., Ahola, T., Martinsuo, M., & Westerholm, T. (2013). Supplier capabilities in large shipbuilding projects. *International Journal of Project Management*, *31*(4), 542-553.

Singh, K. A., & Tiwari, M. K. (2004). Modelling the slab stack shuffling problem in developing steel rolling schedules and its solution using improved Parallel Genetic Algorithms. *International Journal of Production Economics*, 91(2), 135-147.

Stopford, M. (2003), *Maritime Economics*, Routledge, London and New York.

Tang, L., Liu, J., Rong, A., & Yang, Z. (2001). An effective heuristic algorithm to minimise stack shuffles in selecting steel slabs from the slab yard for heating and rolling. *Journal of the Operational Research Society*, 52(10), 1091-1097.

Tang, L., Liu, J., Rong, A., & Yang, Z. (2002). Modelling and a genetic algorithm solution for the slab stack shuffling problem when implementing steel rolling schedules. *International Journal of Production Research*, *40*(7), 1583-1595.

Tang, L., & Ren, H. (2010). Modelling and a segmented dynamic programming-based heuristic approach for the slab stack shuffling problem. *Computers & Operations Research*, *37*(2), 368-375.

Tang, L., Zhao, R., & Liu, J. (2012). Models and algorithms for shuffling problems in steel plants. *Naval Research Logistics (NRL)*, *59*(7), 502-524.

Tierney, K., Pacino, D., & Jensen, R. M. (2014). On the complexity of container stowage planning problems. *Discrete Applied Mathematics*, *169*, 225-230.

Tsirkas, S. A., Papanikos, P., & Kermanidis, T. (2003). Numerical simulation of the laser welding process in butt-joint specimens. *Journal of materials processing technology*, *134*(1), 59-69.

UNCTAD, (2019). *Review of Maritime Transport*, Svizzera: s.n. Available at: https://unctad.org/en/pages/PublicationWebflyer.aspx?publicationid=2563

Ünlüyurt, T., & Aydın, C. (2012). Improved rehandling strategies for the container retrieval process. *Journal of Advanced Transportation*, *46*(4), 378-393.

Van den Berg, J. P., & Zijm, W. H. M. (1999). Models for warehouse management: Classification and examples. International Journal of Production Economics, 59(1), 519–528.

Vlachakis, N., Mihiotis, A., Pappis, C. P., & Lagoudis, I. N. (2016). A methodology for analyzing shipyard supply chains and supplier selection. *Benchmarking: An International Journal*.

Wan, Y. W., Liu, J., & Tsai, P. C. (2009). The assignment of storage locations to containers for a container stack. *Naval Research Logistics (NRL)*, *56*(8), 699-713.

Willoughby, K. A. (2005). Process improvement in project expediting: there must be a better way. *International Journal of Project Management*, *23*(3), 231-236.

Wilson, I. D., & Roach, P. A. (2000). Container stowage planning: a methodology for generating computerised solutions. *Journal of the Operational Research Society*, *51*(11), 1248-1255.

Wortmann, J. C. (1983). A classification scheme for master production scheduling. In *Efficiency of manufacturing systems* (pp. 101-109). Springer, Boston, MA.

Zhang, C., Chen, W., Shi, L., & Zheng, L. (2010). A note on deriving decision rules to locate export containers in container yards. *European Journal of Operational Research*, *205*(2), 483-485.

Sitografia

LBJ (2020). L’impatto del Covid-19 sulla cantieristica. Available at: https://liguria.bizjournal.it/2020/06/limpatto-del-covid-sulla-cantieristica/ [Consultato il giorno 18.06.2020]

La Stampa (30 October 2020)) – Available at: https://finanza.lastampa.it/News/2020/10/30/fincantieri-argina-la-tempesta-covid-produzione-e-redditivita-attese-in-recupero/NjdfMjAyMC0xMC0zMF9UTEI Consulted: 13October2020

Nautilus (2020).Volano di sviluppo per l’uscita dall’emergenza Covid-19. Available at: http://www.ilnautilus.it/porti/2020-04-07/mit-adsp-volano-di-sviluppo-per-l%E2%80%99uscita-dall%E2%80%99emergenza-covid-19_74472/ [Consultato il giorno 18.06.2020]

Appendices

In this Appendix we show the scripts and pseudocodes developed to implement the proposed model and heuristic of the Slab Stack Shuffling problem in the case 1|1 and the detailed results for each instance in terms of objective function and computing time. In particular it is provided:

in Appendix A, the Model 1|1 write in OPL language used on the software "*CPLEX*".

In Appendix B, the "*Matlab"* code for the instance generation procedure.

In Appendix C, the tables of results obtained by the model tested on the generated instances.

In Appendix D, an example of the relation between slabs' *feasible periods* and periods' *feasible slabs*.

In Appendix E, the pseudocode for the *constructive heuristic to generate random feasible solutions*.

In Appendix F, the pseudocode for *constructive heuristic oriented to the minimization of the expired slabs*.

In Appendix G, the pseudocode for *the heuristic's local search phase*.

In Appendix H, the compared results in terms of objective function and computing time between the solution of the Model 1|1 and of the heuristic.

Appendix A

Here we report the whole script related to the model (1c÷7c) presented in Chapter 4, developed in OPL language according to the following notation:

$J = \{1, \dots, n\}$	set of slabs, indexed by j ;
$T = \{ 1, \dots, \bar{T}\}$	set of order periods, indexed by t ;
q^t	number of slabs to retrieve in period t ;
D_j^0	initial position of the slab *j* (positions are indicated from the top (position 1) to the bottom of the stack)
d_j	deadline of the slab j;

and the following decision variables

x_j^t	Binary variable equal to 1 if slab *j* is retrieved at period *t*, 0 otherwise;

S^t Positive integer variable, representing the number of shuffles associated to the stack *f* needed to satisfy requests at period *t;*

```
Input Files
string filename_input ="input_data.txt";
Inizialization
int m = 0;                                   Set  T
int n = 0;                                   Set  J
int P  = 0;                                  Parameter P
execute{
var f = new IloOplInputFile(filename_input);
f.readline();
m = f.readline();                             T size
f.readline();
n = f.readline();                             J size
f.readline();
P = f.readline();                            P value
f.readline();
          }

Sets Definition
{int} T = {};                                Set T
{int} J = {};                                Set J
Sets Composition
execute{
for(var i = 1; i <= m; i++){                 Set T
T.add(i);}
for(var j = 1; j <= n ; j++){
J.add(j);}                                   Set J
         }
Parameters Definition
int q[t in T];                               Requests in period t: q^t
int d[j in J] ;                              Slabs' deadline: d_j
int D0[J] ;                                  Slabs' initial position: D_j^0
Parameters Setting
execute{
var f = new IloOplInputFile(filename_input);
for(var i = 1; i <= 9 ; i++){
f.readline();}
var str = f.readline();
var ar = str.split("\t");
for(var t = 1; t <= M; t++){
q[t] = ar[t-1];}
```

Requests in each period t: $q^t \forall\ t \in T$

```
f.readline();
var str1 = f.readline();
var ar1 = str1.split("\t");
for(var j = 1; j <= N; j++){
d[j] = ar1[j-1];}
```

Slabs' deadline: d_j

```
f.readline();
var str1 = f.readline();
var ar1 = str1.split("\t");
for(var j = 1; j <= N; j++){
D0[j] = ar1[j-1] ;}
```

Slabs' initial position: D_j^0

```
f.close();
}
```

Variables Definition

```
dvar boolean x [j in J][t in T];
```

Slabs' assignment variable: x_j^t

```
dvar int+ S[t in T];
```

Number of shuffles in period t: S^t

Additional Definitions

```
dexpr int D[j in J][t in T] = D0[j] - sum(s in T: s<=
t-1 && t>1, k in J: D0[j]>D0[k])x[k][s] ;
```

Slabs above j in period t: D_j

```
dexpr int Shuf[t in T]= S[t];
```

Support element related to S^t

Objective function

```
dexpr float z1 = sum(t in T) Shuf[t] ;
```

Support element related to the shuffles

```
dexpr float z2 = P*sum( j in J: d[j] <= M)(1-sum(t
in T: t <= d[j])x[j][t]) ;
```

Support element related to the expired slabs

```
minimize z1 + z2;
```

(1c)

Constraints

```
subject to {
forall (t in T: t>= 1, j in J)
```

(2c)

```
   S[t] >= D0[j]*x[j][t] - sum(s in T: s <= t , k in
J:
  D0[j] >= D0[k]) x[k][s];
```

(3c)

```
forall (t in T)
   sum(j in J)x[j][t] == q[t];
```

(4c)

```
forall (j in J)
```

```
    sum(t in T)x[j][t] <= 1 ;
```

(5c)

```
  forall(j in J, t in T: t> d[j])
    x[j][t] == 0;
          }
```

Appendix B

Here we report the whole script related to the *instance generation procedure* reported in Chapter 4 and developed in Matlab language according to the notation reported above.

Input Files

```
function [q,d,D0,cartella,namefile,Nin] = dataINPUT(T,J)
```

Request Setting

```
q = randi([0 4], 1,T) ;
```

Requests in each period t: $q^t \forall\, t \in T$

Deadlines Setting

```
d_D = zeros(J,2);
iS = 1;
appoggio = zeros(1,T+1); variable
```

Inizialization of matrix dD = Jx2 (first colum d_j ; second columns D_j^0)

Support variable

```
for iT=1:T
    while iS<=q(iT)
     d_D(iS+sum(appoggio),1)   = randi([iT, iT+2])    ;
     iS  = iS+1  ;
     end
     iS=1;
     appoggio(iT+1)=q(iT);
end
```

Deadlines setting for the q^t slabs

```
for iii= sum(q)+1:J
 p=rand();
    if 0.7<=p
    d_D(iii,1) = randi([1,T])
    else
    d_D(iii,1) = T+1;
    end
end
```

Deadline setting for the last $|J| - q^t$ slabs

Position Setting

```
d_D(:,2)=randperm(J)'; %
[~,ind]=sort(d_D(:,2),'descend');
d_D=d_D(ind,:);

soglia= round(T/2); %deadline limit
```

Random sorting

Evaluation of the number of slabs with lower deadline to assign in the lower positions

```
N=sum(d_D(:,1)<=soglia);
[irup,~]=find(d_D(1:round(size(d_D,1)/2),1)<=soglia);
[irdow,~]=find(d_D((round(size(d_D,1)/2)+1):end,1)>soglia);
irdow=irdow+round(size(d_D,1)/2);
cond = sum(d_D(end:-
1:(round(size(d_D,1)/2)+1),1)<=soglia);
ii   = 1;

while(cond<=round(0.6*N)&&(isempty(irup)==0)
&&(cond<=round(0.6*N)&&(isempty(irdow)==0))
```

Position swapping according to the conditions set above

```
appo2= d_D(irdow(ii),1);
d_D(irdow(ii),1)=d_D(irup(ii),1) ;
d_D(irup(ii),1) = appo2 ;
[irup,~]=find(d_D(1:round(size(d_D,1)/2),1)<=soglia);
[irdow,~]=find(d_D((round(size(d_D,1)/2)+1):end,1)>soglia);
irdow=irdow+round(size(d_D,1)/2);
cond=sum(d_D(end:-1:(round(size(d_D,1)/2)+1),1)<=soglia);
end
```

Appendix C

Here we report the results of the computational experiences of the model that have been carried out using IBM ILOG CPLEX Optimization Studio 12.9 software on an Intel(R) Core(TM) i7-8550U with 1.80 GHz and 16 GB of RAM.

These regards the 50 random instances generated for each (|*T*|, |*J*|) = {(5, 20); (10, 40); (15, 60); (20, 80); (25,100)}. In particular, in Table .1 and Table .2 are reported the results in terms of objective function and computing time respectively, for each combination of (|*T*|, |*J*|) = {(5, 20); (10, 40); (15, 60) } and P = $\{|T|x|J|, P_m, P_m/2\,, P_m/4\,, 0\}$. While, in Table .3 and Table .4 are reported the results in terms of objective function and computing time respectively, for each combination of (|*T*|, |*J*|) = {(20, 80); (25, 100)}.

Table .1 Computational results in terms of objective function for (|*T*|, |*J*|) = {(5, 20); (10, 40); (15, 60) } and $P = \{\lvert T\rvert\text{x}\lvert J\rvert, P_{\text{m}}, P_{\text{m}}/2\ , P_{\text{m}}/4\ , 0\}$

Instance	*T*	*J*	*P*				
			$\lvert T\rvert$x$\lvert J\rvert$	P_m	$P_m/2$	$P_m/4$	$\lvert T\rvert$x$\lvert J\rvert$
			Computing time (*s*)				
1	5	20	31	27	18	13	3
2	5	20	132	32	23	17	5
3	5	20	411	31	19	13	1
4	5	20	127	27	21	16	4
5	5	20	33	31	18	13	3
6	5	20	26	22	17	14	6
7	5	20	37	33	26	21	9
8	5	20	121	18	13	10	5
9	5	20	32	24	20	15	4
10	5	20	38	30	21	15	8
11	5	20	226	32	25	19	5
12	5	20	227	31	21	15	8
13	5	20	316	27	15	8	1
14	5	20	134	34	26	19	5
15	5	20	47	37	27	17	4
16	5	20	224	28	20	15	1
17	5	20	310	22	12	6	0
18	5	20	322	34	22	16	10
19	5	20	33	29	17	11	3
20	5	20	30	20	16	14	6
21	5	20	710	38	20	10	0
22	5	20	219	31	19	13	2
23	5	20	39	33	27	23	14
24	5	20	317	32	21	14	0
25	5	20	24	16	13	10	7
26	5	20	131	30	21	11	0
27	5	20	413	32	20	14	0
28	5	20	426	46	25	18	2
29	5	20	55	48	30	18	0
30	5	20	135	38	23	17	4
31	5	20	121	21	17	12	2
32	5	20	136	36	24	18	4

Continued on next page

Table .1 – Continued from previous page

Instance	T	J	P				
			$\lvert T\rvert$x$\lvert J\rvert$	P_m	$P_m/2$	$P_m/4$	$\lvert T\rvert$x$\lvert J\rvert$
			Computing time (s)				
33	5	20	136	29	21	17	2
34	5	20	225	31	23	16	0
35	5	20	226	38	25	13	1
36	5	20	131	38	24	12	0
37	5	20	133	32	22	17	6
38	5	20	614	44	27	18	0
39	5	20	22	22	12	8	0
40	5	20	127	31	20	15	5
41	5	20	224	34	21	14	0
42	5	20	128	34	22	12	2
43	5	20	227	33	21	15	1
44	5	20	115	19	12	8	0
45	5	20	19	19	14	11	5
46	5	20	513	28	22	14	6
47	5	20	40	32	25	13	1
48	5	20	26	26	16	10	4
49	5	20	126	26	18	13	1
50	5	20	39	32	22	18	10
1	10	40	2.42	1675	100	63	40
2	10	40	1.44	918	118	76	51
3	10	40	3.09	1367	184	112	56
4	10	40	1.44	2084	109	64	40
5	10	40	3.16	1685	94	67	48
6	10	40	9.95	2887	134	87	61
7	10	40	1.04	2891	139	88	59
8	10	40	6.86	1308	113	77	57
9	10	40	5.10	2119	144	103	77
10	10	40	0.37	2073	101	61	35
11	10	40	1.11	2108	140	90	60
12	10	40	0.60	2862	104	63	35
13	10	40	1.50	2105	118	79	52
14	10	40	0.53	1674	87	52	39
15	10	40	0.58	903	103	68	45
16	10	40	0.67	1684	116	71	41

Continued on next page

Table .1 – Continued from previous page

Instance	T	J	P				
			$\vert T\vert$x$\vert J\vert$	P_m	$P_m/2$	$P_m/4$	$\vert T\vert$x$\vert J\vert$
			Computing time (s)				
17	10	40	1707	140	84	56	14
18	10	40	2084	106	70	45	15
19	10	40	533	118	83	53	20
20	10	40	2099	115	81	49	15
21	10	40	911	120	82	58	33
22	10	40	2102	133	80	53	11
23	10	40	2495	144	87	45	2
24	10	40	508	110	71	38	5
25	10	40	3662	104	62	46	11
26	10	40	3279	142	93	64	19
27	10	40	529	130	90	69	36
28	10	40	1703	120	83	56	14
29	10	40	506	95	66	46	12
30	10	40	1699	122	83	54	9
31	10	40	2490	120	72	42	10
32	10	40	2474	129	82	61	28
33	10	40	3279	135	79	47	15
34	10	40	924	117	78	61	28
35	10	40	2506	153	96	68	26
36	10	40	2110	142	83	57	17
37	10	40	1315	125	86	64	30
38	10	40	2079	108	67	41	15
39	10	40	1703	114	75	51	26
40	10	40	1331	147	101	64	25
41	10	40	2493	125	76	48	18
42	10	40	915	113	76	43	10
43	10	40	128	100	67	39	7
44	10	40	1399	166	118	71	13
45	10	40	2870	106	70	41	9
46	10	40	2051	84	49	29	7
47	10	40	141	117	85	61	34
48	10	40	1273	81	54	33	11
49	10	40	1678	100	69	46	20
50	10	40	1677	108	65	43	10

Continued on next page

Table .1 – Continued from previous page

Instance	T	J	P				
			$\|T\|$x$\|J\|$	P_m	$P_m/2$	$P_m/4$	$\|T\|$x$\|J\|$
			Computing time (s)				
1	15	60	5618	267	161	99	15
2	15	60	9186	275	164	102	31
3	15	60	5601	231	141	80	14
4	15	60	6468	228	156	104	47
5	15	60	9169	243	158	101	38
6	15	60	9149	217	139	95	24
7	15	60	9144	225	135	95	31
8	15	60	13623	268	160	90	12
9	15	60	10015	208	121	75	28
10	15	60	5614	260	159	98	31
11	15	60	4725	251	173	124	69
12	15	60	10959	274	174	100	25
13	15	60	10052	255	162	93	24
14	15	60	3866	276	185	61	49
15	15	60	3849	244	148	91	12
16	15	60	1077	191	122	87	32
17	15	60	9164	252	159	87	12
18	15	60	3808	232	145	95	40
19	15	60	10065	260	176	94	19
20	15	60	5621	254	144	78	10
21	15	60	7342	215	131	72	12
22	15	60	13634	287	171	92	11
23	15	60	6502	264	167	109	25
24	15	60	12726	285	169	106	14
25	15	60	8292	256	153	90	19
26	15	60	9211	364	179	100	19
27	15	60	7420	257	173	110	45
28	15	60	3011	279	191	119	18
29	15	60	10050	251	162	90	18
30	15	60	6461	201	131	100	46
31	15	60	3824	217	156	105	49
32	15	60	10103	356	209	125	37
33	15	60	12709	228	127	82	8
34	15	60	7389	266	170	108	24
35	15	60	9205	280	171	106	13

Continued on next page

Table .1 – Continued from previous page

Instance	T	J	P				
			$\|T\|$x$\|J\|$	P_m	$P_m/2$	$P_m/4$	$\|T\|$x$\|J\|$
			Computing time (s)				
36	15	60	8313	278	177	110	13
37	15	60	9166	255	161	98	32
38	15	60	11847	253	146	98	23
39	15	60	3840	245	155	101	39
40	15	60	9181	274	175	103	31
41	15	60	7407	269	180	119	53
42	15	60	5572	209	136	80	17
43	15	60	8189	153	94	61	7
44	15	60	5582	222	148	94	39
45	15	60	6521	266	175	115	48
46	15	60	10072	262	163	96	24
47	15	60	10070	269	166	94	16
48	15	60	5626	260	156	94	25
49	15	60	9159	224	150	106	38
50	15	60	8281	268	166	105	42

Table .2 Computational results in terms of computing time (s) for (|T|, |J|) = {(5, 20); (10, 40); (15, 60)} and P = $\{|T|\text{x}|J|, P_\text{m}, P_\text{m}/2\,, P_\text{m}/4\,, 0\}$

Instance	T	J	P				
			|T|x|J|	P_m	$P_m/2$	$P_m/4$	|T|x|J|
			Computing time (s)				
1	5	20	0.18	0.20	0.25	0.2 7	0.47
2	5	20	0.19	0.61	0.19	0.21	0.20
3	5	20	0.16	0.19	0.18	0.17	0.17
4	5	20	0.18	0.19	0.2 1	0.24	0.16
5	5	20	0.20	0.24	0.2 1	0.23	0.18
6	5	20	0.17	0.30	0.23	0.18	0.31
7	5	20	0.22	0.25	0.60	0.45	0.36
8	5	20	0.49	0.21	0.23	0.23	0.26
9	5	20	0.86	0.30	0.26	0.31	0.25
10	5	20	0.20	0.22	0.25	0.23	0.41
11	5	20	0.62	0.60	0.36	0.24	0.26
12	5	20	0.84	0.25	0.27	0.62	0.33
13	5	20	0.21	0.18	0.19	0.17	0.20
14	5	20	0.77	0.40	0.37	34.00	0.25
15	5	20	0.61	0.27	0.24	0.23	0.25
16	5	20	0.63	0.25	0.19	0.18	31.00
17	5	20	0.18	0.17	0.22	0.17	0.17
18	5	20	0.27	0.23	0.29	0.25	37.00
19	5	20	0.20	0.20	0.19	0.19	0.24
20	5	20	0.33	0.22	0.19	0.21	0.27
21	5	20	0.24	0.19	0.26	0.19	0.19
22	5	20	0.31	0.20	0.18	0.17	0.23
23	5	20	0.27	0.31	0.20	0.19	0.20
24	5	20	0.23	0.27	0.23	0.27	0.32
25	5	20	0.24	0.30	0.26	0.28	0.21
26	5	20	0.32	0.53	0.19	0.33	0.29
27	5	20	0.28	0.64	0.64	0.33	0.28
28	5	20	0.25	0.66	0.35	0.29	0.31
29	5	20	0.27	0.31	0.20	0.19	0.20
30	5	20	0.28	1.07	0.92	0.44	0.34
31	5	20	0.32	0.46	0.43	0.38	0.35
32	5	20	0.29	0.41	0.44	0.33	0.29

Continued on next page

Table .2 – Continued from previous page

Instance	T	J	P				
			$\lvert T\rvert$x$\lvert J\rvert$	P_m	$P_m/2$	$P_m/4$	$\lvert T\rvert$x$\lvert J\rvert$
			Computing time (s)				
33	5	20	0.62	0.52	0.48	0.44	0.40
34	5	20	0.35	0.31	0.51	0.37	0.27
35	5	20	0.35	0.37	0.29	0.43	0.30
36	5	20	0.34	0.43	0.33	0.36	0.25
37	5	20	0.32	0.48	0.40	0.47	0.41
38	5	20	0.29	0.43	0.22	0.37	0.27
39	5	20	0.26	0.35	0.27	0.33	0.27
40	5	20	0.64	1.25	1.11	0.40	30.00
41	5	20	0.34	0.50	0.47	0.36	0.30
42	5	20	0.35	0.63	0.38	0.32	0.31
43	5	20	0.70	0.40	0.42	0.30	0.30
44	5	20	0.90	0.40	0.45	0.29	0.26
45	5	20	0.80	0.63	0.5 1	0.41	0.46
46	5	20	0.63	0.43	0.24	0.29	0.31
47	5	20	0.50	0.41	0.40	0.36	0.28
48	5	20	0.50	0.74	0.36	0.38	0.30
49	5	20	0.80	0.44	0.31	0.35	0.30
50	5	20	0.47	1.46	0.99	0.63	0.53
1	10	40	2.42	208.32	8.93	6.85	2.03
2	10	40	1.44	7.94	5.30	4.10	1.70
3	10	40	3.09	3.83	2.74	2.99	0.62
4	10	40	1.44	7.66	5.97	5.13	3.20
5	10	40	3.16	7.25	8.22	7.22	6.90
6	10	40	9.95	25.97	24.75	31.64	11.76
7	10	40	1.04	31.47	9.95	7.02	2.49
8	10	40	6.86	12.01	16.15	13.72	9.60
9	10	40	5.10	11.92	14.85	10.76	8.83
10	10	40	0.37	8.11	6.46	4.20	1.93
11	10	40	1.11	18.13	9.93	8.97	2.15
12	10	40	0.60	4.48	4.54	3.67	2.39
13	10	40	1.50	17.31	11.81	9.66	4.48
14	10	40	0.53	4.19	2.93	2.42	2.05
15	10	40	0.58	7.20	6.28	4.95	2.42
16	10	40	0.67	24.31	9.11	5.80	2.38

Continued on next page

Table .2 – Continued from previous page

Instance	T	J	P				
			$\lvert T\rvert$x$\lvert J\rvert$	P_m	$P_m/2$	$P_m/4$	$\lvert T\rvert$x$\lvert J\rvert$
			Computing time (s)				
17	10	40	0.64	18.45	5.11	2.66	2.19
18	10	40	0.95	11.61	7.91	5.11	2.19
19	10	40	5.16	11.87	11.17	6.63	4.97
20	10	40	2.43	24.10	10.48	5.62	1.74
21	10	40	7.69	678.19	55.88	46.02	22.79
22	10	40	1.40	4.34	4.05	4.11	2.17
23	10	40	0.22	7.38	6.16	4.42	1.72
24	10	40	0.72	4.54	3.74	3.91	2.36
25	10	40	0.91	6.25	4.68	3.54	2.48
26	10	40	2.82	15.12	12.61	10.11	2.78
27	10	40	7.92	96.19	28.45	15.55	10.84
28	10	40	1.03	11.98	11.51	6.94	2.16
29	10	40	1.95	12.05	12.57	7.63	1.99
30	10	40	0.49	18.53	7.39	6.19	1.55
31	10	40	1.13	7.47	7.26	6.76	2.37
32	10	40	6.27	361.61	37.04	14.59	8.18
33	10	40	0.98	36.46	8.05	7.16	2.30
34	10	40	4.35	11.25	15.47	8.98	9.13
35	10	40	5.22	45.46	15.27	11.41	8.38
36	10	40	3.01	7.58	9.59	5.39	3.60
37	10	40	6.23	91.17	43.75	16.87	9.51
38	10	40	0.90	12.74	10.99	5.43	1.95
39	10	40	6.70	24.80	37.66	17.25	9.62
40	10	40	8.28	20.52	18.51	10.39	11.41
41	10	40	1.71	21.64	12.36	8.31	5.24
42	10	40	1.17	6.06	*6) 5*	4.22	1.88
43	10	40	0.68	9.95	7.99	2.65	1.75
44	10	40	1.35	27.05	19.23	10.49	2.25
45	10	40	0.58	3.55	4.58	1.64	1.66
46	10	40	1.00	4.38	5.56	4.31	1.88
47	10	40	4.79	40.25	15.12	9.95	9.64
48	10	40	0.77	9.46	5.49	5.05	1.89
49	10	40	2.51	12.56	9.91	7.57	2.37
50	10	40	1.25	8.40	8.40	5.34	1.98

Continued on next page

Table .2 – Continued from previous page

Instance	T	J	P				
			$\lvert T\rvert$x$\lvert J\rvert$	P_m	$P_m/2$	$P_m/4$	$\lvert T\rvert$x$\lvert J\rvert$
			Computing time (s)				
1	15	60	112.31	1077.07	65.95	36.45	6.45
2	15	60	101.56	372.01	32.75	34.55	11.54
3	15	60	3301.68	2258.94	100.00	54.19	5.77
4	15	60	1071.62	3929.76	311.08	1077.70	141.52
5	15	60	360.89	992.07	163.98	121.66	29.87
6	15	60	17.50	135.19	30.96	27.52	8.72
7	15	60	1267.97	1174.30	109.49	43.85	21.76
8	15	60	329.89	51.67	13.21	21.04	6.50
9	15	60	5220.24	1176.90	117.20	128.97	10.38
10	15	60	19.44	1689.73	177.16	182.31	18.20
11	15	60	6.76	901.18	346.10	221.36	362.09
12	15	60	39.31	467.62	47.66	73.35	6.30
13	15	60	505.97	3343.50	58.42	635.07	12.94
14	15	60	10.36	1634.93	1022.25	76.41	94.65
15	15	60	8.59	527.50	66.27	405.26	4.96
16	15	60	1517.45	7877.27	232.35	137.17	35.80
17	15	60	32.81	59.49	12.00	142.46	4.52
18	15	60	794.62	20527.34	296.71	11.86	35.07
19	15	60	451.05	3715.68	229.31	89.95	6.73
20	15	60	102.82	1777.29	24.38	148.36	4.82
21	15	60	25.82	40.37	31.68	25.60	7.46
22	15	60	2 16.92	203.32	27.58	23.59	4.08
23	15	60	18.91	842.34	122.50	45.77	11.81
24	15	60	15.58	53.25	20.56	80.01	4.31
25	15	60	57.83	538.29	140.36	15.46	12.80
26	15	60	7636.44	13974.52	302.63	150.85	8.87
27	15	60	517.92	5261.45	450.11	218.43	158.57
28	15	60	7.96	1446.24	478.18	194.65	11.77
29	15	60	87.42	350.16	33.50	304.40	7.50
30	15	60	120.80	1291.17	811.34	19.90	118.05
31	15	60	120.41	36130.07	1475.22	228.83	98.01
32	15	60	58.18	6916.84	431.54	646.91	77.74
33	15	60	59.78	35.35	17.78	490.09	4.59
34	15	60	58.34	480.78	63.81	18.00	6.17
35	15	60	5.97	117.03	44.33	66.29	5.59

Continued on next page

Table .2 – Continued from previous page

Instance	T	J	P				
			$\vert T\vert$x$\vert J\vert$	P_m	$P_m/2$	$P_m/4$	$\vert T\vert$x$\vert J\vert$
			Computing time (s)				
36	15	60	36.32	1730.69	412.98	25.15	7.88
37	15	60	10.84	94.98	33.22	168.99	30.80
38	15	60	40.80	104.14	43.30	18.54	12.07
39	15	60	9.29	5533.96	417.03	31.48	82.21
40	15	60	347.60	10467.70	163.8 1	240.44	18.15
41	15	60	232.23	19718.64	1055.92	99.70	114.19
42	15	60	890.91	1939.93	136.17	434.72	8.25
43	15	60	19.07	27.98	34.19	53.33	4.70
44	15	60	638.44	3766.62	469.50	11.87	110.48
45	15	60	7712.66	17855.53	6448.24	138.86	111.12
46	15	60	573.10	2936.57	103.87	572.69	19.52
47	15	60	17128.73	21717.63	180.54	55.27	8.78
48	15	60	4441.47	3042. 15	220.67	197.35	18.04
49	15	60	637.25	808.88	169.44	90.59	29.21
50	15	60	1377.58	41250.87	596.92	519.68	44.71

Table .3 Computational results in terms of objective function for ($|T|$, $|J|$) = {(20,80); (25, 100)} and $P = \{|T|\mathrm{x}|J|, 0\}$(in red the instances with a not optimal solution).

Instance	*T*	*J*	*P*	
			$\lvert T\rvert$x$\lvert J\rvert$	0
			Objective function	
1	20	80	19536	65
2	20	80	6735	37
3	20	80	16341	45
4	20	80	17889	41
5	20	80	10069	104
6	20	80	14661	27
7	20	80	13148	13
8	20	80	27475	25
9	20	80	13161	119
10	20	80	14762	197
11	20	80	16262	3
12	20	80	22687	40
13	20	80	13169	90
14	20	80	16327	18
15	20	80	17921	27
16	20	80	25885	31
17	20	80	25882	40
18	20	80	21084	38
19	20	80	27411	54
20	20	80	17850	24
21	20	80	19440	32
22	20	80	21096	45
23	20	80	27462	37
24	20	80	16278	22
25	20	80	25889	21
26	20	80	13170	69
27	20	80	11641	56
28	20	80	14750	52
29	20	80	25862	35
30	20	80	13141	48
31	20	80	16385	101
32	20	80	24266	25
33	20	80	24264	18

Continued on next page

Table .3 – Continued from previous page

Instance	*T*	*J*	*P*	
			|*T*|x|*J*|	0
			Objective function	
34	20	80	13075	29
35	20	80	19470	44
36	20	80	14721	39
37	20	80	21062	27
38	20	80	27470	69
39	20	80	13189	21
40	20	80	13167	22
41	20	80	19555	80
42	20	80	19528	35
43	20	80	16359	63
44	20	80	19565	47
45	20	80	25857	27
46	20	80	13187	82
47	20	80	14716	22
48	20	80	17979	28
49	20	80	18016	125
50	20	80	22673	51
1	25	100	47905	80
2	25	100	33099	65
3	25	100	40420	70
4	25	100	28045	71
5	25	100	33011	86
6	25	100	33080	119
7	25	100	23051	109
8	25	100	30491	41
9	25	100	35500	64
10	25	100	25574	101
11	25	100	47894	46
12	25	100	45547	172
13	25	100	23099	103
14	25	100	37987	26
15	25	100	47965	82
16	25	100	18102	111

Continued on next page

Table .3 – Continued from previous page

Instance	*T*	*J*	*P*	
			|*T*|x|*J*|	0
			Objective function	
17	25	100	35479	98
18	25	100	57915	96
19	25	100	47894	35
20	25	100	33161	60
21	25	100	33103	80
22	25	100	35533	116
23	25	100	18113	139
24	25	100	37986	83
25	25	100	28193	169
26	25	100	35468	80
27	25	100	32921	97
28	25	100	30634	167
29	25	100	32946	93
30	25	100	38057	26
31	25	100	43021	145
32	25	100	47890	36
33	25	100	45486	123
34	25	100	50440	69
35	25	100	23066	51
36	25	100	50506	79
37	25	100	28031	111
38	25	100	28043	73
39	25	100	30439	97
40	25	100	38050	97
41	25	100	52950	71
42	25	100	33017	121
43	25	100	28086	122
44	25	100	45409	47
45	25	100	30546	71
46	25	100	37929	91
47	25	100	40377	40
48	25	100	37939	65
49	25	100	38028	195
50	25	100	52950	61

Table .4 Computational results in terms of computing time(s) for ($\lvert T\rvert$, $\lvert J\rvert$) = {(20,80); (25, 100)} and $P = \{\lvert T\rvert \mathrm{x} \lvert J\rvert, 0\}$

Instance	*T*	*J*	*P*	
			$\lvert T\rvert$x$\lvert J\rvert$	0
			Computing time (*s*)	
1	20	80	64.56	1749.74
2	20	80	3605.19	48.93
3	20	80	1000.21	272.85
4	20	80	3601.41	545.13
5	20	80	3607.14	3599.27
6	20	80	673.71	28.18
7	20	80	483.42	6.31
8	20	80	92.36	12.29
9	20	80	3607.24	3600.21
10	20	80	3607.61	3600.15
11	20	80	17.59	0.76
12	20	80	1001.73	78.89
13	20	80	3600.20	3603.88
14	20	80	1490.72	8.46
15	20	80	3607.42	360.42
16	20	80	1000.19	30.13
17	20	80	202.10	102.13
18	20	80	680.82	43.23
19	20	80	1002.36	587.74
20	20	80	30.53	10.62
21	20	80	61.56	38.58
22	20	80	37.16	3600.2
23	20	80	43.25	70.69
24	20	80	3300.42	12.31
25	20	80	1388.00	27.67
26	20	80	3603.82	3352.10
27	20	80	1783.52	486.42
28	20	80	10.38	488.54
29	20	80	1000.23	51.97
30	20	80	28.87	77.20
31	20	80	3600.18	3600.44
32	20	80	64.56	14.57
33	20	80	3605.19	0.52

Continued on next page

Table .4 – Continued from previous page

Instance	*T*	*J*	*P*	
			|*T*|x|*J*|	0
			Objective function	
34	20	80	1545.93	19.69
35	20	80	299.88	116.49
36	20	80	175.42	106.33
37	20	80	1000.65	7.82
38	20	80	381.66	2927.66
39	20	80	17.41	12.50
40	20	80	12.22	22.95
41	20	80	3602.58	3600.19
42	20	80	1000.64	347.45
43	20	80	1418.20	1562.61
44	20	80	3605.36	81.92
45	20	80	115.75	22.79
46	20	80	3602.23	3605.16
47	20	80	19.20	18.03
48	20	80	12.90	35.02
49	20	80	53.19	3604.23
50	20	80	363.73	410.33
1	25	100	3601.92	3601.62
2	25	100	465.77	1704.87
3	25	100	3601.78	3601.96
4	25	100	3600.38	3602.07
5	25	100	3607.97	3605.40
6	25	100	3603.52	3602.05
7	25	100	3601.28	3601.62
8	25	100	3602.67	3600.54
9	25	100	3600.51	3607.63
10	25	100	3607.42	3602.61
11	25	100	3600.51	3224.03
12	25	100	3600.47	3603.84
13	25	100	1170.85	3603.61
14	25	100	3600.65	25.98
15	25	100	3601.63	3618.01
16	25	100	3604.00	3603.85

Continued on next page

Table .4 – Continued from previous page

Instance	*T*	*J*	*P*	
			|*T*|x|*J*|	0
			Objective function	
17	25	100	247.65	3607.4
18	25	100	3603.82	3603.32
19	25	100	3600.5	153.48
20	25	100	40.19	3600.54
21	25	100	706.93	3602.85
22	25	100	288.57	3603.02
23	25	100	3600.41	3602.71
24	25	100	3601.51	3604.19
25	25	100	3600.38	3603.39
26	25	100	3601.96	3604.08
27	25	100	3600.53	3606.18
28	25	100	2631.33	3602.42
29	25	100	3603.13	3602.57
30	25	100	3603.86	100.32
31	25	100	3600.52	3602.88
32	25	100	3600.43	274.89
33	25	100	3600.39	3602.73
34	25	100	3601.2	3600.61
35	25	100	1121.73	749.61
36	25	100	3601.95	3603.42
37	25	100	3600.45	3602.41
38	25	100	3603.51	3604.78
39	25	100	1718.09	3604.33
40	25	100	3601.35	3603.14
41	25	100	3605.15	3602.18
42	25	100	2743.02	3602.33
43	25	100	443.03	3604.61
44	25	100	3608.23	3600.41
45	25	100	3601.64	3600.46
46	25	100	3601.64	3607.04
47	25	100	3602.99	3601.28
48	25	100	3601.95	3603.16
49	25	100	3260.06	3603.28
50	25	100	3600.44	3607.01

Appendix D

Here we report an example to explain the relation between the set of *feasible* periods for a slab *j* (T_j) and the set of *feasible* slabs for a period *t* (J_t) according to the following notation:

$J = \{1, \dots, n\}$	Set of slabs, indexed by *j* ;
$T\ = \{\ 1, \dots, \bar{T}\}$	Set of order periods, indexed by t
q^t	Number of slabs to retrieve in period t ;
t_j	Assignment period of slab *j* ($t_j = 0$ if the slab is not assigned to any period)
d_j	Deadline of the slab j;
q^t_{res}	Number of requests of the period *t* still not satisfied;
J_t	Set of slabs with a deadline equal or greater than *t* and with $t^j = 0$
T_j	Set of periods equal or lower than the deadline of the slab *j*
E_t	Subset of slabs with d_j = *t*
E^0_t	Subset of slabs with d_j = *t* and $t^j = 0$;
E^0_{tj}	if *t* ≠ d_j $E^0_{tj} = E^0_t$; otherwise $E^0_{tj} = E^0_t$ −$\{j\}$;

Consider a horizon time of 3 periods (T={1,2,3}) and the presence of only one stack with *n* = 8 slabs of the same item; at each slab is associated a deadline whose value is between 1 and 4. Suppose a set of requests for each period $q^1 = 1$; $q^2 = 4$; $q^3 = 1$. Figure .1 illustrates the elements of this example where nuances of increasing intensity are used for different values of deadlines. Suppose we are at the first iteration of an assigning procedure aiming at associating the slabs to the periods in order to satisfy the requests and do not violate the deadline constraint of the slabs. At the first iteration any slab is assigned to any period: $t_j = 0\ \forall\ j \in J$ and $q^t_{res} = q^t\ \forall\ j \in T$. Hence, let's suppose we try to assign slab *j* = 6 with a deadline in *2* ($d_6 =\ 2$) to the period *t* = 1. According to the notation the set of *feasible* periods of the slab *j* = 6, i.e. the set of periods equal or lower than the deadline of the slab, is composed of period 1 and 2 (T_6= {1,2}); while the set of *feasible* slabs of the period *t* = 1, i.e. the set of slabs with a deadline equal or greater than the period, is composed of all the slabs in the stack (J_1= {1,2,3,4,5,6,7,8}). According to these sets the slab *j* = 6 can be temporary assigned to the period *t* = 1, but a further check is necessary to definitively assign this slab to the period. Indeed, the slab *j* can be definitively assigned to the period *t* if it is possible to satisfy the requests of the periods successive to *t* with feasible slabs still not assigned. In other words, it is necessary to check if the slab *j* = 6 is not necessary to satisfy any request after *t* = 1. It is quite easy to verify that the slab *j* = 6 is necessary to satisfy one of the requests in period 2, since the requests from period 2 to period 3 are 5, as the slabs with a deadline equal or greater than *t* = 2. In general, to make this check it is possible to consider a backward procedure from the last period $\bar{T}$ to the period *t+1*: at generic period $s|\ t+1 \leq s \leq \bar{T}$, it is sufficient to verify that the sum between the feasible slabs at *s* ($|J_s|$) and the eventual difference between the sum of feasible slabs for the successive periods and the sum of the requests for these periods is at least equal to the request at *s*. Defined this sum as A^s, the condition to verify is:

$$A^s \geq q_{res}^s \quad \forall\, s \in [\, t, \ldots, \bar{T}\,] \qquad (1);$$

and A^s can be evaluated according to the following formula:

$$A^s = |E_{sj}^0| + A^{s+1} - q_{res}^{s+1} \quad \forall\, s \in [\, t, \ldots, \bar{T}\,] \qquad (2);$$

where if $s \neq d_j$ $|E_{sj}^0|$ is the number of slabs not assigned, with a deadline equal to *s*; otherwise (if $s = d_j$) $|E_{sj}^0|$ is the number of slabs not assigned, with a deadline equal to *s* minus 1. This latter subtraction, when the period s is equal to the deadline of the slab *j*, represents the absence of slab *j* for the retrievals from *t*+1 to $\bar{T}$ in the case in which *j* is definitively assigned to *t*.

According to the formula (2), let's check that the slabs *j* = 6 do not verify the condition (1) in each period *s* and cannot be definitively assigned to *t* = 1:

Since *t* = 1, *s* ∈ [2, 3], hence starting from *s* = 3 the condition (1) is verified:

$$A^3 = |E_{36}^0| + A^4 = |E_{36}^0| + |E_{46}^0| = 1 + 1 = 2 \ ;$$

$$A^3 = 2 \geq q_{res}^3 = 1\,;$$

When s = 2, considering the eventual absence of the slabs *j* = 6, the condition (1) is not verified:

$$A^2 = |E_{26}^0| + A^3 - q_{res}^3 = 2 + 2 - 1 = 3 \ ;$$

$$A^2 = 3 < q_{res}^2 = 4\,;$$

Hence, as anticipated, the slab *j* = 6 is necessary to fulfil the requests in the second period and cannot be definitively assigned to *t* =1, since if it was, there will be not enough slabs not expired to satisfy further periods.

	$t = 1$	$t = 2$	$t = 3$
q^t	1	4	1

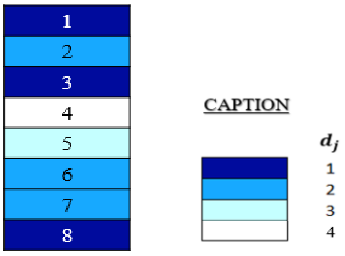


Figure .1 - Stack representation with deadlin

Appendix E

Here we report the pseudocode related to the heuristic to the generation of a random feasible solution developed according to the notation reported in Appendix D:

➢ ***Constructive heuristic to generate random feasible solutions***

```
$t_j \leftarrow 0$ for all $j \in J$
$q_{res}^{t} \leftarrow q^{t}\ \forall\ t \in T$
$\underline{J}_t \leftarrow \{\, j \in J{:}\ d_j \geq t$ and $t_j = 0 \,\}$
while $\sum_{t \in T} q_{res}^{t} > 0$ do
  $t^* \leftarrow 0$
  while $t^* = 0$
    $t \leftarrow$ extract from a uniform distribution in the range $(1,|T|)$
    if $q_{res}^{t} > 0$ then
      $t^* \leftarrow t$
    end if
  end while
  $j^* \leftarrow 0$
  while $j^* = 0$ do
    $j \leftarrow$ extract from a uniform distribution in the range $(1,|\underline{J}_{t^*}|)$
    $j^* \leftarrow j$
    if $j^* > 0$ then
      $s \leftarrow \bar{T} + 1$
      while s > $t^*$
        if $d_{j^*} = s$ then
          $|E_{sj^*}^{0}| \leftarrow |E_{s}^{0}| - 1$
        Else
          $|E_{sj^*}^{0}| \leftarrow |E_{o}^{s}|$
        end if
        if $s \leq \bar{T}$ then
          if $s < \bar{T}$ then
            $A^{s} \leftarrow A^{s+1} + |E_{sj^*}^{0}| - q_{res}^{s+1}$
          Else
            $A^{s} \leftarrow |E_{\bar{T}j^*}^{0}| + |E_{sj^*}^{0}|$
          end if
          if $A^{s} \geq q_{res}^{s}$ then
            if $s = t^*$ then
              $t_j \leftarrow t^*$
              $q_{res}^{t^*} \leftarrow q_{res}^{t^*} - 1$
              $J_{t^*} \leftarrow J_{t^*} - \{\, j^* \}$
              $j^* \leftarrow 0$
            else
              $s \leftarrow s - 1$
            end if
          Else
            $J_{t^*} \leftarrow J_{t^*} - \{\, j^* \}$
            $j^* \leftarrow 0$
            $s \leftarrow 0$
          end if
        Else
          $s \leftarrow s - 1$
      end while
    end if
  end while
end while
```

Step 1-3:

The retrieval time of each slab is initialized to 0 ($t_j \leftarrow 0$ for all $j \in J$) and the residual orders of each period are initialized to the number of requests of each period ($q_{res}^{t} \leftarrow q^{t}\ \forall\ t \in T$).

Steps 4-11:

Until there is still a request to satisfy (**while** $\sum_{t \in T} q_{res}^{t} > 0$ **do**), the reference period t^* is initialized to zero and the set of the feasible slabs in t^* is initialized to $\bigcup_{t \in [t;\bar{T}+1\}} E_t^0$

While (**while** $t^* = 0$) this condition is verified, a period t is extract from a uniform distribution in the range $(1, \bar{T})$ If the residual quantity to satisfy in t is greater than 0 (**if** $q_{res}^{t} > 0$ **then**), the reference period $t^*$is set equal to t ($t^* \leftarrow t$) the set of the slabs not assignable in t^* is initialized to the empty set ($\underline{J}_{t^*} \leftarrow \emptyset$)

Step 12-15:

Then, a slab j is extracted from a uniform distribution in the range (1,| $\underline{J}_{t^*}$|) and reference slab j^* is set equal to j ($j^* \leftarrow j$).

Step 16-49:

If a reference slab has been selected (**if** $j^* > 0$), it is evaluated if the j^* is retrievable in t^*.

Hence, the period s is set equal to $\bar{T} + 1$ ($s \leftarrow \bar{T} + 1$) and until s is different from t^* the number of slabs with deadline in s still not assigned to any period exept for j^* $\left(E_{sj^*}^{0}\right)$ is evaluated.

If $s \neq d_{j^*}$, $e_{oj^*}^{s}$ is equal to the number of slabs with deadline in s still not assigned to any period ($|E_{sj^*}^{0}| \leftarrow |E_s^0|$). Otherwise, $|E_{sj^*}^{0}|$ is the number of slabs with deadline in s subtract by one ($|E_{sj^*}^{0}| \leftarrow |E_s^0| - 1$).

If $s \leq \bar{T}$ it is necessary to check if the slabs still available in s are greater or at least equal to the request in s (**if** $A^s \geq q_{res}^{s}$ **then**).

A^s is equal to the number of slabs available in $s + 1$ minus the requests in $s + 1$ plus the number of slabs with deadline in s still not assigned (Set $A^s \leftarrow A^{s+1} + |E_{sj^*}^{0}| - q_{res}^{s+1}$).

Hence, in each period s from $\bar{T}$ to $t^* + 1$ (**if** $s \leq \bar{T}$ **then**), it is check if $A^s \geq q_{res}^{s}$.

If this condition is verified for each s, the slab j^* is assigned to t^*($t^{j^*} \leftarrow t^*$), the residual orders of $t^*$are updated ($q_{res}^{t^*} \leftarrow q_{res}^{t^*} - 1$) and the subtract $j^*$from J_{t^*} ($J_{t^*} \leftarrow J_{t^*} - \{ j^* \}$). Otherwise, j^* is added to this set J_{t^*} ($J_{t^*} \leftarrow J_{t^*} - \{ j^* \}$), the reference slab j^* is reinitialized to 0 ($j^* \leftarrow 0$) and the procedure to satisfy a request in t^* tries to select a new reference slab present in J_{t^*}

Appendix F

Here we report the pseudocode related to the constructive heuristic oriented to the minimization of the expired slabs according to the notation reported in Appendix D:

Constructive heuristic oriented to the minimization of the expired slabs

```
$t \leftarrow 1$ and $t_j \leftarrow 0$ for all $j \in J$
while $t \leq \bar{T}$ do
  if $|E_t| < q^t$ then
    $t_j \leftarrow t$ for all $j \in E_t$
    $q_{res}^t \leftarrow q^t - |E_t|$
    $t \leftarrow t + 1$
  else
    while $q_{res}^t > 0$ do
      Extract $k$ from a uniform discrete distribution in the range $\lceil 1, |E_t^0| \rceil$
       where $t_{E_t^0(k)} = 0$
      $t_{E_t^0(k)} \leftarrow t$ and $q_{res}^t \leftarrow q_{res}^t - 1$
    end while
    $t \leftarrow t + 1$
  end if
end while
$l \leftarrow 1$
while $l \leq \bar{T}$ do
  if $|E_0^l| > 0$ then
    $j \leftarrow 1$
  while $\exists!\, s \in \lceil 1, l \rceil$ where $q_{res}^s > 0$ do
    while $\exists!\, j \in E_t : t_j = 0$ do
      if $d_j \leftarrow l$ and $t_j \leftarrow 0$ then
        Extract $s$ from a uniform discrete distribution in the range $\lceil 1, l \rceil$ where $q_{res}^s > 0$
        $t_j \leftarrow s$
        $q_{res}^s \leftarrow q_{res}^s - 1$
        $j \leftarrow j + 1$
      Else
        $j \leftarrow j + 1$
      end if
    end while
  end while
end while
$t \leftarrow 1$ and $j \leftarrow 1$
while $t \leq \bar{T}$ do
  if $q_{res}^t > 0$ then
    while $q_{res}^t > 0$ do
      if $d_j \geq t$ and $t_j = 0$ then
        $t_j \leftarrow t$
        $q_{res}^t \leftarrow q_{res}^t - 1$
        $j \leftarrow j + 1$
      Else
        $j \leftarrow j + 1$
      end if
    end while
    $t \leftarrow t + 1$
  Else
    $t \leftarrow t + 1$
  end if
end while
```

Step 1:

It is considered the first period (t ←1), and the value of retrieval time of each slab is initialized to 0 ($t_j \leftarrow 0$ for all $j \in J$).

Step 2-15:

One by one, all the periods are taken under consideration (**while** $t \leq \bar{T}$ **do**).

For each period *t*, if the number of slabs that expired in *t* ($|E_t|$) is equal or lower than the number of slabs requested in that period (q^t) all the slabs with deadline in *t* will be assigned to this period ($t_j \leftarrow t$ for all $j \in E_t$). Hence, the number of requests still not satisfied in *t*, q_{res}^t, will be set by the difference between q^t and $|E_t|$ ($q_{res}^t \leftarrow q^t - |E_t|$).

Instead, if the number of slabs that expired in *t* ($|E_t|$) is greater than the number of slabs requested in that period ($|E_t| > q^t$), until q_{res}^t becomes equal to 0 (**while** $q_{res}^t > 0$ **do)**, a value *k* is extracted from a uniform discrete distribution in the range $\lceil 1, |E_t^0| \rceil$, Hence, the slab corresponding to the *k-th* element of the set e_0^t is assigned to the period *t* ($t_{E_t^0(k)} \leftarrow t$), where E_t^0 represents the ordered set of slabs with deadline *t* still not assigned ($E_t^0 = \{s \in e^t : t_s = 0\}$).

Step 16-32:

Once that all the possible slabs have been assigned to their deadline, *l* is set to 1, and one by one all the periods are taken under consideration (**while** $l \leq \bar{T}$ **do**). Hence, if there are slabs with deadline in *l* still not assigned to any period ($|E_t^0| > 0$), and there is at least a period *s* between 1 and *l,* in which some order need to be satisfied (**while** $\exists!$ s $\in \lceil 1, l \rceil$ where $q_{res}^s > 0$ **do)** from the top of the stack, in a descending order, all these slabs are considered and are assigned to a period *s* (extract from a uniform distribution in the range $\lceil 1, l \rceil$ where $q_{res}^s > 0$).

Step 33-49:

In the end, *t* is set to 1, and, once again, all the period, one by one, are taken under consideration (**while** $t \leq \bar{T}$ **do)** and if they present still some unsatisfied order (**if** $q_{res}^t > 0$ **then**), until the period *t* is completely satisfied (**while** $q_{res}^t > 0$ **do),** from the top of the stack, in a descending order, all the slabs with a deadline greater than *t* and still not assigned (**if** $d_j \geq t$ and $t_j = 0$ **then**) are assigned in *t* (Set $t_j \leftarrow t$).

Appendix G

Here we report the pseudocode related to the heuristic's *local search* according to the notation reported in Appendix E and the further notation:

Initial_solution	Starting solution *;*
Current_solution	Current best solution ($t_j \, \forall j \in J$, where t_j= 0 if j is not assigned to any request
Current_value	Objective function value of the *Current_solution* ;
New_solution	Solution generated by a swap between two slabs;
New_value	Objective function value of the *New_solution* ;
Swapped	Subset of slabs that have been swapped during the procedure;
Best_new_solution	The best solution found in any point of the procedure;
Best_new_value	Objective function value of the *Best_new_solution*;
Continue	Variable equal to 1 if the procedure must be repeated, 0 otherwise

Improving swap procedure

```
$Current\_solution \leftarrow Initial\_solution$
$Current\_value \leftarrow f.o.(Initial\_solution)$
$Best\_new\_solution \leftarrow Initial\_solution$
$Best\_new\_value \leftarrow f.o.(Initial\_solution)$
$Swapped \leftarrow \emptyset$
$Continue \leftarrow 1$
while $Continue > 0$ do
  $Continue = 0$
  $j \leftarrow 1$
  while $j \leq |J|$ do
    $k \leftarrow 1$
    $R_j \leftarrow t_j$
    while $k \leq |J|$ do
      $R_k \leftarrow t_k$
      if $k > j$ or $k \in Swaped$ than
        if $t_j > 0$ then
          if $t_k = t_j$ then
            $k \leftarrow k + 1$
          else
            if $d_j \geq t_k$ and $d_k \geq t_j$ then
              $New\_solution \leftarrow \{t^s \; \forall \; s \in J: t_k = R_j \; t_j = R_k\}$
              $New\_value \leftarrow f.o.(New\_solution)$
              if $New\_value > Best\_new\_value$ then
                $Best\_new\_solution \leftarrow New\_solution$
                Set $Best\_new\_value \leftarrow New\_value$
                $k^* \leftarrow k$
              else
                $k \leftarrow k + 1$
              end if
            else
              $k \leftarrow k + 1$
            end if
          end if
        else
          if $t_k > 0$ and $d_j \geq t_k$
            $New\_solution \leftarrow \{t_s \; \forall \; s \in J: t^k = R_j \; t_j = R_k\}$
            $New\_value \leftarrow f.o.(New\_solution)$
            if $New\_value > Best\_new\_value$ then
              $Best\_new\_solution \leftarrow New\_solution$
              $Best\_new\_value \leftarrow New\_value$
              $k^* \leftarrow k$
            else
              $k \leftarrow k + 1$
            end if
          else
            $k \leftarrow k + 1$
          end if
        end if
      else
        $k \leftarrow k + 1$
      end if
    end while
    if $Best\_new\_value > Current\_value$ then
      $Current\_solution \leftarrow Best\_new\_solution$
      $Current\_value \leftarrow Best\_new\_value$
      $Swapped \leftarrow Swapped \cup \{ j , k^* \}$
      $Continue \leftarrow 1$
    end if
    $j \leftarrow j + 1$
  end while
end while
```

Step 1-6:

Starting from an *Initial_solution* (represented by the set of assignment periods of each slab (*Initial_solution* = $\{t_j\ :\ j\ \in J\}$), procedure sets *Current_solution* and *Best_new_solution* equal to the *Initial_solution* and evaluate the relative objective function values ($Current\_value \leftarrow f.o.(Initial\_solution)$, *Best_new_value* ← $f.o.(Initial\_solution)$ Then, the set of the slabs swapped during the procedure is initialized to the empty set (*Swapped* ← ∅) and the variable *Continue*, that indicates if the procedure must continue (*Continue* = 1) or not (*Continue* = 0) is set equal to 1.

Step 7-9:

While the *Continue* variable is equal to 1, the procedure selects the first slab of the stack, *j* = 1, sets the variable *Continue* equal to 0 and proceeds.

Step 10-13:

Until all the slabs have been considered (**while** $j \leq |J|$), the procedure starts to consider again all the slabs (**while** $k \leq |J|$) starting from the first, *k* = 1, and records the value of t_j as R_j.

Step 14-48:

The current retrieval time of the slab *k* under consideration is recorded as R_k ($R_k \leftarrow t^k$).

if the slab *k* is in a lower position than *j* or the slab *k* has been already swapped (**if** $k > j$ or $k \in Swaped$ **than**) the possible swap between *k* and *j* is evaluated.

If $t_j\ = 0$, this swap is evaluated only when $t_k\ \geq 0$ and $d_j \geq t_k$ Otherwise, if $t_j\ \geq 0$, the swap is evaluated when $t_k\ \ \neq t_j$, $d_j \geq t_k$ and $d_k \geq\ t_j$ If these conditions are verified, the swap between *j* and *k*, is evaluate, creating a *New_solution*, where $t_j\ = R_k$ and $t_k\ = R_j$, and calculating the relative objective function value, *New_value* = *o.f.*(*New_solution*). If the *New_value* under consideration is better than the previous *Best_new_value* found, the *Best_new_solution* and the relative *Best_new_value* are updated (Step 23-24 if $t_j\ > 0$ or Step 37-38 if $t_j\ = 0$) (Set *Best_new_solution* ← *New_solution* and Set *Best_new_value* ← *New_value*)

Step 49-54:

After that all the possible swaps between the slab *j* and the other slabs have been evaluated, if the *Best_new_value* is better than the *Current_value*, the *Current_solution* and the *Current_value* are updated, and the variable *Continue* is set to 1 (Set *Current_solution* ← *Best_new_solution* , Set *Current_value* ← *Best_new_value* and Set *Continue* ← 1)

Step 54-56:

Then, *j* is incremented, ($j \leftarrow j + 1$) until it reaches the value |*J*|, and if at end, at least one update of the *Current_solution* has been performed, having reset to 1 the variable *Continue*, the procedure will start again, otherwise, the procedure will stop

Appendix H

Here we report the results of the computational experiences of the h that have been carried out using a script in *Python* on an Intel(R) Core(TM) i7-8550U with 1.80 GHz and 16 GB of RAM.

These regards the same 50 random instances generated for each (|*T*|,|*J*|) = {(20, 80); (25,100)} used to test the model(1|1). In particular, in Table .5 and Table .6 are reported the comparisons of the results obtained by the heuristic and by the model in terms of objective function and computing time for each combination of (|*T*|,|*J*|) = {(20, 80); (25,100)} for *P* = |*T*|x|*J*| and *P* = 0 respectively. The better value between the model and heuristic both in terms of objective function and computing time is green coloured.

Table .5 Computational results in terms of objective function and computing time(s) for (|*T*|, |*J*|) = {(20,80); (25, 100)} and *P* = |*T*|x|*J*|

Instance	*T*	*J*	*Model*	*Heuristic*	*Model*	*Heuristic*
			Objective function		Computing time (*s*)	
1	20	80	19536	19551	64.56	148.54
2	20	80	6735	6748	3605.19	218.14
3	20	80	16341	16347	1000.21	225.10
4	20	80	17889	17910	3601.41	203.42
5	20	80	10069	10084	3607.14	212.81
6	20	80	14661	14688	673.71	219.33
7	20	80	13148	13165	483.42	204.03
8	20	80	27475	27478	92.36	225.14
9	20	80	13161	13167	3607.24	186.92
10	20	80	14762	14794	3607.61	131.64
11	20	80	16262	16293	17.59	114.99
12	20	80	22687	22693	1001.73	169.41
13	20	80	13169	13186	3600.2	245.60
14	20	80	16327	16336	1490.72	192.34
15	20	80	17921	17942	3607.42	231.50
16	20	80	25885	25893	1000.19	141.11
17	20	80	25882	25904	202.10	187.67
18	20	80	21084	21089	680.82	213.17
19	20	80	27411	27423	1002.36	194.54
20	20	80	17850	17861	30.53	190.32
21	20	80	19440	19441	61.56	216.40
22	20	80	21096	21103	37.16	178.88
23	20	80	27462	27468	43.25	190.40

Continued on next page.

Table .5 – Continued from previous page

Instance	T	J	*Model*	*Heuristic*	*Model*	*Heuristic*
			Objective function		Computing time (s)	
24	20	80	16278	16298	3300.42	239.67
25	20	80	25889	25899	1388.00	174.70
26	20	80	13170	13210	3603.82	229.12
27	20	80	11641	11671	1783.52	190.86
28	20	80	14750	14772	10.38	141.15
29	20	80	25862	25868	1000.23	168.40
30	20	80	13141	13164	28.87	181.73
31	20	80	16385	16385	3600.18	182.63
32	20	80	24266	24266	64.56	204.04
33	20	80	24264	24285	3605.19	195.47
34	20	80	13075	13097	1545.93	221.91
35	20	80	19470	19484	299.88	200.93
36	20	80	14721	14741	175.42	169.45
37	20	80	21062	21062	1000.65	214.04
38	20	80	27470	27480	381.66	194.37
39	20	80	13189	13192	17.41	165.09
40	20	80	13167	13177	12.22	121.12
41	20	80	19555	19584	3602.58	227.72
42	20	80	19528	19533	1000.64	209.60
43	20	80	16359	16369	1418.2	134.27
44	20	80	19565	19596	3605.36	211.42
45	20	80	25857	25861	115.75	197.91
46	20	80	13187	13212	3602.23	187.32
47	20	80	14716	14731	19.20	214.00
48	20	80	17979	17997	12.90	197.23
49	20	80	18016	18019	53.19	174.82
50	20	80	22673	22684	363.73	160.29
1	25	100	47905	47925	3601.92	454.69
2	25	100	33099	33125	465.77	526.10
3	25	100	40420	40434	3601.78	598.44
4	25	100	28045	28058	3600.38	579.12
5	25	100	33011	33048	3607.97	535.29
6	25	100	33080	33102	3603.52	464.93
7	25	100	23051	23109	3601.28	492.26
8	25	100	30491	30513	3602.67	482.44

Continued on next page

Table .5 – Continued from previous page

Instance	*T*	*J*	*Model*	*Heuristic*	*Model*	*Heuristic*
			Objective function		Computing time (s)	
9	25	100	35500	35510	3600.51	456.61
10	25	100	25574	25603	3607.42	515.74
11	25	100	47894	47901	3600.51	378.73
12	25	100	45547	45576	3600.47	436.75
13	25	100	23099	23112	1170.85	566.85
14	25	100	37987	38001	3600.65	407.31
15	25	100	47965	47990	3601.63	422.09
16	25	100	18102	18107	3604.00	643.54
17	25	100	35479	35484	247.65	585.20
18	25	100	57915	57925	3603.82	516.36
19	25	100	47894	47910	3600.50	521.12
20	25	100	33161	33166	40.19	593.64
21	25	100	33103	33109	706.93	594.91
22	25	100	35533	35536	288.57	538.38
23	25	100	18113	18126	3600.41	566.71
24	25	100	37986	38012	3601.51	583.98
25	25	100	28193	28202	3600.38	580.32
26	25	100	35468	35478	3601.96	639.85
27	25	100	32921	32948	3600.53	580.65
28	25	100	30634	30672	2631.33	463.76
29	25	100	32946	32976	3603.13	658.23
30	25	100	38057	38078	3603.86	408.68
31	25	100	43021	43033	3600.52	470.00
32	25	100	47890	47896	3600.43	371.66
33	25	100	45486	45510	3600.39	499.74
34	25	100	50440	50443	3601.20	549.23
35	25	100	23066	23102	1121.73	460.45
36	25	100	50506	50523	3601.95	417.32
37	25	100	28031	28043	3600.45	444.91
38	25	100	28043	28049	3603.51	532.70
39	25	100	30439	30465	1718.09	673.02
40	25	100	38050	38060	3601.35	609.50
41	25	100	52950	52952	3605.15	577.88
42	25	100	33017	33021	2743.02	599.38
43	25	100	28086	28113	443.03	485.72

Continued on next page

Table .5 – Continued from previous page

Instance	*T*	*J*	*Model*	*Heuristic*	*Model*	*Heuristic*
			Objective function		Computing time (*s*)	
44	25	100	45409	45422	3608.23	537.44
45	25	100	30546	30562	3601.64	439.42
46	25	100	37929	37936	3601.64	541.56
47	25	100	40377	40388	3602.99	644.80
48	25	100	37939	37982	3601.95	437.83
49	25	100	38028	38041	3260.06	516.18
50	25	100	52950	25543	3600.44	548.15

Table .6 Computational results in terms of objective function and computing time(s) for (|*T*|, |*J*|) = {(20,80); (25, 100)} and *P* = 0

Instance	*T*	*J*	*Model*	*Heuristic*	*Model*	*Heuristic*
			Objective function		Computing time (*s*)	
1	20	80	65	68	1749.74	487.99
2	20	80	37	40	48.93	350.39
3	20	80	45	50	272.85	330.58
4	20	80	41	44	545.13	409.09
5	20	80	104	123	3599.27	533.17
6	20	80	27	32	28.18	564.56
7	20	80	13	13	6.31	563.53
8	20	80	25	26	12.29	209.98
9	20	80	119	127	3600.21	208.43
10	20	80	197	202	3600.15	186.18
11	20	80	3	3	0.76	160.63
12	20	80	40	43	78.89	188.68
13	20	80	90	86	3603.88	283.06
14	20	80	18	19	8.46	192.48
15	20	80	27	30	360.42	220.54
16	20	80	31	31	30.13	177.88
17	20	80	40	41	102.13	188.06
18	20	80	38	38	43.23	206.5
19	20	80	54	58	587.74	198.98
20	20	80	24	27	10.62	190.95
21	20	80	32	32	38.58	237.87
22	20	80	45	46	3600.2	162.95
23	20	80	37	42	70.69	175.58

Continued on next page

Table .6 – Continued from previous page

Instance	T	J	*Model*	*Heuristic*	*Model*	*Heuristic*
			Objective function		Computing time (s)	
24	20	80	22	22	12.31	213.27
25	20	80	21	21	27.67	193.97
26	20	80	69	72	3352.1	262.34
27	20	80	56	56	486.42	276.19
28	20	80	52	55	488.54	200.93
29	20	80	35	36	51.97	167.3
30	20	80	48	50	77.2	253.91
31	20	80	101	112	3600.44	237.62
32	20	80	25	27	14.57	171.85
33	20	80	18	22	7.35	173.22
34	20	80	29	33	19.69	228.94
35	20	80	44	50	116.49	192.61
36	20	80	39	40	106.33	185.13
37	20	80	27	27	7.82	196.39
38	20	80	69	76	2927.66	198.26
39	20	80	21	21	12.5	195.25
40	20	80	22	22	22.95	353.66
41	20	80	80	88	3600.19	399.34
42	20	80	35	40	347.45	293.11
43	20	80	63	63	1562.61	356.14
44	20	80	47	47	81.92	411.36
45	20	80	27	33	22.79	287.11
46	20	80	82	80	3605.16	370.86
47	20	80	22	25	18.03	346.99
48	20	80	28	30	35.02	186.72
49	20	80	125	126	3604.23	200.87
50	20	80	51	54	410.33	204.8
1	25	100	80	91	3601.62	392.95
2	25	100	65	67	1704.87	734.81
3	25	100	70	75	3601.96	566.56
4	25	100	71	73	3602.07	591.66
5	25	100	86	82	3605.4	527.67
6	25	100	119	123	3602.05	550.6
7	25	100	109	117	3601.62	652.36
8	25	100	41	45	3600.54	568.88

Continued on next page

Table .6 – Continued from previous page

Instance	*T*	*J*	*Model*	*Heuristic*	*Model*	*Heuristic*
			Objective function		Computing time (*s*)	
9	25	100	64	68	3607.63	483.04
10	25	100	101	86	3602.61	528.6
11	25	100	46	46	3224.03	417.32
12	25	100	172	172	3603.84	575.46
13	25	100	103	100	3603.61	585.66
14	25	100	26	29	25.98	459.57
15	25	100	82	93	3618.01	428.45
16	25	100	111	112	3603.85	694.49
17	25	100	98	84	3607.4	607.08
18	25	100	96	106	3603.32	531.07
19	25	100	35	39	153.48	427.75
20	25	100	60	66	3600.54	649.05
21	25	100	80	83	3602.85	620.23
22	25	100	116	121	3603.02	679.15
23	25	100	139	115	3602.71	833.41
24	25	100	83	92	3604.19	531.36
25	25	100	169	153	3603.39	756.85
26	25	100	80	68	3604.08	551.67
27	25	100	97	112	3606.18	566.73
28	25	100	167	177	3602.42	567.51
29	25	100	93	100	3602.57	655.1
30	25	100	26	32	100.32	484.69
31	25	100	145	141	3602.88	550.46
32	25	100	36	39	274.89	346.63
33	25	100	123	116	3602.73	535.52
34	25	100	69	70	3600.61	465.02
35	25	100	51	58	749.61	634.42
36	25	100	79	86	3603.42	457.26
37	25	100	111	92	3602.41	505.3
38	25	100	73	77	3604.78	591.14
39	25	100	97	110	3604.33	650.23
40	25	100	97	82	3603.14	568.8
41	25	100	71	77	3602.18	625.91
42	25	100	121	123	3602.33	638.14
43	25	100	122	139	3604.61	480.69

Continued on next page

Table .6 – Continued from previous page

Instance	T	J	*Model*	*Heuristic*	*Model*	*Heuristic*
			Objective function		Computing time (s)	
44	25	100	47	50	3600.41	584.66
45	25	100	71	71	3600.46	518.31
46	25	100	91	99	3607.04	538.72
47	25	100	40	49	3601.28	540.16
48	25	100	65	73	3603.16	406.38
49	25	100	195	205	3603.28	643.51
50	25	100	61	72	3607.01	641.42