UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
" *INGEGNERIA INDUSTRIALE* "

CICLO XXXIII

in CO-TUTELA con
HOCHSCHULE FÜR TECHNIK STUTTGART

IMPROVEMENT OF LOGISTICS AUTOMATION: A FOCUS ON UNCONVENTIONAL SOLUTIONS

Coordinatore:
Prof. Gianni ROYER CARFAGNI

Tutore:
Prof. Massimo BERTOLINI

Co-Tutore:
Prof. Dieter UCKELMANN

Dottorando:
Mattia NERONI

Anni Accademici 2017/2018 – 2019/2020

Alla mia mamma,
a cui non interessa il mio titolo di studio
o l'argomento di una tesi scritta in una lingua che non capisce,
ma per la quale sarò sempre speciale.

Ai miei amici, la famiglia che mi sono scelto,
le prime persone a cui penso quando sto bene e quando sto male,
i primi a cui dedicherò ogni successo.

A Massimo,
che sapevo essere un grande professore
ed ora so essere una grande persona.

Summary

Publications

I declare that this thesis has been composed by myself and that the work has not been submitted for any other degree or professional qualification. I confirm that the work submitted is my own, except where work which has formed part of jointly-authored publications has been included. My contribution and those of the other authors to this work have been explicitly indicated below. I confirm that appropriate credit has been given within this thesis where reference has been made to the work of others. Part of the work in this thesis is therefore documented in the following publications and papers currently under review.

Published:

Bertolini, M., Neroni, M., Zammori, F. (2020). A flexible operating tool to provide an efficient project's staffing and resource allocation. *International Journal of Project Organisation and Management*.

Bertolini, M., Melloni, R., Neroni, M. (2020). Order picking: a comparison of heuristic and meta-heuristic approaches. *25th Summer school Francesco Turco*.

Bertolini, M., Mezzogori, D., Neroni, M. (2019). Allocation of items considering unit loads balancing and joint retrieving. *24th Summer School Francesco Turco*.

Bertolini, M., Esposito, G., Mezzogori, D., Neroni, M. (2019). Optimizing Retrieving Performance of an Automated Warehouse for Unconventional Stock Keeping Units. *Procedia Manufacturing*.

Bertolini, M., Neroni, M., Romagnoli, G. (2018). A new heuristic algorithm to improve the design of a vertical storage system. *23rd Summer School" Francesco Turco"-Industrial Systems Engineering*.

Under review:

[1] Zammori, F., Neroni, M., Mezzogori, D. Cycle time calculation of shuttle-lift-crane based automated storage and retrieval system. *IISE Transactions*.

[2] Bertolini, M., Neroni, M., Uckelmann, D. A survey of literature on automated storage and retrieval systems from 2009 to 2019. *International Journal of Logistics Systems and Management*.

[3] Bertolini, M., Braglia, M., Marrazzini, L., Neroni, M. Project Time Deployment: A new lean tool for losses analysis in Engineer-to-Order production environments. *International Journal of Production Research*.

[4] Bertolini, M., Juan, A., Neroni, M. A biased randomised discrete event heuristic algorithm to improve the performance in automated storage and retrieval systems. *Computers and Industrial Engineering*.

[5] Bertolini, M., Neroni, M., Zammori, F. A dynamic operative framework for allocation in automated storage and retrieval systems. *Expert Systems with Application*.

Declaration

I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where states otherwise by reference or acknowledgment, the work presented is entirely my own.

Abstract

The benefits coming from logistics and warehousing are unanimously considered of paramount importance in industrial environments. A well-designed logistics allows important advantages, such as a reduced risk of stock-out, a mitigated bullwhip effect, and a reduced lead time. Moreover, the spread of automation which has taken hold in the last years can lead to further advantages (i.e. labour saving, greater specialisation of the employees, increased storage capacity, reduced throughput time, reduced recurrence of errors and damages, etc.). For this reason, during the last years, the scientific community focused on the implementation of automated solutions in the logistics field, addressing several design, management and control issues. Many automated solutions have been studied, although it seems that some industrial fields have been neglected. One of them is the steel industry, which is characterised by a high-level automation in production and manufacturing tasks, boasts over USD 500 billion value-added per year, and employs more than 6 million people. Nevertheless, it is poorly studied from the logistic point of view, and the automated storage and retrieval solutions are few and little studied in relation to its importance in the global economy. The objective of this work is to fill this gap by proposing several algorithms, methodologies, and operative policies focused on this neglected sector to improve different aspects of logistics. More in detail, three different crucial aspects such as (i) logistics of small parts, (ii) logistics of bulky parts, and (iii) managerial issues are considered, and, for each of them, the most widespread problems are addressed. In doing this, not only control policies and operational aspects are considered, conversely, the implementation of unconventional automated storage and retrieval solutions are analysed and their functioning is improved by proposing new algorithms, design decisions and control policies. Each time a new algorithm, operating policy, or methodology is proposed, a case study is carried out to validate it in a real industrial case, or, alternatively, comparing it to the solutions already proposed in literature.

Keywords: Logistics, Warehousing, Automated Storage and Retrieval System, Algorithm, Automation.

1. Introduction

1.1. The value of logistics

Logistics, in general business sense, is the management of the flows between a point of origin and a point of destination, and it broadly consists in the process of coordinating the movement of resources (e.g. people, materials, inventory, and equipment) (Buurman, 2002). The resources managed in logistics may include tangible goods, such as materials, equipment, and supplies, as well as work-in-progress manufactured in outsourcing and other consumable items. Anyway, this flow of physical items also involves the integration of information flows concerning for instance the transportation in itself, the handled goods, the state of the inventory, or other shareable information. Usually, the point of origin corresponds with a producer and the destination is a point of consumption, although, with the emergence of concepts such as the *closed loop supply chain* and the *reverse logistics*, the process of moving goods from their typical final destination to a new starting point for the purpose of proper disposal is becoming more and more popular (Lambert et al., 2008).

The first implementations of logistics concern the military field, in which logistics was concerned with maintaining army supply lines while disrupting those of the enemy. This application was already practiced in the ancient world and is still used even if more advanced solutions have been developed. Nowadays, logistics is widely used in industrial environments and it is unanimously considered as a discipline of paramount importance to gain competitive advantage and meet the customers' needs. In fact, a well-designed logistics increases the economic added-value of products, making them available at the right time in the right place (Penteado and Cicarelli, 2016). Logistics involves many different issues and decisions, most of which regard the storage and handling of raw materials, components, work-in-progress (WIP), or finished goods (Zhang and Lai, 2006). For sake of clarity, adopting a classification already supported by Van den Berg and Zijm (1999), the activities involved might be classified into two categories:

- Inventory management. The activities concerning the products in stock, their quantity and their quality level, such as controlling and overseeing ordering inventory, storage of inventory, and controlling the amount of product for sale. It includes order policies (e.g. economic order quantity, economic order interval, vendor managed inventory, drop shipping, etc.), sales reports, ABC analysis, and rotation index calculation.
- Warehouse management. The processes related to maintaining and controlling the warehouse, such as layout definition, put away operations, retrieving operations, packing, picking, sorting, shipping, and managing returns. These processes are very

important to integrate the warehouse with other elements in the facility, such as production lines, and customers orders.

Both aspects are equally important, since a well-designed and well-managed warehousing system can lead to great benefits, such as a reduced risk of stockouts, a mitigated bullwhip effect, and a reduced lead time of final products (Zhang and Lai, 2006). This is just an introductory consideration, for more details on the value of logistics and how to quantify it, the author suggests the work proposed by Lambert and Burduroglu (2000).

1.2. First steps towards automation

Automation is one of the keywords that best represents the last decade. It became transversal in all the industrial sectors, giving rise to a real industrial revolution which in occasion of the Hannover fair in 2011 was called '*Industry 4.0*' (Davies, 2015). The most surprising notion is the decrease in the cost of robot units, CNC machines, and automated tools. Between 2007 and 2014 the cost of a robot arm has decreased from $550,000 to $20,000, and, proportionally, a similar decrease might also be found in Internet of Things (IoT) devices for ent-to-end communication (GP Bullhound, 2019). The growth not only regards automated physical machines, but software, data, and telecommunication too. Nowadays, factories and manufacturing industry generate 1.8 petabytes of data each year, twice as much as the governments (GP Bullhound, 2019), and this leads to a great need of structured data collection and data analysis. The rapid growth of automation has been leading to a transformation of jobs. The human component still covers a key role, however, it will be more and more important throughout the activities of training and design, by gradually abandoning more repetitive tasks. According to the Oxford Economics (2019), on a global scale, we could suffer a loss of up to 20 million jobs because of automation by 2030, since each robot replaces on average 1.7 job places. These statistics may be alarming, but it should also be specified that the loss of jobs is directly related to the most repetitive functions, and that, according to the Boston Consulting Group (BCG), 70% of people surveyed prefer that the most repetitive and least interesting parts of their jobs be automated with artificial intelligence. Moreover, automation also brings great benefits; for instance, in logistics, the new automated warehousing solutions, generally known as Automated Storage and Retrieval Systems (AS/RS), when compared to classic manual warehouses, provide undeniable advantages, such as labour saving, increased storage capacity, increased throughput (or reduced cycle time), greater specialization of the employees, and reduced recurrence of errors and damages (Roodbergen and Vis, 2009). Despite the benefits an AS/RS can lead to, it is a complex machine, in which many logistic processes must be

executed with no human interaction, and in the shortest possible time. The implementation of an AS/RS might be detrimental if the design decisions and control policies are not consistent with needs, and, in order to avoid this, many different problems such as the scheduling of retrievals, the scheduling of storage operations, the assignment of items in stock to customers orders, the assignment of delivery trucks to output points, the routing of machines involved, and many else must be considered. Furthermore, because of the complexity of the system, the problems reported above must be frequently tackled together, since any decision might have a consistent impact on the whole system (Chen et al., 2010). For this reason, the support of the scientific community and the collaboration between companies and universities is essential to guarantee a proper implementation of the automated solutions, so as to guide the growth and development of enterprises.

1.3. The value of the steel industry

The steel industry, according to an analysis by the World Steel Association (2019) (WORLD STEEL IN FIGURES 2019), is in the heart of global development. In 2017, the steel industry sold US$2.5 trillion worth of products and created US$500 billion added-value. Moreover, the same study shows that for every $1 of value that is added by work within the steel industry itself, a further $2.50 of value-added activity is supported across other sectors of the global economy because of purchases of raw materials, goods, energy and services. This generates over US$1.2 trillion of added-value and proves the global relevance of an often forgotten and poorly considered sector. Concerning the employment side, the steel industry employs more than 6 million people and for every 2 job places in the steel sector, 13 more job places are supported throughout its supply chain.

As reported by World Steel Association (2019) and shown in Table 1.1 and represented in Figure 1.1, the global crude steel production has grown exponentially since the fifties. In particular it has grown from 189 Mt to 2008 Mt. Note for transparency that the value reported for 2020 is a forecast made in 2018 which has not been verified yet.

The production leaders are China, Indian, and Japan, which together in 2017 and 2018 were approximately generating more than 50% of global production, followed by the United States, Russia and South Korea. Among the European countries that stand out, there are Italy and Germany, that still remain among the top ten global producers.

Table 1.1. Global production of crude steel from 1950 to 2020 (World Steel Association, 2019).

Years	Global production [Million of tonnes]
1950	189
1955	270
1960	347
1965	456
1970	595
1975	644
1980	717
1985	719
1990	770
1995	753
2000	850
2005	1148
2010	1433
2015	1620
2020	2008 (forecast)

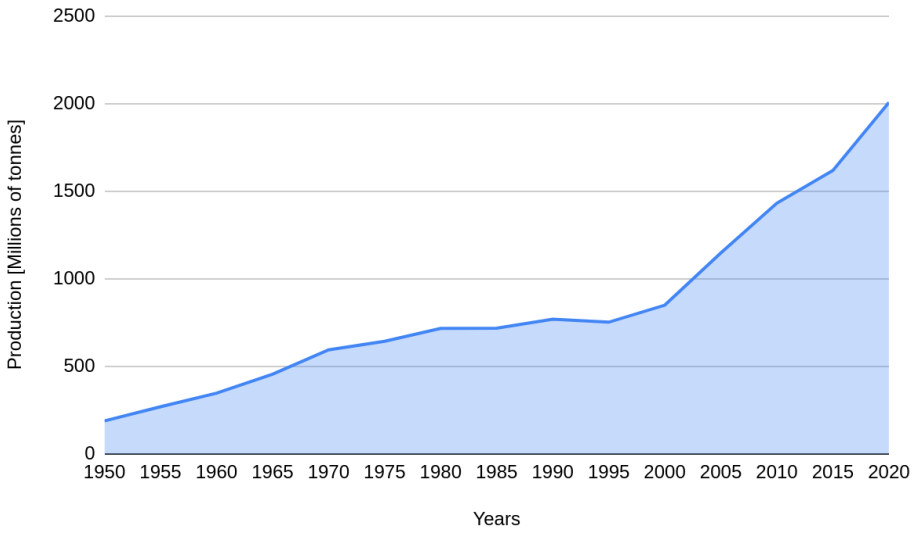


Figure 1.1. The global crude steel production from 1950 to 2020 (World Steel Association, 2019).

The classification of the top ten main producers is reported in Table 1.2, where it is clearly visible that China dominates this market (World Steel Association, 2019).

Table 1.2. Main crude steel producing countries in 2018 and 2017.

Country	2018		2017	
	Rank	Million of tonnes	Rank	Million of tonnes
China	1	928	1	870
India	2	106	3	101
Japan	3	104	2	104
United States	4	86	4	81
South Korea	5	72	6	71
Russia	6	71	5	70
Germany	7	42	7	43
Turkey	8	37	8	37
Brazil	9	34	9	34
Italy	10	24	10	24

Conversely, comparing the production data with import and export ones (Table 1.3 from World Steel Association (2019)), it is possible to see that, in relation to crude steel production, the European nations are those which export the most. In fact, Germany exported around 62% of its annual production in 2018, while Italy exported the 75%. European Union (EU) is also the major importer, and most of the European countries import a quantity similar to that exported. This represents a situation where trade between companies of EU member countries is very frequent and where companies of EU often prefer a foreign supplier to that of a supplier in their area.

The steel sector is therefore a very important element for the global economy and its relevance is unanimously verified, although, concerning the development and investment in automation is important to make some clarifications. The manufacturing in the steel industry is characterised by high-level automation, and almost all production lines are fully or partly automated. The implementation of automated solutions in the production lines started in the early sixties, with first programmable computers controllers of individual processes, and it has always been driven by physical needs (Williams, 1992), i.e. the weight of the handled items, their size, the great efforts required in production, and the necessity to move incandescent items. On the other hand, the logistics in the steel industry is still in its early stages: the automated solutions are few and little studied, and the flow control decision-making processes are much less structured than those adopted in the Large-Scale Retail Trade (LSRT).

Table 1.3. Main exporters and importers of steel in 2018 (World Steel Association, 2019).

Exports			Imports		
Rank	**Country**	**Million of tonnes**	**Rank**	**Country**	**Million of tonnes**
1	China	68.8	1	European Union *	44.9
2	Japan	35.8	2	United States	31.7
3	Russia	33.3	3	Germany	26.6
4	South Korea	30.1	4	Italy	20.6
5	European Union *	28.4	5	Thailand	15.5
6	Germany	26.0	6	South Korea	14.9
7	Turkey	19.9	7	France	14.9
8	Italy	18.2	8	Belgium	14.8
9	Belgium	18.0	9	China	14.4
10	Ukraine	15.1	10	Viet Nam	14.1
11	France	14.4	11	Turkey	14.0
12	Brazil	13.9	12	Mexico	13.1
13	Taiwan, China	12.3	13	Poland	12.1
14	India	11.1	14	Indonesia	11.7
15	Netherlands	11.0	15	Spain	10.8
16	Iran	9.3	16	Nederland	10.3
17	Spain	8.6	17	Canada	9.1
18	United States	8.6	18	Philippines	9.1
19	Austria	7.5	19	India	9.0
20	Canada	6.4	20	Malaysia	8.0

* Excluding intra-European trade. Conversely, data for European countries include intra-European trade.

However, because of the advantages they provide, the AS/RSs have quickly spread in many different contexts during the last decade, and they found application in the steel industry too. The AS/RSs adopted in this sector are very different from those adopted in other environments, such as AS/RS for pallets (Roodbergen and Vis, 2009), miniloads (Foley et al., 2004), shuttle-based AS/RS (Kosanić et al., 2018), and many others. The differences are mostly since in the steel sector the systems have to move unconventional stock-keeping units, generally heavier and bulkier, such as slab, bloom, billet, tube and bar bundles, metal sheets bundles and so on. Unfortunately, because of physical and structural differences, solutions already proposed for classic AS/RSs are hardly implementable in systems designed for the steel sector, and, conversely, the solutions specific for the steel sector seem to have been ignored by the scientific community.

1.4. The value of the agricultural automotive

The companies operating in the steel industry supply raw materials to many other relevant sectors. For this reason, each dollar invested in the steel industry indirectly generates an added value in other industrial environments. One of the main manufacturing sectors and consumers of steel is the automotive. Because of this, an overview on the importance of this sector would make even more clear the importance of the steel industry. In particular, in this section the attention is focused on the agricultural automotive: a highly competitive market characterised by the presence of few prominent players (e.g. John Deere, AGCO, CNH Industrial), along with several medium and small-scale players accounting for the market share. In addition to being one of the steel sector customers, the agricultural automotive has also very similar logistics needs. Some of the automated solutions presented in the next chapter, as well as many of the procedures proposed in this work could be reimplemented in agricultural automotive too. For this reason, to the author's view it is important to highlight the importance of agricultural automotive. The spread of automation and technology in agricultural environments is proceeding quickly and seems to be the new frontier of automation and industry 4.0 (Size, 2019). Strong economic growth in developing countries such as China, India, and Middle Eastern countries is projected to drive the farm machinery industry. Asia Pacific is expected to emerge as the largest market and witness the fastest growth over the forecast period. China alone held over 30.0% of the regional revenue in 2018. Regional growth can be ascribed to low levels of mechanization and large areas of agricultural land. However, mechanization of various agricultural processes is evolving progressively in Asia Pacific, which is estimated to spur demand for agricultural machinery over the forecast period. On the other hand, North America held a share of over 22.0% in 2018 and is anticipated to exhibit strong growth by 2025. This is ascribed to introduction and utilization of machines with better fuel efficiency and improved features. Additionally, shortage of farm labor is expected to be one of the factors driving demand for farm equipment in North America (Size, 2019).

Harvesters and tractors are the leading products of this market. For instance, the tractors contributed to over 25.0% of the total revenue in 2018, are essential and used for several activities, both as a standalone equipment and in tandem with other implements. Additionally, their sales are expected to increase as well as their level of automation and their productivity. Concerning the harvesters, always according to Size (2019), they are expected to account for a share of over 21.0% in the agriculture equipment market by 2025. Moreover, more and more agricultural vehicles are equipped with technologies such as

Global Positioning System (GPS) that might be used for real time positioning to implement intelligent logics and obtain further improvements in productivity.

1.5. The objective of this study

Logistics is therefore an aspect of paramount importance to obtain competitive advantage and meet the customers needs (Zhang and Lai, 2006), and automation can increase the benefits for all actors along the supply chain. These considerations are valid in most industrial fields, including the steel sector, which plays a key role in the global economy and the supply of raw materials to other sectors. Unfortunately, the logistics and the implementation of automated solutions in the steel industry are way behind if compared to the production and the logistics in other sectors. There are few vendors of Automated Storage and Retrieval Systems (AS/RS) for the steel products and the core business of most of them consists of other activities such as the processing of steel itself or the sale of AS/RS for different products (e.g. classic AS/RS for pallets, miniloads, etc.). Moreover, to the author's best knowledge, the scientific community has neglected the study of AS/RS for steel and there are very few publications about them. Even the logistics processes in general are little studied and, because of the importance of the field, they should be analysed in more detail.

The objective of this work is to fill this gap by proposing practical solutions to logistics and managerial issues that characterise the steel industry. The proposed solutions mainly consist of algorithms, procedures, methodologies, and frameworks, and they aim to improve several aspects of logistics such as the design of the warehouse, the implemented control policies, the organisation of the work, and the managerial decisions too. The problems faced by the proposed solutions have been identified *(i)* analysing the gaps in the scientific literature, and *(ii)* observing, day after day, the problems encountered by companies in northern Italy that had embarked on a path of collaboration with the University of Parma. Each proposed solution is then validated through a real industrial implementation, a simulation, or a comparison with similar solutions already proposed in scientific literature.

1.6. The workflow of the thesis

This work is essentially divided into 6 main parts, each of which constitutes a chapter in itself. At first, in Chapter 2 a digression on the main automated solutions adopted in the steel sector is presented. In this part, the most widespread AS/RSs are briefly described, some technical notions concerning their software infrastructure and their electrical and mechanical

functioning are provided. Then, how they integrate with the Enterprise Resource Planning (ERP) and the processing of input and output requests are described.

In Chapter 3 a literature review of the last 10 years of publications on AS/RS is presented. More than one thousand scientific publications are analysed, classified, and summarised to deliver new academic insights to the field, provide clear guidelines for researchers and practitioners, and prove that the logistics in the steel sector is actually a poorly studied topic. This part is also enriched by a particular analysis originally presented by Kulik, Kulik and Cohen (1980) to map and visualise thematic evolution of the considered research field.

In Chapter 4 an overview of the key performance indexes that might be used in the steel sector is presented. The presented indexes allow to measure the performance, the energy consumption, the maintenance costs, and the risk of damages.

Chapter 5 is dedicated to the logistics of small parts, hence, an overview of the Vertical Lift Modules (VLM) is presented, because it is the most used AS/RS for this kind of products. Then a brief description of the performance measurement in VLM is reported. Finally, two solutions to improve respectively the design and the allocation in case of VLM are proposed. Since in case of small parts the picking is a recurrent topic, a comparison of some possible algorithms for manual order picking is also proposed, in order to find out the most efficient and reliable solution.

The logistics of bulky parts is treated in Chapter 6. A particular AS/RS for storage of long metal bars bundles is described. The usage of this AS/RS is frequent in the steel industry and involves many additional constraints not considered in other warehouses, thus, most of the solutions proposed in this chapter are designed for it. The performance calculation in this AS/RS via analytical approach is presented, and then, three different algorithms for performance improvement are proposed and validated.

Managerial aspects are dealt with in Chapter 7, where, after an overview of managerial problems which might take place, two different tools for a target group of practitioners and managers are proposed. They respectively deal with the reduction of wastes, and the allocations of resources to tasks (also known as project staffing).

Finally, conclusions and future perspectives are presented in Chapter 8. A representation of the workflow is shown in Figure 1.2, where, for each chapter are reported its content and the reason that justify its inclusion in this work.

1. **Introduction**
2. **Digression on the implementation of an automated solution**

   Content. Automated solutions in the steel sector, how they work and how they interact with ERP.

   Motivation. Describing the existing automated solutions and how they work.

3. **Literature review**

   Content. A literature review on AS/RS that analyses more than 1000 papers.

   Motivation. Proving the scientific community has neglected the automated solutions implementable in the steel sector.

4. **Key performance indexes**

   Content. Key performance indexes that might be used to evaluate the efficiency of the logistic policies.

   Motivation. Explaining the most important parameters to optimize.

5. **Logistics of small parts**

   Content. Solutions to improve design, allocation, and retrieving of small parts.

   Motivation. Most of the algorithms proposed for Vertical Lift Modules do not consider important constraints.

6. **Logistics of bulky parts**

   Content. Analytical and simulation approaches to measure the performance in AS/RS for steel and algorithms to improve them.

   Motivation. AS/RS for steel bulky parts are a poorly studied topic.

7. **Managerial decisions**

   Content. Two innovative tools to support the project managers in reducing wastes and properly allocating resources to project's activities during the realization of an AS/RS.

   Motivation. The realization of an AS/RS is an engineering-to-order project which involves many risks, costs, and important long-term decisions.

8. **Conclusions**

Figure 1.2. The workflow of this thesis.

References

Buurman, J. (2002). *Supply chain logistics management*. McGraw-Hill.

Chen, L., Langevin, A., & Riopel, D. (2010). The storage location assignment and interleaving problem in an automated storage/retrieval system with shared storage. *International Journal of Production Research*, *48*(4), 991-1011.

Davies, R. (2015). Industry 4.0 Digitalisation for productivity and growth. *European Parliamentary Research Service*, *1*.

Foley, R. D., Hackman, S. T., & Park, B. C. (2004). Back-of-the-envelope miniload throughput bounds and approximations. Iie Transactions, 36(3), 279-285.

GP Bullhound Independent Technology Research. (2019). All change at the top. Accessed August 2019. https://www.gpbullhound.com/insights/titans-tech-2019/.

Kosanic, N.Z., Milojevic, G.Z., and N.D. Zrnic (2018). A survey of literature on shuttle based storage and retrieval systems. *FME Transactions*, 46(3), 400-409.

Kulik, C. L. C., Kulik, J. A. and P. A. Cohen (1980). Instructional Technology and College Teaching. *Teaching of Psychology*, 7(4), 199-205.

Lambert, D. M., & Burduroglu, R. (2000). Measuring and selling the value of logistics. *The International Journal of Logistics Management*, *11*(1), 1-18.

Lambert, D. M., García-Dastugue, S. J., & Croxton, K. L. (2008). The role of logistics managers in the cross-functional implementation of supply chain management. *Journal of business logistics*, *29(*1), 113-132.

Oxford Economics (2019). How robots change the world. *What Automation Really Means for Jobs and Productivity*.

Penteado M., M. and Chicarelli A., R. (2016). Logistics activities in supply chain business process. *The International Journal of Logistics Management*, 27(1), 6-30.

Roodbergen, K. J., & Vis, I. F. (2009). A survey of literature on automated storage and retrieval systems. *European journal of operational research*, *194*(2), 343-362.

Size, E. O. M. (2019). Share & Trends Analysis Report By Application (Cleaning & Home, Medical, Food & Beverages, Spa & Relaxation), By Product, By Sales Channel, And Segment Forecasts, 2019-2025. Report ID, 978-1.

Van den Berg, J. P., & Zijm, W. H. (1999). Models for warehouse management: Classification and examples. *International journal of production economics*, *59*(1-3), 519-528.

Williams, T. J. (1992). The Remodernization of Automation in the Iron and Steel Industries in North America. *IFAC Proceedings Volumes*, *25*(19), 23-34.

World Steel Association. (2019). *World Steel in Figures 2009*. Brussels: World Steel Association, 2009: 15.

Zhang, G. Q. and Lai, K. K. (2006). Combining path relinking and genetic algorithms for the multiple-level warehouse layout problem. *European Journal of Operational Research*, 169(2), 413-425.

2. Digression on the implementation of an automated solution

In this chapter the information useful to understand how an automated warehouse is implemented are reported. In particular, the focus is on fully automated solutions specifically designed for the steel sector. All the Automated Storage and Retrieval Systems (AS/RS) typically used in other environments are described in chapter 3, where a review of literature on AS/RS is proposed. The AS/RSs described below in this chapter are strictly designed to store typical products of the metallurgical industries, hence, their implementation in other contexts is not possible. Of course, depending on the handled products, classic AS/RSs can be used by companies operating with the steel, but a warehouse ad hoc is often needed. After an overview of the utilizable warehouses, the infrastructure and the software implementation are described to understand from a technical point of view how the system actually works. The focus of this study consists in operations, procedures, control policies, and managerial decisions, for this reason no strictly electronic and mechanical aspects are concerned. Finally, it is accurately described how the system communicates with the Enterprise Resource Planning (ERP) to understand how the requests are processed and the information elaborated. Even in this case, the focus is not on IT aspects and Application Programming Interfaces (API), but procedures and operational constraints are considered, in order to provide the reader a clear idea of what could and could not be improved by implementing new algorithms and operative policies.

2.1. A brief overview of automated solutions for the steel sector

The most popular and widespread fully automated solutions for the steel industry are essentially of five typologies. Each of them is designed for a particular product, characterised by fairly unusual shape and dimensions. Thus, the AS/RS adopted in the steel sector are the following:

- Vertical Lift Module (VLM). It is an AS/RS inspired by classic VLM used in other industrial environments. It consists in an enclosed system made of two vertically arranged storage areas divided by an aisle, where a delivery lift moves to bring the unit loads back and forth from the I/O point. Unlike usual VLM, such as Vertimag (https://www.ferrettogroup.com/index.cfm/it/soluzioni/magazzini-automatici-verticali/magazzini-verticali-vertimag/) or Modula (https://www.modula.eu/ita/), VLM for the steel sector is designed to move and store unit loads up to 7 meters long and 5000

kilograms heavy. A representation of a VLM for steel is shown in Figure 2.1, and a more detailed description is provided in chapter 5.

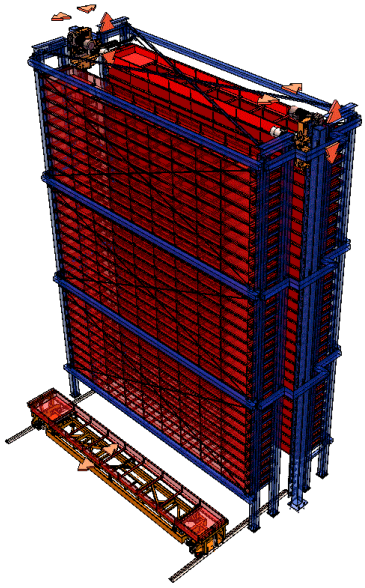


Figure 2.1. Representation of a VLM for the steel sector.

- Honeycomb. A stock intensive system of big dimensions designed for storage of particular unit loads known as 'skid'. Unlike usual unit loads such as those handled by the VLM, the skids are designed to slide and crawl on the floor and inside the storage locations. This makes them particularly suitable for storage of very long products, and the possibility to drag them instead of lifting them increases their useful weight. The honeycomb can be up to 26 meters height and can store around 8000 skids of 5-8 tonnes each. Its width is always 3 times the length of the handled skids, which ranges from 3 to 14 metres. The system is made of 2 different storage areas situated on both sides of an Storage and Retrieval (S/R) machine. The S/R machine can perform vertical and horizontal movements in parallel, and it is equipped with a conveyor or a mobile hook to insert and retrieve the skids from the locations. In some cases, it can handle 2 unit loads at the same time. From the operational point of view, the honeycomb is very similar to a classic AS/RS for pallets, although, for the weight and the length of the unit loads, its kinematic, mechanical, and electrical characteristics are totally different. The representation of a honeycomb is provided in Figure 2.2.

Figure 2.2. Representation of a honeycomb.

- Metal sheets AS/RS. An AS/RS designed to store metal sheets. These sheets are usually stacked on top of each other and placed on wooden unit loads similar to pallets. These unit loads are therefore stored inside the warehouse by using a S/R machine similar to classic S/R machines for pallets. The difference between a standard pallet and a unit load of the metal sheet AS/RS lies in dimensions and weight capacity. The representation of a metal sheet AS/RS is shown in Figure 2.3. The metal sheets are hard to handle especially on occasion of picking and packing operations, when the single sheets must be taken from the unit load. For this reason, the I/O points of the metal sheet AS/RS are often equipped with a machine like that in Figure 2.4, which helps the operator to pick and place the sheets in absolute safety.

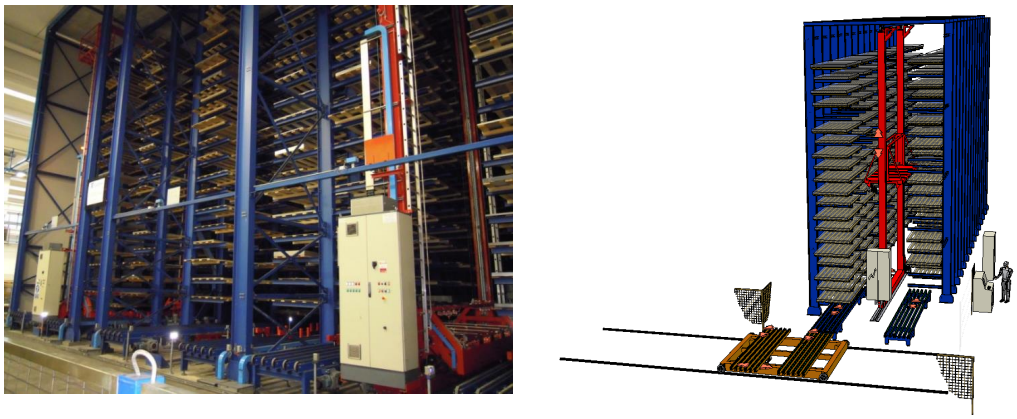


Figure 2.3. Real picture and representation of a metal sheet AS/RS.

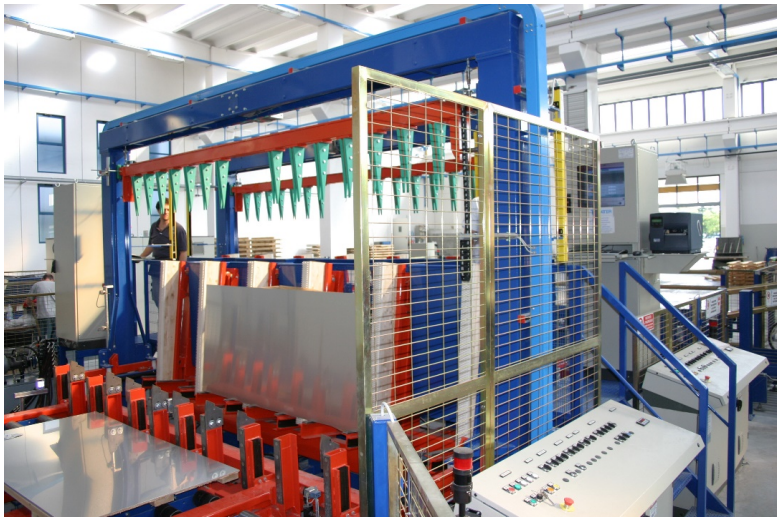


Figure 2.4. Machine for picking in metal sheet AS/RS.

- Shuttle-Crane AS/RS (SC-AS/RS). An AS/RS for long metal boxes like those handled by the VLM. The system is served by two different machines: a shuttle that moves on the floor under the storage area, and a crane that moves on top of the storage area moving on two railway guides supported by bearing metal structure. The unit loads are arranged in columns and every two columns there is an aisle. The crane is equipped with two forks, able to go down along the aisles to retrieve/store the unit loads. The crane and the shuttle are independent, and they can take care of two different operations at the same time. Typically, the crane retrieves a unit load, it waits for the shuttle and places the unit load on it. Then, the shuttle goes to an I/O point, while the crane goes to retrieve the next unit load, or, in case of input

operation, to the aisle where the next unit load will be stored. Having two independent machines may lead important benefits in terms of throughput (i.e. operations per hour).

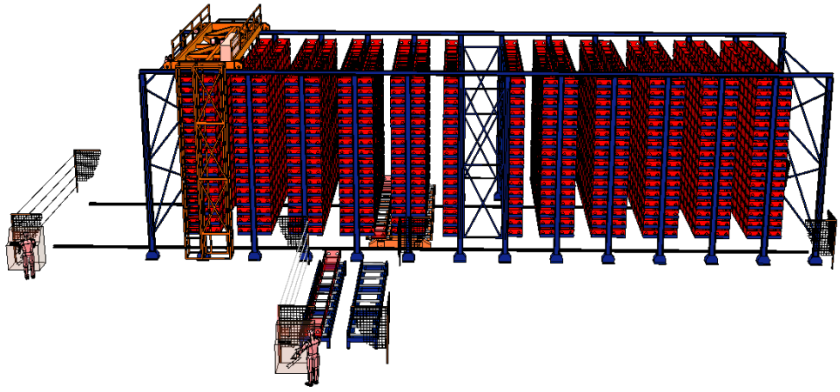


Figure 2.5. Representation of a Shuttle-Crane AS/RS.

- Shuttle-Lift-Crane AS/RS (SLC-AS/RS). A storage system served by three different kinds of machines: (i) shuttles, (ii) lifts, (iii) cranes. The shuttles perform horizontal movements only and bring the products from the I/O points to the lifts and vice versa. The lifts take the products from the shuttles and bring them in height and vice versa. The cranes move on top of the storage areas (i.e. racks) and bring the products from the lifts to the storage locations and vice versa. The whole system may be divided into several areas or racks, and each of them is served by a crane, or, sometimes, two cranes sharing the same path. All the racks are connected to each other and to the I/O points by the shuttles that usually move perpendicularly to the racks' length. The system is designed to store long metal bars' bundles and billets, and there are no unit loads, since these products are stored as is on simple shelves. Because of the characteristics of the billets and the bundles, the picking is not allowed in this system, although many more complications are involved concerning the placement of bundles on the shelves, retrieving, quality of handled items, and so on. A more accurate description of the system is provided in chapter 6, and a picture is presented in Figure 2.6.

Figure 2.6. Picture of a Shuttle-Lift-Crane AS/RS [4].

This work is mainly focused on the VLM and the SLC-AS/RS, as they are the most widespread solutions respectively for small and bulky products. Both involve several constraints and complications which, to the author's best knowledge, have never been treated before, since they do not concern classic AS/RSs for pallets. Conversely, the other solutions concern niche contexts, and, especially for the honeycomb and the metal sheets AS/RS, their mechanism is partially referable to a classic pallet warehouse.

2.2. Infrastructure and software implementation

The AS/RSs mentioned above are based on a multi-tier architecture, which in software engineering means that the functionalities of the software are logically split or divided into several layers in communication with each other. Five main layers might be identified and are represented below in Figure 2.7:

- Enterprise Resource Planning (ERP). It is the company's management software, generally shared between several corporate functions. It receives and processes the customers orders, the purchases, and the accounting in general. At this level, the only available information concerning the warehouse is the quantity in stock per each handled product, plus possible forecasts of demand and consumption. The communication between the ERP and the other layers does not necessarily have to happen in real time, for instance it might take place at the end of each working shift and depends on how quickly the company wants to respond to the customers.
- Warehouse Management System (WMS). It is the centralised software where the main control decisions take place. The WMS knows where each single item in stock

is stored, which customer order it is assigned to, which shipment it is assigned to. Moreover, information concerning the entering unit loads and products are processed too. The algorithms and the control policies are implemented in this layer, where problems such as location assignment, operations scheduling, assignment of retrievals to the output points, and many other control issues are tackled (a more accurate description of the control issues concerning an AS/RS is proposed in chapter 3). The WMS needs to know in real time (i.e. a few seconds or even less) all the information concerning the products in stock, the customers orders, and the entering products. Concerning the machines, usually they do not communicate their position in real time to the WMS, conversely, they send a request or a communication to the centralised system only when they reach a specific position or a predefined destination. The WMS controls the operative policies and the organization of the system, low-level operations such as movements and physical controls are delegated to the lower layers. Sometimes, the WMS is integrated with Supervisory Control And Data Acquisition (SCADA), a software for high-level process supervisory management. In this way, information concerning the failures and the alarms are registered in the same database used by the WMS, and can be used by the WMS to take control decisions.

- Warehouse Control System (WCS). It directs the real-time activities and is responsible for keeping everything running smoothly, maximizing the efficiency of the material handling subsystems and often, the activities of the warehouse associates themselves. It coordinates all the machines involved such as shuttles, S/R machines, lifts, carousels, conveyors, sorters, etc. It drives the single machines for each single operation by tackling issues like routing, sorting, or sequencing, and provides real-time directives to operators to accomplish the order fulfillment. In some cases, it collects statistical data on the operational performance of the system diagnostic and maintenance. It is also responsible for barcodes reading, labelling, and physical controls such as quality, shape and weight checks.
- Programmable Logic Controller (PLC). It is a programmable computer that processes the digital and analogical signals coming from sensors and directed to the actuators on the machines. Each machine is equipped with a PLC on board, which is responsible for repetitive operations, low-level controls, safety stops, etc. The PLC communicates with the WCS via TCP/IP protocols and with others PLCs via standard protocols (e.g. Profibus, Profinet, DeviceNet, Ethercat, Modbus).

- Electrical System. The set of sensors and remote devices used to collect data from the field.

ERP Sales, purchases, accounting.
WMS Identification and collocation of products picking organization and shipment organization.
WCS Labelling, RFID scanning, shape and weight control.
PLC Movements control.
Electrical system Electrical panels, sensors, field devices.

Figure 2.7. Software architecture.

Theoretically, each layer is a stand alone software characterised by a web server and a database, and eventually an interface (i.e. front end). The communication between the different layers is made through standard communication protocols such as TCP/IP or HTTP. However, in some application cases, the distinction between the layers may be not so clear, especially for high-level layers (i.e. ERP, WMS, WCS). There are many examples in real industrial cases where the WMS and the WCS are integrated in a single program characterised by its own database. Unusual is instead the integration of the WMS with the ERP, because the suppliers of the two systems are often different and because the ERP needs to be an independent software shared between different corporate functions. Sometimes, the same control or the same task is carried out by two different layers. This is frequent in case of safety/security tasks, for instance, in AS/RS where two machines share the same path, the control to avoid obstructions and crashes is made twice at WCS and PLC level.

2.3. Integration with the ERP and requests processing

An AS/RS is subject to several material and information flows. These information might be classified in incoming flows (i.e. arrivals from suppliers, arrivals from production lines), inventory management flows (i.e. products identification, weighing, safety controls, storage operations, inventory reorganization during non-working shifts), and outgoing flows (i.e. retrievals for customers orders, retrievals for production orders). Each transaction must be registered and communicated to the ERP in order to update the data concerning the

products in stock. According to these information, the ERP will organise the replenishment, the Production Plan (PP), the Master Production Schedule (MPS), and the Material Requirements Planning (MRP). Each notion concerning batches, castings, serial numbers, quality and suppliers must be preserved to ensure the tracking and tracing of each single item along the whole supply chain. Hence, the information history must be saved to preserve the traceability of materials and unit loads, which is unanimously considered as an added value by the customers and the salers. In fact, through traceability of materials, the firsts are aware of the origin of the purchased product, while the second ones have enough information for data analysis, shipments organization, and demand forecasting. There are two main kinds of traceability (Pinheiro, 2004): *(i)* the forward traceability and *(ii)* the backward traceability. The first one is in the interest of the suppliers that use this information for data analysis, demand forecasting, and shipments organisation. It concerns everyday activities from the customer order to the delivery. Conversely, backward traceability is useful when a problem occurs, and it allows the identification of the shipped products, the relevant lot and the indicted process to investigate the cause. The identification of a batch or a process allows quick actions to improve services and quality, which leads to higher and more stable product quality and ensures greater reliability in the eyes of the customers. To better explain the concept of traceability a schematic representation is given in Figure 2.8.

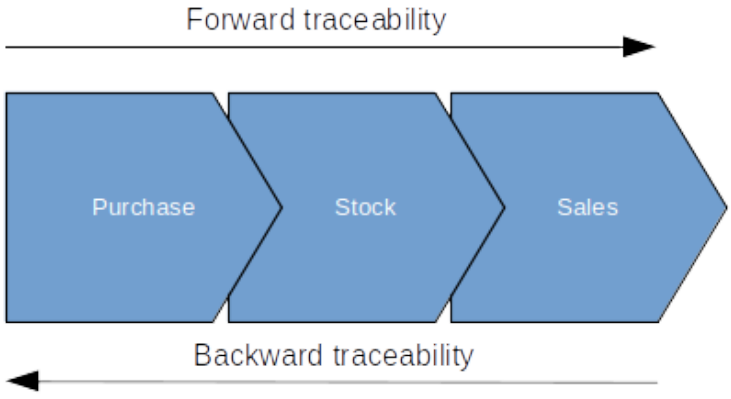


Figure 2.8. Forward and backward traceability.

The data collection and the communication with the ERP may happen in many different ways, depending on the context where the company operates and overall design decisions. Usually, the ERP and the WMS are two different programs running on different servers which communicate via specific API. Each of them has its own database, which is structured in a different manner and contains different data. Given a generic item, the ERP typically keeps track of the supplier, the batch of origin, the production line, the customer who buys it,

and the correspondent shipment. More detailed information concerning the unit load and the location where the item is stored, the movements due to the inventory reorganization, and the actions that usually take place between its arrival and its shipment are managed by the WMS. The clear distinction between the WMS and the ERP, and the necessity to organise the information in two different ways is due to the different function of the two programs. There are, however, real cases where data are split or handled differently.

The behaviour of an AS/RS and the operations that may be taken to improve its functioning depend on the communication between the ERP and the WMS. In particular, they depend very much on the frequency the information is transferred with. The communication between these programs can be in real time (i.e. a few seconds or even less), at fixed intervals, or priority dependent. In the latter case, the communication, especially from the ERP to the WMS, is supposed to happen at fixed intervals, although, for prioritized events is possible a real time communication. For instance, if the ERP receives a customer order requiring a specific item or a product in short inventory, a priority request can be generated, and the WMS froze that item/product (i.e. it avoids assigning it to other customer orders).

The communication with the ERP has a great importance in the design of the control policies. For instance, if the ERP communicated the customers orders only once a day (typically in the evening or at the end of each working shift), during the non-working shift the AS/RS could reorganize the inventory moving the required item as close as possible to the output points. Conversely, if the customers orders are transmitted to the WMS in real time only when the delivery truck must be uploaded, there is no possible preparation and the retrieving operations must be organised just-in-time. While in the first case there is a long time to find the solution, in the latter case the solution must be provided in a short computational time without any possible preparation. This has a significant impact on the control policies, for instance, in the first case the algorithm engaged to organise the retrieving operations could seek the optimum, while, in the latter case, it would be better to implement an heuristic that provides a suboptimal solution in a reasonably short time.

References

Pinheiro, F. A. (2004). Requirements traceability. *Perspectives on software requirements* (pp. 91-113). Springer.

3. Literature review on AS/RS

In this chapter a comprehensive literature review on Automated Storage and Retrieval Systems (AS/RS) is presented with the purpose of understanding what might be useful to reimplement in the steel industry and what is missing instead. The proposed literature review is focused on the last 10 years of publications (i.e. from 2009 to 2019), because the papers prior to 2009 are already treated and described by Roodbergen and Vis (2009). Another purpose of this chapter is to properly classify and analyse the articles lately published on AS/RS, because, in the last decade, numerous papers have been published without receiving a correct classification, and, to the author's best knowledge and according to Scopus, which claims to be *"the largest abstract and citation database of peer-reviewed literature"*, a holistic literature review is missing. Most of the surveys published after 2009, as already noted by Shah and Khanzode (2017), analyse a single aspect or a single typology of AS/RS. More in detail, Shah and Khanzode (2017) address the aspect of performance measurement. Boysen and Stephan (2016) consider scheduling problems in single crane AS/RS. Ekren and Heragu (2012a), Janilionis and Bazaras (2010a), and Janilionis and Bazaras (2010b) focus on autonomous vehicles automated storage and retrieval systems only. Kosanic, Milojevic and Zrnic (2018) consider shuttle based automated storage and retrieval systems, while Dukic et al. (2018) consider vertical lift module systems. None of them provides a comprehensive classification which makes the point concerning all the publications on AS/RS, hence, the paper published by Roodbergen and Vis (2009) needs an update and an extension, which is therefore provided in this chapter.

3.1. Methodology

The AS/RS, because of the large number of involved issues, is a very discussed topic. Thus, to analyse the multitude of articles on AS/RS, a systematic selection process which allows the identification of the most relevant contributions is needed. The selection was therefore made applying the principles of the systematic review (Hart, 1998; Tranfield et al., 2003; Bigliardi and Galati, 2018). As reported in Figure 3.1, to carry out the selection, a methodology of six phases is used.

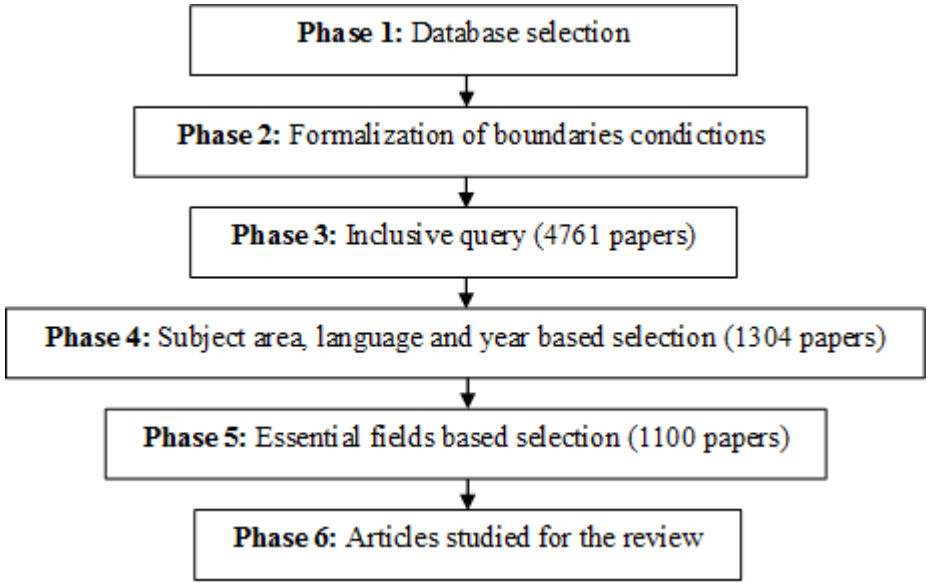


Figure 3.1. Flow chart of the review process adopted [2].

In phase 1, the database selection was carried out and Scopus (https://www.scopus.com/), which claims to be *"the largest abstract and citation database of peer-reviewed literature"*, was chosen. In phase 2, the boundary conditions of research were outlined (e.g. Tranfield et al., 2003) as follows:

- the focus had to be on automated storage and retrieval systems;
- only English papers published between 2009 and 2019 were considered;
- only papers with correct EID and DOI were considered trustworthy;
- only engineering and computer science related subjects were taken into account.

In phase 3, the following query was performed (in June 2020):

TITLE-ABS-KEY(("AS/RS" OR "Automated Storag*" OR "Automated Warehous*") AND NOT ("Non-Automat*" OR "Non - Automat*" OR "Not Automat*")).*

Note that the Scopus search engine is not case sensitive and interprets the asterisks (e.g. "*") as a replacement character to find words with divergent endings. Thus, for instance, the query *TITLE-ABS-KEY ("Automated Warehous*")*, search for all the articles having in title abstract or keywords terms such as "automated warehouse", "automated warehousing", and else. By doing so, 4761 documents were identified. In the next phase (i.e. phase 4) a further

filter has been applied by using the following query (application of all remaining boundaries conditions):

LIMIT-TO (PUBYEAR, 2019) OR LIMIT-TO (PUBYEAR, 2018) OR LIMIT-TO (PUBYEAR, 2017) OR LIMIT-TO (PUBYEAR, 2016) OR LIMIT-TO (PUBYEAR, 2015) OR LIMIT-TO (PUBYEAR, 2014) OR LIMIT-TO (PUBYEAR, 2013) OR LIMIT-TO (PUBYEAR, 2012) OR LIMIT-TO (PUBYEAR, 2011) OR LIMIT-TO (PUBYEAR, 2010) OR LIMIT-TO (PUBYEAR, 2009)) AND (LIMIT-TO (LANGUAGE, "English")) AND (LIMIT-TO (SUBJAREA, "ENGI") OR LIMIT-TO (SUBJAREA , "COMP")).

The result was a selection of 1304 articles published between 2009 and 2019, written in or at least translated to English, and belonging to engineering or computer science subject areas. The papers were further filtered in phase 5 to eliminate those with inappropriate essential fields such as DOI or EID. By so doing, 1100 trustworthy documents remained. The resulting papers were weighted in phase 6. Priority was given to the documents with a greater citation impact I, where, given c the number of citations received and y the years that the paper had been published, it can be calculated by using equation (3.1):

$$I = \frac{c}{y+1} \tag{3.1}$$

3.2. The bibliometric analysis

Before going into the details of the publications extracted, a quantitative study known as bibliometric analysis and never applied before to AS/RS, is presented. As mentioned by Cobo et al. (2011), the objective of this particular analysis originally presented by Kulik, Kulik and Cohen (1980) is to map and visualise thematic evolution of considered research fields. The aim of this section is therefore to analyse the articles on AS/RSs published between 2009 and 2019 by using graphic and bibliometric analysis tools essentially based on the keywords and the citations. This approach is not going into the content of each paper as a systematic literature review might do, conversely, it focuses on very high-level aspects hard to observe by simply reading the articles one by one.

The bibliometric analysis is not based solely on the 1100 articles extracted in the previous section, conversely, papers cited by them are considered too. In the remainder of this section, the papers extracted as described in the previous section are called seed-1 and the papers cited by them are called seed-0. More in detail, starting from the 1100 seed-1 obtained in the previous section, 5259 seed-0 are extracted. It is interesting to note that, as

the topic of selected papers is concerned, seed-0 mostly describe algorithms, control policies, and methods, while seed-1 study their application to AS/RSs.

First of all a keyword-based analysis is presented. Fadlalla and Amani (2015) have already applied a similar analysis to Enterprise Resource Planning (ERP) to derive major areas of emphasis in ERP research. According to procedure defined by Fadlalla and Amani (2015), keywords cited in the considered papers are classified in a matrix like that one described in Figure 3.2, and classified according to the number of times they are mentioned in the considered papers (i.e. frequency), and how old is their first apparition (i.e. age). This classification allows to analyse the keywords in terms of popularity and lifetime, and, since the keywords might be considered a good indicator of the topic treated in the papers (Fadlalla and Amani (2015)), it identifies the most popular topics related to AS/RS and their evolution over time.

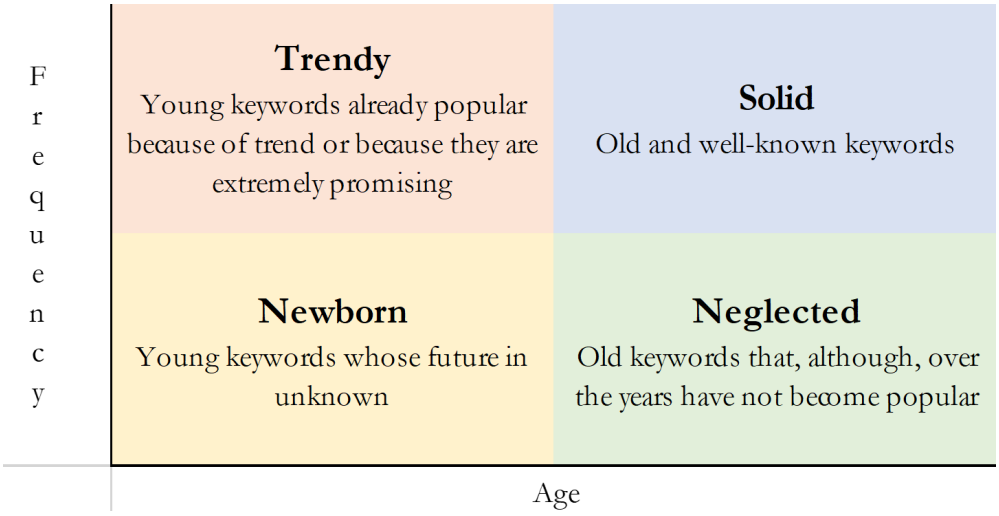


Figure 3.2. Keywords classification matrix [2].

The keywords are classified in a matrix like that one represented in Figure 3.2, and collocated in one of the dials according to their age and frequency. The matrix is supposed to be interpreted as follows:

• In the lower right dial, there are the keywords/topics which have been existing for a long time, but have been cited a few times. They have therefore gained little importance and are classified as neglected.

• In the upper right dial, there are the keywords/topics which have been existing for a long time and have collected a high number of citations. They represent strongly discussed topics in the past and possibly still popular; we call these keywords solid.

• In the upper left dial, there are the highlights of the moment. The keywords/topic in this dial represent the trendy topics on which the research community has been focusing in the last period.

• In the lower left dial, there are the keywords cited a few times because they have been existing for a short time. They are the new-born keywords, and, in the future, they can evolve into trendy or neglected. The topics represented by these keywords which could represent future research topics, as well as nothing interesting.

Since many different keywords refer to similar concepts, before classifying them into the matrix, a clustering is done. For the clustering process the algorithm proposed by Coulter, Monarch, and Konda (1998), and well-described also by Cobo et al. (2011), is used. The clustering algorithm basically looks for keywords that often appear together and collects them in a single cluster, which is named as the most recurring keyword. The functioning of the algorithm is not relevant for the purpose of this paper, and, for a more detailed description of it, the authors suggest to refer to its first presentation (Coulter, Monarch, and Konda, 1998) or the detailed description provided by Cobo et al. (2011).

The keywords analysis was however limited to the seed-1 papers, because the inclusion of the seed-0 would have caused confusion by introducing many keywords, most of which not very relevant. Indeed, the seed-0 are the papers cited by the seed-1 and there was no way to verify in this step if they concern AS/RSs or other topics close to them (e.g. operations research tools, manual warehouses, vehicle routing problems, etc.). Hence, to avoid complications and misinterpretation of results, the keywords-based analysis described above was limited to seed-1 papers. The resulting matrix is presented in Table 3.1. It is important to point out that, for the definition of the thresholds between upper and lower dials or right and left dials, the average frequency and age were respectively used.

Table 3.1. The result of keywords analysis [2].

vertical lift module; **hybrid storage system**; **sbs/rs**; **analytical modelling**; caching; constraint programming; cyber physical systems; dynamic tiering; **multi aisle**; automatisation; **autonomous vehicle storage and retrieval system**; average retrieval time; flow rack; **green design**; **multi deep as/rs**; **multi shuttle as/rs**; **multi-tier shuttle warehousing system**; processing; rack shape; robotics; **shuttle based storage and retrieval system**; storage retrieval scheduling; **swarm intelligence**; **tabu search algorithm**; throughput model; warehouse picking; classification; crane scheduling; internet of things; order batching; performance evaluation; retrieval time; scheduling modeling; shuttle based systems; compact storage systems; dynamic sequencing; forklift; industry 4.0; pallet detection; puzzle based storage; retrieval scheduling; shaking table test; **free fall flow rack as/rs**; **particle swarm algorithm;** optical detection system.	advanced manufacturing technology; **agv**; **ant algorithm**; **ant colony algorithm**; assistance system; automated storage and retrieval system; automatic storage; automatic storage and retrieval system; barcode; carousel; class based storage; **control policies**; **discrete event simulation**; **genetic algorithm**; modeling; **monte carlo simulation**; multi agent system; navigation; order picking; programmable logic controller; queuing model; regression analysis; **single and dual cycle times**; storage location assignment; **system design**; **travel time model**; tsp; optimal operation; routing; travel time analysis; collection management; cylindrical automated storage retrieval system; dual shuttle; energy efficiency; integrated optimization; integrated scheduling; intelligent storage and retrieval systems; material handling systems; **miniload**; **multi criteria decision making**; multi objective optimization; multi shuttle; path optimization; **performance analysis**; queueing networks; single command; slotting

	optimization; stacker; **storage allocation**; unit load storage and retrieval; discrete event systems; dual command; indoor positioning; **markov chain**; photoelectric sensor; semi open queuing network; **sequencing**; serial communication; acceleration deceleration; **job scheduling**; **mathematical modeling**; **queuing theory**; **random storage assignment**; **simulated annealing algorithm**; single cycle; compact storage system; end of aisle; energy consumption; informatics; **order scheduling**; **storage optimization**; supply chain.
carbon footprint; case level order picking; cellular warehousing; closed form expressions; closed form travel time; competitive analysis; **condition based maintenance**; constrained optimisation problem; constructive heuristic; cycle time; **design optimisation**; double deep; **dwell point location**; dynamic change; **ecommerce**; elitist non dominated sorting genetic algorithm; expected distances; **failure mode effect analysis**; **fast genetic algorithm**; frozen food; grid warehousing; life cycle picking; modified multi command cycle; monitor and diagnostic; **multiple input output points**; **multiple regression analysis**; offline vehicle routing; orientation; palletized ground stacking; pick and pass system; pick up path; planning; polychromatic sets; preliminary; processed grain warehouse; product identification; product life cycle; quadruple command cycle; **rack dimension optimisation**; **sustainable supply chain management**; system expected travel time; task scheduling problem; the unified modeling language; tier captive system; trunk rail network; two phase tabu search; variable neighborhood search; ant colony clustering algorithm; arena simulation; bi-directional flow rack; cable driven parallel mechanism; cloud based material handling system; command sequence sorting; duration of stay policy; **energy**; **energy demand**; equipment control system(ecs); equipments failing; **fault diagnosis**; fill grade factor; force distribution; grouping matching heuristic; modified miniload automated storage retrieval system; multi objective; multi stage heuristic algorithm; multivariate methods of analysis; operation management framework; performance diagnosis; pharmaceutical logistic; value of stochastic solution; vibration control; virtual sensors; visual navigation; wire driven robot; work order; workshop distribution; answer set programming; **anti-collision principle**; as/rs fitted with gravity conveyor; atmospheric optical communication line; cross entropy method; **dashboard design; data visualization**; **double crane**; drug distribution; **energy sustainability**; ergonomics in logistics; **evolutionary algorithms**; **conflict free.**	access to collections; aggregate model; balance; bead sort algorithm; **capacitated vehicle routing problem**; cellular automaton (ca); circulation procedures; combinatorial optimization problem; common zone; computer integrated manufacturing (cim); continuous models; conveyors; data access; **dedicated storage**; demand side management; demand variation; **dijkstra**; display in real time; effective process time; expected operation time models; expert system; **fixed storage racks**; flow time prediction; immune genetic algorithm; incremental encoder; integrated multi-level conveying device; intrinsically safe; inverter; ladder diagram; **lagrangian relaxation**; library operations; light dependent resistor (ldr); loading and unloading scheduling; loadshuffling; local optimization; low power consumption; manufacturing parameters; math programming; mobile storage system; modern logistics; multi-channel; node formulation; open rack structure; optimal goods distribution; optimal result convergence; pallet loading problem; palletizing system; permutation graphs; pick up robots; picking optimization; recursive algorithm; retailing; small and medium sized enterprises; stacker crane scheduling problem; transportation; warehouse layout; acceleration deceleration; algorithms; allocation of cargo space; automated stereoscopic warehouse; batch order picking; bend aisle; bit stuffing; cellular manufacturing; class works; colored timed petri net; combine; communication module; compound management; digital pallet; digital shelf; digital warehouse management system; dispatching optimization problem; distributed problem solving; distribution center.

By looking at the keywords listed into the matrix some considerations can be made. Observing the dial of trendy keywords, it is possible to speculate which are the new most discussed topics. As shown by the keywords highlighted in bold in Table 3.1, most of the trendy keywords regard several typologies of AS/RS (see for instance vertical lift module, hybrid storage system, shuttle-based storage and retrieval system, autonomous vehicle storage and retrieval system, free fall flow rack AS/RS, etc.). Any of these warehouses is different from the others, and all of them are different from the classic AS/RS for pallet, which was the most studied before 2009. Hence, concerning the typologies of AS/RS on which the research community is focused, it seems that in recent years, scientific interest

has shifted from classic AS/RS for pallets to new configurations that stand out for efficiency or a different application context. Always focusing on the same dial, it is possible to see three keywords referring to optimisation tools such as swarm intelligence, tabu search algorithm, and particle swarm algorithm. The Tabu Search (TS) algorithm is one of the most widely used metaheuristic in several different aspects of warehousing, such as scheduling, sequencing, or location assignment (see for instance Chen et al., 2010; Chen et al., 2011; Bessenouci et al., 2012; Yang et al., 2015). Conversely, the particle swarm algorithm (also known as Particle Swarm Optimization (PSO)) is a relatively new nature-inspired metaheuristic optimisation technique recalling bird flocks' behaviour that belongs to the family of swarm intelligence algorithms. Its first contribution is by Kennedy and Eberhart (1997) and dates back to 1997, although, the PSO gained popularity in the last years for its effectiveness in contexts such as Neural Networks (NN) training and Bayesian Network learning (Aouay et al., 2013), and, nowadays, many authors are proposing a discrete version of it by rearranging the original version to solve complex combinatorial problems in the field of logistics.

Observing the dial of solid keywords it is possible to see many keywords which represent well-known and broadly analysed aspects of warehousing (see for instance control policies, single and dual command cycle times, travel time model, system design, multi-criteria decision making, performance analysis, storage allocation, queuing theory, job scheduling, etc.). These keywords, for both researchers and practitioners of the field, do not need any explanation, since they refer to well-known topics related to the AS/RS. There are also some well-known optimisation techniques, such as ant algorithm (or ant colony algorithm), and simulated annealing. The first is a metaheuristic inspired by the behaviour of ants, which belongs to the family of swarm intelligence algorithms and performs very well in problems that might be formalised with a graph (see for example Xing et al., 2010 or Brezovnik et al., 2015). Conversely, the Simulated Annealing (SA) is a metaheuristic which boasts several implementations in AS/RS as well as in many other scientific fields (Bessenouci et al, 2012). Finally, it is possible to see some tools adopted for measuring the performance of the AS/RS and essential for testing a new proposed algorithm, such as discrete event simulation and monte carlo simulation. There are also some statistical stochastic models like the Markov chain, which is essential for measuring the performance in a complex system where each event depends on the previous ones.

In the dial of neglected keywords, most of the elements are rarely discussed arguments or, at least, known topics whose name has changed and the old representative keywords abandoned. There are some rarely discussed topics, such as dedicated storage, as well as

old tools of operations research and linear integer programming (e.g. Dijkstra algorithm and Lagrangian relaxation), which have been replaced by modern dynamic programming. To the author's best view, it would be premature to say that all these arguments must be ignored, in this dial there might be forgotten topics that it would be better to study in more detail. For instance, the authors are surprised to see in this dial the capacitated vehicle routing problem, which, to the authors best knowledge and according to Ralphs et al. (2003) finds many applications in the industrial and logistic fields.

Lastly, the keywords classified as newborns highlight three main concerns: (i) the energy consumption, represented by keywords such as energy, energy demand, energy sustainability, sustainable supply chain management, (ii) the data analysis and prediction mainly aimed at improving the maintenance (e.g. dashboard design, data visualization, condition based maintenance, failure mode effect analysis, multiple regression analysis, faults analysis), and, finally, (iii) again some new warehouse configurations, as expressed by keywords like multiple input output points and rack design optimization, or anti-collision principle and conflict free, that shift the focus to autonomous vehicles AS/RS or, in general, AS/RS where some machines share the same path (or railway). It is difficult to say which of these topics classified as newborns is destined to grow and which one will slowly disappear. The energy consumption, for example, boasts many publications (e.g. Meneghetti and Monti, 2013; Wang, Tang, and Shao, 2016), although, to the author's best knowledge it makes more sense in supply chain than in warehousing, since, according to Beckschäfer et al. (2017), energy consumptions are very low. According to one of the biggest sellers of AS/RS in Europe, the machines used in some AS/RSs such as the grid-based warehouse system absorb only 0.1 kWh (it means that 6 machines consume the same energy as a toaster).

After the keywords-based analysis, the second high-level analysis is instead based on the citations. The hypothesis which justifies this analysis is the same supported by Bertolini et al. (2019): the number of citations between two groups of papers is a good indicator of how similar discussed topics are. Precisely, the greater the number of reciprocal citations the more similar the topics treated are. Thus, assuming the papers (both seed-1 and seed-0) as nodes, and the relative citations as edges, a visual representation can be provided by using a commercial software (i.e. Gephi). The result is visible in Figure 3.3, where the seed-1 papers are represented in green, and the seed-0 (i.e. the papers cited by the seed-1, but not concerning AS/RS) in pink. For the collocation of the nodes (i.e. papers) in the space, an algorithm already incorporated in the program and called ForceAtlas2 (Jacomy et al., 2014) is used. The algorithm carries out a collocation of the nodes which fits well with the need of

placing groups of nodes (i.e. papers) with multiple mutual connections (i.e. citations) next to each other.

Figure 3.3. Citations-based representation of the extracted sample of papers [2].

All the papers float around a single centre, and this is a symptom that AS/RS is still a nice topic when compared to other research themes that touch more and more aspects. In the middle, in correspondence with the centre, there is the survey on AS/RS by Roodbergen and Vis (2009), which is mentioned in most other papers, while in zones outside the agglomeration there are the newest and most recent publications, which have been placed there by the algorithm because they cite but are not cited. Simply by reading the abstracts of these papers it is possible to identify some of the newest aspects explored by the research community. Some of them are just borderline topics, such as scheduling truck loads, AGVs, or RFID, which are somehow related to AS/RS but are to be considered different. A good part of them is a well-known topic, which, as already seen in the keywords matrix, is still popular (e.g. sequencing, route optimization, simulation, performance analysis, shortest path finding). Conversely, some of them are representative of concerns already classified as

newborns by the keywords-based analysis (e.g. environment, split platforms AS/RS, cranes path sharing). What this analysis shows, and the keywords-based one is missing, is that the newest publications are citing many more seed-0. In part, this is due to the increasing number of citations in recent publications when compared to the past, however, it also shows that the literature on AS/RS is increasingly benefiting from other kinds of publications. For instance, most of the algorithms recently applied to AS/RS are strongly inspired by algorithms already proposed for different contexts.

3.3. AS/RS types

As shown above, most of the recent publications on AS/RS focus on alternative configurations and new typologies of warehouse rarely studied before. The most studied ones are AS/RS with double or multiple storage depth, known as multi-deep AS/RS, storage systems with more racks and aisles, i.e. multiple-aisles AS/RS, and AS/RS with multiple I/O points, i.e. multi-depot AS/RS. In the last case, some works analyse the impact of the position of these I/O points too, as in case of AS/RS with mid-point I/O. Other scientific contributions analyse completely different warehouses, such as:

- *Automated Vehicle Storage and Retrieval System (AVS/RS)*. An evolution of AS/RS which, through the use of autonomous vehicles, platforms, and lifts, decouple the movements of the machines for a greater scalability (Janilionis and Bazaras, 2010a; Janilionis and Bazaras, 2010b).
- *Shuttle Based AS/RS (SB-AS/RS)*. An AS/RS which handles the inventory via shuttles that run on tracks between racking structures. It can operate on single level or multiple levels, and, in the latter case, the vertical movement of the shuttles takes place thanks to lift platforms usually located at the beginning of the aisles. A recent literature review on SB-AS/RS is proposed by Kosanic et al. (2018).
- *Miniload*. An AS/RS very similar to the classic one, but specifically designed for handling lightweight loads, with the advantage to have a great reliability and a light aluminium structure requiring low power consumption (Lerher et al., 2015a).
- *Vertical Lift Module (VLM)*. An enclosed system consisting of two vertically arranged storage areas divided by an aisle, where an inserter/extractor moves to bring the unit loads back and forth from the I/O point. The unit loads are usually made of trays or long metal boxes, and the inserter/extractor automatically locates, retrieves, and delivers them to the operator at an ergonomically positioned pick window (Dukic et al., 2018).

- *3D compact AS/RS*. Stock intensive system which makes use of a S/R machine and several conveyors working in pairs responsible for the depth movements. It is very efficient in terms of space usage and is becoming popular for items with a relatively low turnover (Yu and De Koster, 2012). An exhaustive description of the system and a model for the travel-time calculation in 3D compact AS/RS are provided by Xu et al. (2018).
- *Fall-flow rack AS/RS*. Stock intensive system where the storage locations are basically conveyors. They are characterised by the alternation on aisles exclusively dedicated to storage operations and aisles dedicated only to retrieval operations. In this way, once an item (typically a pallet) has been stored, it moves along the conveyor until it reaches the other side, closed to the aisle dedicated to retrieving, where it can be retrieved and taken out. An alternative version is the free-fall-flow rack AS/RS, in which the conveyors are slightly tilted and the items move over them because of gravity.
- *Cylindrical AS/RS (C-AS/RS)*. A particular system where the storage locations are arranged all around an S/R machine forming a cylindrical structure. The S/R machine is able to rotate and vertically move, so that it can reach any storage location. See for instance Janilionis and Bazaras (2013).
- *Carousel*. A rotating carriage of unit loads where the moving parts correspond to the storage locations. It is ideal for storage of small parts, where this AS/RS saves a relevant amount of time and energy if compared to classic methods of rack and shelving type storage. See for instance Chun Park (2009).

The AS/RS typologies listed above are the most discussed in scientific literature, and the increasing popularity of rarely studied configurations and new AS/RS types has led to several changes in the machines used, as well as in the handling policies and in the racks. For these reasons, the classification of AS/RS introduced over ten years ago by Roodbergen and Vis (2009) needs an update and a revision. The classification by Roodbergen and Vis (2009) was based on three aspects: (i) the cranes, (ii) the handling policies, and (iii) the racks. For sake of clarity, the classification criteria adopted in this chapter are the same, although the term ‘crane’ is replaced with the term ‘machines’, because not only cranes are used in the newly studied AS/RS. The newly proposed classification is shown in Figure 3.4, where the aspects already mentioned by Roodbergen and Vis (2009) have been extended with additional aspects sketched in red colour.

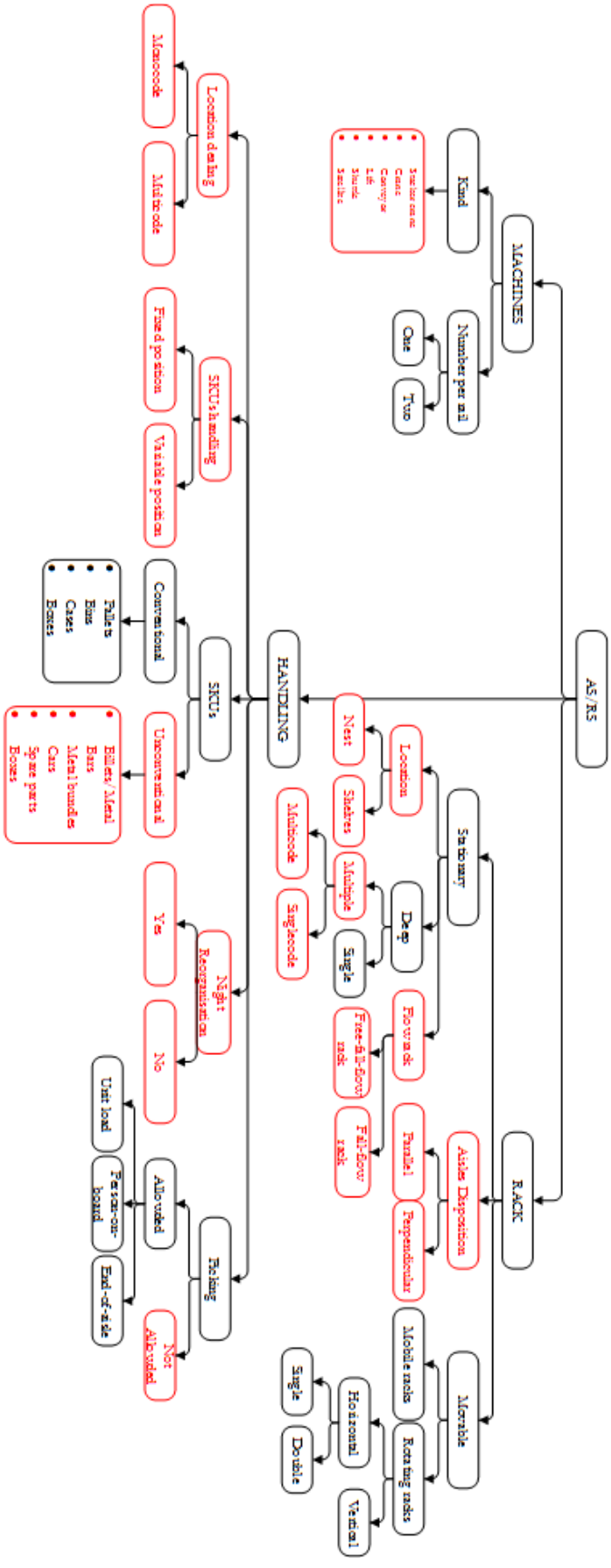


Figure 3.4. Extended classification of AS/RS [2].

3.3.1 Machines

In the typologies of AS/RS listed above, machines rarely studied before 2009 are used and must therefore be integrated in the classification proposed by Roodbergen and Vis (2009), which, on the other hand, was considering only classic cranes able to move horizontally and vertically at the same time. In many types of AS/RS, movements have been fragmented and decoupled, and this aspect allows a greater throughput and more flexibility when compared to classic AS/RS. The replacement of a single machine with many different ones and the clear separation of the movements in the three directions leads to a quicker release of each machine involved in the operations. Thus, when a machine is completing an operation, the others are free to take care of a different retrieve or storage. This leads to more flexibility, better performance (i.e. throughput), and, indirectly, makes it possible to create larger warehouses without impacting cycle times (Xia et al., 2018).

Among the machines mentioned in Figure 3.4 there are the conveyors, that often consist of a belt stretched across two or more pulleys. The belt forms a closed loop around the pulleys so it can continually rotate. One pulley, known as the drive pulley, drives or tows the belt, moving items from one location to another. They are usually combined with I/O points to work as buffers, or, alternatively, used to move and sort the retrieved products, making them available in the right picking, packing, or truck-loading position. Moreover, they might be used for many different scopes, too. For instance, in 3D compact AS/RS, the conveyors are used as a proper storage location to store the product in multiple depths.

Another machine which is very recurrent in the recently studied AS/RS is the shuttle (also known as vehicle or bot in some applications). The shuttles typically perform horizontal movements only and are equipped with an engine on board. Information and commands are transmitted to the shuttles via radiating cables (see for instance the IWLAN Rcoax Cable). Sometimes, they are moved by lift platforms between the different levels of the warehouse and this solution decreases the investment cost requiring fewer vehicles than levels (Ozaki et al., 2016). The satellite is another machine which typically does only horizontal movements. It is a sort of telescopic vehicle always incorporated in the shuttle, which releases it in correspondence of the right storage location to pick up the products in depth. More shuttles can share the same satellites thereby reducing waiting times. A good description of how shuttles and satellites can work is provided by Bruno and D'Antonio (2018) and Kaczmarek et al. (2014).

Conversely, the lifts (also indicated as elevators or lift platforms) typically perform exclusively vertical movements. They are usually located at the end-of-aisle, and they perform elevations, lowerings, or both. There are many industrial implementations of the lifts, and

each of them has its own characteristics. For instance, some of them are equipped with an engine-on-board, while, in other cases the engine is kept on the ground for safety and for simplifying the maintenance activities, and the transmission of motion is via belts or chains. Another aspect where the lifts might be different from one case to another is the object they move, which might be a product or a unit load, as well as another machine.

Finally, there are the cranes, which can be the classic S/R machines able to move in two directions at the same time, as well as different handling machines able to rotate like those used in the C-AS/RS. Even concerning this machine there are many industrial implementations and each of them has different technical characteristics. Of course, the size and the design of the crane are made according to the handled unit loads and, in case of light-weight products, as in case of a miniload, the crane structure is much lighter and more economical too.

An aspect concerning the machines, which had already been described by Roodbergen and Vis (2009) and is growing in popularity, is the utilization of more machines sharing the same path. An example is proposed by Kung et al. (2014), who consider a warehouse served by multiple cranes by considering the possible collisions and improving the work efficiency of 95% in case of two cranes, and 78% in case of three cranes. Similarly, Cai et al. (2018) improved the scheduling of operations by considering two cranes that share the same railway. A further contribution is by Beckschäfer et al. (2017), who focused on an grid-based AS/RS (e.g. AutoStore), a grid-based storage systems where products are stored in bins, stacked on top of each other, and laid out into a grid of rows and columns. Some robots are engaged to store and retrieve bins by traveling on top of the grid avoiding colliding with each other.

3.3.2 Handling

As for machines, also for the handling operations, some aspects not clearly classified by Roodbergen and Vis (2009) must be considered. During the last decade, a consistent number of papers considering multiple deep allocation (e.g. multi-deep AS/RS) have been published (see for instance Ghormi and Cardin (2018)), and a further complication has been introduced by others in which the allocation in depth of different products is considered. Other changes are due to the Stock Keeping Units (SKU) which have been labeled by the authors as unconventional (i.e. metal bars' bundles, cars, spare parts, different kinds of boxes). The evolution of AS/RS is linked with non-industrial problems too. For instance, the limited space for parking lots in cities led to studies on new automated car parking systems for big cities which are for all intents and purposes AS/RS: see for example Sumathi et al.

(2013) or Brumpton et al. (2014). Storage of different SKUs, mostly because of physical needs, elicits different handling procedures too. Indeed, for some of these unconventional SKUs such as cars or metal bundles, picking is usually not allowed because of physical constraints, and this is another handling aspect which had been neglected until 2009.

3.3.3 Racks

Concerning the racks, the first difference that must be considered is between stationary and movable ones. Movable racks can be very time and energy-consuming, and, because of this, their study has not had a relevant follow-up. One of the last publications which consider movable racks is by Hu et al. (2009). However, similar warehouses such as carousels and rotating racks are still studied (see for instance Chun Park (2009) or Wang et al.(2015a)). The 3D compact AS/RS might be considered a sort of AS/RS with movable racks, because each storage location is a movable conveyor (Yu and De Koster, 2012; Xu et al., 2018). Conversely, concerning classic stationary racks, it is important to point out the growing popularity of multiple deep location racks or multi-deep location racks. A typology of static rack. which has never been studied before 2009, is free-fall-flow rack. The free-fall-flow rack AS/RS (Metahri and Hachemi, 2018a; Metahri and Hachemi, 2018b) is an evolution of the fall-flow rack AS/RS. In both, instead of normal locations, there are long conveyors, which allow the storage of several items one behind the other. The free-fall-flow and fall-flow rack AS/RS are characterised by the alternation on aisles exclusively dedicated to storage operations and aisles dedicated to retrieval operations only. In this way, once an item (typically a pallet) has been stored, it moves along the conveyor until it reaches the other side, closed to the aisle dedicated to retrieving, where it can be retrieved and taken out. The only difference between fall-flow and free-fall-flow is that in the second ones, the conveyors are slightly tilted, this way, the items move over them because of gravity. Free-fall-flow rack AS/RS ensure consistent money-saving but their implementation is limited to a restricted number of products. A comparison of performance between fall-flow and free-fall-flow rack AS/RS is provided by Metahri and Hachemi (2017).

Finally, in Figure 3.4 are mentioned the kind of storage location (i.e. nest or shelves) and the disposition of aisles (i.e. parallel to the rack or perpendicular to the rack), because there are some examples of AS/RS where the storage locations are made of metal shelves and the aisles are perpendicular to the rack length (Bertolini et al. (2019)).

A further aspect which is not directly related to the racks, but is relevant for the rack configuration is the position of the I/O points. Before 2009, most of the authors had usually considered AS/RSs where I/O points were close to each other and located at the end of the

aisle, although their position and their dimensioning have a big impact on performance and have not been completely analyzed (Ramtin and Pazour, 2014). In industrial environments, positioning of I/O points is often influenced by the layout of system interchanging with the warehouse and, sometimes, it is chosen for improving the performance of the warehouse itself. In recent years, new configurations of AS/RS have been studied, and, as a matter of fact, some of them focused on unconventional ways to position the I/O points. Some authors studied configurations where there are only one input and one output point, which are at opposite ends of the aisle (Yu and Yu, 2019; Tanaka and Araki, 2009). Others considered middle cross-aisle I/O points (Hu et al., 2009) or multiple I/O points (Song et al., 2016; Lantschner et al., 2013). Unfortunately, the scientific community has neglected the problem of dimensioning the I/O points, which is important to properly integrate the AS/RS with the rest of the plant (e.g. the truck loading area, the yard, or the production lines).

3.4. Overview of design decisions

The two main aspects concerning the design of an AS/RS are the (i) physical design and the (ii) control policies. Both have a great impact on the system in terms of performance, energy consumption, investment cost, maintenance, and utilization cost. However, the realisation of an AS/RS is often an Engineering-To-Order (ETO) or Design-To-Order (DTO) project. For this reason, it is important to highlight a further phase that characterises every project: the proposal. The proposal phase is the part of a project between the customer's request and the beginning of the design. It is essential because it is characterized by strong contact with the customers, who usually know their own needs without having a clear idea of which AS/RS is most suitable and how much the new investment will impact their business. During the proposal phase, many configurations of the system must be evaluated in order to measure and forecast the key indexes of interest to the customers, then the best configuration becomes the object of the design phase. In Figure 3.5, the figure concerning the design decisions presented by Roodbergen and Vis (2009) has been extended including the proposal phase. The choice of the configuration impacts on investment costs, lead time (i.e. the time between the customer order and the commissioning of the warehouse), operative costs, energy consumption, and performance, which may be expressed in terms of cycle time or operations per time unit (i.e. throughput). Concerning the investment costs and the lead time, it is reasonable to believe that their estimation might be possible from the beginning, or, at least, they might be defined in agreement with the customer. Conversely, the operational parameters such as the performance, the operating costs, and the energy consumption are difficult to know. A preliminary evaluation of performance in the case of

AS/RS for pallets can be obtained with the F.E.M. 9.851. standards, but if dealing with other kinds of warehouses, the performance calculation of a system that has not yet been designed requires a different tool. Here are several mathematical models discussed and validated by the scientific community that come to the aid at this stage of the proposal. Another tool frequently used for its simplicity is the simulation, although, in the proposal phase, several data are missing and cannot be collected from the field. For this reason, it is important to point out two different typologies of simulation: (i) kinematic and (ii) dynamic. The first one is usually adopted in the proposal phase and is made by managing just a single code, and adopting for both input and output operations a random logic with equal probability of access to the compartment, i.e. each compartment has the same probability of being chosen. The throughput is usually obtained by reducing as much as possible the time between the arrival of a request and the next (i.e. inter-arrival time) and observing how many operations per unit of time the AS/RS can carry out. To achieve this kind of simulation, in general, it is enough to have a layout of the warehouse and to know the basic functioning of the machines and their kinematic parameters (e.g. speed, acceleration, and deceleration). However, even in the proposal phase, if the customer is willing to share his market demand's history and his production data, it is possible to anticipate the decisions relative to the control of the warehouse and to calculate the performances through more accurate simulation. Indeed, the second approach replicates the real functioning of the warehouse considering the calendar, the working hours, the Overall Equipment Effectiveness (OEE) and most real environmental conditions.

Figure 3.5. Design and proposal of an AS/RS system [2].

Table 3.2 contains the list of configuration aspects and decisions that should be considered in the design of an AS/RS. The table is an extension of that one already proposed by Roodbergen and Vis (2009), and, starting from it, the necessary changes included have been included and sketched in italics underlined. The introduction of new machines, handling issues, and rack typologies involves additional decisions to be evaluated during the proposal. For the purposes of a good proposal all possible decisions should be evaluated and their impact should be estimated. To conclude this section, it is important to say that the proposal is an important phase which involves any big project, and there is unanimous agreement that, in this phase, multi-criteria decision making methods might be very helpful approaches (Braglia et al., 2019), thus, more approaches like those proposed by Tosun and Aktan (2016), Alac (2019) or Erkan et al. (2014) should be investigated.

Table 3.2. Updated overview of design decisions.

CLASS OF DECISION	DECISIONS TO BE MADE
System configuration	Number of aisles Height of the storage racks Equally sized or modular storage locations Number and location of the I/O points Buffer capacity at the I/O points

	Number of cranes per aisle Number of order pickers per aisle *Typology of racks (i.e. stationary, movable, rotating)* *Number of racks* *Disposition of aisles (i.e. parallel or perpendicular)* *Kind of locations (i.e. nests or shelves)* *Machines used (e.g. shuttles, lifts, cranes, etc.)*
Storage assignment	Storage assignment method Number of storage classes The positioning of the storage classes *SKUs handling (i.e. fixed or variable)* *Night-time reorganization of stock or not*
Batching	Type of batching (static or dynamic) Batch size (capacity or time based) Selection rule for the assignment of orders to batches
Sequencing	Sequencing restrictions (e.g. due dates) Type of operation (single or dual command) Scheduling approach (block or dynamic) Scheduling method
Dwell point	Type of positioning (static or dynamic) The location where idle cranes will be placed
Locations dealing	*Type of locations (i.e. nests or shelves)* *Number of codes per location (i.e. monocode or multicode)* *Deep (i.e. single or multiple)*
Unit loads configuration	*Conventional or unconventional* *Type of unit loads*

3.5. Physical design

The design of an AS/RS is therefore made of two phases: the physical design and the control policies design (Dong et al., 2018). The first includes long-term decisions, since changing the physical structure is difficult and expensive. The second includes short-term decisions which concern the software components and can be subsequently or periodically modified without incurring excessive investments. In this section the long-term decisions and the physical design are considered.

The physical design consists of two main steps: the system choice and the system configuration. The system choice is the selection of the most suitable AS/RS type. Conversely, the system configuration consists in the definition of layout, dimensions, positioning, and number of physical components (e.g. racks, locations, cranes, shuttles, elevators, aisles, and I/O points). The physical design has a strong impact on performance, costs, and storage capacity. Typically, the problem to be addressed, as shown in Figure 3.6, concerns the definition of a layout that optimizes performances and that respects certain budget and capacity constraints. Since the main objective is often to maximise the throughput, physical and control policies design are often jointly handled. The processes

adopted to properly select the design of an AS/RS are represented in Figure 3.7: (i) analysing different control policies under the same design conditions, or (ii) comparing different design solutions under the same control policy.

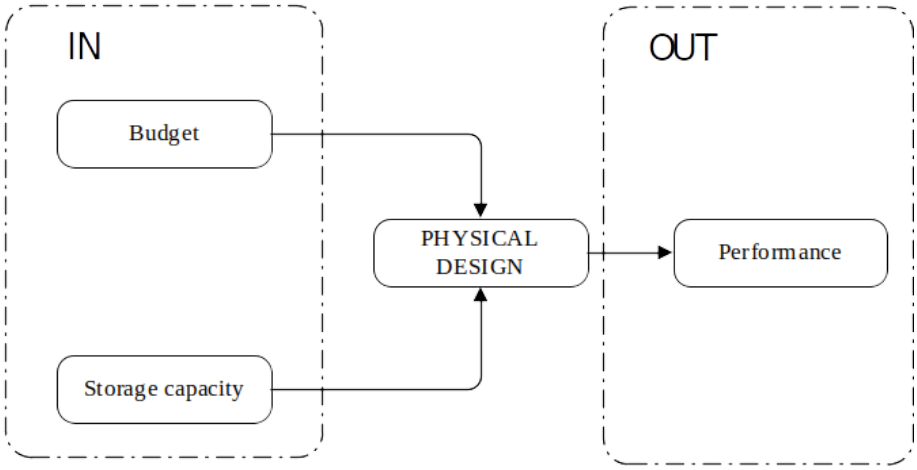


Figure 3.6. Usual decision process for the physical design [2].

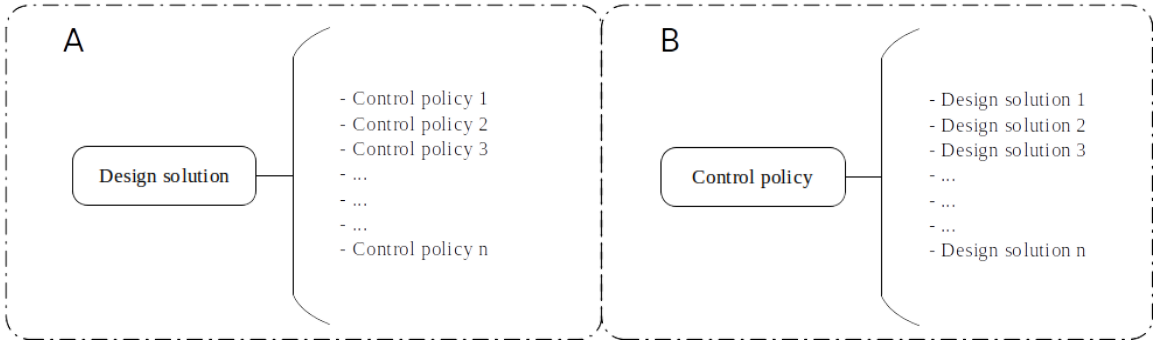


Figure 3.7. Approaches that are usually used to make design decisions [2].

Several solutions to improve the design are presented in scientific literature and most of them are based on a simulative approach. For instance, Gagliardi, Renaud and Ruiz (2012) study multi-aisle AS/RS considering complicated aspects such as double storage depth and non-captive aisle cranes, i.e. cranes able to move from one corridor to another. Their research compares different combinations of control policies and physical design configurations, considering cranes' freedom, I/O points position, and number, length, height, depth, and spacing of racks. Yang et al. (2015) analyse the impact of dimensions, factor shape, and S/R speed, in the design of a 3D compact AS/RS In this way, they extend the study by De Koster, Le Duc and Yugang (2008), considering acceleration and deceleration of S/R machine. Conversely, a purely mechanical physical design study aimed at improving

the dynamics of the crane is proposed by Zheng et al. (2012). Finally, Bahrami, Aghezzaf and Limere (2014) analyse different design configurations and their impact on performance.
Another widely used approach is the analytical one, adopted for instance by Ramtin and Pazour (2014) to define the positioning of the I/O points, the number of racks, and the number of aisles to improve the performance. The analytical approach is used for SB-AS/RS by Borovinsek et al. (2017), who might be considered as pioneers in the utilisation of a multi-objective optimisation, which takes into account several aspects (i.e. cycle time, throughput, energy consumption, investment cost). The study is therefore extended considering a classic AS/RS in Rajkovic et al. (2019).
Further works focus on less common types of warehouses such as Vertical Lift Modules (Bertolini et al., 2018), Carousels (Janilionis and Bazaras, 2012), or 3D compact AS/RS (Hao et al., 2015; Yu and De Koster, 2009).

3.6. Storage assignment

Storage assignment essentially consists: (i) in the assignment of empty storage locations to the incoming unit loads and (ii) in the selection of unit loads, available in stock, to match retrieval requests. Implementing a well-designed storage assignment policy is fundamental, as it allows exploiting the available stock capacity and it reduces distances covered by the handling machines, with immediate benefits in terms of throughput time and energy consumption. According to Roodbergen and Vis (2009), the most common storage assignment policies are:

- Dedicated: each item is stored always in the same location;
- Random: storage locations are randomly assigned and, typically, each one of them has the same probability to be selected;
- Closest location: each time a new input or output request is received, the closest storage location (with enough room to store the entering unit load or with adequate materials to satisfy the customer request) is selected;
- Full-turnover-based: each product is stored in a dedicated area of the warehouse and each area is dimensioned to accommodate the maximum inventory level of the stock keeping unit it must hold;
- Class-based: the products are grouped into specific classes, depending on the frequency they are required with. Each class is assigned to a specific area of the warehouse, with classes corresponding to fast-moving items usually located as close as possible to the exit points.

Recent publications show that class-based policies ensure the highest benefits, especially when the number of classes is close to three (Yuan, Graves and Cezik, 2019). However, in case of mixed loading, a random storage-based policy can outperform both the dedicated and the class-based policy, as shown by Ishigaki and Hibino (2014).

All the above-mentioned policies are still widely used, although, as AS/RS configurations spread, alternative approaches have emerged. Some of them are completely new, while others are just a readjusted version of standard policies. For instance, Moon et al. (2009) considered the insertion of a common area between two class zones, to limit the 'rack shortage problem' occurring when the demand pattern changes. Meneghetti and Monti (2014) proposed an improved class-based strategy that can be effectively used in case of multiple-weight stock-keeping units. Different authors (e.g. Bortolini et al., 2017) focused on storage assignment policies aimed at reducing energy consumption. Others have integrated class-based policies with meta-heuristic algorithms, such as Atmaca and Ozturk (2013), who propose a simulated annealing. A similar approach is proposed by Pan et al. (2015) and Wang et al. (2018), who instead use a genetic algorithm.

It is important to note that, being AS/RS a complex system, it is often impossible to achieve significant improvements unless multiple aspects are conjointly considered (Chen et al., 2010). This is the reason why most of the above-mentioned solutions tackle not only storage assignment, but interleaving, sequencing, or scheduling too.

To the author's best knowledge, most of the proposed solutions have a crucial limit, as they lead to a static behaviour of the system, independent of changes in the context in which it operates. The storage assignment policies should be dynamic and the system should be able to define the correct policy depending on the current boundary conditions. For instance, the system could take advantage of the periods with fewer requests to be met, to reorganise the stock in view of busier moments. Another crucial aspect is the staticity of the classes in case of class-based assignment policy. A worth mentioning solution which takes in consideration this criticality is the one proposed by Chou et al. (2012), who developed what they defined a 'recency-based' storage allocation policy, which makes it possible to dynamically and automatically modify the classes in which stock keepings units are grouped, based on the analysis of customers' demand. Further neglected considerations which could be studied concern the storage assignment based on the equal distribution of the workload in case of more resources (e.g. cranes, shuttles, etc.). Moreover, the storage assignment could also consider the correlation in customers demand between products, for instance taking into account that two products usually required together might be stored one beside the other, or in a specific position which allows the cycle time reduction. This might be useful

in case of dual capacity handling AS/RS, i.e. an AS/RS whose crane can handle two unit loads at the same time (Dorr, 2016; Dorr and Furmans, 2016; Dorr, 2018).

3.7. Batching

Through the process of batching, the storage and retrieving operations are collected in groups (i.e. batches), and then processed together according to the predefined control policy. Batching is not an obligation, although, in some circumstances, a good batching strategy can reduce the cycle time and improve the performance. Observing the relationship between customer orders and retrieval operations, as described by Chirici and Wang (2014), three distinct cases can be identified: (i) the customer order requires a too big quantity to be satisfied with a single retrieval operation, and, to fulfill it, several retrieves must be carried out; (ii) the customer order might be fulfilled with a single retrieval operation (in this case no batching strategy is needed); or (iii) several customer orders are agglomerated and satisfied with a single retrieving. The possibility of batching depends on the type of AS/RS, the customer orders, the allocation of stock and the possibility of sorting and picking after retrieving.

A recent survey concerning batching strategies has been published by Boysen, De Koster and Weidinger (2018), who focused on an e-commerce context characterized by a big number of small orders, where the aggregation of multiple orders in a single retrieval is essential to meet delivery times. A batching problem, due to its complexity and the many variables involved, is always approached as an operational search problem. The composition of a batch can be associated with many operational search problems such as the Knapsack Problem (KP), the Cutting Stock Problem (CSP), the Travelling Salesman Problem (TSP). Because of this, the solution to this problem defined NP-hard is often a meta-heuristic algorithm. Wang et al. (2015a) propose a Genetic Algorithm (GA) to seek the optimal batching strategy in two different part-to-picker systems: an AS/RS and a Carousel. Lenoble, Frein and Hammami (2016) propose an algorithm to obtain optimal batching strategy in a Vertical Lift Module (VLM) and then extend the study to a storage system consisting of several VLMs in Lenoble, Frein and Hammami (2018). Lenoble, Frein and Hammami (2017) study the optimal batching in storage composed by more Carousels, formulating the problem as a Mixed-Integer Linear Program (MILP) and solving it to the optimum. Chen and Li (2016) study how to improve the performance in an AS/RS with bi-directional flow racks and a dual-command operation policy, also considering the issue of

batching. Finally, Liu, Sun and Wang (2014) provide a GA to find optimal batching in a generic tiered warehouse.

3.8. Dwell point location

The dwell point is the position where the machines wait for the next request when they have no operations to process and its positioning is generally aimed to reduce the cycle time or the energy consumption. The impact of the dwell-point on the cycle time of an AS/RS is simulatively demonstrated by Regattieri et al. (2013), who made more than 1000 simulations in several different scenarios. The choice of the dwell point is strongly influenced by two aspects studied in depth by Sari and Hamzaoui (2013): the location where the I/O points are and the warehouse shape factor.

An aspect which is evident but not mentioned in literature is that the importance of the dwell point decreases as the workload increases: if a resource (i.e. machine) is always busy or does not have time to reach its resting position, this policy becomes superfluous.

The choice of dwell point is of two types: static or dynamic. Static means that the dwell point of each machine (e.g. crane, shuttle, lift, etc.) is always the same, while dynamic means that it is calculated every time the machine needs to rest. In particular, in the latter case, each time the machine is going to rest, the resting point is defined considering aspects such as the probabilistic arrival point of the next operation, the class-based distribution of products, etc.

Liu et al. (2016) study the dwell point for SB-AS/RSs in the case of dual command cycle, and Janilionis and Bazaras (2010b), remaining within the SB-AS/RSs range, report an initial analysis of control policies and dwell point. Feng, Chen and Ding (2012) study the effect of the dwell point when a single crane must run in more aisles. Other researchers, such as Park (2009), focus instead on the positioning of the dwell point in the case of vertical and horizontal carousels. Hale et al. (2015) first analyse the problem for the classic AS/RSs for pallets and then developed an extension of the paper dedicated to carousels (Huq, Hale, Lutz, and Pujari, 2018) considering many allocation policies.

3.9. Sequencing of storage and retrieval requests

According to Roodbergen and Vis (2009), sequencing of storage requests is not time-critical and its impact on the overall performance of the warehouse can usually be neglected. Conversely, the sequencing of retrievals leads great improvements in the overall throughput of the AS/RS. The list of retrievals is continuously changing over time, as performed retrievals are deleted from the list and new retrieval requests arrive; for this reason, two

ways of dealing with this dynamic problem are usually adopted: (i) the first is called *block sequencing* and consists in selecting a block of most urgent requests, sequencing them, and then, when they are completed, selecting the next ones, and so on. Alternatively, (ii) a procedure called *dynamic sequencing* can be adopted. Dynamic sequencing re-sequences the whole request list after a predefined interval of time or every time a new request arrives. The dynamic sequencing allows the optimum to be found, while the block sequencing refuses the optimum in favor of a shorter computational time.

Parameters observed in the literature to evaluate a sequencing procedure are mainly the travel-time, the travel distance, or the tardiness (i.e. interval of time between request arrival and its processing).

The most used sequencing policy is the First-In First-Out (FIFO) method. We can find it in most AS/RSs' industrial applications and simulations adopted in the research. However, Yu and De Koster (2012) prove that the Shortest Leg (SL) policy can outperform FIFO by 20–70%. Even Popovic, Vidovic and Bjelic (2014), after studying an AS/RS whose crane can handle three unit loads at the same time and comparing three greedy heuristics such as Nearest Neighbour (NN), Reverse Nearest Neighbour (RNN) and SL, state that SL outperforms others. In the end, they also propose a genetic algorithm able to guarantee even better results. Gagliardi, Renaud and Ruiz (2014) explain that better performance is obtained by adopting a single procedure for jointly optimizing both location assignment and sequencing, instead of improving them in two different steps, and they quantify this improvement at 25%. In fact, many authors studied sequencing and assignment jointly. Hachemi, Sari and Ghouali (2012) were the pioneers proposing an algorithm to speed up the cycle time in a warehouse where the same code is stored in several not predetermined locations. A similar solution is proposed by Hachemi and Besombes (2013), who extend the problem of allocation assignment and retrieval sequencing by integrating the product expiry date. Location assignment and requests sequencing are also mutually considered by Yang et al. (2017), who propose two Tabu Search algorithms and validate them in many different configurations of multi-shuttle AS/RSs. Chen et al. (2010) jointly consider storage allocation and interleaving in a single-aisle classic AS/RS. They approach the problem in three different ways: with integer programming, with a two-step heuristic, and then with a tabu search. The results are gratifying, although, even if the paper is one of the most cited, a real implementation would be difficult, mainly because of two aspects: (i) they suppose to know the exact deterministic instant of time in which an item will leave the location even before stocking it, and (ii) they supposed to handle single unique items. Wang et al. (2015b) focus on a multi-tier shuttle-based AS/RS and improve the performance, sometimes reaching the

global optimum, with the proposed genetic algorithm. Carlo and Vis (2012) study warehouses where two different machines share the same path and sequence requests to minimize the number of obstructions.

Finally, Gagliardi, Renaud and Ruiz (2015) demonstrate that a multi-aisle system cannot be accurately represented by multiple single-aisle systems. They also show that given a multi-aisle AS/RS, improvement of sequencing made considering the whole system can outperform the improvement made by independent optimization of every single aisle up to 48%. Another innovative solution is proposed by Foumani et al. (2018), who define sequencing rules to minimise the travel time of an AS/RS with a cartesian robot. They also define the movement sequence to detect and reduce collisions.

3.10. Performance measurement

A proper estimation of the performance of an AS/RS provides good support during the layout selection, improves the organization of the shipments, and supports the design of the elements interfacing with the warehouse (e.g. the production lines and the shipment area). The two main alternatives to estimate the performance in AS/RS are essentially (i) the discrete event simulation and (ii) the analytical models. The simulation is easier and might be very cheap thanks to the low cost of computers and the existence of free and open-source tools (see for instance the Python 3 simulation library called Simpy©). However, it requires much data from the field, and, sometimes, a good knowledge in computer science too. Moreover, if the choice of the company had to fall on commercial software (usually because of a marketing choice), this would lead to considerable investment costs in the purchase of the software and the training of the personnel. Or all these reasons, small and medium enterprises usually opt for analytical models, which, once designed, could be easily implemented for example in a spreadsheet, with no need of hard-to-learn tools and with no need of a huge amount of data from the field (Garcia de Jalon and Bayo, 1994).

Performance is usually expressed in terms of operations per hour (i.e. throughput), or average cycle time (usually seconds per cycle), but both alternatives are correct. The estimation can consider several aspects such as faults, waiting times, minor causes of machines unavailability, and handling operations, although, most of the works in scientific literature focus only on the travel-time, whose impact is majoritarian, at least in well-designed AS/RS where the machines spend the most of the time moving from a position to the other. Handling operations have a minor impact and are generally approximated with a fixed time that can be easily introduced in the model in a second step.

Performance estimation is probably the most studied aspect in warehousing, and many publications are proposed every year. A comprehensive review of scientific papers published before 2009 is proposed by Cai et al. (2009), while for newest publications a good reference can be the article by Kosanic et al. (2018). As for other aspects, the research on performance estimation is moving from classic AS/RS to alternative and more complicated configurations, such as multi-deep AS/RS (Lehrer, 2016), or multi-aisles AS/RS (Shi and Li, 2017). Even different warehouses are considered; for instance Eder (2019) uses a time-continous queuing model combined with a discrete spatial approach to reproduce the functioning of lifts and shuttles in SB-AS/RS, and properly account for possible interactions among them without using the most used Markov chain. Another contribution based on SB-AS/RS is by Borovinsek et al. (2017), who propose a design optimisation tool considering seven variable (i.e. number of aisles, number of tiers, number of columns, speed of shuttles, acceleration/deceleration of shuttles, speed of lifts, acceleration/deceleration of lifts) and three objective functions (i.e. throughput, energy consumed, investment cost). In Table 3.3 a comprehensive and schematic classification of recent publication on performance estimation in AS/RS is shown. For each article, the characteristics of the warehouse are specified such as the type of AS/RS, the layout, the racks, the number of I/O points, the management of stock, the cycles considered and, finally, the operational characteristics of the machines. Even in this case, the purpose of the table is to update and extend the table already published by Roodbergen and Vis (2009).

All the papers listed in Table 3.3 propose an analytical model for performance measurement and then validate it by simulation. Observing the table, the research has focused on certain aspects rather than others. As far as racks are concerned, few consider the double depth or the possibility that the rack is interrupted, as often happens, by load-bearing columns or structural components. Moreover, no one considers the possibility of having storage locations of different sizes.As far as allocation policies are concerned, the most studied is undoubtedly a random allocation and few consider class-based or dedicated allocations; at most, some assign a different probability of being chosen to different areas of the warehouse. As far as the operational characteristics are concerned, most authors adopt a transitory, considering that each movement is characterized by three main phases: an acceleration, an eventual stretch at constant speed, and a deceleration. The way how this operational characteristic is integrated in the analytical model changes from article to article, but the calculation of the travel-time is almost the same and might be summarised as follows.

Table 3.3. Classification of articles on performance estimation.

Article	AS/RS type	Layout				Rack				I/O points		Storage							Operational characteristics	
		A	B	C	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
(Lerher et al. 2015c)	Miniload		X		X	X		X			X	X						X	X	X
(Lerher et al., 2010)	AS/RS		X		X		X		X		X	X					X			X
(Basile et al., 2012b)	AS/RS and carousels		X		X		X	X			X	X		X		X				X
(Basile et al., 2012a)	AS/RS and carousels		X		X		X	X			X	X		X		X				X
(Lerher, 2016)	SB-AS/RS		X		X		X		X	X		X				X	X			X
(Heragu et al, 2011)	AS/RS and AVS/RS		X		X	X		X			X	X				X				X
(Lerher, et al., 2015b)	SB-AS/RS	X			X		X	X		X		X					X			X
(Roy et al., 2012)	AVS/RS		X	X		X		X		X					X	X			X	
(Lerher et al., 2015a)	SB-AS/RS	X			X		X	X		X		X				X	X			X
(Xu et al., 2016)	AS/RS		X		X	X		X		X		X				X	X		X	
(Khojasteh and Son, 2016)	AS/RS		X		X	X		X			X	X				X	X		X	
(Liu et al., 2016)	Split-platform AS/RS	X			X	X		X			X	X					X		X	
(Lerher et al., 2017)	SB-AS/RS	X			X		X	X		X		X				X	X			X
(D'Antonio et al., 2018)	AVS/RS	X	X			X		X		X		X						X		X
(Xu et al.,, 2018)	3D compact AS/RS				X	X		X		X		X				X	X			X
(Rosi et al., 2016)	Miniload		X		X	X		X			X		X			X	X		X	
(Xu et al., 2015)	Dual-shuttle AS/RS		X		X	X			X	X		X						X	X	
(Epp et al., 2017)	AVS/RS	X			X	X		X		X		X				X			X	

Legenda:									
Single Aisle	A	Continuous rack	F	Single I/O	J	Full-turnover storage	N	More than dual	R
Multiple Aisle	B	Discrete rack	G	Multiple I/O	K	n-class based storage	O	Time = space/speed	S
Square-in-time	C	Single deep storage	H	Random storage	L	Single command	P	Transitory	T
Non-square-in-time	E	Multiple deep storage	I	Dedicated storage	M	Dual command	Q		

To get a precise estimation of the travel-time, we will consider also the acceleration and deceleration of each machine involved according to the velocity profile of Figure 3.8, valid for any direction k. The first cycle represents the standard situation: the machine speeds up from zero to the maximum velocity v_k, with constant acceleration a_k. Next, it proceeds at constant speed, until it starts to decelerate with constant deceleration d_k to stop, exactly, at the destination point.

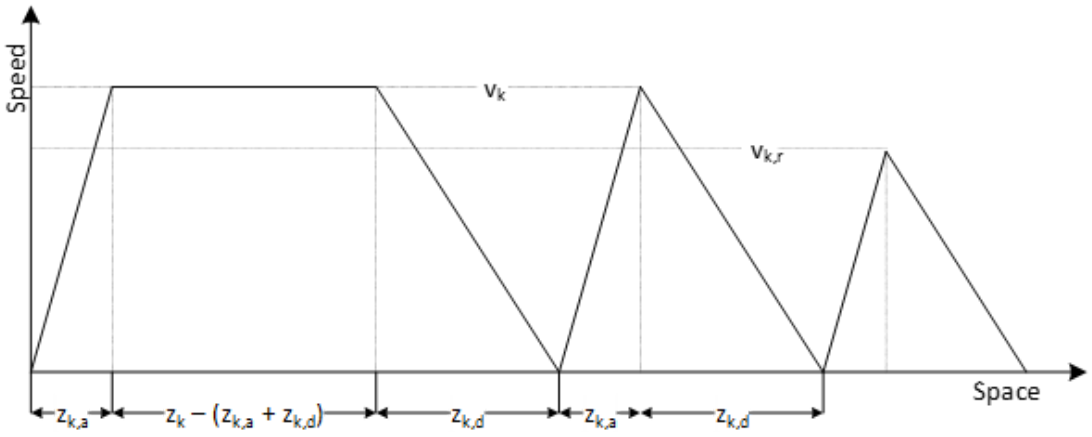


Figure 3.8. The velocity profile of the machines [1].

To reach v_k, the distance z_k to the destination point must be greater than the sum of the distance $z_{k,a}$ and $z_{k,d}$, needed to accelerate from zero to v_k, and to decelerate from v_k to zero, respectively. If not so, there is no movement at constant speed, as deceleration immediately follows the acceleration phase. This extreme condition is shown by the second and third cycle of Figure 3.8. Specifically, in the second cycle $z_k = z_{k,a} + z_{k,d}$ and the maximum speed $v_{k,r}$ reached by the machine equals exactly v_k. Conversely, in the third cycle $z_k < z_{k,a} + z_{k,d}$ and $v_{k,r}$ is lower than v_k.

Accordingly, the time $\tau_k(z_k)$, needed to travel a distance z_k along direction k, can be computed as in eq. (3.2):

$$\tau_k(z_k) = \begin{cases} \frac{v_{k,r}}{2a_k} + \frac{v_{k,r}}{2d_k}, & z_k \leq z_{k,a} + z_{k,d} \\ \frac{v_k}{a_k} + \frac{v_k}{d_k} + \frac{z_k - z_{k,a} - z_{k,d}}{v_k}, & z_k > z_{k,a} + z_{k,d} \end{cases} \tag{3.2}$$

Where:

- $z_{k,a} = \frac{v_k^2}{2a_k}$ is the distance needed to accelerate from zero to v_k;

- $z_{k,d} = \frac{v_k^2}{2d_k}$ is the distance needed to decelerate from v_k to zero;

- $z_{k,r} = \sqrt{\frac{2z_k a_k d_k}{a_k + d_k}}$ is the maximum speed, when the distance z_k is not enough to reach v_k.

3.11. Energy

Studies concerning energy consumption mainly focus on the system design, the machines and engines design, the consumptions because of movements and accelerations, and the energy consumption for keeping the system temperature into certain limits (for instance when stock consists of food). The energy consumption is generally expressed as total emissions of CO2 equivalent and its value depends on many aspects. Wang, Tang and Shao (2016) observed how a different and innovative crane reduces energy consumption. Liu, Wang and Sun (2013) studied the energetic impact of bearing and rail corrosion. Meneghetti and Monti (2013) studied how it is possible to recycle the energy dissipated by the machines in the form of heat. An interesting trade-off has been done by Hahn-Woernle and Gunthner (2018), who studied the link between the performance's reduction due to the slowdown of the machines (in speed and acceleration) and the corresponding reduction in energy consumption. The result is a logarithmic curve where an initial small reduction of machines' speed and acceleration corresponds to a great energy saving. These and other articles concerning energy consumption are reported in Table 3.4 where the type of warehouse studied, the problems considered, and the approach used are reported.

Table 3.4. Articles on energy consumption in AS/RS [2].

PAPER	WAREHOUSE	ISSUE	APPROACH
(Ekren et al., 2018)	SB-AS/RS	Design	Analytical
(Hahn-Woernle and Günthner, 2018)	Miniload	Energy consumption in movement	Analytical
(Eder and Kartnig, 2018)	SB-AS/RS	Energy consumption in movement	Analytical
(Austermann et al., 2016)	AS/RS and Conveyors	Machines design	Analytical
(Roshan et al., 2018)	AS/RS	Design and energy consumption in movement	Analytical and algorithms
(Soyaslan et al., 2017)	AS/RS	Energy consumption in movement	Simulation, analytical and algorithms
(Meneghetti et al.,	AS/RS	Design and energy	Algorithms

2015)		consumption in movements	
(Wang et al., 2016)	AS/RS	Design of machines	Analytical
(Windmann et al., 2015)	Conveyors	Energy consumption in movement and energy consumption for keeping system temperature in limits	Simulation
(Meneghetti and Monti, 2013)	AS/RS	Energy consumption in movement	Analytical
(Liu et al., 2013)	SB-AS/RS	Bearing corrosion	Algorithms
(Lerher, Edl, and Rosi, 2014)	Miniload	Design	Simulation

References

Aláč, P. (2019) ‘Multi Criteria Decision Making Model for Logistics Processes in Particular Enterprise’, System Safety: Human-Technical Facility-Environment, Vol. 1 No. 1, pp. 522-531.

Aouay, S., Jamoussi, S., and Ayed, Y. B. (2013) ‘Particle swarm optimization based method for Bayesian Network structure learning’, In 5th International Conference on Modeling, Simulation and Applied Optimization (pp. 1-6).

Atmaca, E., and A. Ozturk (2013) ‘Defining order picking policy: A storage assignment model and a simulated annealing solution in AS / RS systems’, Applied Mathematical Modelling, Vol. 37 No. 7, pp 5069–5079.

Austermann, J., Borcherding, H., Stichweh, H., and V. Grabs (2016) ‘High efficient modular drive system - An ideal approach for green intralogistics applications’, In 18th European Conference on Power Electronics and Applications, EPE 2016 ECCE Europe.

Bahrami, B., Aghezzaf, El-H., and V. Limere (2014) ‘Simulation based performance analysis of an end-of-aisle automated storage and retrieval system’, In 3rd International Conference on Operations Research and Enterprise Systems (pp. 334–341).

Basile, F., Chiacchio, P., and J. Coppola (2012a) ‘A Hybrid Model of Complex Automated Warehouse Systems — Part I: Modelling and Simulation’, IEEE Transactions on Automation Science and Engineering, Vol. 9 No. 4, pp. 640–653.

Basile, F., Chiacchio, P., and J. Coppola (2012b) ‘A Hybrid Model of Complex Automated Warehouse Systems — Part II: Analysis and Experimental Results’ IEEE Transactions on Automation Science and Engineering, Vol. 9 No. 4, pp. 654–668.

Bayliss, C., Juan, A. A., Currie, C. S., & Panadero, J. (2020) ‘A learnheuristic approach for the team orienteering problem with aerial drone motion constraints’, Applied Soft Computing, Vol. 92 No. 106280.

Beckschäfer, M., Malberg, S., Tierney, K., & Weskamp, C. (2017) ‘Simulating storage policies for an automated grid-based warehouse system’ In International Conference on Computational Logistics (pp. 468-482).

Bertolini, M., Neroni, M., and G. Romagnoli (2018) ‘A new heuristic algorithm to improve the design of a vertical storage system’, In 23rd Summer School ‘‘Francesco Turco’’ - Industrial Systems Engineering (pp. 22–27).

Bertolini, M., Esposito, G., Mezzogori, D., and Neroni, M. (2019) ‘Optimizing Retrieving Performance of an Automated Warehouse for Unconventional Stock Keeping Units’, Procedia Manufacturing, Vol. 39, pp. 1681-1690.

Bertolini, M., Esposito, G., Neroni, M., Rizzi, A., and Romagnoli, G. (2019) ‘A meta-analysis of industry 4.0-related technologies that are suitable for lean manufacturing’, In 24th Summer School Francesco Turco Vol. 1, pp. 150-156.

Bessenouci, H. N., Sari, Z., & Ghomri, L. (2012) ‘Metaheuristic based control of a flow rack automated storage retrieval system’, Journal of Intelligent Manufacturing, Vol. 23 No. 4, pp. 1157-1166.

Bigliardi, B., and F. Galati (2018) ‘Family firms and collaborative innovation: Present debates and future research’, European Journal of Innovation Management, Vol. 21 No. 2, pp. 334-358.

Borovinšek, M., Ekren, B.Y., Burinskiene, A., Lerher, T. (2017) ‘Multi-objective optimisation model of shuttle-based storage and retrieval system’, Transport, Vol. 32 No. 2, pp. 120-137.

Bortolini, M., Faccio, M., Ferrari, E., Gamberi, M., and F. Pilati (2017) ‘Time and energy optimal unit-load assignment for automatic S / R warehouses’, International Journal of Production Economics, Vol. 190, pp. 133–145.

Boysen, N., and K. Stephan (2016) ‘A survey on single crane scheduling in automated storage/retrieval systems’, European Journal of Operational Research, Vol. 254 No. 3, pp. 691-704.

Boysen, N., De Koster, R., and F. Weidinger (2018) ‘Warehousing in the e-commerce era: A survey’, European Journal of Operational Research, Vol. 277 No. 2, pp. 396-411.

Braglia, M., Frosolini, M., Gallo, M., & Marrazzini, L. (2019) ‘Lean manufacturing tool in engineer-to-order environment: Project cost deployment’, International Journal of Production Research, Vol. 57 No. 6, pp. 1825-1839.

Brezovnik, S., Gotlih, J., Balič, J., Gotlih, K., & Brezočnik, M. (2015) ‘Optimization of an automated storage and retrieval systems by swarm intelligence’, Procedia Engineering, Vol. 100, pp. 1309-1318.

Brumpton, S., Trujillo, D., and S. W. Meng (2014) ‘Designing and delivering ‘the cube’ mixed-use building in birningham, uk’, Proceedings of the Institution of Civil Engineers: Civil Engineering, Vol. 167 No. 3, pp. 115-122.

Bruno, G. and G. D’Antonio (2018) ‘Flexible reconfiguration of AVS/RS operations for improved integration with manufacturing processes’, Procedia CIRP, Vol. 78, pp. 196-201.

Cai, J., Liu, X., Xiao, Z., Liu, J. (2009) ‘Improving supply chain performance management: A systematic approach to analyzing iterative KPI accomplishment’, Decision Support Systems, Vol. 46 No. 2, pp. 512-521.

Cai, A., Cai, Y., Guo, S., and J. Wang. (2018) ‘Scheduling policy of double-crane in automated warehouse’, Computer Integrated Manufacturing Systems, Vol. 24 No. 6, pp. 1483–1493.

Carlo, H. J., and I. F. A. Vis (2012) ‘Sequencing dynamic storage systems with multiple lifts and shuttles’, International Journal of Production Economics, Vol. 140 No. 2, pp. 844–853.

Chen, L., Langevin, A., and Riopel, D. (2010) ‘The storage location assignment and interleaving problem in an automated storage/retrieval system with shared storage’, International Journal of Production Research, Vol. 48 No. 4, pp. 991-1011.

Chen, L., Langevin, A., & Riopel, D. (2011) ‘A tabu search algorithm for the relocation problem in a warehousing system’, International Journal of Production Economics, Vol. 129 No. 1, pp. 147-156.

Chen, Z., and Y. Li (2016) ‘Dual-command operation generation in bi-directional flow-rack automated storage and retrieval systems with random storage policy’, Proceedings of the 2015 Chinese Intelligent Systems Conference (pp. 125-131).

Chirici, L. and K. S. Wang (2014) ‘Tackling the storage problem through genetic algorithms’, Advances in Manufacturing, Vol. 2 No. 3, pp. 203–211.

Chou, Y. C., Chen, Y. H., and H. M. Chen (2012) ‘Recency-based storage assignment and warehouse configuration for recurrent demands’, Computers and Industrial Engineering, Vol. 62 No. 4, pp. 880–889.

Chun Park B. (2009) ‘Turnover distribution and carousel system performance’, International Journal of Production Research, Vol. 47 No. 22, pp. 6455-6467.

Cobo, M. J., López-Herrera, A. G., Herrera-Viedma, E., and Herrera, F. (2011) ‘An approach for detecting , quantifying , and visualizing the evolution of a research field : A practical application to the Fuzzy Sets Theory field’, Journal of Informetrics, Vol. 5 No. 1, pp. 146–166.

Coulter, N., Monarch, I. and Konda, S. (1998) ‘Software Engineering as Seen through Its Research Literature: A Study in Co-Word Analysis’, Journal of the Association for Information Science and Technology, Vol. 49, pp. 1206–1223.

D’Antonio, G., De Maddis, M., Bedolla, J. S., Chiabert, P., and F. Lombardi (2018) ‘Analytical models for the evaluation of deep-lane autonomous vehicle storage and retrieval system performance’, International Journal of Advanced Manufacturing Technology, Vol. 94, pp. 1811–1824.

De Koster, R. B. M., Le-Duc, T., and Y. Yugang (2008) ‘Optimal storage rack design for a 3-dimensional compact AS/RS’, International Journal of Production Research, Vol. 26 No. 1, pp. 38-46.

Dong, W., Jin, M., Wang, Y., and L. Xing (2018) ‘Retrieval scheduling for crane-based AS/RS with compact storage system and autonomous shuttle’, In IISE Annual Conference and Expo 2018 (pp. 623–628).

Dorr, K. (2016) ‘Determination of cycle times for double deep storage systems using a dual capacity handling device’, In 14th IMHRC Proceedings.

Dorr, K., and K. Furmans (2016) ‘Throughput analysis of double deep storage using dual capacity load-handling devices’, Logistic Journal.

Dorr, K. (2018) ‘Travel time models and throughput analysis of dual load handling automated storage and retrieval systems in double deep storage’, KIT Scientific Publishing.

Dukic, G., Rose, L., Gajsek, B., Opetuk, T., and H. Cajner (2018) ‘Space, time and ergonomic assessment of order picking using vertical lift modules’, In 14th International Conference of Industrial Logistics (pp. 68-74).

Eder, M., and G. Kartnig (2018) ‘Calculation method to determine the throughput and the energy consumption of S/R shuttle systems’, FME Transactions, Vol. 46 No. 3, pp. 424–428.

Ekren, B.Y., and S.S. Heragu (2012) ‘A new technology for unit-load automated storage system: Autonomous vehicle storage and retrieval system’, Springer London, pp. 285-339.

Ekren, B. Y., Akpunar, A., Sari, Z., and T. Lerher (2018) ‘A tool for time , variance and energy related performance estimations in a shuttle-based storage and retrieval system’, Applied Mathematical Modelling, Vol. 63, pp. 109–127.

Erkan, T. E., Can, G. F., Turan, E., & Erkan, G. F. (2014) ‘Selecting the best warehouse data collecting system by using AHP and FAHP methods’, Technical Gazette, Vol. 21 No. 1, pp. 87-93.

Epp, M., Wiedemann, S., and K. Furmans (2017) ‘A discrete-time queuing network approach to performance evaluation of autonomous vehicle storage and retrieval systems’, International Journal of Production Research, Vol. 55 No. 4, pp. 960-978.

Fadlalla, A. and Amani, F. (2015) ‘A keyword-based organizing framework for ERP intellectual contributions’, Journal of Enterprise Information Management, Vol. 28 No. 5, pp. 637–657.

Feng, A., Chen, Y., and W. Ding (2012) ‘Research on the optimal dwell point location of single stacker in multiple aisles automated storage and retrieval systems’, International Conference of Logistics Engineering and Management (pp. 513–518).

Foumani, M., Moeini, A., Haythorpe, M., and K. Smith-miles (2018) ‘A cross-entropy method for optimising robotic automated storage and retrieval systems’, International Journal of Production Research, Vol. 56 No. 19, pp. 6450–6472.

Gagliardi, J., Renaud, J., and A. Ruiz (2012) ‘A simulation modeling framework for multiple-aisle automated storage and retrieval systems’, Journal of Intelligent Manufacturing, Vol. 25 No. 1, pp. 193–207.

Gagliardi, J. P., Renaud, J., and A. Ruiz (2014) ‘On sequencing policies for unit-load automated storage and retrieval systems’, International Journal of Production Research, Vol. 52 No. 4, pp. 1090–1099.

Gagliardi, J. P., Renaud, J., and A. Ruiz (2015) ‘Sequencing approaches for multiple-aisle automated storage and retrieval systems’, International Journal of Production Research, Vol. 53 No. 19, pp. 5873–5883.

Garcia de Jalon J. and Bayo E. (1994) ‘Kinematic and Dynamic Simulation of Multibody Systems: The Real Time Challenge’, Springer-Verlag, Berlin, Germany.

Ghormi, L., and O. Cardin (2018) ‘Determination of an empirical model of average rank for multi-deep AS/RS based on simulation’, Winter Simulation Conference (pp. 3184–3195).

Hachemi, K., Sari, Z., and N. Ghouali (2012) ‘A step-by-step dual cycle sequencing method for unit-load automated storage and retrieval systems’, Computers and Industrial Engineering, Vol. 63 No. 4, pp. 980–984.

Hachemi, K., and B. Besombes (2013) ‘Integration of products expiry dates in optimal scheduling of storage/retrieval operations for a flow-rack AS/RS’, International Journal of Industrial and Systems Engineering, Vol. 15 No. 2, pp. 216–233.

Hale, T. S., Hanna, M. E., Huq, F., and A. Gil (2015) ‘Closed form models for dwell point locations in a multi-aisle automated storage and retrieval system’, International Journal of Industrial and Systems Engineering, Vol. 19 No. 3, pp. 364–388.

Hahn-Woernle, P., and W. A. Gunthner (2018) ‘Power-load management reduces energy-dependent costs of multi-aisle mini-load automated storage and retrieval systems’, International Journal of Production Research, Vol. 56 No. 3, pp. 1269–1285.

Hao, J., Yu, Y., and L. L. Zhang (2015) ‘Optimal design of a 3D compact storage system with the I/O port at the lower mid-point of the storage rack’, International Journal of Production Research, Vol. 53 No. 17, pp. 5153-5173.

Hart, C. (1998) ‘Doing a literature review: Releasing the social science research imagination’, Sage Publications, London.

Heragu, S. S., Cai, X., Krishnamurthy, A., and C. J. Malmborg (2011) ‘Analytical models for analysis of automated warehouse material handling systems’, International Journal of Production Research, Vol. 49 No. 22, pp. 6833–6861.

Hu, K. Y., Chang, T. H., Fu, H. P., and H. Yeh (2009) ‘Improvement order picking in mobile storage systems with a middle cross aisle’, International Journal of Production Research, Vol. 47 No. 4, pp. 1089-1104.

Ishigaki, A., and H. Hibino (2014) ‘Optimal Storage Assignment for an Automated Warehouse System with Mixed Loading’, International Conference on Advances in Production Management Systems (pp. 475–482).

Jacomy, M., Venturini, T., Heymann, S., and Bastian, M. (2014) ‘ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software’, PloS one.

Janilionis, V. V., and Z. Bazaras (2010a) ‘Comparison of models for analysis of split-platform automated storage and retrieval systems’, Transport Means – Proceedings of the International Conference (pp. 183-187).

Janilionis, V. V., and Z. Bazaras (2010b) ‘Investigation of split-platform automated storage/retrieval system performance’, Intelligent Technologies in Logistics and Mechatronics Systems (pp. 142–148).

Janilionis, V. V, and Z. Bazaras (2012) ‘Impact of input/output position locations to the performance of C-AS/RS with autonomous load handling devices’, Intelligent Technologies in Logistics and Mechatronics Systems (pp. 86–90).

Janilionis, V. and Bazaras, Z. (2013) ‘Performance comparison of cylindrical as/rs with single and multi-load handling devices’, Transport Problems, Vol. 8 No. 4, pp. 93-102.

Juan, A. A., Grasman, S. E., Caceres-Cruz, J., & Bektaş, T. (2014) ‘A simheuristic algorithm for the single-period stochastic inventory-routing problem with stock-outs’, Simulation Modelling Practice and Theory, Vol. 46, pp. 40-52.

Kaczmarek, S., Goldenstein, J., M. Ten Hompel (2014) ‘Performance analysis of autonomous vehicle storage and retrieval systems depending on storage management policies’, IEEE International Conference on Industrial Engineering and Engineering Management (pp. 1424-1428).

Kennedy, J., and Eberhart, R. C. (1997) ‘A discrete binary version of the particle swarm algorithm. In 1997 IEEE International conference on systems, man, and cybernetics’, Computational cybernetics and simulation, Vol. 5, pp. 4104-4108.

Khojasteh, Y., and J. D. Son (2016) ‘A travel time model for order picking systems in automated warehouses’, The International Journal of Advanced Manufacturing Technology, Vol. 86, pp. 2219–2229.

Kosanic, N.Z., Milojevic, G.Z., and N.D. Zrnic (2018) ‘A survey of literature on shuttle based storage and retrieval systems’, FME Transactions, Vol. 46 No. 3, pp. 400-409.

Kulik, C. L. C., Kulik, J. A. and P. A. Cohen (1980) ‘Instructional Technology and College Teaching’, Teaching of Psychology, Vol. 7 No. 4, pp. 199-205.

Kung, Y., Kobayashi, Y., Higashi, T., Sugi, M., and J. Ota (2014) ‘Order scheduling of multiple stacker cranes on common rails in an automated storage / retrieval system’, International Journal of Production Research, Vol. 52 No. 4, pp. 1171–1187.

Lantschner, D., Atz, T., and W. A. Gunthner (2013) ‘Simulation based evaluation of concepts and strategies for automated storage and retrieval systems with multiple i/o points’, In 25th European Modeling and Simulation Symposium (pp. 544-550).

Lenoble, N., Frein, Y., and R. Hammami (2016) ‘Optimization of order batching in a picking system with a vertical lift module’, In 6th International Conference on Information Systems, Logistics, and Supply Chain (pp. 153–167).

Lenoble, N., Frein, Y., and R. Hammami (2017) ‘Optimization of order batching in a picking system with Carousels’, European Journal of Operational Research, Vol. 267 No. 3, pp. 958–976.

Lenoble, N., Frein, Y., and R. Hammami (2018) ‘Order batching in an automated warehouse with with several vertical lift modules: Optimization and experiments with real data’, European Journal of Operational Research, Vol. 267 No. 3, pp. 958–976.

Lerher, T., Potrc, I., Sraml, M., and T. Tollazzi (2010) ‘Travel time models for automated warehouses with aisle transferring storage and retrieval machine’, European Journal of Operational Research, Vol. 205 No. 3, pp. 571–583.

Lerher, T., Edl, M., and B. Rosi (2014) ‘Energy efficiency model for the mini-load automated storage and retrieval systems’, International Journal of Advanced Manufacturing Technology, Vol. 70, pp. 97–115.

Lerher, T., Sraml, M., and I Potr. 2015a. “Simulation analysis of mini-load multi-shuttle automated storage and retrieval systems.” International Journal of Simulation Modelling, 14(1): 48–59.

Lerher, T., Ekren, B. Y., Dukic, G., and B. Rosi (2015b) ‘Travel time model for shuttle-based storage and retrieval systems European Federation of Materials Handling’, International Journal of Advanced Manufacturing Technology, Vol. 78 No. 9-12, pp. 1705–1725.

Lerher, T. (2016) ‘Travel time model for double-deep shuttle-based storage and retrieval systems’, International Journal of Production Research, Vol. 54 No. 9, pp. 2519-2540.

Lerher, T., Borovinsek, M., Ficko, M., and I. Palcic (2017) ‘Parametric Study of Throughput Performance in SBS/RS Based on Simulation’, International Journal of Simulation Modelling, Vol. 16 No. 1, pp. 96–107.

Liu, S., Wang, Q., and J. Sun (2013) ‘Integrated optimization of storage allocations in automated storage and retrieval system of bearings’, In 25th Chinese Control and Decision Conference (pp. 4267–4271).

Liu, S. A., Sun, J., and Q. Wang (2014) ‘Optimization of storage performance for generic tiered warehouse by Genetic Algorithm’, In 26th Chinese Control and Decision Conference (pp. 2934–2939).

Liu, T., Xu, X., Qin, H., and A. Lim (2016) ‘Travel time analysis of the dual command cycle in the split-platform AS / RS with I / O dwell point policy’, Flexible Services and Manufacturing Journal, Vol. 28 No. 3, pp. 442–460.

Meneghetti, A., and L. Monti (2013) ‘How energy recovery can reshape storage assignment in automated warehouses’, IFIP Advances in Information and Communication Technology (pp. 33–40).

Meneghetti, A., and L. Monti (2014) ‘Multiple-weight unit load storage assignment strategies for energy efficient automated’, International Journal of Logistics Research and Applications, Vol. 17 No. 4, pp. 304–322.

Meneghetti, A., Dal Borgo, E., and L. Monti (2015) ‘Rack shape and energy efficient operations in automated storage and retrieval systems’, International Journal of Production Research, Vol. 53 No. 23, pp. 7090–7103.

Metahri, D., and Hachemi, K. (2017) ‘Automated storage and retrieval systems: a performances comparison between Free-fall-flow-rack and classic flow-rack’, In 6th International Conference on Systems and Control (pp. 589-594).

Metahri, D., and K. Hachemi (2018a) ‘Retrieval-travel-time model for free-fall-flow-rack automated storage and retrieval system’, Journal of Industrial Engineering International, Vol. 14, pp. 807-820.

Metahri, D., and K. Hachemi (2018b) ‘Optimization of free-fall-flow-rack automated storage and retrieval system dimensions’, International Conference on Embedded and Distributed Systems.

Moon, G., Kim, G. P., and W. J. Moon (2009) ‘Improvement of AS/RS performance using design and application of common zone’, International Journal of Production Research, Vol. 47 No. 5, pp. 1331-1341.

Ozaki, M., Higashi, T., Ogata, T., Hara, T., Rubrico, J., I, U., and J. Ota (2016) ‘Design of AVS/RS under group constraint’, Advanced Robotics, Vol. 30 No. 22, pp. 1446-1457.

Pan, J. C., Shih, P., Wu, M., and J. Lin (2015) ‘A storage assignment heuristic method based on genetic algorithm for a pick-and-pass warehousing system’, Computers and Industrial Engineering, Vol. 81, pp. 1–13.

Park, B. C. (2009) ‘Turnover distribution and carousel system performance’, International Journal of Production Research, Vol. 47 No. 22, pp. 6455–6467.

Popovic, D., Vidovic, M., and N. Bjelic (2014) ‘Application of genetic algorithms for sequencing of AS/RS with a triple-shuttle module in class-based storage’, Flexible Services and Manufacturing Journal, Vol. 26 No. 3, pp. 432–453.

Ralphs, T. K., Kopman, L., Pulleyblank, W. R., and L. E. Trotter (2003) ‘On the capacitated vehicle routing problem’, Mathematical programming, Vol. 94 No. 2-3, pp. 343-359.

Ramtin, F., and J. A. Pazour (2014) ‘Analytical models for an automated storage and retrieval system with multiple in-the-aisle pick positions’, IIE Transactions, Vol. 46 No. 9, pp. 968–986.

Rajković, M., Zrnić, D.Đ., Kosanić, N., Borovninšek, M., Lerher, T. (2019) ‘A multi-objective optimization model for minimizing investment expenses, cycle times and CO2 footprint of an automated storage and retrieval systems’, Transport, Vol. 34, No. 2, pp. 275-286.

Roodbergen, K. J., & Vis, I. F. (2009). A survey of literature on automated storage and retrieval systems. *European journal of operational research*, *194*(2), 343-362.

Roshan, K., Shojaie, A., and M. Javadi (2018) ‘Advanced allocation policy in class - based storage to improve AS / RS efficiency toward green manufacturing’, International Journal of Environmental Science and Technology, Vol. 16 No. 10, pp. 5695-5706.

Rosi, B., Grasic, L., Dukic, G., Opetuk, T., and T. Lerher (2016) ‘Simulation-based performance analysis of automated single-tray vertical lift module’, International Journal of Simulation Modelling, Vol. 15 No. 1, pp. 97–108.

Roy, D., Krishnamurthy, A., Heragu, S. S., and C. J. Malmborg (2012) ‘Performance analysis and design trade-offs in warehouses with autonomous vehicle technology’, IIE Transactions, Vol. 44 No. 12, pp. 1045–1060.

Sari, Z., and A. M. Hamzaoui (2013) ‘Optimization of a Single Machine Flow Rack AS / RS for Minimum Expected Travel Time’, IFAC Proceedings Volumes, Vol. 46 No. 7, pp. 57-62.

Shah, B., and V. Khanzode (2017) ‘A comprehensive review of warehouse operational issues’, International Journal of Logistic Systems and Management, Vol. 26 No. 3, pp. 346-378.

Shi, Z. and Li, L. (2017) ‘Travel time analysis of the single and dual command of AS/RS’, In Proceedings of the 29th Chinese Control and Decision Conference (pp. 3407–3413).

Song, Y., Jiang, Z., and B. Sun (2016) ‘Integrated optimization of operations in multiple-I/O points automated storage and retrieval system’, Journal of Central South University, Vol. 47 No. 6, pp. 1930-1939.

Soyaslan, M., Kozkurt, C., and A. Fenercioglu (2017) ‘A new truck based order picking model for automated storage and retrieval system (AS/RS)’, Journal of Engineering Research, Vol. 5 No. 4, pp. 169–194.

Sumathi, V., Pradeep Varma, N.V., and M. Sasank (2013) ‘Energy efficient automated car parking system’, International Journal of Engineering and Technology, Vol. 5 No. 3, pp. 2848-2852.

Tanaka, S., and M. Araki (2009) ‘Routing problem under the shared policy for unit-load automated storage and retrieval systems with separate input and output points’, International Journal of Production Research, Vol. 47 No. 9, pp. 2391-2408.

Tosun, Ö., and Aktan, H. E. (2016) ‘A multi criteria decision-making approach to evaluate automated storage and retrieval systems’, International Journal of Applied Decision Sciences, Vol. 9 No. 2, pp. 182-195.

Tranfield, D., Denyer, D., and P. Smart (2003) ‘Towards a methodology for developing evidence informed management knowledge by means of systematic review’, British Journal of Management, Vol. 14 No. 3, pp. 207-222.

Vasili, M. R., Tang, S. H., & Vasili, M. (2012) ‘Automated storage and retrieval systems: a review on travel time models and control policies’, In Warehousing in the Global Supply Chain (pp. 159-209).

Wang, Y., Zhou, Y., Shen, C., and Y. Wu (2015a) ‘Applicability selection method of two parts-to-picker order picking systems’, Journal of Mechanical Engineering, Vol. 51 No. 4, pp. 206–212.

Wang, Y., Mou, S., and Y. Wu (2015b) ‘Task scheduling for multi-tier shuttle warehousing system’, International Journal of Production Research, Vol. 53 No. 19, pp. 5884-5895.

Wang, W., Tang, X., and Z. Shao (2016) ‘Study on Energy Consumption and Cable Force Optimization of Cable-Driven Parallel Mechanism in Automated Storage/Retrieval System’, Second International Conference on Soft Computing and Machine Intelligence (pp. 144–150).

Wang, L., Jia, S., Song, J., and G. Jiang (2018) ‘A location allocation method based on association rules’, IPPTA: Quarterly Journal of Indian Pulp and Paper Technical Association, Vol. 30 No. 6, pp. 49–57.

Windmann, S., Niggermann, O., and H. Stichweh (2015) ‘Energy efficiency optimization by automatic coordination of motor speeds in conveying systems’, IEEE International Conference on Industrial Technology (pp. 731–737).

Xia, Z., Yaouha, W., Delong, X., and C. Yunxia (2018) ‘Dynamic modelling of an automated vehicle storage and retrieval system and a simulation analysis of its efficiency’, International Journal for Engineering Modelling, Vol. 31 No. 4, pp. 29-42.

Xing, B., Gao, W. J., Nelwamondo, F. V., Battle, K., and Marwala, T. (2010) ‘Ant colony optimization for automated storage and retrieval system’, In IEEE Congress on Evolutionary Computation (pp. 1-7).

Xu, X., Zou, B., Shen, G., and Y. Gong (2016) ‘Travel-time models and fill-grade factor analysis for double-deep multi-aisle AS / Rss’, International Journal of Production Research, Vol. 54 No. 14, pp. 4126–4144.

Xu, X., Gong, Y., Fan, X., Shen, G., and B. Zou, B. (2018) ‘Travel-time model of dual-command cycles in a 3D compact AS/RS with lower mid-point I/O dwell point policy’, International Journal of Production Research, Vol. 56 No. 4, pp. 1620-1641.

Xu X., Gong Y., Shen G., Zou, B. (2018) ‘Travel-time model of dual-command cycles in a 3D compact AS/RS with lower mid-point I/O dwell point policy’, International Journal of Production Research, Vol. 56 No. 4, pp. 1620-1641.

Xu, X., Shen, G., Yu, Y., and W. Huang (2015) ‘Travel time analysis for the double-deep dual-shuttle AS/RS’, International Journal of Production Research, Vol. 53 No. 3, pp. 757-773.

Yang, P., Miao, L., Xue, Z., and Qin, L. (2015) ‘An integrated optimization of location assignment and storage/retrieval scheduling in multi-shuttle automated storage/retrieval systems’, Journal of Intelligent Manufacturing, Vol. 26 No. 6, pp. 1145-1159.

Yang, P., Peng, Y., Ye, B., and L. Miao (2017) ‘Integrated optimization of location assignment and sequencing in multi-shuttle automated storage and retrieval systems under modified 2 n - command cycle pattern’, Engineering Optimization, Vol. 49 No. 9, pp. 1604–1620.

Yu, Y., and R. B. M. De Koster (2009) ‘Designing an optimal turnover-based storage rack for a 3D compact automated storage and retrieval system’, International Journal of Production Research, Vol. 47 No. 6, pp. 1551-1571.

Yu, Y. and de Koster, M.B.M. (2012) ‘Sequencing Heuristics for Storing and Retrieving Unit Loads in 3D Compact Automated Warehousing Systems’, IIE Transactions, Vol. 44 No. 2, pp. 69-87.

Yu, H., Y. Yu (2019) ‘Optimising two dwell point policies for AS/RSs with input and output point at opposite ends of the aisle’, International Journal of Production Research, Vol. 57 No. 21, pp. 6651-6633.

Yuan, R., Graves, S. C., and T. Cezik (2019) ‘Velocity-Based Storage Assignment in Semi-Automated’, Production and Operations Management, Vol. 28 No. 2, pp. 354–373.

Zhang, G. Q., and Lai, K. K. (2006) ‘Combining path relinking and genetic algorithms for the multiple-level warehouse layout problem’, European Journal of Operational Research, Vol. 169 No. 2, pp. 413-425.

4. Key Performance Indexes

In warehousing, the term '*performance*' is often used to indicate the state of health of the system (i.e. AS/RS or manual warehouse). The state of health of a warehouse might be measured in several different ways, and each of them provides a different view of the system. As it is possible to understand from the literature review proposed in this work, many studies focus on performance in the sense of operations completed per unit of time. This is actually the most important aspect to evaluate a warehouse, because a greater and faster processing of operations ensures a greater customer satisfaction and gives companies the opportunity to expand by increasing their sales (Abushaikha et al., 2018). However, it is not the only one, and in this chapter a brief overview of the most important Key Performance Indexes (KPI) useful to understand the state of health and AS/RS are shown.

4.1. Operations per time unit

As mentioned above, the number of operations per unit of time is an important parameter to be measured and improved. Each AS/RS must be able to deal with the input and output requests otherwise queues of entering items and retrieving operations would continue to accumulate. By increasing the number of operations that the system can complete, the delivery trucks would be loaded in a shorter time and the customers served faster, reducing the lead time. Similarly, by increasing the operations that can be completed per time unit, the companies increase their capacity, becoming able to carry out more shipments per day. This, in case of corresponding market demand, allows them to sell more, with a consequent growth of profits. Furthermore, even if the amount of sales does not require an increased capacity, if the system completed its operations in a shorter time, it would have more time for reorganizing the stock or anticipating the next day's operations.

The operations per time unit are usually expressed in terms of operations per hour or, otherwise, operations per day. The result is called *throughput* and is therefore representative of how fast is the AS/RS in completing the operations required. Usually the throughput for input and output operations are reported separately.

A complementary and equally well-known index is the *cycle time*, which is usually expressed in seconds and represents the average time the AS/RS takes to complete a cycle. The cycle time is less used, because it introduces further complications such as the necessity to observe (i) the single command cycle in input operations, (ii) the single command cycle in output operations, and (iii) the dual command cycle too. Moreover, when the storage area is

served by several different machines moving in parallel, the average time to complete a cycle is difficult to trace back to the performance of the entire system.

Moreover, for both throughput and cycle time, in order to provide a more reliable analysis, not only the average is computed, but also the standard deviation and, in some cases, the entire probabilistic distribution (see for instance [1]).

4.2. Storage capacity

The storage capacity is essentially the quantity of products that the system is able to store. it can be expressed in many different ways depending on the products in inventory, e.g. kilograms, tonnes, cubic meters, unit loads, etc. Concerning the steel sector, since the products in inventory are often very heavy, the most popular drivers are the kilograms or tonnes, and the unit loads (especially when the products in stock are all different from each other).

Sometimes, the storage capacity is expressed as a quantity in stock per unit of surface/volume. In this way, it becomes an indicator of superficial/volumetric exploitation. This is very useful during the design since it immediately provides information concerning the necessary space to store the handled quantity.

4.3. Modularity

Modularity is an indicator of flexibility. This indicator was designed for softwares and, in particular, it was born to measure the strength of division of a network into modules (i.e. clusters). The higher is the modularity, the greater are the possibilities of expansion and modification.

In literature there are some models which define how to quantify and define the modularity for software implementations (see for instance Xiang et al. (2019)), although, to the author's best knowledge, the same study specifically focused on AS/RS is missing.

An AS/RS that makes modularity its strength is the Automated Grid-Based Warehouse (Beckschäfer et al., 2017).

4.4. Selectivity

Selectivity indicates the number of unit loads or products directly accessible by the machines without moving any other unit load or item in stock. It matches the information provided by the throughput with those of the storage capacity, because, in general, if all the items in stock are directly accessible, the time in which the operations are completed is shorter;

otherwise, in case of low selectivity the space dedicated to the aisles and the machines railways is smaller, hence the storage capacity of the system is higher.

Unlike for the modularity, the selectivity is unanimously defined as a value between 0 and 1, where 1 means that all the items/products/unit loads in stock are directly accessible, and 0 the borderline case in which none of them is directly accessible.

4.5. Energy consumption

The energy consumption can be expressed in kWh, however, in many scientific papers this value is then translated into total emissions of CO_2 equivalent. This is because the criticality of energy consumption in AS/RS is not the cost saving, but it is more an environmental aspect. Many important AS/RS sellers and producers affirm that the consumptions of an AS/RS have a neglectable impact on the operative costs, are traceable to that of normal household appliances.

4.6. Maintenance costs

The maintenance costs are mainly concerned by the works focused on purely mechanical and electrical aspects. Although, operational decisions and control policies may also have a relevant impact on the maintenance costs. In these terms, a good indicator to prevent the maintenance costs is the distance travelled by the machines. In fact, in the steel sector are adopted big and heavy machines, which absorb a large part of maintenance costs. By reducing the distance travelled by these machines, the maintenance costs can be considerably reduced (see for instance [5]).

References

Abushaikha, I., Salhieh, L., & Towers, N. (2018). Improving distribution and business performance through lean warehousing. *International Journal of Retail & Distribution Management*.

Beckschäfer, M., Malberg, S., Tierney, K., & Weskamp, C. (2017). Simulating storage policies for an automated grid-based warehouse system. *International Conference on Computational Logistics* (pp. 468-482). Springer, Cham.

Xiang, Y., Pan, W., Jiang, H., Zhu, Y., & Li, H. (2019). Measuring software modularity based on software networks. *Entropy*, *21*(4), 344.

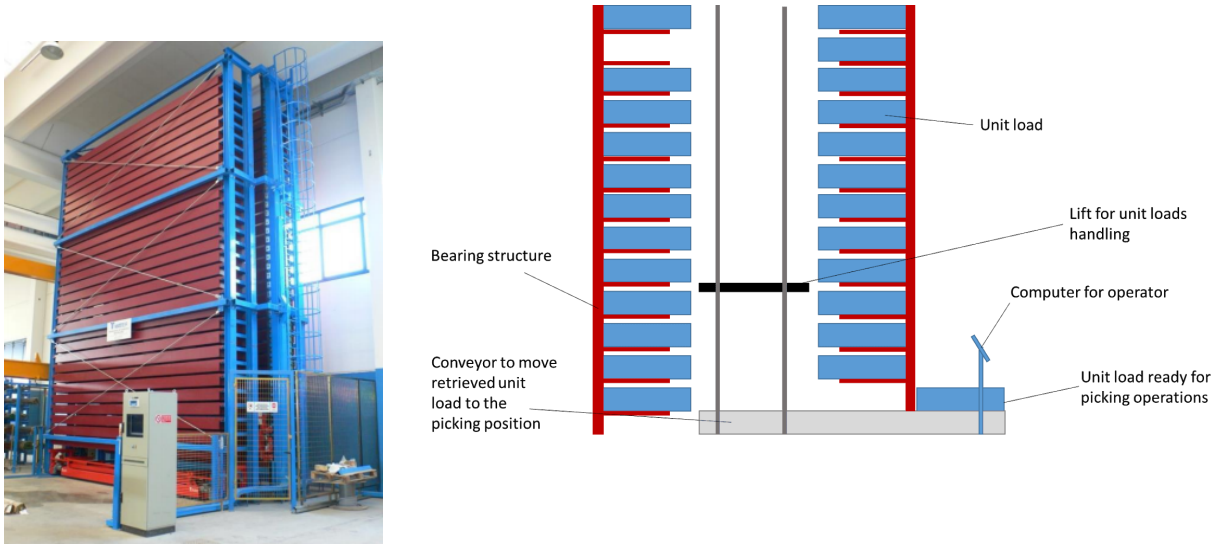
5. Logistics of Small Parts

This chapter is focused on the logistics of small products, which, in the steel sector consist of metal dies, individually stored metal bars, metal moulds, components such as bolts and screws, etc. All these products are generally stored inside metal boxes or metal trays that, depending on the size of the items in stock, might be splitted into several sections each of which is dedicated to a different product. Even if the products are considered ‘small’, the length of these unit loads may vary from 3 up to 7 meters, hence, their handling requires systems specifically designed for this aim. A representation of a classic unit load for small parts is provided in Figure 5.1.

Figure 5.1. Typical unit load for small products.

5.1. Storage Solutions: Vertical Lift Module (VLM)

The VLM is a stock intensive storage system which generally consists of two vertically arranged storage areas divided by a single aisle where a delivery lift moves, while the operator is called to wait for picking units standing in front of a picking position or an ergonomic workstation. In classic VLM (see for instance the well-known Modula: https://www.modula.eu/ita/), the unit loads where goods are stored consist of metal trays with a maximum capacity of 1000 kilograms. In the steel sector, since the system has been rearranged for storage of long metal bars or metallic dies, each unit load has a maximum capacity around 3000-5000 kilograms (i.e. even 5 times bigger than usual) and a kerb weight of 400 kilograms. To facilitate the comprehension of the system a real picture and a schematic representation are respectively provided in Figure 5.2 (a) and (b).



(a) (b)

Figure 5.2. Real picture (a) and scheme (b) of the VLM [3].

The functioning of the system is the following. Each time the operator asks for a unit load, the delivery lift moves in vertical direction, when it reaches the height where the desired unit load is stored, it hooks the unit load and brings it down to the conveyor. Then, the conveyor moves the retrieved unit load to the picking position. Once the operator has finished, he/she can ask for a new unit load or simply ask to return the current one inside. At this time the conveyor moves the unit load to the lift, which takes it back to the storage location.

Hence, the functioning is linear, with no overlaps or activities performed in parallel (i.e. at the same time). This is also the main difference with the classic VLM such as Modula (https://www.modula.eu/ita/). Unlike automated solutions for bulky parts (chapter 6), which in case of the steel industry are characterised by additional complications and constraints if compared to classic solutions (e.g. AS/RS for pallets), in case of VLM, the system adopted in the steel sector is easier than the classic one from an operational perspective. In fact, in classic VLM like Modula, while the operator is picking from the retrieved unit load, the lift is allowed to retrieve the next one and place it in a buffer position under the picking zone. There is therefore a sort of parallelism which makes the simulation of the system and its performance calculation more complicated.

Since the system is relatively simple, the controls are given to the PLC directly by a Supervisory Control and Data Acquisition (SCADA), whose Human Machine Interface (HMI) is usually accessible from a computer in the control station beside the machine, although remote control is possible too.

An alternative version of the system might be defined as double-deep VLM. In this case two unit loads are placed one behind the other in each storage location. The main difference in case of double-deep VLM concerns the necessity to remove the unit load in first depth, every time that one in double depth is required. Because of this necessity, a storage location must always be kept empty, so that it can be used as a temporary support for the unit load in the first depth to be moved.

Of course many different control policies can be implemented and the functioning of the system might be modified. For instance, a storage location closed to the picking position might be kept empty and used as the buffer position (i.e. *buffer location*), in this way, while the operator is picking, the delivery lift could take care of the next unit load moving it to the buffer location. Conversely, no fixed storage location might be assigned to the unit loads, so that, every time a unit load is taken inside, it is placed in the first empty location, implementing a sort of full-turnover allocation policy. However, to the author's best knowledge there is no need for complicated control policies in VLM. The buyers who implement this system are not very interested in the throughput and in the cycles per hour; the main reason because they buy a VLM is for its storage capacity and reduced surface area. Indeed, the VLM allows the storage of great quantities occupying very little surface and developing mostly in height.

5.2. Performance Measurement

As mentioned above, there are no conspicuous advantages that can be obtained by the implementation of complicated control policies in a VLM. First of all, because the buyers who decide to use this system are more interested in its increased storage capacity than in the possible throughput. Secondly because its relative simplicity and its dimensions make routing, scheduling, and sequencing decisions superfluous. In a system such as the Shuttle-Lift-Crane AS/RS, the travel-time is notoriously the most consistent component of operating time, since, in a well-designed system, the machines spend the most of their time moving from a position (i.e. location, I/O point, interchange point) to the other. Conversely, in VLM, the distance that the lift runs in each cycle is not relevant if compared to the picking time of the operator, which is much longer. Moreover, the VLM usually does not communicate in real time with the ERP or the Manufacturing Execution System (MES), thus the organisation of retrievals is not automatically made by the warehouse, but it depends on the operator's decisions. For these reasons, choices concerning the scheduling or sequencing of operations can be neglected, as well as the routing of the lift.

However, in most cases the handled unit loads contain many different products, hence, important time and cost savings might be obtained by improving the (i) location assignment, (ii) the unit loads filling, and (iii) the collocation of products inside the unit loads. For instance, a great saturation method can reduce the number of the needed unit loads, reducing in this way the investment cost. As the number of unit loads increases, a better saturation can lead to more and more relevant money saving. Similarly, the utilisation of a reduced number of unit loads means that the lift has to travel a shorter distance at each cycle, with a consequent improvement in both input and output performance. Another example of allocation policy that improves the performance in VLM is the allocation in the same unit loads of products which are often required together (i.e. joint retrieving). In fact, by placing the jointly required items in the same unit loads, the retrieving operations necessary to fulfill a customer order can be reduced, with a consequent time saving.

Concerning the KPIs described in chapter 4, the following considerations can be made. The calculation of the throughput is therefore simple, since all movements are sequential. The main expedients that can be taken to improve the precision of the travel-time calculation are kinematic; for example it can be computed considering the acceleration and deceleration time as in Eq.(3.2). The throughput may therefore be expressed in unit loads retrieved per hour or fulfilled orders per hour.

The selectivity is always equal to 1, since all the unit loads are directly accessible. However, in case of double-deep VLM, the selectivity is always 0.5 since the number of accessible unit loads is exactly half of those in stock.

Concerning modularity, being the VLM a small warehouse where each unit load can occupy each storage location it is notoriously a modular solution. If the storage capacity must be increased more modules can be installed too. However, the integration of the new modules with the old one is not easy, especially if the control system is not centralised, once a customer order is processed, the operator has to know in which module the required product is stored. Even the physical integration is not easy: the modules might be placed one beside the other, but the handling machines are hardly integrable. Furthermore, the single modules cannot be extended and equipped to store more unit loads, because this would require intervention on the supporting structure with unsustainable and not convenient costs.

With respect to the energy consumption, as it is possible to observe in chapter 3 section 3.11, there are no scientific contributions on the energy consumption in VLM. Given the

dimensions and the standardization of the system, it is reasonable to believe that the energy aspects do not have such a considerable impact in this case.

5.3. Design Improvement

5.3.1. Introduction

In VLM, as in any AS/RS, the shape of the storage area (i.e. racks) affects performance and an effective trade-off between width, height, and length can minimize travel-time (Bozer, Y.A. and White, 1996; De Koster, Le-Duc, and Yugang, 2008). However, in VLM width and length are fixed and the design problem is reduced to one dimension: the height. The height depends on the number of stored unit loads, and the lower the height the lower the purchase costs and operating travel-time. For this reason, it is important to properly define the collocation of items inside unit loads, so that the number of used unit loads is reduced. For this reason, a constructive heuristic algorithm is proposed in this section. Given a list of items to be stored, the aim of the algorithm is to set out the best allocation with the purpose of reducing the unit loads used. The algorithm might be implemented in both the (i) design and the (ii) filling phase: in the first case, it provides an estimation of the number of unit loads needed to store a specific set of items, while, in the second case, it provides a good collocation of items so that only the unit loads in the lower levels of the warehouse are used, reducing the cycle times for storage and retrieving. An additional parameter such as the height of the unit loads is also considered. This parameter can be relaxed and let the algorithm define it (useful in case of design problems), but it can also be fixed by the user and adopted as a constraint (useful in case of allocation problems). Hence, the aspects taken into account by the proposed algorithm are: parallelepiped-shaped items of different size, the mass capacity of the unit loads, the volume capacity of the unit loads, and the height of the unit loads which might be used as a variable or a constraint depending on the application case. Note that in literature some authors extend the problem of collocation to irregularly shaped items (see for instance Egeblad (2009)). The proposed solution just considers parallelepiped-shaped items, although this is not an implementation limit, because the algorithm would provide a solution even with irregular items, the only need is an inscription of irregular objects in rectangles before iterating the algorithm.

5.3.2. The proposed algorithm

Given a set of 3-dimensional parallelepiped-shaped items to stock inside an automated vertical storage system, the algorithm finds out the arrangement that minimizes the number

of containers needed, in order to improve performance and reduce costs. Firstly, the items to be stored are sorted by decreasing height. Then, beginning from the top of the list, they are placed in containers using a procedure similar to the Finite Bottom Left (Berkey & Wang, 1987). To verify if an object can fit in the surrounding area, the algorithm approximates items and unit loads by scanlines, and hands the two-dimensional BPP as a variant of one-dimensional BPP, exactly as Okano (2002). Before placing an item, the algorithm checks the exposed perimeter (i.e. the item's perimeter that is not in touch with container's borders or other objects) in both possible orientations, by performing a 90° rotation on the height-axis, and it selects the orientation with the minimum exposed perimeter. This criteria, according to Ma and Zhou (2017) is an efficient way to find the best orientation. Then, once all the items have been placed, an optimisation process relocates objects inside the same container and switches items between different containers to further improve the surface exploitation.

The inputs and constraints relative to the unit loads and the warehouse are the following:

- Unlimited usable number of unit loads;
- Width and Length of the unit loads;
- Weight capacity of the unit loads;
- Maximum height of the unit loads;
- Maximum number of different heights (if set equal to 1 the maximum height is used as a constraint).

The inputs linked to the items to allocate are the following:

- Dimensions (i.e. height, length and width);
- Mass;
- A priority level.

It is also assumed that the items can spin on the basement but they cannot lie on their side. In other words, items cannot be tipped over or placed on one side.

The aim of the algorithm is to reduce as much as possible the number of unit loads used. Firstly, the algorithm sorts the items by descending height and items of equal height are sorted by descending mass and, lastly, objects characterized by equal mass and height are sorted by decreasing priority level. Then it works as follows:

1. Select the first item in the list (beginning from top).

2. Select the first unit load (prioritising those already used).
3. Select first possible item orientation.
4. Beginning from the bottom left corner, look for an empty area.
5. If the area is big enough to accommodate the item, calculate the exposed perimeter.
6. Change orientation and go back to 4.
7. If both orientations provide a possible placement, select that one which ensures the smaller exposed perimeter and go to 9; if only one orientation is possible choose it; otherwise, if none orientation is possible, consider the next unit load in list and go to 4.
8. If the item placed was not the last, take the next and go to 2.

Figure 5.3 shows a generic solution provided by the algorithm. A plan view of five unit loads is reported. In the example reported, 150 items (red, yellow, green and orange colours) have been placed in 5 unit loads. Blue colour represents empty spaces.

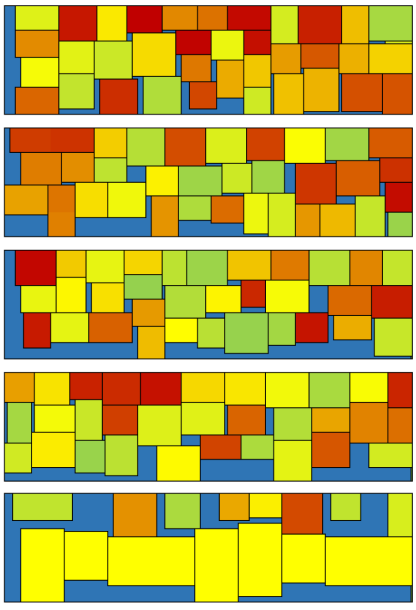


Figure 5.3. Plan view of a solution provided by the algorithm (Bertolini et al., 2018).

Then an optimisation procedure is used to redistribute the items, removing them from the unit loads characterized by lowest volumetric exploitation. The objective is to empty the less exploited unit loads. Essentially, the optimization consists of 3 different loops. The first loop simply tries to move items from the emptiest unit loads to others just double-checking what the previous constructive procedure did. In the second loop, items which are not in touch with others are moved toward the borders of the unit load where they are. This procedure rearranges the items, eventually generating new empty areas where other items (coming from the emptiest unit loads) might be placed. The third loop takes the items from the

emptiest container, looks into others for a smaller item with empty space around it, and switches the two items to improve the solution. The optimization algorithm steps work as follows:

FIRST LOOP

1. Select emptiest unit load.
2. Take the first item inside the selected unit load.
3. Select first possible orientation.
4. Try to plug it into other unit loads.
5. If any space hasn't been found and the second possible orientation has not been tried yet, change orientation and go to 4, otherwise go ahead.
6. If the item selected was the last STOP, else take the next item in the selected unit load and go to 3.

SECOND LOOP

7. Select the first unit load different from the emptiest one.
8. Beginning from the bottom left corner, move every item that is not obstructed by others and it is not in touch with the unit load's edges, in a direction perpendicular to the long edge as represented in Figure 5.4.
9. Beginning from the bottom left corner, move every item in a direction perpendicular to the short edge as represented in Figure 5.5.
10. Repeat FIRST LOOP to try to transfer an item from the emptiest unit load to the selected one.
11. If the unit load was the last STOP, otherwise select the next one and go to 8.

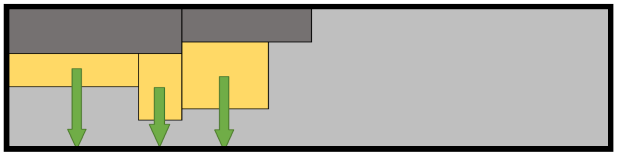


Figure 5.4. Representation of procedure described in step 8 (Bertolini et al., 2018).

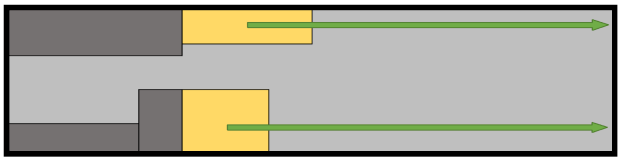


Figure 5.5. Representation of procedure described in step 9 (Bertolini et al., 2018).

THIRD LOOP

12. Select the first item in the emptiest unit load and check its surface (i.e. S1).
13. Select the first unit load (i.e. C2) different from the emptiest one.
14. Check the surface (i.e. S2) of the first item in the selected unit load.
15. If S1 > S2 and there is empty space around S2, then try to place the biggest item instead of the smaller one. If placing is possible, switch their position and go to 17, in any other case go to 16.
16. If items in C2 are finished go on, otherwise select the next item in C2 and go back to 15.
17. If unit loads different from emptiest one are finished go on, otherwise select the next one and go to 14.
18. If items in the emptiest unit load are not finished, select the next one, check its surface, and go back to 13. Otherwise, repeat the FIRST LOOP and STOP.

An example to better explain how the third loop works is provided. In Figure 5.6 the unit load (UL) number 2 is the emptiest one. At the moment it is impossible to place item B in any empty space such as C. The algorithm firstly switches B with a smaller item with an empty area around such as A (Figure 5.7). At this point, it is evident that the surface exploitation of unit load 1 has been improved. Additionally, with a final repetition of the first loop (as explained in step 18), it is possible to place object A in empty space C as shown in Figure 5.8 to further improve the solution.

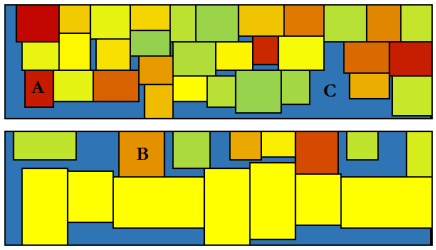


Figure 5.6. Starting point before applying the optimisation.

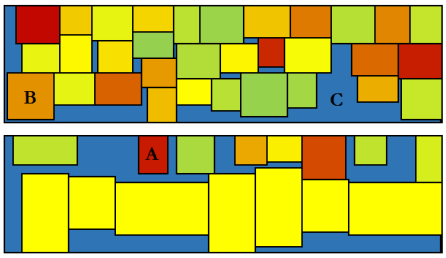


Figure 5.7. Intermediate point: switch between two items.

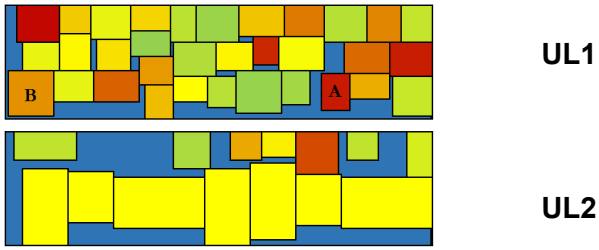


Figure 5.8. Final point after implementing the optimisation.

5.3.3. The validation

To analyse the results of the algorithm it was tested in several iterations with different numbers of items, whose average dimensions were ranging between 50 and 800 millimetres. In the experiment, unit loads considered were 3 meters long and 0.8 meters large.

The algorithm was tested in two different situations. In the first one, the unit load's mass capacity constraint was relaxed to check the actual volume and surface saturation. In the second situation, the mass capacity was reduced a lot to check the actual saturation in terms of mass.

In 16.05% of all runs, the number of unit loads required by the algorithm was equal to the minimum (Figure 5.9). The concentration of these optimal results was bigger when the items managed were smaller.

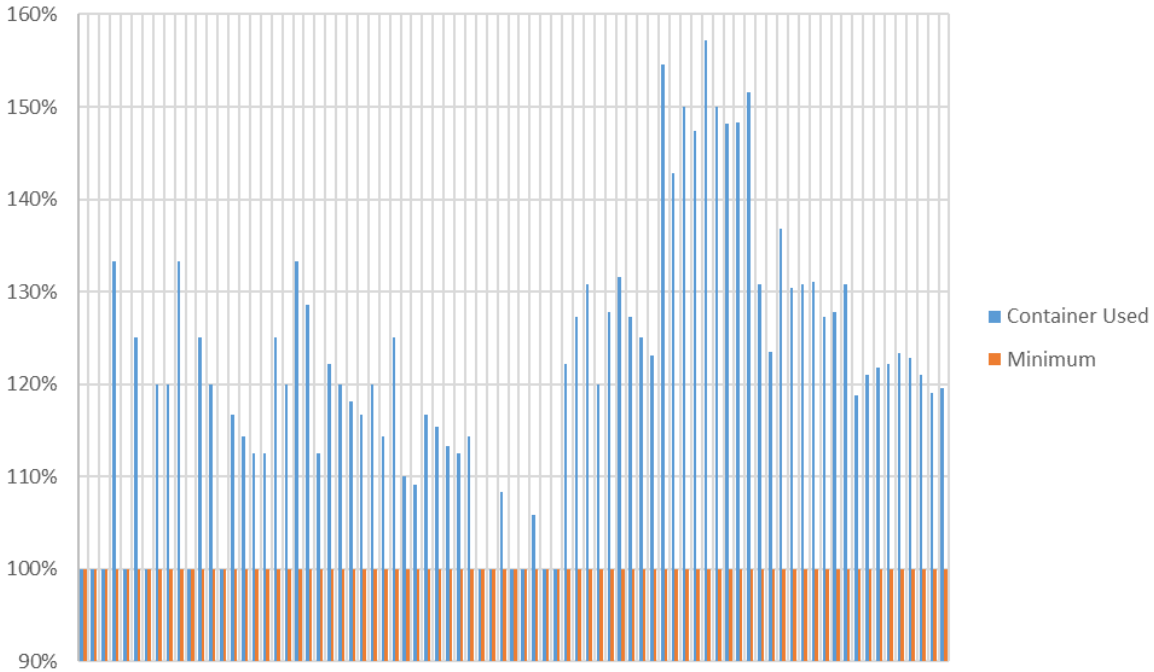


Figure 5.9. Comparison between unit loads used and minimum number required to store all the items (Bertolini et al., 2018).

The surface and volumetric exploitation levels, obtained by relaxing the mass capacity constraint, are reported in Table 5.1. While those obtained with the mass capacity constraint are reported in Table 5.2.

Table 5.1. Results in surface and volume saturation (Bertolini et al., 2018).

	Max	Medium	Min
Surface	99.9%	82.7%	64.3%
Volume	99.1%	80.5%	63.8%

Table 5.2. Results in mass saturation (Bertolini et al., 2018).

	Max	Medium	Min
Mass	99.1%	80.6%	63.9%

The volumetric and surface exploitation increase as the items' average dimension grows, as represented in Figure 5.10.

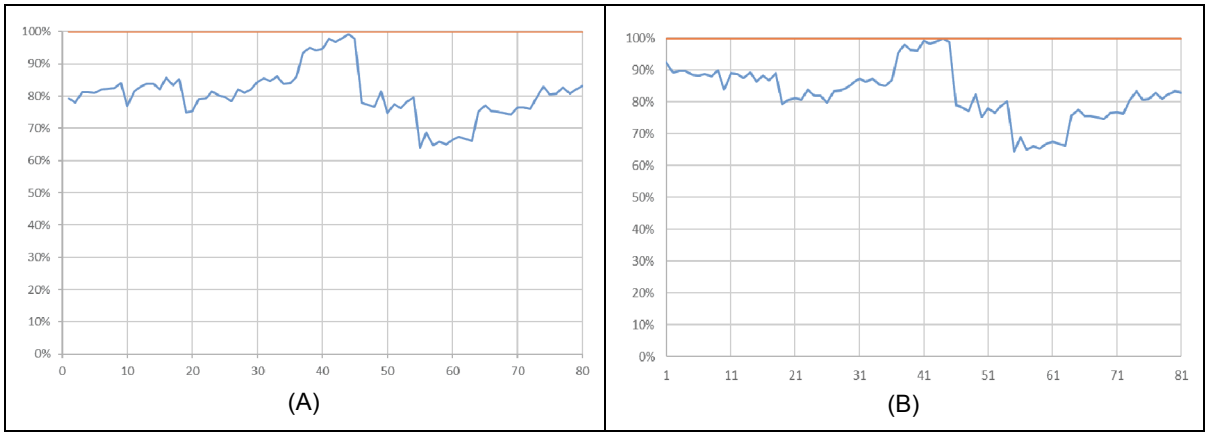


Figure 5.10. The overall volume (A) and surface (B) exploitation as the average size of the items (x-axis) increases (Bertolini et al., 2018).

5.4. Allocation Improvement

5.4.1. Introduction

A correct allocation has great importance in automated warehouses where routing decisions are limited, and the route taken by the machines cannot be optimized beyond a certain threshold (Atmaca and Ozturk, 2013), as in Vertical Lift Modules (VLMs) (Bertolini, Neroni and Romagnoli, 2018; Bertolini, Mezzogori and Neroni, 2020). Concerning the allocation and assignment of the products to the unit loads, only few authors consider the correlation between products and the advantages allowed by close allocating two products which are often ordered together (see for instance Erkip, Hausman and Nahmias (1990); Manzini, 2006; Xiao and Zheng, 2008; Zhang, Wang and Pan, 2019). Because of this, the objective of this section is to extend research in this area, dealing with this aspect rarely considered before.

5.4.2. Problem description and model formalization

The algorithm proposed in this section tries to optimize the allocation of goods inside a VLM. The final solution, respecting the weight capacity of the unit loads, must guarantee that (i) unit loads are balanced in terms of weight and (ii) the codes frequently requested together are placed in the same unit load. The first is a mechanical requirement that increases safety and reduces machines' wear, while the latter reduces the average number of retrieves required to fulfil a picking order. As usually happens in vertical lift modules used for small components, each unit load is divided into a fixed number of compartments. Inside each compartment, just one code is stored, and goods are usually contained in boxes whose size

coincides with that of the compartment. In order to avoid misunderstandings due to nomenclature, elements mentioned above are better described. A unit load is the single element stored into the warehouse and transported by the lift. Each unit load can be split by metallic separators in compartments. Each compartment can be filled with one and just one box (typically a paper box), which contains more items characterised by the same code. In Figure 5.11 a conceptual representation of mentioned elements is represented.

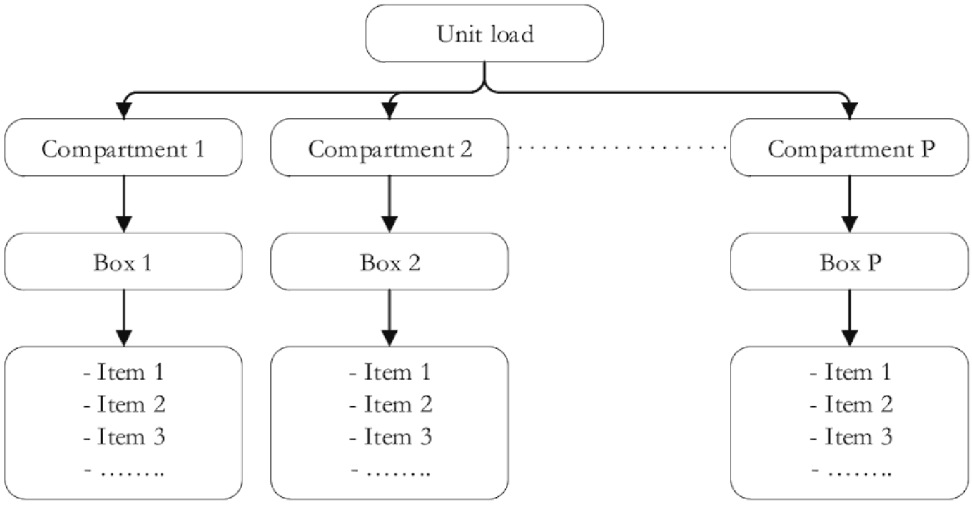


Figure 5.11. Conceptual hierarchical representation of unit loads and their components (Bertolini et al., 2019).

After defining (i) a list of single-code boxes to be placed inside the warehouse, (ii) the current state of the warehouse (a list of previously filled compartments), and (iii) the historical demand from which the correlation between the managed codes can be calculated; the algorithm provides a good allocation of boxes that balances the unit loads and reduces the number of retrievals used to fulfil orders. Given $\{i_1, ..., i_N\}$ the single-code boxes to be stored, the solution is formalized as a list of length $N+M$ in which each box occupies one only position and where $\{j_1, ..., j_M\}$ are the dummy boxes. Dummy boxes are codeless and zero-weight boxes, which represent the compartments that will remain empty even after the filling procedure. The evaluation of the solution is done by iterating the list of unit loads in stock $\{U_1, ..., U_k\}$ keeping a pointer p on the solution. At first p is set equal to 0. For each unit load, given v the number of its empty compartments, elements of solution whose position is within the range $[p, p+v)$ will be allocated into the unit load. Then,

$p = p + v$ and the next unit load is considered. A representation of solutions interpretation is given in Figure 5.12.

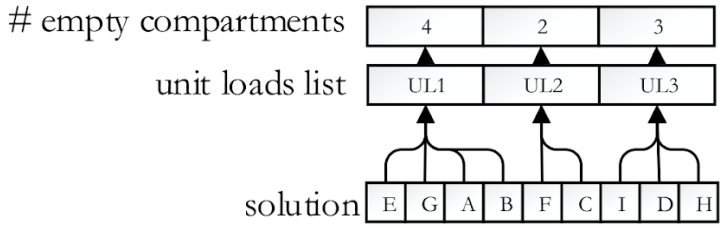


Figure 5.12. Representation of solutions interpretation (Bertolini et al., 2019).

Each unit load has a fitness value that considers its balance and the correlation between the codes within it. The overall fitness of the solution is given by the average between fitnesses of all the unit loads. The fitness of each unit load f is calculated by the Eq. 5.1, where *B* is its balancing index, *C* its correlation index, and *α* and *β* are weights empirically attributed whose sum is 1.

$$f = \alpha B + \beta C \tag{5.1}$$

Given ψ the unbalance of the box, the index of unbalance *B* is a value included in the range $(0, 1]$ and it is calculated with the Eq. 5.2. The higher value of *B*, the more balanced unit load is.

$$B = a^{-\psi} \tag{5.2}$$

In Eq. 5.2 a is a scalar calculated with Eq. 5.3 in such a way that, for the maximum unbalance ψ_{max} the index *B* is equal to a very small value ε close to zero.

$$a = \sqrt[v_{max}]{1/\varepsilon} \tag{5.3}$$

The unbalance of the unit load is calculated with a formula that fits well whatever the number of compartments and whatever their arrangement. In the proposed model, the compartments assigned to each unit load are considered, however, it is not considered how they will be arranged inside it to ensure effective balancing. The solution could be a specific program which considers the centre of gravity of the unit loads and their geometric characteristics, however, this is not treated for the purposes of this section. The unit loads' unbalance ψ is the sum of absolute differences between each compartment's weight and the average weight of the unit load. Given a set of *P* compartments within a unit load, namely $\{s_1, ..., s_P\}$, w_s the weight of a compartment and μ the average weight of the unit load, the unbalance is calculated with Eq. 5.4.

$$\psi = \sum^{P} |w_s - \mu| \tag{5.4}$$

The maximum unbalance ψ_{max} is calculated using Eq. 5.5 and it can be obtained in the worst case, when a single compartment is enough to saturate the capacity of the unit load *W*, and all other compartments must be empty.

$$\psi_{max} = (W - \mu) + \mu(P - 1) \tag{5.5}$$

For the correlation between codes, the well-known correlation coefficient by Pearson *r* has been used. However, while the value of *r* between two perfectly correlated codes would be 1, in the proposed model this value is set to 0. This is necessary to avoid single-code unit loads, which, in case of small retrieves to which VLMs are often subjected, would slow down the picking. For the calculation of *C*, a sigmoid function as in Eq. 5.6 is used, where ρ is the correlation inside the unit load.

$$C = \frac{1}{1+e^{-b\rho}} \tag{5.6}$$

In Eq. 5.6, the scalar *b* is calculated using Eq. 5.7 to make sure that for the maximum correlation ρ_{max}, value of *C* is equal to a value θ close to 1.

$$b = \frac{ln(\frac{1}{\theta} - 1)}{\rho_{max}} \tag{5.7}$$

The correlation within a unit load is the sum of the correlations between its codes, calculated considering each pair of compartments only once. Given $\{s_1, ..., s_P\}$ compartments within the unit load and $r_{i,j}$ the correlation coefficient between the codes contained in the compartments s_i and s_j, the correlation within the unit load can be calculated with Eq. 5.8. Knowing that the maximum value of the correlation coefficient between two variables is 1, the maximum correlation obtainable can also be easily calculated using Eq. 5.9.

$$\rho = \sum_{i=1}^{P-1} \sum_{j=i+1}^{P} r_{ij} \tag{5.8}$$

$$\rho_{max} = 1 \cdot \frac{P(P-1)}{2} \tag{5.9}$$

5.4.3. The proposed algorithm

The algorithm proposed is a meta-heuristic known as Genetic Algorithm (GA) (Mayer et al., 1999). However, some expedients have been taken to ensure a correct implementation. First, before the algorithm starts, as many completely empty load units (i.e. they consist exclusively of dummy boxes) as possible are built using a constructive procedure. The built boxes are frozen and inserted in what will be the final solution, and they will no longer be

subject to any modification. Then, given *N* the number of boxes to allocate, the number of possible solutions is equivalent to *N!*; for this reason, when the number of boxes is too high, the number of solutions explored by a normal GA in an acceptable computational time would be too small and insufficient to find an acceptable local optimum solution. Because of this, the following consideration is made: if the number of boxes to allocate is lower than a threshold a simple genetic algorithm is executed (namely *Procedure-Y*); otherwise, a different procedure (namely *Procedure-Z*) is used. In Figure 5.13 the *Procedure-Y* is represented.

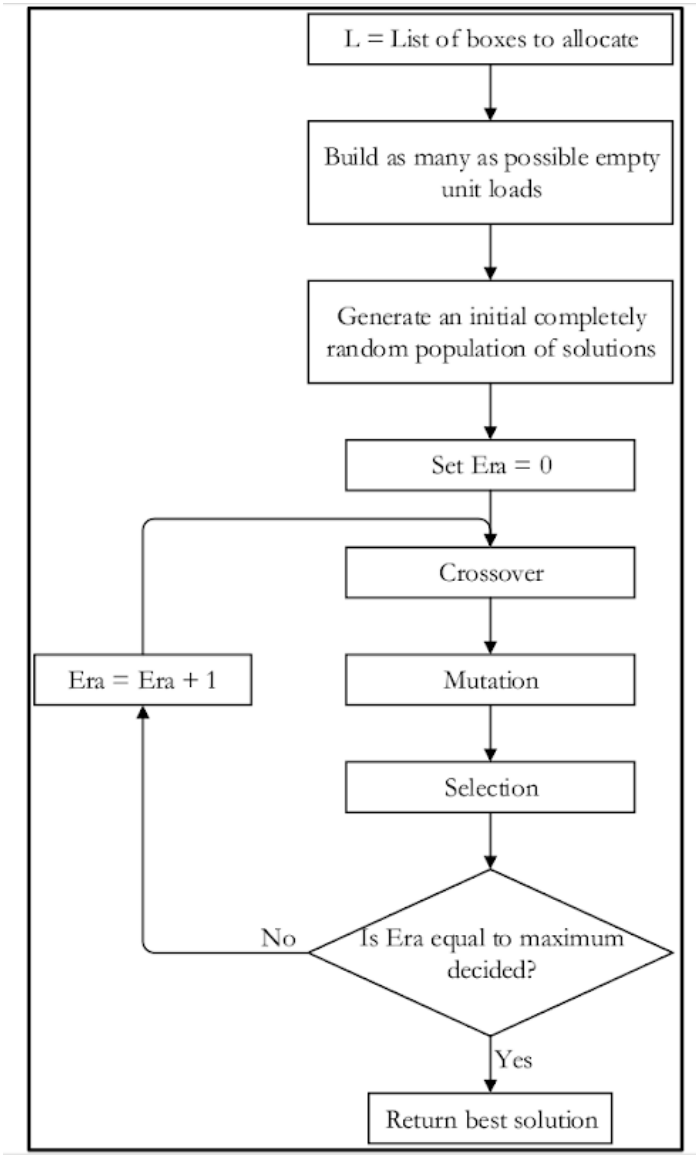


Figure 5.13. Procedure-Y (Bertolini et al., 2019).

As it happens in every GA, once defined the number of eras and the number of solutions *G* in the initial population, for each era three phases are performed: *crossover*, *mutation* and

selection. These three phases are subsequently described in more details. Conversely, Procedure-Z is a constructive greedy procedure. First, a fitness threshold value *T* is defined to identify which unit loads are acceptable and which are not. Until the final solution is not complete, the genetic is executed more times and, each time, observing the best returned solution, the unit loads whose fitness value f exceeds the threshold *T*, are inserted in the final solution. Every time the number of accepted unit loads is zero, the threshold *T* is decreased by a predefined value **Δ**. The procedure is performed cyclically until the final solution is completed. Procedure-Z is represented in Figure 5.14.

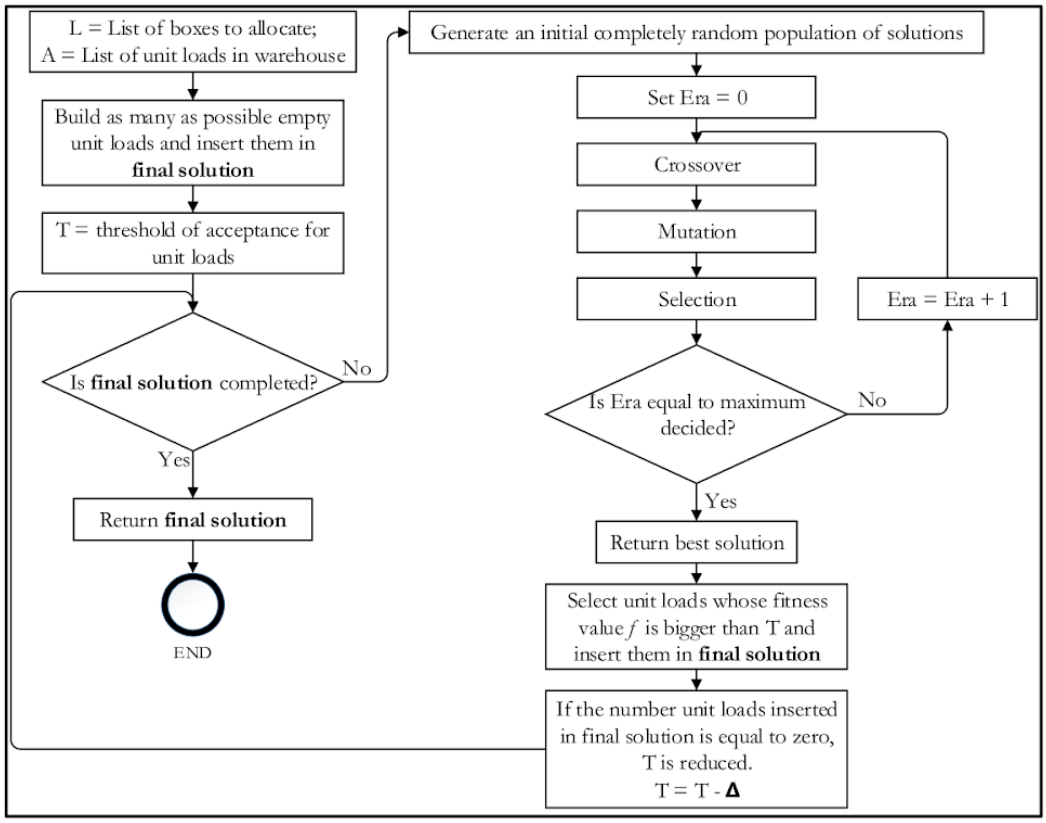


Figure 5.14. Procedure-Z (Bertolini et al., 2019).

5.4.4. Crossover, mutation, and selection

The crossover is a procedure, inspired by the crossing between chromosomes, according to which solutions in the current population (i.e. parents) generate new solutions (i.e. children), which will share partial common traits with their originating parents. In the proposed algorithm, the number of solutions generated by crossover (children) is equal to the number of solutions in the starting population (parents). The parents are coupled two by two and starting from them, two children are generated. Each solution of the starting population can

be coupled only once. Parents selection is made using a method known as roulette wheel selection adopted, for example, by Quyen et al. (2017). Roulette wheel selection assigns to each parent a probability to be chosen proportional to its fitness, which, given *G* the population size, is calculated by using Eq. 5.10. In this way, the best solutions have a higher probability to mate together.

$$probability = \frac{f}{\sum_{i=1}^{G} f_i} \tag{5.10}$$

Children generation is made using a procedure frequently adopted in literature to solve scheduling problems (Murata, Ishibuchi and Tanaka, 1996), which is represented in Figure 5.15. Two random numbers q_1 and q_2 are generated to define the cutting points. By convention, a parent is identified as the father, the other as the mother. The solution that represents the son will be identical to the father from its beginning until q_1 and from q_2 to the end. The elements in the middle will be inserted into the solution in the same order as they appear in the mother solution. Vice versa, it happens for the daughter.

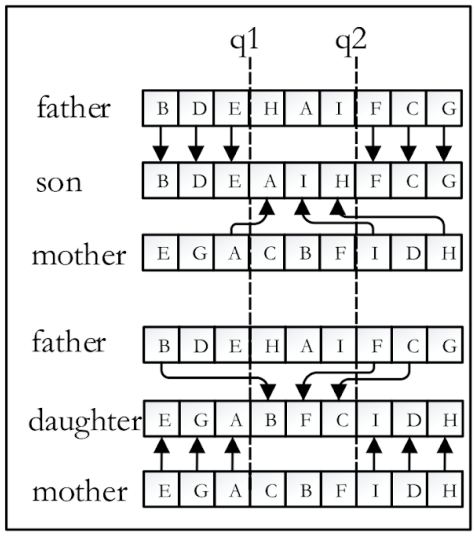


Figure 5.15. Crossover representation (Bertolini et al., 2019).

Selection of cutting points q_1 and q_2 is based on a triangular distribution. The triangular distributions used for q_1 and q_2 selection have mode respectively centred on 1/3 and 2/3 of

the solution length. The distribution used for q_1 goes from 0 to 2/3 of solution length and the distribution of q_2 goes from 1/3 of solution length till 3/3 of solution length. If q_2 results smaller than q_1, they are switched. In this way, each child is on average for 2/3 of its length equal to one of parents, to guarantee a neighbour search instead of a random generation of new different solutions. Probability distribution for cutting point choice is represented in Figure 5.16.

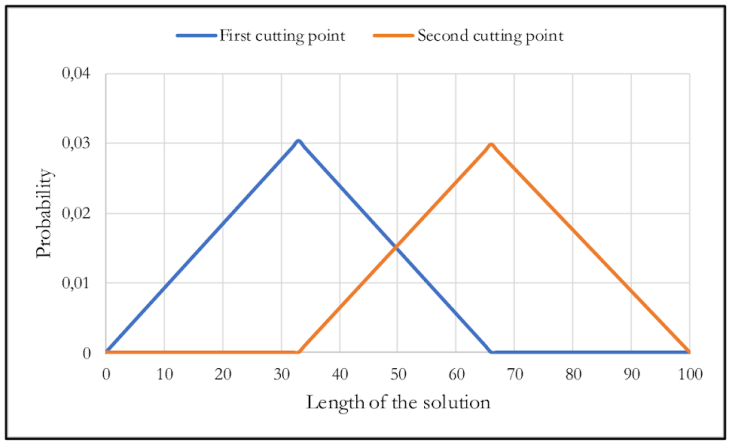


Figure 5.16. Probability distribution for deciding cutting points (Bertolini et al., 2019).

The mutation prevents the algorithm from being stuck on local optima (Gracia, Andrés and Gracia, 2013). It consists of a little modification of the new solution obtained by crossover. Each time a new solution is generated, a random number is generated too. If the random number is lower than the mutation probability p_M, a mutation takes place. As stated by Mayer et al. (1999), the mutation probability must be very low to prevent the procedure from turning into a random generation of solutions. In the proposed algorithm the mutation occurs by selecting two random load units of the solution according to uniform probability. Next, an exchange of two compartments between the two selected load units takes place. The selection of the compartment inside each box is done according to uniform probability, too. An example of mutation is given in Figure 5.17.

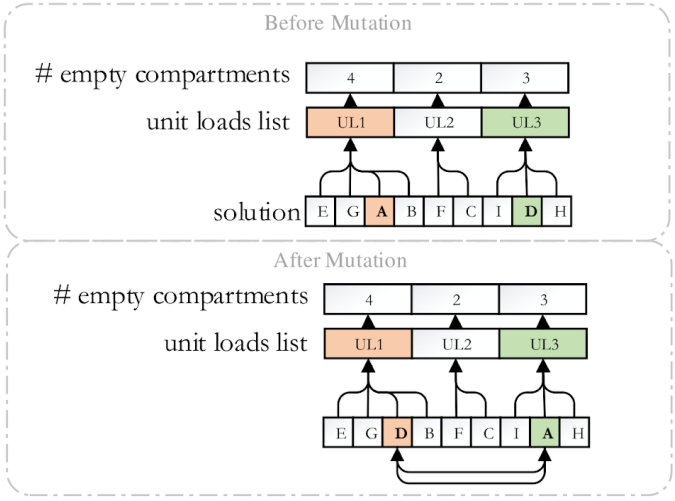


Figure 5.17. Mutation representation (Bertolini et al., 2019).

After the crossover and the mutation of the new-generated solutions, a selection process is carried out. First, the children solutions where the weight of at least one of the load units exceeds their capacity (not-acceptable solutions) are eliminated. Conversely, there is no control of clones (children identical to parents), indeed, during the tests, the number of clones generated was always zero or insignificant. For this reason, no procedure for modifying or eliminating clones is carried out. After purging not-acceptable solutions, the new population is built up. New population is made up of 50% of the best parents and 50% of the best children. This selection criteria guarantees, in case some discarded parents were better than some selected children, the acceptance of worse solutions, thus, it prevents the algorithm to stop on local optima. As mentioned above, it may happen that some of the children generated are eliminated as not acceptable. If the number of children is not enough to constitute the 50% of the new population, the ratio between parents and children in the new population will be unbalanced, but always ensuring that the number of solutions is equal to that of the previous population (i.e. *G*).

5.4.5. Case study

To validate the algorithm, it was implemented in Python 3.6, compiled using the CPython interpreter, and tested on a standard personal computer, Intel QuadCore i7 CPU at 2.4GHz with 4Gb RAM and Ubuntu 18.04 OS. Firstly, the two procedures have been compared to each other in case of equal conditions with the objective to find a criteria for automatically deciding the best procedure at each run of the algorithm. As shown in Figure 5.18, as the number of boxes to be allocated increases, the Procedure-Z provides better results. However, the computational time of Procedure-Z is longer: while the computational time of

Procedure-Y was ranging between 50 and 150 seconds depending on the number of boxes, the average computational time of Procedure-Z was always over 200 seconds. Because of this, it is convenient to use the Procedure-Y until it provides an acceptable solution. Empirically, it was observed that with a number of compartments to be filled equal to 96, Procedure-Y was still providing an acceptable result. In Figure 5.19 it is possible to observe the trend of the objective function when the Procedure-Y was tested in that specific case. Note that in the graphs in Figure 5.18 and Figure 5.19 the value 1 does not refer to the global optimum of the problem, but to a reference value specific for the case of perfect balance and with the maximum correlation inside each unit load.

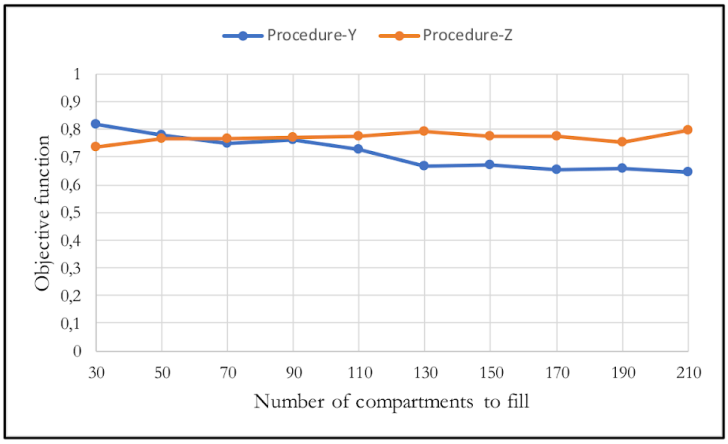


Figure 5.18. Procedures compared (Bertolini et al., 2019).

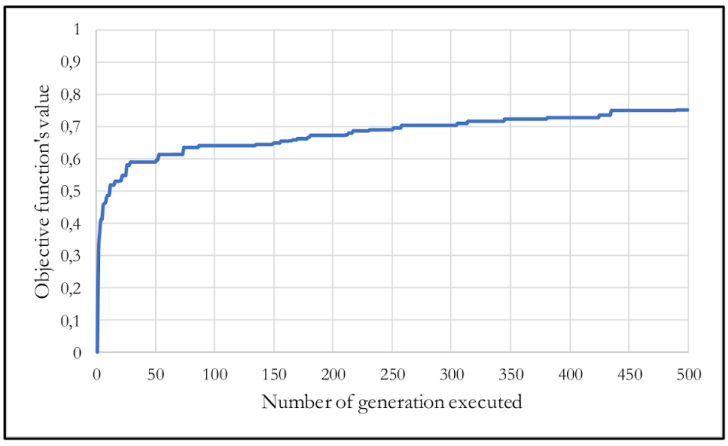


Figure 5.19. Objective function's trend using Procedure-Y for storage of 96 boxes (Bertolini et al., 2019).

For testing both procedures in a real case study, two different VLMs sold by a partner company were considered. For privacy reasons more information cannot be made explicit,

although, both the warehouses are VLM like that described at the beginning of this chapter. The delivery lift can deal with just a unit load at a time and there is no buffer to take the next unit load closer to the I/O point while the operator is picking. The company which owns the warehouses is operating in the steel sector and fills them on average every three weeks. Different periods of three weeks were therefore studied, comparing the results provided by the proposed algorithm with those taken from the Warehouse Management System (WMS). Each period was independently studied, i.e. before filling, the current situation of the warehouse in the simulation model was always made identical to the real warehouse. Moreover, it has not been possible to study consecutive periods. The comparison was made by simulating the same retrieving orders registered in the WMS, and by observing the balance of the unit loads and the number of retrieves during the period to fulfil the picking orders. The algorithm adopted for location assignment in retrieving phase is the following: (i) order lines are sorted for quantity required (to each order line corresponds a specific code); (ii) the first not fulfilled order line is selected; (iii) the unit load closest to I/O point which store that code is retrieved; (iv) all the order lines which can be fulfilled picking from the retrieved unit load are fulfilled; (v) mentioned steps are repeated until the order has been fulfilled. Concerning the VLMs used for testing, the smaller one had the capacity to stock 100 compartments (i.e. 24 unit loads, arranged on 13 levels, with 4 compartments per each), and it was used to test Procedure-Y. The other warehouse had a capacity of 360 compartments (i.e. 90 unit loads, arranged on 46 levels, with 4 compartments per each) and it was used to test the Procedure-Z. In the tests, empirical parameters α and β of objective function were set equal to 0.35 and 0.65 giving more importance to the correlation, while the mutation probability was set to 0.05.

Results concerning the first warehouse are reported in Table 5.3, while results concerning the second implementation case are reported in Table 5.4. In both tables, each row represents a different test, and, moving from left to right, in columns are reported: (i) the test number, (ii) the warehouse saturation before filling, (iii) number of new boxes to stock, (iv) average unbalance provided by currently implemented algorithm, (v) number of retrieves made to satisfy orders occurred before next filling due to allocation provided by currently implemented algorithm, (vi) average unbalance provided by algorithm proposed in this paper, (vii) number of retrieves made to satisfy orders occurred before next filling due to allocation provided by proposed algorithm. The average unbalance, even in the real case, was calculated using the Eq. 5.4. The results are satisfactory, although the allocation suggested by Procedure-Z sometimes requires more retrieves than those provided by the

program currently implemented in the real warehouse. However, the error can be due to the variability of the demand.

Table 5.3. Results for Procedure-Y (Bertolini et al., 2019).

Test	Warehouse's starting filling	Boxes to allocate	Average unbalance [kg] (real VLM)	#Retrieves (real VLM)	Average unbalance [kg] (proposed algorithm)	#Retrieves (proposed algorithm)
1	24%	75	~727	816	**211**	**766**
2	4%	96	~583	972	**169**	**927**
3	30%	51	~599	459	**152**	**473**
4	15%	79	~775	1153	**193**	**1084**
5	2%	96	~353	1142	**180**	**1007**
6	4%	91	~395	1047	**161**	**904**

Table 5.4. Results for Procedure-Z (Bertolini et al., 2019).

Test	Warehouse's starting filling	Boxes to allocate	Average unbalance [kg] (real VLM)	#Retrieves (real VLM)	Average unbalance [kg] (proposed algorithm)	#Retrieves (proposed algorithm)
7	10%	163	~572	1743	**136**	**1740**
8	18%	160	~571	1662	**149**	**1544**
9	23%	180	~631	1310	**142**	**1441**
10	24%	201	~696	1536	**203**	**1459**
11	35%	232	~607	1410	**198**	**1429**
12	48%	184	~636	1623	**127**	**1557**
13	29%	256	~682	1829	**97**	**1778**
14	2%	350	~731	2025	**148**	**2047**

5.5. Retrieving Improvement: Order Picking

5.5.1. Introduction

Since this chapter is dedicated to logistics of small parts, the work would be incomplete without talking about picking, a strategic aspect of internal logistics that is very discussed in scientific literature and in industry (Penteado and Chicarelli, 2016). Automation and picking are not mutually exclusive elements, although, the picking is an activity where the introduction of automated handling solutions is not yet efficient, and the 'automation' consists mainly of individual devices and softwares to support the human operator. Even if the fully automated handling solutions lead to important benefits, their implementation is limited to contexts where the items are handled in groups (usually inside unit loads) and not individually (Janssen et al., 2019), and their implementation is convenient when the handled quantities are big and the variety of items is small (Janssen et al., 2019). For this reason, in contexts characterised by small light-weight items and spare-parts-handling, manual warehouses still represent the standard solution, and manual picking strategies are the main element to be improved. In most distribution centres (in the steel sector as in many others) the classic picker-to-parts strategy is still the most widespread, because operations mainly consist in decomposition of Stock Keeping Units (SKU), short period storage, re-composition and shrink-wrapping of new SKU. Since, the pickers spend the most of their working time moving from a warehouse location to another, effective picking strategies may reduce the travel-time and increase productivity. Each warehouse is different, and the picking strategy must be chosen based on layout, items, number of resources, and equipment that the workers are utilizing. Among the most known picking strategies are batch picking, zone picking, and wave picking. Many other strategies, such as vision picking and voice picking, also found application thanks to innovative electronic solutions. However, all these strategies require particular conditions or equipment to be efficiently implemented. For instance, batch picking and wave picking need the possibility to sort the items after retrieving, zone picking is efficient in case of very diverse inventory, vision picking and voice picking need an headset and a mobile computer attachment. For these reasons, classic order picking, even though it guarantees less chance of improvement, is often the most sensible choice. Since, in case of order picking there is no batches composition problem, the main aspect to focus on is the routing of pickers. The scientific literature has been studying for a long time the routing strategies to reduce the travel-time for pickers under order picking conditions,

although, each time a new approach is proposed, it is often compared on equal terms with just one or two more solutions (Roodbergen and De Koster, 2001). Moreover, to the author's best knowledge, only few publications exhaustively study the impact of different layouts on the proposed approach. Because of this, during the design phase of a warehouse, it is hard to understand which is the most feasible approach. In this section, the aim is to partially fill this gap. Focusing on order picking, three heuristic routing strategies (i.e. S-Shape, Largest Gap, and Combined) are firstly compared each other, and then compared to a meta-heuristic algorithm owning to swarm intelligence family (i.e. Ant Colony Optimization). The choice of the ant colony is due to the fact that some publications sponsor it as the most effective in these contexts. Finally, a new solution is proposed by introducing a re-adapted Particle Swarm Optimization, and comparing it with previously introduced approaches. The Particle Swarm Optimization (PSO) is a relatively new metaheuristic, which, up to now, has always provided great benefits when applied to different contexts such as Neural Networks training (Suresh et al., 2015) and Bayesian Networks (Aouay et al., 2013). Its application to industry, logistics, and other contexts where a discrete version of the algorithm is needed is still premature, however, in the era of modern computing, the PSO is promisingly more suitable than other metaheuristics for both parallelization techniques: multiprocessing and multithreading (Malakhov et al., 2018). All the analysed approaches have been implemented using both Python© and Cython© programming languages, and then compared using the same layout, and observing their behaviour when the complexity of the problem and the storage assignment policy change. The layout considered in this case is that of a classic manual warehouse, although, with slight modifications all the proposed approaches could be implemented in other contexts, such as a battery of VLMs or a partially automated warehouse.

5.5.2. The heuristic approaches: S-Shape, Largest Gap, Combined

The S-Shape strategy (Marchet, 1994) is also known as 'traversal'. It defines a route in which the aisles that are to be visited, are totally traversed. Conversely, aisles where nothing has to be picked are skipped. The advantage of this strategy consists in its simplicity and easy implementation. In case of multiple blocks and cross-aisles, after traversing an aisle, to decide which aisle to run, only adjacent blocks are considered.

In the Largest Gap (LG) strategy, firstly introduced by Hall (1993), every time the picker enters an aisle, the distance between the current picking position and the next one is estimated, if it is bigger than the distance between the current position and the beginning of

the aisle, the picker go back to the beginning of the aisle, otherwise he/she goes to the next picking position. In case of multiple blocks and cross-aisles, as for the S-Shape strategy, every time a new picking point is to be defined, only adjacent blocks are considered.

The Combined strategy (Roodbergen and De Koster, 2001) introduces a sort of try-and-error evaluation to define the best route. Every time all picking positions in an aisle have been visited, the alternatives to go to the rear end of the aisle and to return to the front end are compared with each other. The solution resulting in the shortest path is chosen. As for the strategies described above, in case of multiple blocks, only the adjacent ones are considered.

5.5.3. Ant Colony Optimization (ACO)

The ACO is a meta-heuristic optimization technique inspired by ants behaviour. When an ant must choose a route instead of the other, he/she looks at the quantity of pheromone left by other members of the colony. A higher level of pheromone means a better route, usually because it is shorter if compared to the others. This curious behaviour inspired the creation of a probabilistic technique of operational research for solving computational problems, which can be formalised with a graph. The first version was proposed by Dorigo et al. (1996), and it was originally called Ant System. Then, over the years, several improvements and adjustments to different contexts were proposed (Bell and McMullen, 2004), and the ACO was rearranged to work with discrete problems and is now note to be one of the best performing algorithms for routing and Travel Sales Problem (TSP). The ACO execution consists of loops. At each loop *t*, a new ant is generated and the algorithm takes into account the set of *n* picking positions (*i = 1,…,n*) to be visited. The edge connecting two picking positions *i* and *j*, where *i ≠ j*, can be denoted by tuple *(i, j)*, and its length is d_{ij}.

Hence, the cost of a solution *D* may be calculated as $\sum_{i=1}^{n-1} d_{i,\,i+1}$.

Each ant provides a new solution by building it element by element. The new solution is then compared to the best solution found so far and, if its cost is lower than the best solution's cost, it is made the new best solution. Every time a new best solution is found, the pheromone on each edge is updated. Given τ_{ij} the pheromone on edge *(i, j)*, it is updated according to Eq. 5.11, where ρ and *Q* are parameters of the algorithm.

$$\tau_{i,j} = \begin{cases} \rho\tau_{ij} + \frac{Q}{d_{ij}} \text{ , if (i, j) } \in \text{best solution} \\ \\ \rho\tau_{ij} \text{ , otherwise} \end{cases} \quad (5.11)$$

While the ant is building a new solution, given *i* the last element of the current incomplete solution and *I* the set of picking positions not visited yet, the next position is selected by using a roulette wheel (if not confident with it check the description provided by Shtovba, 2005), where the probability to move to picking position *j*, namely p_{ij} is calculated as in Eq. 5.12.

$$p_{ij} = \frac{\tau_{ij}{}^{\alpha}\left(\frac{1}{d_{ij}}\right)^{\beta}}{\sum_{k}\tau_{ik}{}^{\alpha}\left(\frac{1}{d_{ik}}\right)^{\beta}} \text{ , if } j \in I \text{ (and 0 otherwise)} \quad (5.12)$$

5.5.4. Background on Particle Swarm Optimization (PSO)

The Particle Swarm Optimization (PSO) is a nature-inspired metaheuristic optimization technique recalling bird flocks' behavior. The first contribution to PSO is attributed to Kennedy and Eberhart (1995), and it is now classified as an evolutionary and population-based algorithm. The basic version of the PSO starts from a population of candidate solutions; this initial population is generally called swarm, while the solutions are usually called particles, as the name of the algorithm recalls. The swarm is therefore represented by a set of particles ($i = 1, ..., N$), and the movement of each particle i represents the exploration of a new solution. Each particle i moves in an *n*-dimensional space of feasible solutions *S*, where $S \subseteq R^n$. Each movement is made taking into account two main elements: *(i)* the global best solution found so far by the whole swarm (i.e. $gbest$), and *(ii)* the best solution found so far by the particle (i.e. $pbest_i$). In particular, given $p_i(t)$ the

current position of a generic particle i under the t iteration, the movement made by the particle is defined by the following Eq. 5.13.

$$p_i(t+1) = p_i(t) + v_i(t) \tag{5.13}$$

In equation (1), $v_i(t)$ is a parameter called *speed*, which is calculated at each iteration according to Eq. 5.14 below.

$$\begin{aligned} v_i(t) = v_i(t-1) \\ + rnd() \cdot C_1 \cdot (p_i(t-1) - pbest_i(t-1)) \\ + rnd() \cdot C_2 \cdot (p_i(t-1) - gbest\,(t-1)) \end{aligned} \tag{5.14}$$

In Eq. 5.14, $rnd()$ is a random generated number between 0 and 1, while C_1 and C_2 are parameters of the algorithm, which are often set equal to 2 in order to overtake the intention of the particle (i.e. $v_i(t-1)$), whose weight is implicitly equal to 1, about half the time. Basically, the concept behind this formula is that each particle tends to maintain its own direction, although it is affected by small and random deviations due to the best solution found by itself and the best solution found by the whole swarm.

The implementation of the algorithm is quick and immediate. At each iteration t, until stop criteria set at the beginning are met, each particle i of the swarm moves according to Eq. 5.13, eventually replace its personal best solution $pbest_i(t)$ in case it finds a better solution, and updates its current speed according to Eq. 5.14. Once all the swarm has moved, global best $gbest(t)$ is updated and the process is reiterated.

Up to now, several variants of the PSO have been proposed and some relevant changes have been brought to the basic version. One of the main contributions came from Shi and Eberhart (1998), who introduced a new parameter ranging between 0 and 1 called inertia (i.e. w). Thanks to this new term, Eq. 5.15 below replaces Eq. 5.14 for the speed updating.

$$\begin{aligned} v_i(t) = v_i(t-1) \cdot w \\ + rnd() \cdot C_1 \cdot (p_i(t-1) - pbest_i(t-1)) \end{aligned} \tag{5.15}$$

$$+ rnd() \cdot C_2 \cdot (p_i(t-1) - gbest\,(t-1))$$

The aim of inertia is to mitigate the effect of particles' intention. As a matter of facts, being w usually between 0 and 1, the impact of the speed in the previous iteration (i.e. $v_i(t-1)$) is reduced. In many applications, the inertia w decreases as the computed iterations increase. This behaviour probabilistically leads the particles to explore a wider space of solutions during the first iterations, and to finally focus on local optima when the procedure is ending.
Some years later, an alternative to inertia which increases the stability of the algorithm and improves the optimal solution seeking, was proposed by Clerc and Kennedy (2002), who named it constriction factor (i.e. χ). Again this time, Eq. 5.16 replaces the previous Eq. 5.15 for the speed update.

$$\begin{aligned} v_i(t) = \chi[&v_i(t-1) \\ &+ rnd() \cdot C_1 \cdot (p_i(t-1) - pbest_i(t-1)) \\ &+ rnd() \cdot C_2 \cdot (p_i(t-1) - gbest\,(t-1))] \end{aligned} \tag{5.16}$$

All the contributions described above are designed to work with continuous variables; thus, none of them is suitable to solve discrete or binary optimization problems. Since most real industrial and logistic applications must be formalized as a discrete problem, it is worth highlighting the first contributions towards the discretization of PSO, which are ascribed to Kennedy and Eberhart (1997), and Laskari et al. (2002). The first presented a binary version of PSO by converting continuous variables into binary ones by using the sigmoid function, while the second authors showed that PSO with slight modifications might be used to solve discrete optimization problems by working with continuous variables and rounding off the real optima to the nearest integer position.
Recently, the discrete PSO has gained interest from the scientific community, and many solutions to deal with discrete problems have been proposed. Focusing on TSP, it is important to highlight Wang et al. (2003), Shi et al. (2007) and Chen et al. (2009). For more implementations and additional information, the authors would like to suggest two comprehensive literature reviews proposed by Banks et al. (2007) and Banks et al. (2008), or a more recent publication by Garcia-Gonzalo and Fernandez-Martinez (2012).

5.5.5. The proposed Particle Swarm Optimization (PSO)

In the proposed version of PSO, at each iteration *t* each particle has (i) a current solution $current(t)$, (ii) an intention $pintention(t)$ constituted by a random solution, (iii) the greedy solution $pgreedy$, (iv) the personal best solution $pbest(t)$. Moreover it knows the global best solution found so far $gbest(t)$. To each of the solutions that the particle knows, except for the current solution, a weight is assigned (i.e. w_1 for the $pintention(t)$, w_2 for the $pgreedy$, w_3 for the $pbest(t)$, and w_4 for the $gbest(t)$). During each iteration, each particle explores a new solution, then all particles share with others their $pbest(t)$. If a $pbest(t)$ is better than the $gbest(t)$, it is made the new global best. In order to better understand the proposed neighbour search, the authors would like to induce the following notation. Given *i* a generic picking position and *t* the current iteration, the authors define:

- $(i,j_1)_t$ the edge which connect *i* to the next picking position in the $pintention(t)$ in iteration *t*;
- (i,j_2) the edge which connects *i* to the next picking position in the $pgreedy$;
- $(i,j_3)_t$ the edge which connects *i* to the next picking position in the $pbest(t)$ in iteration *t*;
- $(i,j_4)_t$ the edge which connects *i* to the next picking position in the $gbest(t)$ in iteration *t*.

Subsequently, are defined $d_1(t)$ the length of $(i,j_1)_t$, d_2 the length of (i,j_2), $d_3(t)$ the length of $(i,j_3)_t$, and $d_4(t)$ the length of $(i,j_4)_t$.

Hence, in each iteration *t*, each particle explores a new solution by changing its current solution $current(t)$. The new solution $current(t+1)$ is built position after position, beginning from *i=1* and going on until its completion. In particular, the next picking position is selected between j_1, j_2, j_3, and j_4. The selection is made by using a roulette wheel, where the probability to choose the next picking position depends on its distance from *i*: bigger is the distance, lower is the probability. More in detail, given *t* the current iteration and $m \in \{1,2,3,4\}$, the probability p_m to choose j_m the next node is calculated as in Eq. 5.17.

$$p_m = \begin{cases} \frac{k_m w_m e^{T/d_m}}{\sum_{s=1}^{4} w_s e^{T/d_s}} \text{ , if } d_m > 0 \\ k_m \text{ , if } d_m = 0 \end{cases} \quad (5.17)$$

where $T = \sum_{s=1}^{4} d_s$ and k_m is a binary parameter which defines the possibility to choose j_m as the next picking position. Thus, given *I* the set of picking positions not included in the new current solution yet, $k_m = 1$ if $(i, j_m)_t$ exists (i.e. *i* is not the last picking position of the solution observed), and $j_m \in I$. If after calculating the probabilities it comes that the roulette wheel is not possible because $p_1 = p_2 = p_3 = p_4 = 0$, the next picking position is randomly selected from the set *I*. After defining the new current solution, according to Zhong, Zhang and Chen (2007) a mutation is carried out with low probability. The mutation is made by using a single iteration of the well-known 2-opt algorithm.

5.5.6. Setting of parameters

Concerning the ACO, since a design of experiments was already carried out in a recent publication concerning a similar problem (De Santis et al., 2018), the set of parameters defined in that publication is used. Conversely, concerning the PSO, an empirical design was carried out to find out the best combination. The algorithm was tested on 40 randomly generated picking lists made of 20 picking locations. The algorithm was iterated 10 times per each picking list, and the combination of parameters which was providing the best average result on most of the picking lists was selected. The layout considered is the same adopted in the case study. The final optimal set of parameters is the following:

- number of particles = 40;
- w_1 = w_3 = w_4 = 1;
- w_2 = 0.1;
- mutation probability = 0.1.

5.5.7. Setting of parameters

The comparisons were made doing following assumptions: *(i)* no capacity limit for pickers is considered; *(ii)* every picking list is associated to one and only one order; *(iii)* the improvement strategies are executed singularly on each picking list; *(iv)* possible physical obstructions between pickers are not considered; *(v)* activities to refill the storage locations are not considered; *(vi)* when a picker has visited all the locations associated to a picking list, before taking care of the next one, he/she must go back to the I/O point, thus, given *(i = 1,…,n)* the set of picking positions to visit, it is always true that *1* = *n* = I/O.

5.5.8. Layout

In the selection of the layout, a real industrial case was observed. The considered warehouse is made of 2 blocks divided by a cross-aisle. In each block there are 7 aisles, with 11 storage locations on each side and 2 aisles at the two opposite ends with 11 storage locations only on one side. Given *u* the distance unit, each storage location is *2u* deep and *2u* large, aisles are *4u* large, and the cross aisle is *4u* large as well. The Input/Output (I/O) point is in the bottom left corner. A representation of the warehouse is provided in Figure 5.20. To translate the warehouse layout into a distance matrix reporting the minimum path between locations, the well-known Floyd-Warshall (FW) algorithm was used. FW algorithm is an exhaustive procedure, which compares each possible path between two given locations (or nodes), in order to find the minimum. Of course, before running FW, additional nodes were appended where the aisles cross each other, otherwise FW would have not worked (as explained by Pansart et al. (2018)).

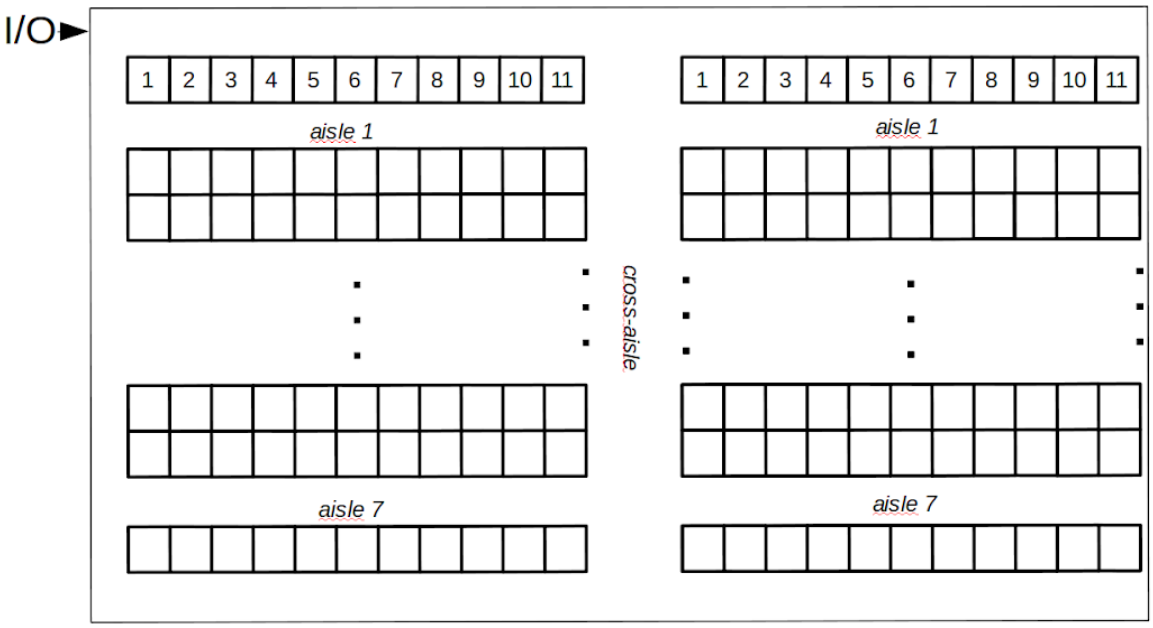


Figure 5.20. Layout considered (Bertolini et al., 2020).

5.5.9. Comparison and results

The 5 investigated procedures were compared for 3 different complexities of the problem (i.e. picking list of 10 items, picking list of 20 items and picking list of 30 items) and 2 different location assignment policies (i.e. random and class-based). The author is aware that a longer picking list increases the number of feasible solutions, but not necessarily entails a greater complexity of the problem. To define the complexity of the problem, the number of aisles and the arrival frequency of orders have to be considered, too. However, in the proposed case study, the layout used is always the same, each order is singularly considered, and the picking lists are randomly generated, but always controlling at the same time that the locations required were not to be closed and not all the aisles have to be visited. These assumptions avoid falling into borderline cases, where the length of the picking list would not have any impact on the complexity of the problem. For all these reasons, the length of the picking list can be considered a sufficiently accurate index of complexity of the problem.

In Table 5.5 and 5.6 the comparison of the 5 procedures is carried out respectively in case of random and class-based location assignment policy. In this case each picking list is made of 10 items. For each case, 10 picking lists were used, and the length of the route provided by each algorithm for each list is reported in the tables. Concerning the metaheuristics, both were tested 10 times on each picking list, and in the table the average result and the

standard deviation are reported. Observing the results is clear that both metaheuristics always outperformed the other procedures, and both are characterized by great accuracy, since the standard deviation is null. By observing the S-Shape, Largest Gap and Combined policies, some interesting considerations can be made. As it is clearly visible in Figure 5.21, in case of random location assignment, the combined method outperforms the others in most tests. Conversely, in case of class-based location assignment, as represented in Figure 5.22, the results lead to a different consideration. Since a class-based assignment policy leads by itself to a shorter route, the difference between the analysed routing policies is less evident. However, in a relevant number of tests the largest gap policy outperformed the combined one, and, even the S-Shape is providing results closer to those that the Combined is able to guarantee. Because of this, and due to relatively easier implementation of the S-Shape and Largest Gap policies, they may be considered more convenient than the combined policy.

Table 5.5. Comparison in case of random allocation policy for picking lists of 10 positions.

Picking list	ACO		PSO		S-Shape	LG	Combined
	Avg.	Dev.St	Avg.	Dev.St			
1	236	0	236	0	324	304	272
2	160	0	160	0	270	198	208
3	220	0	216	0	324	296	264
4	200	0	200	0	368	346	232
5	196	0	196	0	256	280	222
6	192	0	192	0	358	236	234
7	212	0	212	0	348	274	248
8	196	0	196	0	276	276	230
9	200	0	200	0	324	304	210
10	172	0	172	0	256	240	210

Table 5.6. Comparison in case of class-based allocation policy for picking lists of 10 positions.

Picking list	ACO		PSO		S-Shape	LG	Combined
	Avg.	Dev.St	Avg.	Dev.St			
1	104	0	104	0	114	112	112
2	152	0	152	0	240	178	196
3	160	0	160	0	180	212	174
4	168	0	168	0	258	222	228
5	144	0	144	0	246	170	285
6	180	0	180	0	248	210	208
7	200	0	200	0	324	230	318
8	164	0	164	0	256	218	180
9	200	0	200	0	256	240	212
10	180	0	180	0	264	220	208

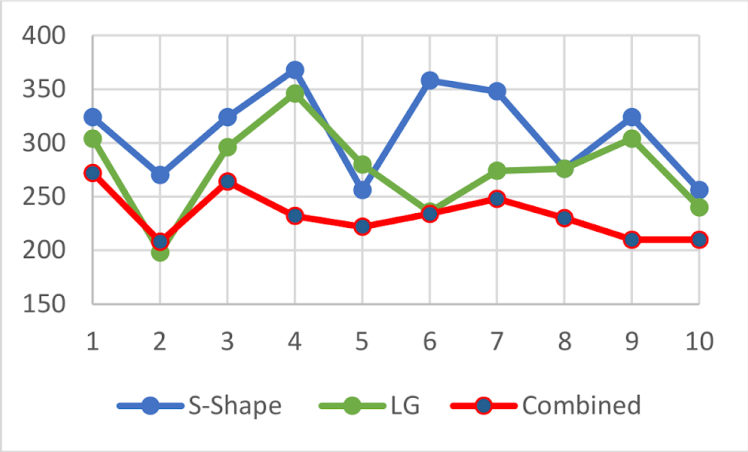


Figure 5.21. Comparison of heuristics under random assignment (Bertolini et al., 2020).

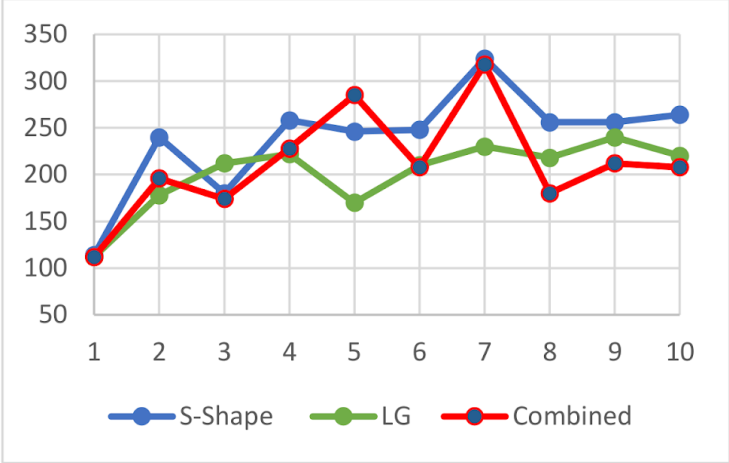


Figure 5.22. Comparison of heuristics under class-based assignment (Bertolini et al., 2020).

In view of provided results, metaheuristic approaches are still the most efficient and effective. Both outperformed the other routing strategies in 100% of tests carried out, and both had an approximate accuracy of 99% returning the same solution in all the 10 executions made per each picking list. For the purpose of this work it is not relevant, although, given the results and the accuracy, it could be possible to check if the metaheuristics are finding the best solution. In order to provide a further analysis, the heuristic procedures are therefore abandoned, and the metaheuristics are compared for increasing complexity of the problem. At first, the length of the picking list was increased to 20 elements, and the results are reported in Table 5.7 and 5.8. Lastly, the length of the picking list was increased again till 30 items, and the results for random and class-based assignment policies are respectively reported in Table 5.9 and 5.10. For each case described by a specific length of the picking lists and a specific location assignment policy,

both algorithms were tested on 20 different picking lists. For each picking list, both algorithms were executed 10 times, hence, in tables are reported the average length of the best route found (i.e. Avg.), the standard deviation (i.e. Dev.St.), and the well-known coefficient of variation defined as the standard deviation divided by the average. Finally, the numerical comparison is computed in the last 2 columns of each table.

Table 5.7. Results for picking lists of 20 items and random assignment policy.

Test	ACO			PSO			Δ Avg.	Δ Coeff. Var.
	Avg.	Dev.St	Coeff. Var	Avg.	Dev.St	Coeff. Var		
1	286	4	1,25%	284	0	0,00%	-0,56%	-1,25%
2	295	4	1,48%	295	4	1,48%	0,00%	0,00%
3	248	0	0,00%	248	0	0,00%	0,00%	0,00%
4	282	4	1,27%	285	3	1,18%	1,12%	-0,10%
5	310	5	1,73%	317	9	2,74%	2,02%	1,01%
6	260	0	0,00%	262	4	1,37%	0,61%	1,37%
7	338	2	0,65%	333	2	0,54%	-1,42%	-0,11%
8	296	0	0,00%	298	2	0,74%	0,54%	0,74%
9	212	0	0,00%	212	0	0,00%	0,00%	0,00%
10	290	2	0,75%	290	2	0,75%	0,00%	0,00%
11	380	0	0,00%	382	4	0,94%	0,42%	0,94%
12	300	0	0,00%	300	0	0,00%	0,00%	0,00%
13	308	0	0,00%	311	4	1,41%	1,03%	1,41%
14	286	4	1,25%	286	4	1,25%	0,00%	0,00%
15	248	0	0,00%	256	6	2,21%	3,13%	2,21%
16	264	0	0,00%	264	0	0,00%	0,00%	0,00%
17	260	0	0,00%	263	4	1,66%	1,22%	1,66%
18	260	0	0,00%	260	0	0,00%	0,00%	0,00%
19	306	2	0,72%	304	0	0,00%	-0,78%	-0,72%
20	268	0	0,00%	271	5	1,92%	1,18%	1,92%

Table 5.8. Results for picking lists of 20 items and class-based assignment policy.

Test	ACO			PSO			Δ Avg.	Δ Coeff. Var.
	Avg.	Dev.St	Coeff. Var	Avg.	Dev.St	Coeff. Var		
21	252	0	0,00%	252	0	0,00%	0,00%	0,00%
22	244	0	0,00%	244	0	0,00%	0,00%	0,00%
23	312	0	0,00%	315	5	1,65%	1,02%	1,65%
24	304	0	0,00%	306	2	0,72%	0,52%	0,72%
25	304	0	0,00%	308	7	2,25%	1,30%	2,25%
26	288	0	0,00%	283	4	1,55%	-1,67%	1,55%
27	287	2	0,62%	287	2	0,62%	0,00%	0,00%
28	303	2	0,59%	308	3	0,92%	1,56%	0,33%
29	279	2	0,64%	284	7	2,63%	1,69%	1,99%
30	248	0	0,00%	250	4	1,43%	0,64%	1,43%
31	260	0	0,00%	260	0	0,00%	0,00%	0,00%
32	316	0	0,00%	316	0	0,00%	0,00%	0,00%
33	236	0	0,00%	237	2	0,76%	0,34%	0,76%
34	276	0	0,00%	276	0	0,00%	0,00%	0,00%
35	300	0	0,00%	307	7	2,14%	2,34%	2,14%
36	320	0	0,00%	320	0	0,00%	0,00%	0,00%
37	288	0	0,00%	289	2	0,62%	0,28%	0,62%
38	305	2	0,59%	305	2	0,59%	0,00%	0,00%
39	289	3	1,16%	290	4	1,23%	0,55%	0,07%
40	308	0	0,00%	313	7	2,10%	1,53%	2,10%

Table 5.9. Results for picking lists of 30 items and random assignment policy.

Test	ACO			PSO			Δ Avg.	Δ Coeff. Var.
	Avg.	Dev.St	Coeff. Var	Avg.	Dev.St	Coeff. Var		
1	383	6	1,55%	386	2	0,57%	0,83%	-0,98%
2	342	4	1,04%	346	4	1,03%	1,15%	-0,01%
3	321	2	0,56%	329	7	2,00%	2,43%	1,44%
4	377	3	0,89%	380	6	1,49%	0,84%	0,60%
5	315	2	0,57%	315	5	1,65%	0,00%	1,09%
6	316	0	0,00%	329	16	4,74%	3,89%	4,74%
7	358	4	1,00%	362	7	2,01%	0,88%	1,01%
8	349	2	0,51%	354	6	1,71%	1,58%	1,20%
9	348	0	0,00%	357	6	1,66%	2,47%	1,66%
10	349	5	1,50%	354	7	2,05%	1,58%	0,56%
11	322	4	1,11%	328	7	2,28%	1,71%	1,17%
12	393	2	0,46%	393	5	1,33%	0,00%	0,87%
13	308	0	0,00%	308	0	0,00%	0,00%	0,00%
14	398	5	1,35%	407	8	1,89%	2,16%	0,54%
15	310	7	2,16%	313	9	2,92%	0,77%	0,76%
16	344	6	1,64%	345	11	3,11%	0,23%	1,47%
17	356	0	0,00%	369	9	2,47%	3,47%	2,47%
18	332	0	0,00%	341	4	1,29%	2,58%	1,29%
19	360	0	0,00%	368	6	1,72%	2,17%	1,72%
20	390	7	1,71%	390	9	2,36%	-0,20%	0,65%

Table 5.10. Results for picking lists of 30 items and class-based assignment policy.

Test	ACO			PSO			Δ Avg.	Δ Coeff. Var.
	Avg.	Dev.St	Coeff. Var	Avg.	Dev.St	Coeff. Var		
21	180	0	0,00%	187	7	3,51%	3,85%	3,51%
22	289	2	0,62%	301	7	2,19%	3,99%	1,57%
23	326	2	0,67%	329	11	3,38%	0,97%	2,70%
24	277	2	0,65%	279	2	0,64%	0,86%	-0,01%
25	300	5	1,63%	300	6	1,89%	0,00%	0,25%
26	317	2	0,56%	315	2	0,57%	-0,51%	0,00%
27	293	2	0,61%	302	12	3,82%	2,92%	3,21%
28	252	0	0,00%	258	5	2,08%	2,48%	2,08%
29	252	6	2,24%	252	9	3,55%	0,00%	1,30%
30	308	0	0,00%	322	13	4,09%	4,23%	4,09%
31	258	5	1,77%	258	6	2,35%	0,00%	0,58%
32	300	0	0,00%	306	10	3,14%	2,09%	3,14%
33	288	0	0,00%	298	2	0,73%	3,49%	0,73%
34	288	0	0,00%	296	8	2,87%	2,70%	2,87%
35	238	2	0,92%	242	2	0,91%	1,66%	-0,02%
36	281	2	0,64%	288	10	3,40%	2,50%	2,77%
37	323	7	2,03%	327	17	5,29%	1,22%	3,25%
38	312	3	0,91%	314	13	4,08%	0,76%	3,18%
39	261	2	0,69%	264	4	1,52%	1,21%	0,83%

40	264	0	0,00%	266	4	1,35%	0,60%	1,35%

The aim is to compare two main characteristics: the validity of the route provided and the accuracy of the algorithm. Concerning the accuracy, the author is aware that a bigger accuracy is not a good indicator of the validity of a metaheuristic, since the computational time should be analysed as well. For instance, if an algorithm had a great accuracy and, under equal conditions, it provided every time the same solution; if its computational time is too long, a less accurate but faster algorithm would be preferable. In the same time the first algorithm is executed once, the second one might be iterated more times, exploring different local optimums to choose the best in the end. However, since the compared algorithms are very similar in terms of computational time, only the accuracy is observed. Moreover, both ACO and PSO, are slower if compared to other metaheuristics, thus, a big accuracy can be considered a symptom of quality.

It is also important to point out that, unlike the ACO, the PSO is very feasible for parallel computing, which is a big advantage in 2020. Although the author is not going into it in this work, which is more for practitioners.

Thus, given the objective to compare validity and accuracy of ACO and the proposed PSO, in the last two columns in Tables 5.7, 5.8, 5.9, 5.10, are computed the difference in percentage between the average result and the variance coefficient for each picking list.

Given Avg_{ACO} the average result of the ACO and Avg_{PSO} the average result of the PSO, the percentage difference *ΔAvg.* is computed as in Eq. 5.18.

$$\Delta Avg = \frac{Avg_{PSO} - Avg_{ACO}}{max\{Avg_{PSO}, Avg_{ACO)}} \tag{5.18}$$

Hence, a negative value means the PSO provided better results than the ACO, otherwise, the ACO worked better than the PSO. Analysing the results, no relevant influence of the location assignment policies was detected. Of course, in case of class-based assignment policy the travelled distance is usually shorter, but that is because of the allocation logic, and not because of the routing algorithms. The difference between the two algorithms is minimal. On average the ACO is returning a solution 0.47% better than the PSO in case of picking lists made of 20 items, and 1.59% better in case of picking lists made of 30 items. Thus, even if the difference is minimal, it is possible to highlight a deterioration of the PSO, when

the length of the picking list increases. The same trend may be detected looking at the accuracy, and it is intuitively represented in Figure 5.23. As the length of the picking list increases (i.e. the space of feasible solutions increases), by looking at the variation coefficient, it is possible to see how the accuracy of the PSO decreases faster than in case of the ACO.

Figure 5.23. Effect of the length of the picking list on the accuracy of the metaheuristics (Bertolini et al., 2020).

References

Aouay, S., Jamoussi S., and Y. B. Ayed (2013). Particle swarm optimization based method for Bayesian Network structure learning. *5th International Conference on Modeling, Simulation and Applied Optimization*, pp. 1-6.

Atmaca, E. and Ozturk, A. (2013). Defining order picking policy : A storage assignment model and a simulated annealing solution in AS / RS systems. *Applied Mathematical Modelling*. 37(7), 5069–5079.

Banks, A., Vincent, J., & Anyakoha, C. (2007). A review of particle swarm optimization. Part I: background and development. *Natural Computing*, 6(4), 467-484.

Banks, A., Vincent, J., & Anyakoha, C. (2008). A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing*, 7(1), 109-124.

Berkey, J. O., & Wang, P. Y. (1987). Two-Dimensional Finite Bin-Packing Algorithms. *Journal of the Operational Research Society*, 38(5), 423–429.

Bertolini, M., Neroni, M., Romagnoli, G. (2018). A new heuristic algorithm to improve the design of a vertical storage system. *23rd Summer School" Francesco Turco"-Industrial Systems Engineering*.

Bertolini, M., Mezzogori, D., Neroni, M. (2019). Allocation of items considering unit loads balancing and joint retrieving. *24th Summer School Francesco Turco*.

Bertolini, M., Melloni, R., Neroni, M. (2020) Order picking: a comparison of heuristic and metaheuristic approaches. *25th Summer School" Francesco Turco"-Industrial Systems Engineering*.

Bozer, Y.A. and White, J. A. (1996). A generalized design and performance analysis models for end-of-aisle order-picking systems. *IIE Transactions.* 28(4), 271–280.

Chen, W. N., Zhang, J., Chung, H. S., Zhong, W. L., Wu, W. G., and Shi, Y. H. (2009). A novel set-based particle swarm optimization method for discrete optimization problems. *IEEE Transactions on evolutionary computation*, 14(2), 278-300.

De Koster, R. B. M., Le-Duc, T., and Yugang, Y. (2008). Optimal storage rack design for a 3-dimensional compact AS/RS. *International Journal of Production Research*, 46(6), 1495–1514.

De Santis, R., Montanari, R., Vignali, G., and E. Bottani (2018). An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses. *European Journal of Operational Research*. 267, 120-137.

Dorigo, M., Maniezzo, V., and A. Colorni (1996). Ant System: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Men, and Cybernetics*. 26(1), 29-41.

Egeblad, J. (2009). Placement of two- and three-dimensional irregular shapes for inertia moment and balance. *International Transactions in Operational Research*, 16(6), 789–807.

Erkip, N., Hausman, W. H. and Nahmias, S. (1990). Optimal decentralized ordering policies in multi-echelon inventory systems with correlated demand. *Management Science*, 36(3), pp. 381–393.

Garcia-Gonzalo, E., & Fernandez-Martinez, J. L. (2012). A brief historical review of particle swarm optimization (PSO). *Journal of Bioinformatics and Intelligent Control*, 1(1), 3-16.

Gracia, C., Andrés, C. and Gracia, L. (2013). A hybrid approach based on genetic algorithms to solve the problem of cutting structural beams in a metalwork company. *Journal of Heuristics.* 19(2), 253–273.

Hall, R. W. (1993). Distance approximations for routing manual pickers in a warehouse. *IIE Transactions*. 20(1), 53-62.

Janssen, C.P., Donker, S.F., Brumby, D.P., Kun, A.L. (2019). History and future of human-automation interaction. *International Journal of Human-Computer Studies*. 131, 99-107.

Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks* (pp. 1942–1948).

Kennedy, J., and Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. *Computational cybernetics and simulation.* 5, 4104-4108.

Laskari, E. C., Parsopoulos, K. E., & Vrahatis, M. N. (2002). Particle swarm optimization for integer programming. *In Proceedings of the 2002 Congress on Evolutionary Computation*. (Vol. 2, pp. 1582-1587).

Ma, N., and Zhou, Z. (2017). Mixed-Integer Programming Model for Two-Dimensional Non-Guillotine Bin Packing Problem with Free Rotation. *2017 4th International Conference on Information Science and Control Engineering (ICISCE)*, pp. 456–460.

Malakhov, A., Liu, D., Gorshkov, A., and Wilmarth, T. (2018). Composable Multi-Threading and Multi-Processing for Numeric Libraries. *Scipy Conference.*

Manzini, R. (2006). Correlated storage assignment in an order picking system. *International Journal of Industrial Engineering*. 13(4), pp. 384–394.

Marchet, G. (1994). Confronto tra diverse tipologie di percorsi nei sistemi di picking manuali. *Logistica Management*.

Mayer, D. G. et al. (1999). Survival of the fittest: genetic algorithms versus evolution strategies in the optimization of systems models. *Agricultural Systems*, 60, 113–122.

Murata, Ishibuchi and Tanaka (1996). Multi-objective genetic algorithm and its applications to flowshop scheduling. *Computers Industrial Engineering*. 30(4), 957–968.

Okano, H. (2002). A scanline-based algorithm for the 2D free-form bin packing problem. Journal of the Operations Research Society of Japan, 45(2), 145-161.

Pansart, L., Catusse, N., and H. Cambazard. (2018). Exact algorithms for the order picking problem. *Computers and Operations Research*. 100, 117-137.

Penteado M., M. and Chicarelli A., R. (2016). Logistics activities in supply chain business process. *The International Journal of Logistics Management*, 27(1), 6-30.

Quyen, N. T. P., Chen, J. C. and Yang, C.-L. (2017). Hybrid genetic algorithm to solve resource constrained assembly line balancing problem in footwear manufacturing. *Soft Computing*. 21(21), 6279–6295.

Roodbergen, K. J. and R. De Koster (2001). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*. 133(1), 32-43.

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. *IEEE world congress on computational intelligence* (pp. 69-73).

Shi, X. H., Liang, Y. C., Lee, H. P., Lu, C., and Wang, Q. X. (2007). Particle swarm optimization-based algorithms for TSP and generalized TSP. *Information processing letters*, 103(5), 169-176.

Shtovba, S. (2005). Ant algorithms: Theory and applications. *Programming and Computer Software*. 31(4), 167-178.

Suresh, A., Harish K.V., and N. Radhika (2015). Particle Swarm Optimization over Back Propagation Neural Network for Length of Stay Prediction. *Procedia Computer Science*. 46, 268-275

Xiao and Zheng (2008). Storage location assignment in a multi-aisle warehouse considering demand correlations. J*isuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems*, 2447–2451.

Wang, K. P., Huang, L., Zhou, C. G., and Pang, W. (2003). Particle swarm optimization for traveling salesman problem. *In Proceedings of the 2003 international conference on machine learning and cybernetics* (Vol. 3, pp. 1583-1585).

Zhang, Wang and Pan (2019). New model of the storage location assignment problem considering demand correlation pattern. *Computers and Industrial Engineering*, 129, 210–219.

Zhong, W., Zhang J., and W. Chen (2007). A novel discrete particle swarm optimization to solve traveling salesman problem. *IEEE Congress on Evolutionary Computation*.

6. Logistics of bulky parts

This chapter is focused on the logistics of big bulky products, which, in the steel sector consist of long metal bars bundles, billets, tubes bundles, etc (Figure 6.1). They are from 3 to 12 meters long and from 800 to 3000 kilogram heavy depending on the product (i.e. the billets are generally heavier then the bars bundles and the bars bundles are generally heavier than the tubes). These products are stored without using any unit load and they are simply placed on shelves. Their handling involves many complications, mainly due to the impossibility to do picking and to their weight. Moreover, such long items are subject to considerable bending, with consequent handling and safety risks.

Additional complications are related to the frequent mismatch between what the customers ask and what is available in stock. The customers usually ask for a quantity usually expressed in kilograms or any weight unit, and a quality level. Both requirements must be met during the retrieving taking into account that the picking is not possible and bundles cannot be unpacked.

In this chapter, innovative solutions to improve the handling and warehousing of these products are proposed.

Figure 6.1. Bulky products adopted in the steel sector.

6.1. Storage solutions: Shuttle Lift Crane AS/RS

6.1.1. Layout and machines

A Shuttle-Lift-Crane Automated Storage/Retrieval System (SLC-AS/RS) is a widespread solution for storage of bulky products in the steel industry. It is a fully automated solution expressly designed to stock bundles of long metal bars and billets. A typical layout is shown in Figure 6.2, which displays the planar scheme of a SLC-AS/RS.

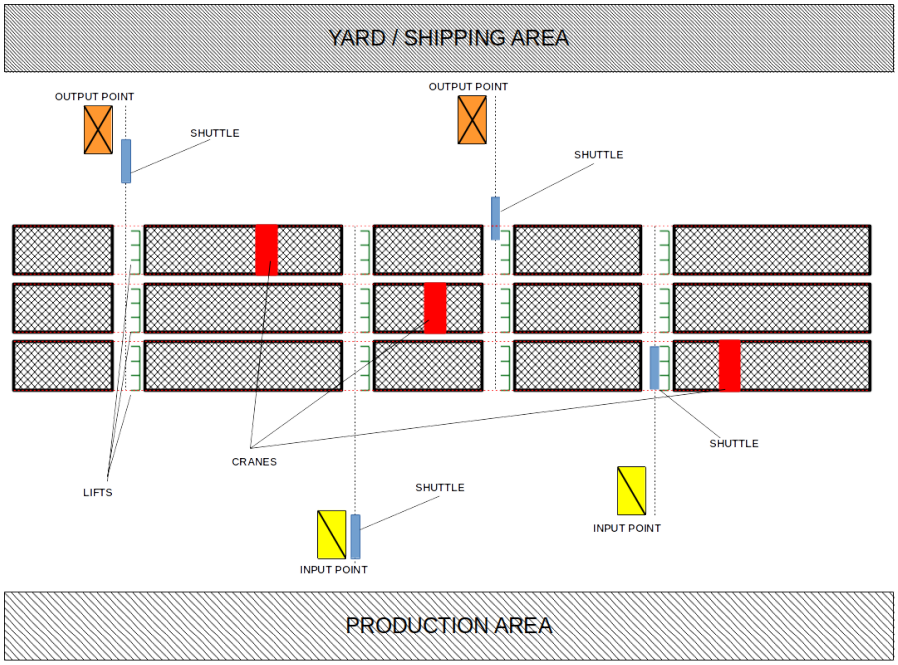


Figure 6.2. Planar scheme of a SLC-AS/RS [4].

Before proceeding with a detailed description of the SLC-AS/RS, it is important to highlight some aspects concerning the terms used and the names assigned to the machines involved. This typology of AS/RS is nowadays rarely studied and it is difficult to find correspondences in literature. The machines involved might be grouped into four categories named respectively Input/Output (I/O) points, Shuttles, Lifts, and Cranes. Even though the names used may seem familiar to scientists and practitioners of the field, the machines are very different from any other machine usually installed in other typologies of AS/RS. Indeed, shuttles and lifts are very different from classic shuttles and lifts used in SB-AS/RS, and, similarly, the crane is not the classic S/R machine which moves simultaneously in vertical and horizontal direction, but, it is in some ways more like a bridge crane. The naming has been made observing the machines' scope and not their design or technical features. In fact, in classic AS/RS, the crane is typically the machine which moves the unit loads from storage positions to an input/output point and vice versa, hence, in this chapter, the "crane" is the machine with the same purpose. Similarly, in SB-AS/RS, the shuttle and the lift are typically

the machines which only perform horizontal and vertical movements, hence, in this chapter, “shuttle” and “lift” are the machines with the same purposes.

Clarified the terminological aspects, a description of the warehouse is now provided. As shown in Figure 6.2, the overall stocking area can be split into separated units called racks (for instance the configuration of Figure 6.2 has three racks), bearing metal structures (made of columns and shelves) where bundles are stored. Also, each rack has several aisles, with metal shelves on both sides, arranged perpendicularly to the rack length. The bundles are stored on these shelves, and, depending on the width of the rack and the length of the bundles, they can be stored one beside the other in the same storage location.

Bundles are handled by automated machines mentioned above (i.e. I/O points, Shuttles, Lifts, Cranes), which are represented in Figure 6.3 and described below.

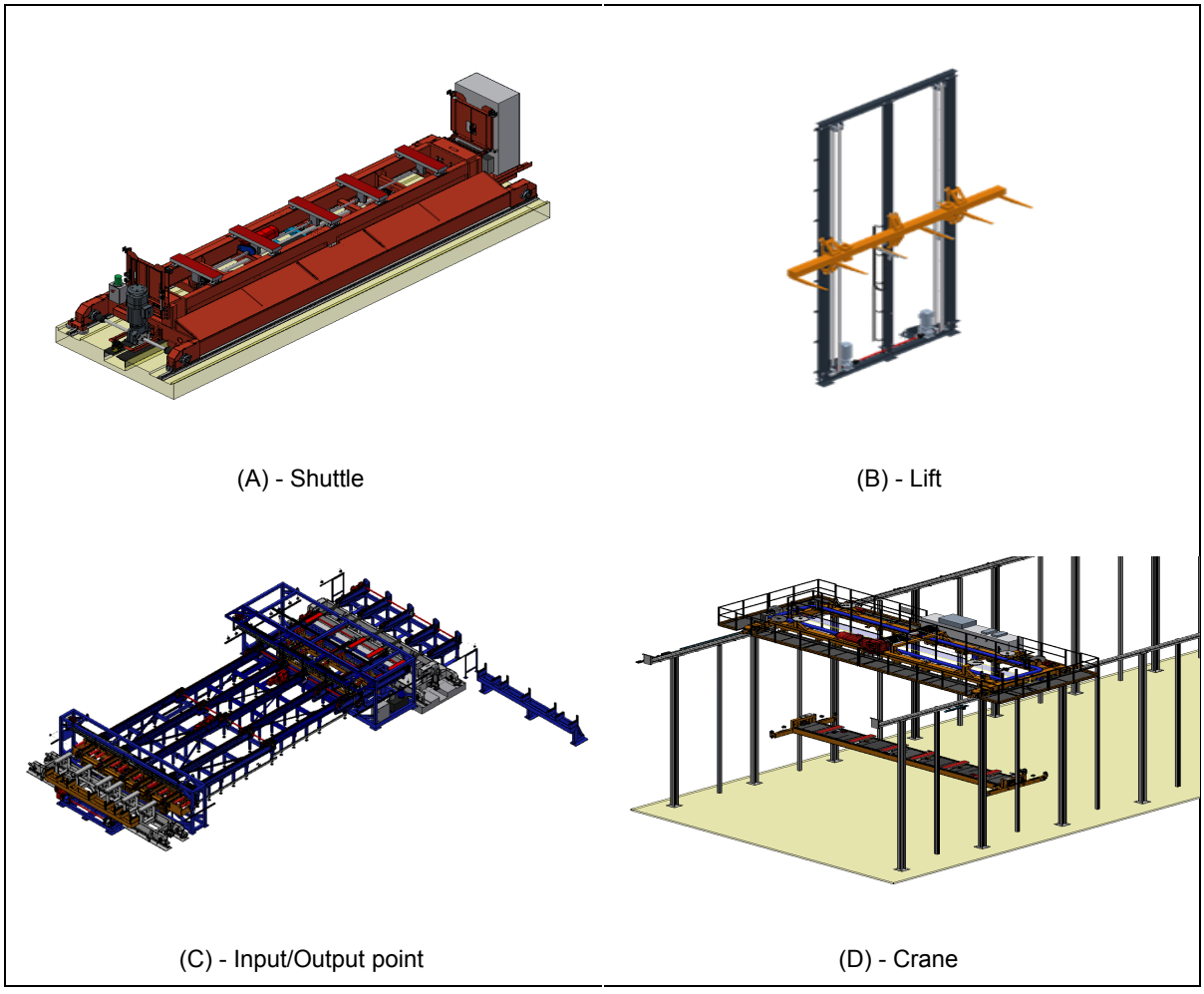


Figure 6.3. Machines installed in a SLC-AS/RS [4].

I/O points are essentially chain conveyors placed on the perimeter of the storage area, used to link the warehouse with production, arrivals and shipments departments. Their primary

function is to move bundles on the horizontal plane, but they can also act as buffers, by accumulating a limited number of bundles. Also, to support storage operations, I/O points are equipped with sensors to automatically detect and/or control bundle shape, weight and length.

Shuttles are rail-based handling systems used to take bundles inside and outside of racks. Each shuttle moves horizontally from the I/O point to the lifts and from the lifts to the I/O point, crossing the racks in a dedicated aisle where bundles cannot be stored.

Lifts are installed in the same aisle where the shuttles cross the racks, and they are used to move bundles from the ground level to the top of the rack, where the crane runs. So, their main task is to connect the shuttle and the crane.

Cranes move on top of the rack and are used to move bundles from the lift to a specific storage location inside the rack, and vice versa. This is clearly shown in Figure 6.2, where there are three cranes, one for each rack of the stocking area. As shown by Figure 6.4, the crane is made of three parts that can operate independently: the bridge crane, the undercarriage and the handling forks.

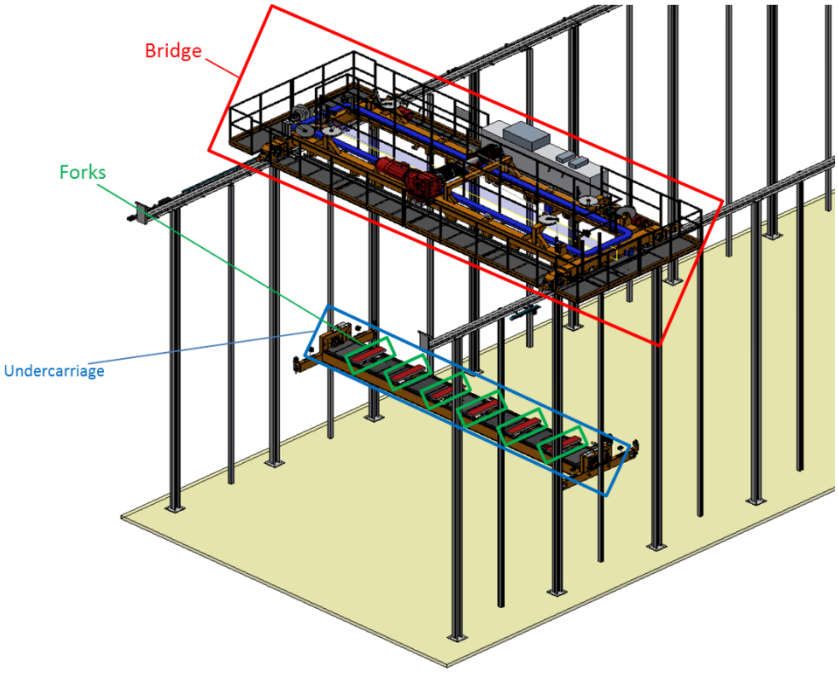


Figure 6.4. The components of the crane [1].

The horizontal displacements take place at the top of the rack and are performed by the bridge crane and by the undercarriage. Specifically, the bridge crane moves along the rack's

length, using two rails installed on the perimetral walls of the rack, while the undercarriage moves along the rack's width, using two rails installed inside the bridge crane. The displacements of the bridge crane and the undercarriage occur simultaneously and require the forks to be in rest position (i.e., at the top of the rack). Only when the bridge crane and the undercarriage have stopped at the proper horizontal position, the forks can move vertically (inside an aisle) to retrieve or to deposit a bundle. Note that the forks can make a small horizontal movement too; this is needed to assure a safe loading/unloading of the bundle.

The movement of the undercarriage is of paramount importance, because the bundles can be stored one beside the other in the same storage location and, each time a bundle is retrieved, it is important to take it in correspondence of its gravity center to prevent it from unhinging and falling during transportation. Similarly, when a bundle is stored, it must be placed in the correct position beside the others. For sake of clarity, a visual example is proposed in Figure 6.5, where is represented the possible collocation of two different bundles (i.e. A1 and B1) in the same storage location. In the figure are also represented two points (i.e. P1 and P2) that represent the two different positions that the crane have to reach to pick up respectively bundle A1 and B1.

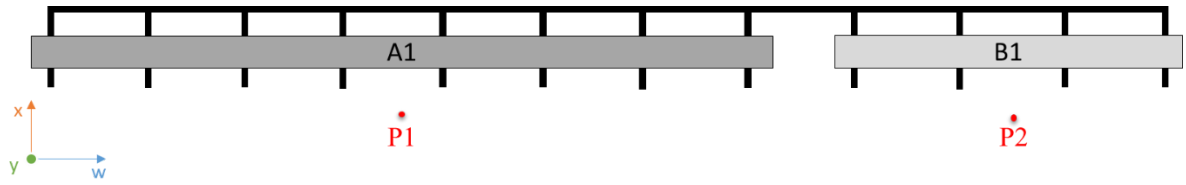


Figure 6.5. Possible collocation of two bundles in the same storage location [5].

6.1.2. Machines synchronization and operating policies

All the machines adopted in SLC-AS/RS are independent and can work in parallel. So, to maximize performance, as soon as a machine completes its mission, it immediately heads to the next destination, or it returns to its origin point waiting for a new mission to start. Nonetheless, a proper coordination of the handling operations is needed to ensure that bundles could be passed from one machine to another one, at the interchange points located at the bottom and at the top of the lifts. Specifically, the lift and the crane have to meet at the interchange point located at the top of the rack (i.e. IP_{Up}) and, similarly, the lift and the shuttle have to meet at the interchange point, located at the bottom of the rack (i.e. IP_{Dw},). Clearly, if there are two shuttles, the total number of interchange points per rack will

be equal to four: that is two interchange points for the input operations and other two for the output operations. Anyhow, during an operating cycle each machine could remain idle at an interchange point waiting for the machine with which the bundle must be exchanged. For a better comprehension, about how the machines operate and interact, two block diagrams representing the time and methods relative to the input and output case are reported in Figures 6.6 and 6.7.

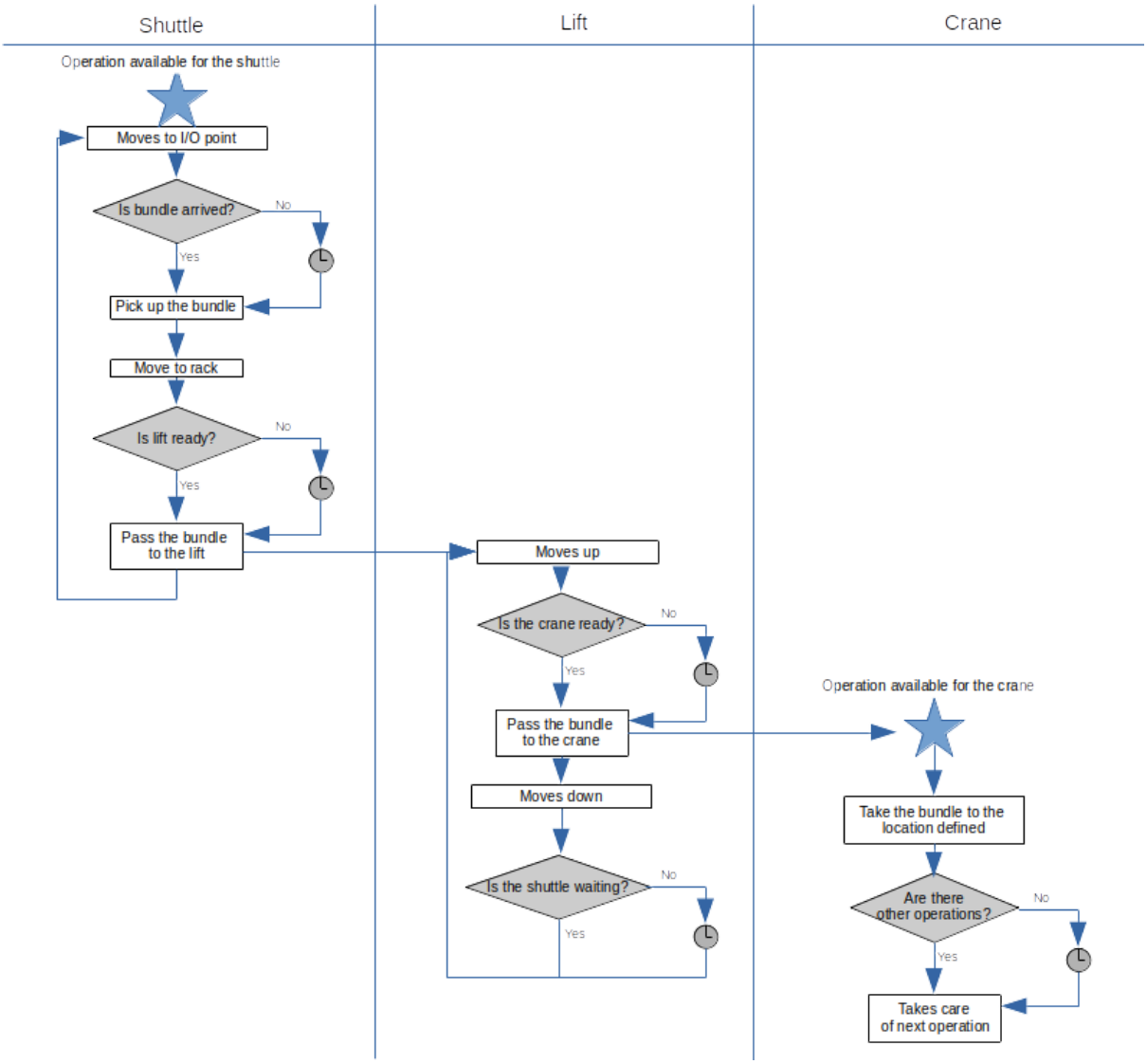


Figure 6.6. Time and methods in a generic input operation [4].

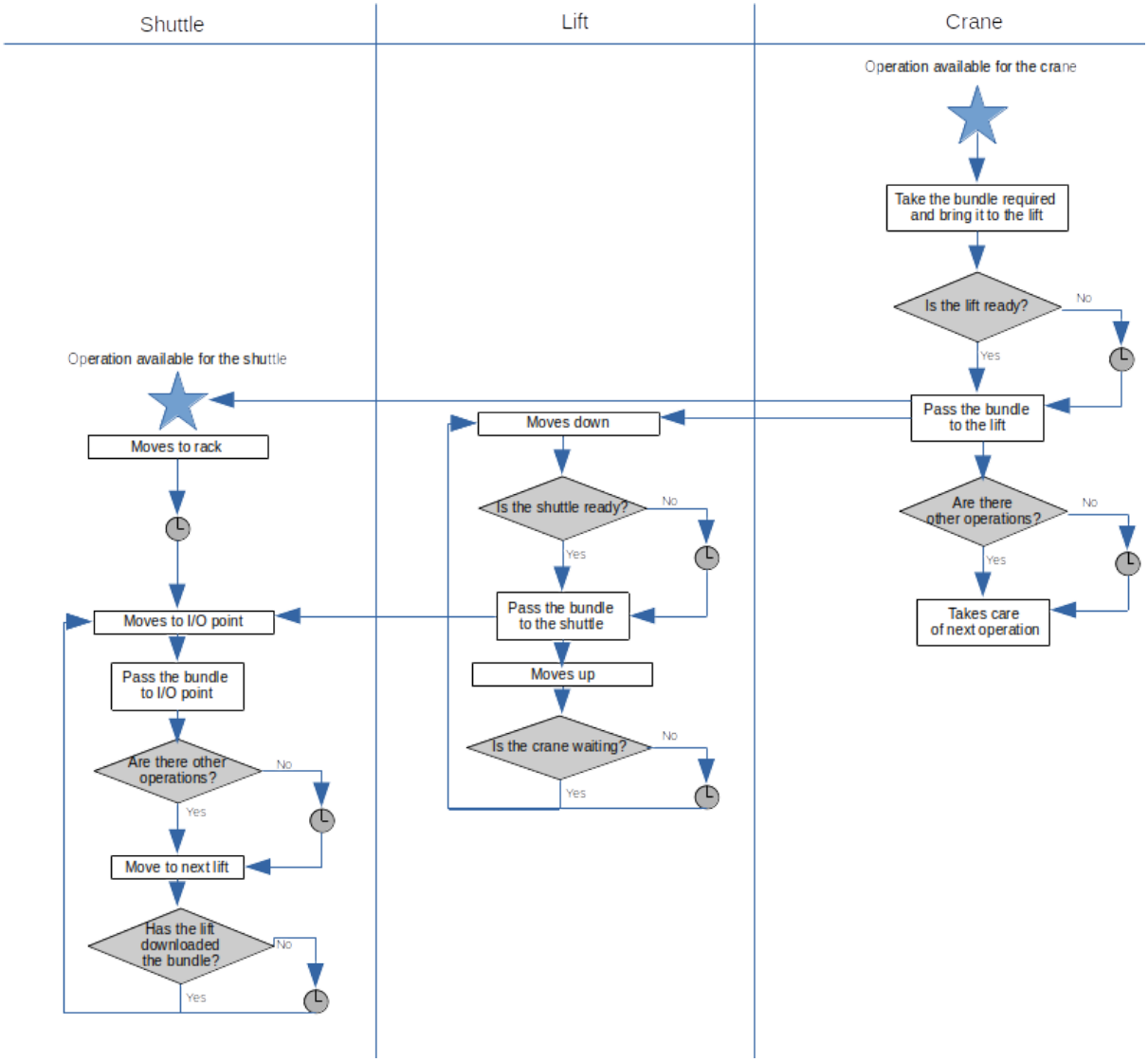


Figure 6.7. Time and methods in a generic output operation [4].

Each input or output operation must be assigned to a specific crane, lift, and shuttle. The selection of the crane depends on the rack where the bundle is stored (in case output) or will be stored (in case of input), while that of the shuttle depends on the input/output bay from which the bundle will enter or exit. The selection of the lift is automatically determined, once the crane and the input/output bay have been chosen. It is also worth mentioning that a SLC-AS/RS can work in single and dual command mode (or cycle). In case of single cycle, storage and retrieval operations are performed separately and, for an input operation, the crane performs the following actions:

- it moves to the interchange point (IP_{UP}) relative to the input bay from which the bundle is coming from,
- eventually it waits for the lift at IP_{UP},
- it loads the bundle from the lift,
- it moves to the storage location assigned to the bundle
- it downloads the bundle and stops at the corresponding *x*-*w* location, waiting for the next operation.

For an output operation the cycle changes as follows:

- the crane moves from its current location to the one where the selected bundles is stored
- it uploads the bundle,
- it moves to the IP_{UP} corresponding to the output bay that required the bundle,
- eventually it waits for the lift,
- it passes (exchanges) the bundle to the lift.

In case of dual command cycle, storage and retrieval operations are jointly performed and, in this case, the crane:

- loads a bundle at the corresponding IP_{UP},
- leaves the bundle at the assigned storage location, say p_1 at (x_1, y_1, w_1),
- moves to a second point, say p_2 at (x_2, y_2, w_2), where it loads a second bundle,
- goes to the IP_{UP} corresponding to the output bay which requires the uploaded bundle,
- eventually waits for the lift,
- passes the bundle to the lift.

6.1.3. The bundles

The bars bundles (or more in general Stock Keeping Units (SKU)) stored in a SLC-AS/RS are characterized by attributes such as length, shape, weight and quality level. Each combination of length and shape constitutes a bar type, which is identified with an univocal code. Conversely, the weight depends on the nominal weight but also on tolerances. For instance, given a bundle of 46 round bars with a nominal diameter of 30 mm (±0.15 mm tolerance) and a length of 4000 mm (±25 mm tolerance) and 7859 kg/m^3 the density of the material, the nominal weight of the bundle is around 1022 kg. However, considering the tolerances, the real weight could range from 982 kg to 1049 kg. Finally, the bundles' quality level depends on the geometric dimensional tolerances the bar is obtained with, the chemical purity of the material, the presence of surface damages and so on.

Given these characteristics, the SKUs in the problem addressed in this chapter may be formalised as follows. Let's assume $i = 1, ..., N$ the SKUs/bundles currently in stock; to each bundle i is associated a set of five values $(c_i, w_i, q_i, r_i, p_i)$, where:

- c_i is the code (that expresses shape and length information);
- w_i is the real weight registered when the bundle entered the system (by the input bays);
- q_i is the quality level of the bundle (often ranging between 1 and 3, where 3 is the highest quality level and 1 the lowest one);
- r_i and p_i are respectively the rack and the place where the item is currently stored.

6.1.4. The customers orders

Each customer order contains, at least, the information about (i) specific code (i.e. section and length of the bar then of the bundle), (ii) quantity in kg required by the customer and (iii) desired quality level. The weight and quality levels must be respected by the seller within certain tolerance margins agreed with each customer. Hence, the following formalisation is suggested. Let's consider $k = 1, ..., K^{out}$ the set of customers' orders to be processed; to each order k is associated a set of six attributes $(\rho_k, c_k, w_k, q_k, s_k, t_k)$, where:

- ρ_k is the arrival time;
- c_k is the code required;
- w_k is the total quantity required expressed in terms of weight (e.g. kilograms);
- q_k is the quality level to be respected;
- s_k is the output point where items to satisfy order k must be collected (note that it also corresponds to the shuttle to be used). We assume here that the output point s_k is not a variable but is given. This means that is the operator that select the output point: this reflects many of the actual conditions under which the warehouse systems have to work (clearly, choosing the output bay where to send the specific order, by the algorithm , could improve performance);
- t_k is a label that identifies the type of the request and distinguishes input and output requests (in case of customer order this value is set equal to "output request").

Each order $k = 1, ..., K^{out}$ can be fulfilled by retrieving one or more bundles according to the quantity required w_k. The quantity retrieved should be exactly that one required, however a small difference (usually expressed as a percentage on the total order weight) is usually considered acceptable. On the other hand, concerning the quality, the set of bundles in stock selected to respond to the customer order k, namely $i = 1, ..., n_k$, must have an average quality level higher or equal than the quality required q_k. In other words, following constraints expressed in Eq.(6.1), Eq.(6.2), and Eq.(6.3) must be respected.

$$w_k - \Delta \leq \sum_{i=1}^{n_k} w_i \tag{6.1}$$

$$\sum_{i=1}^{n_k} w_i \leq w_k + \Delta \tag{6.2}$$

$$q_k \leq \frac{\sum_{i=1}^{n_k} q_i}{n_k} \tag{6.3}$$

Where Δ is usually a percentage on the total quantity required w_k previously defined in the contract between seller and customer.

6.1.5. The storage operations

Concerning the storage operations $k = 1, ..., K^{in}$, since they consist of a single bundle to stock in the warehouse, they are not subject to quantity and quality constraints described above. However, since the machines and in particular the cranes need to deal with storage and retrieval tasks at the same time, storage operations must be considered as well. Exactly

as for customers' orders, for a standardization of nomenclature, to each storage operation is associated a set of six attributes $(\rho_k, c_k, w_k, q_k, s_k, t_k)$, where:

- ρ_k is the arrival time;
- c_k is the code of the input bundle;
- w_k is the real weight of the input bundle (usually recorded by load cells in the input point / bay);
- q_k is the quality level of the input bundle;
- s_k is the input point where the bundle is entering from (which defines also the shuttle to be used);
- t_k is the label which identifies the type of the request (in this case is set equal to "input request").

6.1.6. Sequencing

In classic AS/RS (e.g. where SKUs are "conventional" such as EPAL), it is possible to intervene for improving the performances by changing the sequence in which input and output operations are processed. However, working on SLC-AS/RS, there is for many reasons no much room for improvement via sequencing. First of all, the sequence of the input operations cannot be modified, as they are made of physical bundles on the input point. These bundles, because of their size and weight are difficult to move, and their movement would be too time-consuming for operators. Moreover, the bundles which constitute the input operations usually come directly from a production line, hence they must be processed as soon as possible without many possibilities of buffering. The situation concerning the output operations is similar. Even in this case, due to the difficulty of handling the bundles, the first bundle that is retrieved is the first that is loaded onto the truck. It would be possible to decide a sequence with which to load the truck (and therefore a sequence with which to retrieve the bundles), but, in any case remains the constraint that all the quantities required for each individual code must be delivered to the output bay without mixing them with other codes. Because of these reasons, the problem can be classified as a strictly time dependent problem that might somehow be compared to multi-depot vehicle

routing problems, such as that one approached by Fikar et al. (2016). The time dependencies limit the collection of operations to rearrange their sequence.

Although, many other interventions are still possible, such as interleaving, allocation, assignment, batching, sequencing of requests arrived together and so on. And these are the aspects on which the proposed solutions intervene.

6.2. Performance measurement via analytical model

6.2.1. Introduction

An analytical model for an AS/RS is usually a fast way to estimate the analysed parameters and it is very important for many different reasons. In terms of computational time, it is much less expensive than a simulation, although it provides the same information. For this reason, it can be easily implemented in an algorithm in order to evaluate the new generated solutions in a very short computational time. Moreover, once developed, it can be easily implemented in a spreadsheet without any need of computer knowledge or expensive simulation softwares.

In this section, an analytical model to estimate the time needed by the SLC-AS/RS to complete a given set of operations (i.e. makespan) is presented. Given a set of operations to execute in a predefined sequence, the proposed model returns the starting and ending time for each of them.

6.2.2. The notation and the model

Given $k = 1, ..., K$, where $K = K^{in} \cup K^{out}$, the set of input and output requests with the characteristics described in section 6.1, they might be fulfilled using a set of operations $j = 1, ..., J$, where $J \leq K$, because each input request corresponds to one and only one operation, while an output request might result in more operations depending on the quantity required. Each operation j is characterised by eight attributes $(\rho_j, i_j, s_j, r_j, p_j, t_j, P1_j, P2_j)$, where:

- ρ_j is the arrival of the correspondent input/output request;
- i_j is the bundle moved with this operation;
- s_j the input/output point which is the starting or ending point of the operation (note it corresponds also to the shuttle to be used);

- r_j and p_j are respectively the rack (or the crane) and the storage location where the bundle i_j must be stored (in case of input) or from which the bundle i_j is going to be retrieved (in case of output).
- t_j is the label used to distinguish input and output operations.

Unlike the other attributes, $P1_j$ and $P2_j$ do not provide any new information, they just explicit additional information already implicit in other attributes. In particular, they represent the first and the second position the assigned crane (i.e. r_j) has to visit to complete the operation j. More in detail, if j is an input operation, $P1_j$ is the position of the exchange point between the crane r_j and the lift in rack r_j served by the shuttle to be used (i.e. s_j), while $P2_j$ is the position of the assigned storage location p_j. Conversely, if j is an output operation the situation is opposite: $P1_j$ is the position of the storage location p_j where is stored the item to retrieve i_j, while $P2_j$ is the exchange point between the crane r_j and the lift in rack r_j served by the shuttle to be used (i.e. s_j).
The model proposed in this section calculates the time needed to carry out all the operations $j = 1, ..., J$ (i.e. makespan).
The duration of each operation depends on the machines assigned to it, and the operations previously made by those machines. Because of this, the time needed to complete all the operations can be calculated in a deterministic way, by repeating the following steps for each operation, from the first one to the latter. The following nomenclature (part of which has already been introduced in section 6.1) is introduced.

- j = the current operation.
- ρ_j = arrival time of operation j.
- t_j = label which say if j is an input operation or an output operation.
- s_j, r_j, l_j = respectively the shuttle, the crane, and the lift involved in operation j.

- $P1_j$, $P2_j$ = first and second position the crane must visit to carry out operation j. If j is an input operation $P1_j$ is the interchange point between the lift and the crane, while $P2_j$ is the position of the storage location. In case of output operation, $P1_j$ is the position of the storage location and $P2_j$ is the interchange point between the lift and the crane.

- $j^*_{s_j,r_j,l_j}$ = last operation which involved machines s_j, r_j and l_j.
- c = fixed travel time necessary to upload or download a bundle in storage locations, I/O points, or to pass it from a machine to the next one.
- y = one way travel time for lifts.
- m_{s_j,r_j} = one way travel time for shuttle s_j to reach rack r_j.
- $d_{P1_j,P2_j}$ = time needed by crane to move from $P1_j$ to $P2_j$

Then, for each operation, following values can be iteratively calculated.

- $START_{s_j}(j)$ = when the shuttle s_j starts working for that operation j.
- $END_{s_j}(j)$ = when the shuttle s_j ends working for operation j.
- $END_{l_j}(j)$ = when the lift l_j ends working for operation j.
- $START_{r_j}(j)$ = when crane r_j starts uploading the bundle of operation j.
- $END_{r_j}(j)$ = when crane r_j ends downloading the bundle of operation j.
- $AVAIL_{r_j}(j)$ = when operation j gets available for crane r_j.
- $W_{s_j,l_j}(j)$ = how much shuttle s_j waits for lift l_j in operation j.
- $W_{r_j,l_j}(j)$ = how much crane r_j waits for lift l_j in operation j.

Hence, for each *input operation*:

$$START_{s_j}(j) = max\{\rho_j,\ END_{s_j}(j^*_{s_j})\}$$

$$W_{s_j,l_j}(j) = max\{0,\ y\ +\ c\ + START_{r_j}(j^*_{l_j}) - (START_{s_j}(j) + c + m_{s_j,r_j})\}$$

$$AVAIL_{r_j}(j) = START_{s_j}(j)\ + 2c + y + m_{s_j,r_j} + W_{s_j,l_j}(j)$$

$$END_{s_j}(j) = START_{s_j}(j)\ + 2c + 2m_{s_j,r_j} + W_{s_j,l_j}(j)$$

$$START_{r_j}(j) = max\{AVAIL_{r_j}(j),\ END_{r_j}(j^*_{r_j})\}\ + d_{P2_{j^*_{r_j}},P1_j}$$

$$END_{r_j}(j) = START_{r_j}(j) + 2c + d_{P1_j,P2_j}$$

And for each *output operation*:

$$AVAIL_{r_j}(j) = \rho_j$$

$$START_{r_j}(j) = max\{AVAIL_{r_j}(j),\ END_{r_j}(j^*_{r_j})\}\ +\ d_{P2_{j^*_{r_j}},P1_j}$$

$$W_{c_j,l_j}(j) = max\{0,\ END_{l_j}(j^*_{l_j}) +\ y\ - (c\ + START_{r_j}(j) + d_{P1_j,P2_j})\}$$

$$END_{r_j}(j) = START_{r_j}(j) + 2c + d_{P1_j,P2_j} + W_{c_j,l_j}(j)$$

$$START_{s_j}(j) = max\{END_{r_j}(j),\ END_{s_j}(j^*_{s_j})\}$$

$$END_{l_j}(j) = max\{y + END_{r_j}(j),\ START_{s_j}(j) + m_{s_j,r_j}\} + c$$

$$END_{s_j}(j) = 2c + m_{s_j,r_j} + max\{y + END_{r_j}(j),\ START_{s_j}(j) + m_{s_j,r_j}\}$$

By implementing just mentioned formulas, and using some trigger values in first operations, it is possible to know when every operation in the system is terminated depending on the sequence in which operations are executed.

6.2.3. A numerical example

To provide a better comprehension a numerical example is therefore provided. Let's consider a SLC-AS/RS made of 1 rack, 1 input point with its shuttle and 1 output point with its shuttle, as reported in Figure 6.8.

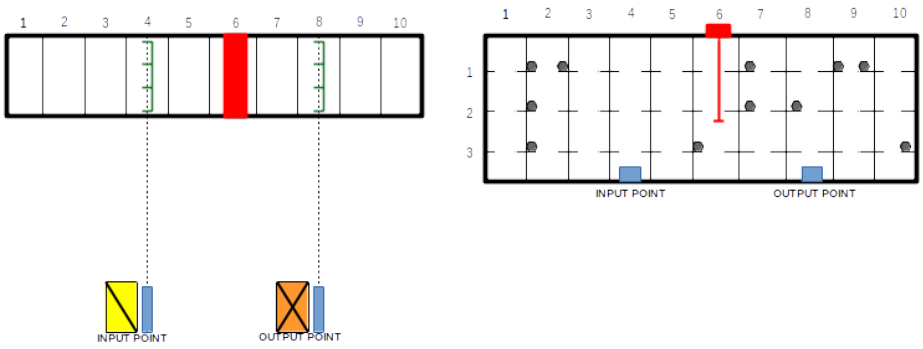


Figure 6.8. Planar (left) and frontal (right) view of the SLC-AS/RS considered in this example [4].

In the rack there are 1 crane and 2 lifts (one in correspondence of each I/O point). In the rack there are 10 aisles 1 meter long, the input shuttle crosses the rack in correspondence of the fourth aisle, while the output shuttle crosses it in correspondence of the eighth one. In each aisle there are 3 levels of storage 1 meter high. The crane moves in a uniform straight movement doing 1 meter per second both in length and height. Let's suppose the one way travel time for the lifts is 3 seconds ($y = 3$), the upload and download time is 5 seconds ($c = 5$), and the one way travel time of shuttles to reach the only rack is 6 seconds ($m_{s_j,r_j} = 6,\ \forall j$). Then, the following operations in the given sequence are considered:

- $t_1 = OUTPUT,\ \rho_1 = 3,\ P1_1 = (5,1),\ P2_1 = (8,0)$
- $t_2 = INPUT,\ \rho_2 = 4,\ P1_2 = (4,0),\ P2_2 = (7,2)$
- $t_3 = INPUT,\ \rho_3 = 10,\ P1_3 = (4,0),\ P2_3 = (1,3)$
- $t_4 = OUTPUT,\ \rho_4 = 11,\ P1_4 = (7,2),\ P2_4 = (8,0)$
- $t_5 = OUTPUT,\ \rho_5 = 20,\ P1_5 = (7,1),\ P2_5 = (8,0)$

Note that points to visit $P1_j$ and $P2_j$ are for simplicity already defined by using a tuple where the first element is the aisle and the second one the level, and remember that for input operations $P1_j$ is the interchange point where the lift pass the bundle to the crane, and for output operations $P2_j$ is the interchange point where the crane passes the bundle to the

lift. By using the formulas mentioned above, all the events in the system can therefore be calculated as in Table 6.1 and the makespan is given.

Table 6.1. Calculation of the makespan in the example presented [4].

Operation 1	**Operation 2**
$AVAIL_{r_1}(1) = \rho_1 = 3$ $START_{r_1}(1) = max\{3,\ 0\} + 0 = 3$ $W_{c_1,l_1}(1) = 0$ $END_{r_1}(1) = 3 + 10 + 4 + 0 = 17$ $START_{s_1}(1) = max\{17,\ 0\} = 17$ $END_{l_1}(1) = max\{3 + 17,\ 17 + 6\} + 5 = 28$ $END_{s_1}(1) = 10 + 6 + max\{3 + 17,\ 17 + 6\} = 39$	$START_{s_2}(2) = max\{4,\ 0\} = 4$ $W_{s_2,l_2}(2) = 0$ $AVAIL_{r_2}(2) = 4 + 10 + 3 + 6 + 0 = 23$ $END_{s_2}(2) = 4 + 10 + 12 + 0 = 26$ $START_{r_2}(2) = max\{23,\ 17\} + 4 = 27$ $END_{r_2}(2) = 27 + 10 + 5 = 42$
Operation 3	**Operation 4**
$START_{s_3}(3) = max\{10,\ 26\} = 26$ $W_{s_3,l_3}(3) = max\{0,\ 3 + 5 + 27 - (26 + 5 + 6)\} = 0$ $AVAIL_{r_3}(3) = 26 + 10 + 3 + 6 + 0 = 45$ $END_{s_3}(3) = 26 + 10 + 12 + 0 = 48$ $START_{r_3}(3) = max\{45,\ 42\} + 5 = 50$ $END_{r_3}(3) = 50 + 10 + 6 = 66$	$AVAIL_{r_4}(4) = 11$ $START_{r_4}(4) = max\{11,\ 66\} + 11 = 77$ $W_{c_4,l_4}(4) = max\{0,\ 28 + 3 - (5 + 77 + 3)\} = 0$ $END_{r_4}(4) = 77 + 10 + 3 + 0 = 90$ $START_{s_4}(4) = max\{90,\ 39\} = 90$ $END_{l_4}(4) = max\{3 + 90,\ 90 + 6\} + 5 = 101$ $END_{s_4}(4) = 10 + 6 + max\{3 + 90,\ 90 + 6\} = 112$
Operation 5	
$AVAIL_{r_5}(5) = 20$ $START_{r_5}(5) = max\{20,\ 90\} + 2 = 92$ $W_{c_5,l_5}(5) = max\{0,\ 101 + 3 - (5 + 92 + 2)\} = 5$	

$$END_{r_5}(5) = 92 + 10 + 2 + 5 = 109$$

$$START_s(5) = max\{109,\ 112\} = 112$$

$$END_l(5) = max\{3 + 109,\ 112 + 6\} + 5 = 123$$

$$END_s(5) = 10 + 6 + max\{3 + 109,\ 112 + 6\} = 134$$

6.3. Responding to customers orders

6.3.1. Introduction

In this section a solution to respond as better as possible to the customers orders is proposed. The algorithm proposed is inspired by the Simulated Annealing (SA), a metaheuristic framework originally proposed by Kirkpatrick et al. (1983). Given a generic customer order $k = 1, ..., K^{out}$, the objective of the proposed algorithm can be identified in 3 main concerns: (i) minimization of the difference between the quantity required w_k and the quantity retrieved $\sum_{i=1}^{n_k} w_i$ by properly selecting the retrieved bundles, i.e. $i = 1, ..., n_k$; (ii) minimization of the difference between the quality level required q_k and the average quality of retrieved bundles (i.e. $\frac{\sum_{i=1}^{n_k} q_i}{n_k}$); and, finally, (iii) minimization of the time needed to fulfill the order (i.e. makespan). To do this, the algorithm collects several orders before processing them all together, and it seeks for the optimal solution for that batch of orders.

6.3.2. Formalization

The proposed algorithm is a metaheuristic inspired by the Simulated Annealing (SA), which is well-known in literature and has already been applied for solving warehouses retrieving problems (see for instance Nadir et al. (2012), Yang et al. (2013), and Bian et al. (2018)). The main procedure of the SA is subject to many parameters, which is important to describe. A parameter of paramount importance is the *temperature*, because its value defines the acceptance probability of a new solution worse than the current one and indirectly defines the size of the explored neighborhood. At the beginning of each era, the temperature is brought back to the initial level selected and then decreases with the increase in iterations till

it reaches the final value. The temperature and its reduction define the number of solutions explored by the algorithm, the acceptance threshold of worse solutions, and the size of neighborhood analysed at each iteration.

Another element to define is the *cooling schedule*, which defines how the temperature T decreases as the number of iterations increases. Every iteration *m*, temperature value is calculated using the formula proposed by Lai and Chan (1997) (Eq. (6.4)):

$$T_m = \frac{T_{m-1}}{1+T_{m-1} \cdot B} \tag{6.4}$$

where *B* is calculated as in Eq. (6.5)

$$B = \frac{T_i - T_f}{T_i \cdot T_f \cdot E} \tag{6.5}$$

where T_i is the starting temperature, T_f is the ending temperature and *E* is the number of eras.

Initial and final temperature are set accordingly to Kirkpatrick et al. (1983). Specifically, it is chosen to assure a high initial acceptance probability and a low ending acceptance probability. Fixed F_{max} the highest (best) of fitness that can characterize a solution and $a_{\%}$ the desired acceptance threshold, the temperature is calculated using equation proposed by Bertolini et al. (2019):

$$T = \frac{-F_{max}}{ln(a_{\%})} \tag{6.6}$$

The fitness function, which the SA tries to optimize, takes into account three elements represented by three different indexes:

- Q_O represents the quality delta between customer order request and planned order retrieving;
- W_O represents the quantity delta between customer order request and planned order retrieving;
- T_O represents the estimated retrieving time.

Each component is designed to have a value ranging between 0 and 1, and each of them is assigned a weight coefficient ε, φ and μ whose values have been empirically defined (ε =

0.35, φ = 0.2, μ = 0.45) to match the case study expectations. The fitness function is thus expressed in Eq. (6.7).

$$Fitness = \varepsilon Q_o + \varphi W_o + \mu T_o \tag{6.7}$$

Q_O represents the difference between the quality level required and average quality of retrieved bundles in the entire solution. It is calculated as the sum of the single order quality indexes Q_k, where $k = 1, ..., K^{out}$ are the customers orders. Given a generic order k asking for a quality level q_k, given $i = 1, ..., n_k$ the bundles retrieved to fulfill it, the quality index Q_k is computed as in Eq. (6.8).

$$Q_k = 1 - \frac{\left| q_k - \sum_{i=1}^{n_k} q_i \right|}{0.5(UB_q - LB_q)} \tag{6.8}$$

where UB_q and LB_q are respectively the quality upper and lower bounds. Index Q_k ranges between 0 and 1, and the higher is the difference between the quantity required and that retrieved, the lower is the index. Likewise, the weight index relative to each order k is calculated as in Eq. (6.9).

$$W_k = 1 - \frac{\left| w_k - \sum_{i=1}^{n_k} w_i \right|}{0.5(UB_w - LB_w)} \tag{6.9}$$

where UB_w and LB_w are respectively the weight upper and lower bounds. Even in this case the index relative to the whole solution W_o is the sum of all the resulting indexes W_k.

The calculation of the time index T_o considers the distance *D* between retrieved items and output point requiring them. It also takes into account the number of required cranes *R* simultaneously involved in the handling tasks (considering that each crane is able to reach just specific areas of warehouse). The idea is based on the hypothesis that the higher is the number of cranes involved the lower is the makespan, since the activities performed in parallel are increased. The distance *D* is calculated as the distance between the output point and the item to be retrieved, multiplied by 2, to predict the worst situation in which the crane has just finished the previous mission. The travel time has no upper bound and, in order to obtain a value between 0 and 1, the maximum distance that could be covered is calculated beforehand. Given *L* the distance between the output point and the furthest item, w_k the quantity required by the order k, and *p* the nominal weight of required code, the maximum distance D_{max} is calculated as in Eq. (6.10).

$$D_{max} = 2L\frac{w_k}{p} \tag{6.10}$$

where here w_k/p is the indicative number of travels needed to fulfill the order line. This pessimistic value is compared with the real distance to be covered, that, given $i = 1, ..., n_k$ bundles selected to fulfill the order k and d_i the distance between bundle *i* and the output point, is calculated as follows (Eq. (6.11)):

$$D = \sum_{i=1}^{n_k} d_i \tag{6.11}$$

Finally, the time index T_k for a generic order k is calculated as in Eq. (6.12):

$$T_k = 1 - \frac{D}{D_{max} \cdot R} \tag{6.12}$$

Even in this case, the time index of the whole solution T_O is the sum of all orders indexes.

Then the *threshold* is considered. In the simulated annealing procedure, at each iteration a new solution is generated from the current working solution neighborhood. The procedure, while it aims to obtain at each iteration a new solution providing a better fitness function value, also considers the possibility to temporarily accept a worsening solution, so as to escape, in following iterations, a local optima. This behavior is managed using a probability threshold, which describe the probability to accept a new solution even if worse than the current one and is calculated using the following formula frequently used in literature:

$$threshold = e^{-\Delta F/T} \tag{6.13}$$

Where *T* represents the current temperature and ΔF the difference between the fitness of two compared solutions.

The last parameter is the *anchor*: every time a new solution is generated adopting a neighbor search, the anchor defines the exact number of elements (bundles) to change in that iteration. Fixed θ the number of items, which compose the current solution and *T* the current temperature, the anchor is calculated by using the following function proposed by Lai and Chan (1997) (Eq. 6.14).

$$anchor = roundup(\theta \cdot e^{-1/T}) \tag{6.14}$$

6.3.3. The proposed algorithm

The macro-procedure is represented in Figure 6.9. Each era a new random solution is generated, accepted and made the current solution. Temperature is set equal to the beginning value defined a priori. Each iteration temperature value is updated, a new solution is generated and rated: if it is accepted it becomes the new current solution. Procedure is repeated till temperature reaches the final value defined a priori. Then, if the maximum number of eras has been reached, the procedure ends and returns the best solution found so far, otherwise the era's number is updated and the whole procedure is repeated.

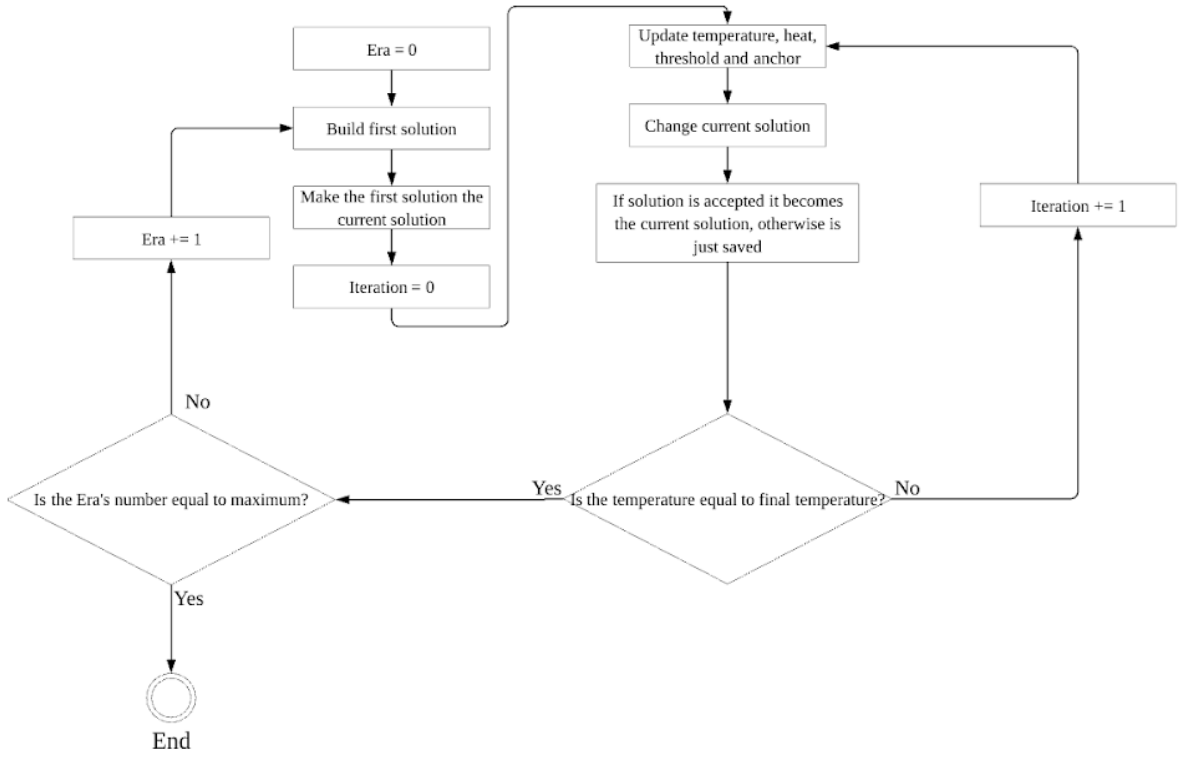


Figure 6.9. Macro-procedure (Bertolini et al., 2019).

The procedure for constructing the initial solution is sketched in Figure 6.10. The objective is to respond to a given set of customers orders, and the response to each order is called indicated as sub-solution (i.e. the set of bundles retrieved to fulfill that order). There are two ways to fulfill an order line:

- Matching: the request is fulfilled with just one bundle;
- Filling: the request is fulfilled with a list of bundles (Figure 6.11).

Matching always takes precedence. In this way, if an order can be completed with a single item, the possibility of fulfilling it with several items is not considered, because, generally, retrieving more items reduces retrieving performances. If matching is not possible the filling procedure described in Figure 6.10 is implemented. However, it must be noted that, even if the order can be fulfilled with the current stock, the filling procedure can result in a deadlock situation where the addition of any available item would violate the weight upper bound allowed (Eq. (6.2)). In this case, an additional previous step is applied, i.e. the elimination of the heaviest item from sub-solution, before repeating the filling procedure. This is repeated at most a number of times equal to the length of feasible items' list.

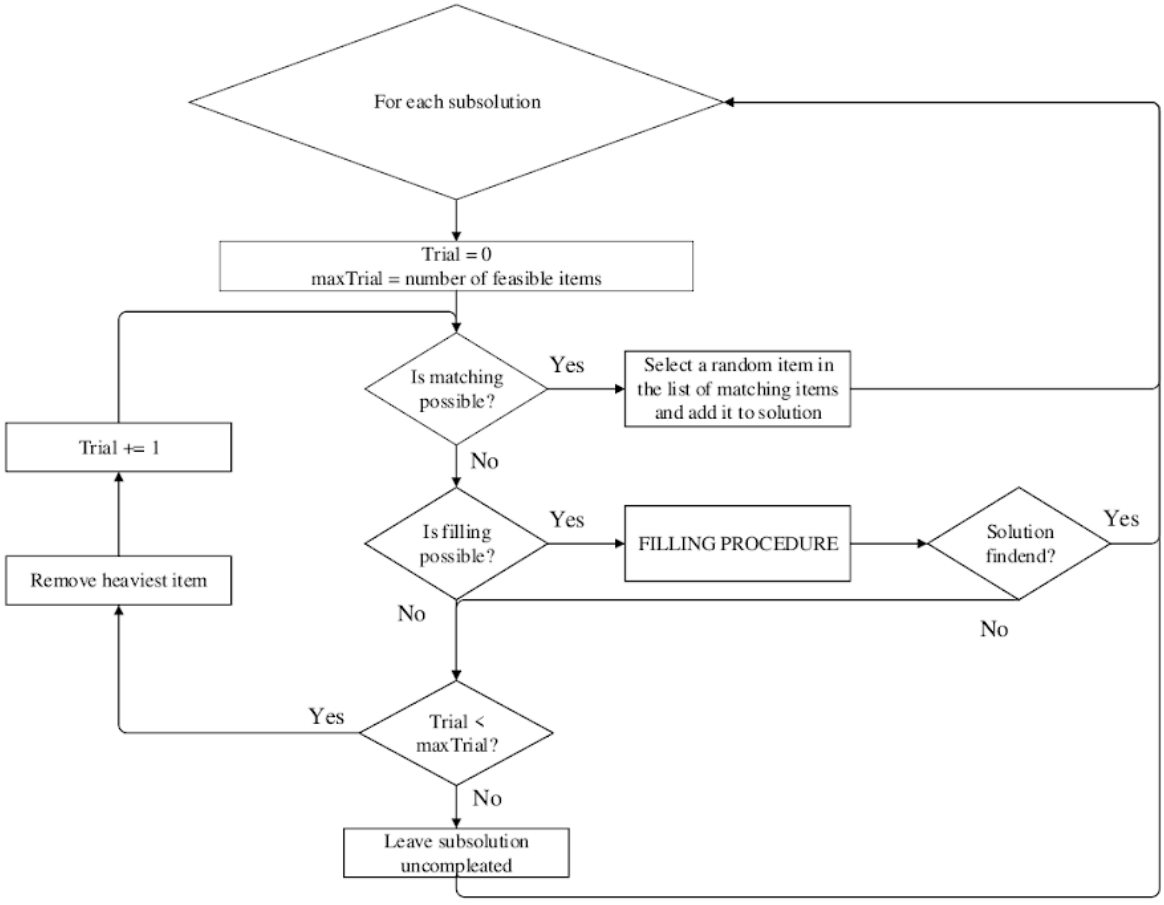


Figure 6.10. Generation of a new solution(Bertolini et al., 2019).

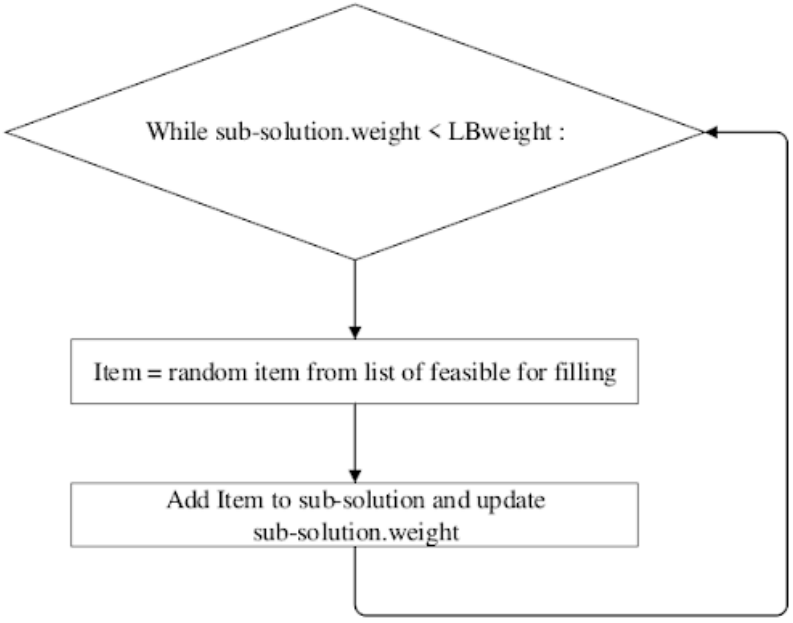


Figure 6.11. Filling procedure (Bertolini et al., 2019).

The neighbourhood exploration is performed exactly as the initial solution's creation but is preceded by the random deletion of an anchor number of bundles from the current solution.

6.3.4. Case study

To test the algorithm, a simulation model of an existing SLC-AS/RS was created in Python 3.6 using the Simpy package, compiled using CPython interpreter and the simulation was run on a standard personal computer Intel QuadCore i7 CPU at 2.4GHz with 8Gb RAM and Ubuntu 18.04 OS. The warehouse consists of two distinct racks served by two different cranes, one output point and one input point connected to the cranes by shuttles and lifts. The algorithm was tested comparing the simulation model with the real system over two weeks of work characterized by a high peak of requests and shifts of 8 hours per day, during which both input and output missions were performed. The warehouse's filling at the beginning of the test was 60% and three different types of codes of different length and nominal weight were managed. Given q_k and w_k respectively the quality level and the quantity required, the quality upper bound UB_q, the quality lower bound LB_q, the weight upper bound UB_w and the weight lower bound LB_w were defined as follows:

- $UB_q = q_k + q_k e^{-1.5}$
- $LB_q = q_k - q_k e^{-1.5}$
- $UB_w = w_k + w_k e^{-1.5}$
- $LB_w = w_k - w_k e^{-1.5}$

Results are reported in Table 6.2 and show how the proposed procedure correctly ensures better performance and less difference between quality or quantity required and retrieved.

Table 6.2. Results of the comparison (Bertolini et al., 2019).

Comparison value	Units of measurement	Proposed algorithm	Currently implemented algorithm
Working hours simulated	hour	80	80
Missions completed	number	2,243	2,018
Missions per hour	number/hour	28	25
Average computation time	seconds	1.526	0.039
Average quality difference	%	0.72	3.07

Average quantity difference	%	8.96	14.06

6.5. Throughput improvement

6.5.1. Introduction

In the previous section, it has been investigated how to efficiently respond to customers orders. In particular the algorithm proposed defines which items in stock should be assigned to which customers orders, taking into account the quantity and the quality level required, and rough estimate of the makespan (i.e. the time needed to fulfill the request). As shown by the case study, a good assignment of items in stock to customers orders involves substantial benefits and increases the performance too. However, the obtained improvement in terms of throughput is not that relevant and the algorithm proposed is not very 'dynamic', since it needs to collect several orders before computing a solution, which, however, will never be optimal.

Instead, in this section the focus is the throughput, and quantity and quality are treated as constrained. The proposed algorithm aims to be fast and cheap in terms of computational time, for dynamically generating a handling plan able to reduce as much as possible the time necessary to fulfill all the requests the system is subject to (i.e. makespan). It might be classified as a Biased Randomised Algorithm (BRA), an alternative that, to the author's best knowledge, is rarely adopted in warehousing. In particular, the author would rather define it as a Biased Randomised Discrete Event Heuristic (BR-DEH). The proposed BR-DEH is then incorporated in a metaheuristic framework, to provide many feasible good solutions in a short time and finally return the best one. If compared to the solution proposed in the previous section, further aspects are also considered, since the BR-DEH is designed to jointly deal with (i) the assignment of stored bundles to customers' retrieval orders, (ii) the allocation of entering bundles to empty storage locations, (iii) the assignment of machines to the moved bundles, and (iv) the sequencing of input and output operations.

6.5.2. Background on Biased Randomised Algorithm (BRA)

The Biased Randomised Algorithm (BRA) owns the plethora of randomised heuristics, which, nowadays, are widely used to solve large scale optimization problems. The BRA is a constructive procedure where each element of the solution under construction is iteratively selected according to a certain probability. The concept is similar to that one already introduced by the well-known roulette wheel selection process. In BRA, each time a new

candidate must be included in the solution, a different probability based on some criteria (e.g. ranking, priority rule, heuristic value, etc.) is associated to each candidate, and the selection is randomly made according to these probabilities. Better is the candidate, higher is the probability to select it. The idea behind this concept is to introduce slight modifications in the greedy constructive behaviour, to escape the local optima by exploring many solutions in a very short computational time, while maintaining the logic behind the heuristic.

Two of the earliest biased randomised procedures were proposed by Arcus (1965) and Tonge (1965). The procedure was called Biased Random Sampling (BRS) and it was used to bias the selection of randomly generated solutions. After that, many priority rules-based heuristics have been designed. However, before seeing the application of a BRA in a metaheuristic framework, the scientific community had to wait for the Probabilistic Tabu Search (PTS) firstly introduced by Glover (1989) and successively extended in Glover (1990). Another metaheuristic which implements the BRA is the Ant Colony Optimization (ACO), which was originally called Ant Colony System when Dorigo & Gambardella (1997) introduced it.

However, all these solutions define the probability by using an empirically-constructed distribution. A reasonable alternative is to adopt a theoretical probability distribution. In this way, to the authors best knowledge, Juan et al. (2010) have been pioneers in the implementation of a skewed theoretical distribution in the BRA. The main advantage coming from the utilisation of a theoretical distribution is the possibility to obtain a random candidate in a less time-consuming way using an analytical expression. In the BRA via theoretical distribution, the candidates are sorted from the best to the worst according to the desired criterion, and then, the probabilities are assigned to the candidates depending on the position they occupy on the list. To the author's best knowledge, the most used theoretical distribution in BRA is the geometric distribution, probably because of its simplicity and its dependence on a single parameter. As a matter of facts, the dependence of the algorithm on a single parameter avoids a long time consuming fine-tuning process (see for instance Juan et al., 2015 or Belloso et al., 2015). Since this part wanted to be just a brief overview of BRA, for more implementations and additional information, the author suggests Grasas et al. (2017): a recent literature review on the topic.

6.5.3. The Biased Randomised Discrete Event Heuristic

A Biased Randomised Discrete Event Heuristic (BR-DEH) algorithm is proposed to quickly provide as many feasible solutions as possible. The input and output requests are considered together ($k = 1, ..., K$) and processed one-by-one using a FIFO (i.e.

First-In-First-Out) logic. For each request, the algorithm tries to provide a good feasible solution, otherwise, if the solution found is not feasible, the request is postponed.

More in detail, if k is an input request and the solution found is not feasible, all the input requests coming from the same input point s_k are postponed. This because, as explained in section 6.1, the inputs requests' queue cannot be changed. In particular, if the non-feasible input requires to store a bundle of type c_k, all the inputs are postponed after the next output that requires to retrieve a bundle of the same type. In this way, since the output free a storage space, the non-feasible request k is certainly satisfied at the next allocation process. An example of this postponement process is represented in Figure 6.12, where the request number 1 (the non-feasible) and request 3 (input coming from the same I/O point of 1), are postponed after 5 (the output which makes space for 1).

Request k	**Type** t_k	**I/O point** s_k	**Bundle type** c_k
1	**IN**	**1**	**A**
2	IN	2	B
3	**IN**	**1**	**C**
4	OUT	3	B
5	**OUT**	**3**	**A**
6	IN	2	C

Request k	**Type** t_k	**I/O point** s_k	**Bundle type** c_k
2	IN	2	B
4	OUT	3	B
5	**OUT**	**3**	**A**
1	**IN**	**1**	**A**
3	**IN**	**1**	**C**
6	IN	2	C

Figure 6.12. List of requests before (left) and after (right) the postponement process [4].

On the other hand, if the non-feasible request is an output it is postponed after the next input that is introducing the same type of bundle in the warehouse.
The pseudocode of the overall BR-DEH is represented in Figure 6.13. The parameters of the procedure have to be interpreted as follows: *requests* is the list of input and output requests to be processed, *racks* is the list of the warehouse's racks, and *beta* is the list of parameters used for the geometric distributions.

```
procedure BR-DEH(requests, racks, beta)
01 solution <- NULL %% general empty solution
02 while length(requests) > 0: %% until all requests are not solved
03   nextR <- extractNextReq(requests) %% next request in time
```

```
04    sol <- NULL %% solution to a single request
05    if type(nextR) is Input: %% nextR is a bundle to store
06      sol <- processInput(nextR, solution, racks, beta)
07    else if type(nextR) is Output: %% nextR is a customer order to retrieve
08      sol <- processOutput(nextR, solution, racks, beta)
09    end if
10    if sol is not feasible:
11       postponementProcess(nextR, requests) %% postpone the request
12    else
13       solution <- add(solution, sol) %% update general solution
14    end if
15 end loop
16 return solution %% return the solution found
end procedure
```

Figure 6.13. Pseudocode of the BR-DEH proposed [4].

In case of input operation, the objective is to find an empty place in the warehouse to host the entering bundle. To host the bundle, the empty place needs to be long enough, and the length of bundles depends on their type (or code). At first, to find a solution, all the racks are considered. At each iteration the racks are sorted prioritizing those whose crane is first available and one of them is selected via biased randomisation. This step is very important to equally distribute the workload across all the cranes.

In particular, concerning the biased randomisation, the authors decided to adopt a theoretical quasi-geometric distribution according to Juan et al. (2010). Given x the position occupied by a candidate in the previously sorted list and β the parameter of the geometric distribution, the probability to choose the candidate $f(x)$ is calculated by Eq. (6.15).

$$f(x) = (1 - \beta)^x \tag{6.15}$$

The procedure used to select a random element from a list is reported in Figure 6.14.

```
procedure findBR(list, beta)
01 rnd <- randomNumber() %% random number between 0 and 1
02 i <- int(log(rnd) / log(1-beta)) % length(list) %% find the index
03 return list[i] %% return the i-th candidate in list
end procedure
```

Figure 6.14. Biased randomised selection of a candidate [4].

If in the selected rack there are some feasible storage locations, one of them is randomly selected according to a uniform distribution, a solution is generated, and a new operation is scheduled to update the state of the system. In this case, the selection is made according to a uniform distribution because, without any information concerning the next operation carried out by the crane, there is no need to prefer a location to the other. Moreover, according to Roodbergen and Vis (2009), the assignment and sequencing of input operations is not prominent for the throughput of an AS/RS. Otherwise, if in the selected rack there is no feasible place, the rack is removed from the list of possible racks and the process is repeated. If the list of possible racks is empty and a solution has not been found yet, a non-feasible solution is returned, and the input request will be postponed by the BR-DEH procedure. The pseudocode for the processing of input operations is reported in Figure 6.15.

```
procedure processInput(nextR, solution, racks, beta)
01 pRacks <- copy(racks) %% list of racks to consider
02 sol <- NULL %% empty solution to nextR (in this case a single operation)
03 while not sol and length(pRacks) > 0: %% repeat until a solution is found
or there are no feasible storage locations
04   pRacks <- sort(pRacks, key:craneAvailability) %% sorts racks
prioritizing those whose crane is available first
05   rack <- findBR(pRacks, beta) %% select a rack according to geometric
distribution
06   pPlaces <- feasiblePlaces(rack) %% feasible storage locations
07   if length(pPlaces) > 0: %% a place will host the bundle
08      place <- randomUniformChoice(pPlaces) %% randomly select a storage
location
09      sol <- new InputOperation(nextR, rack, place) %% instance the
solution to nextR
10      scheduleOperation(solution, sol) %% schedule the operation
11   else:
12      pRacks <- remove(pRacks, rack) %% remove rack from list of racks to
be considered
13   end if
14 end loop
15 return sol
end procedure
```

Figure 6.15. Pseudocode for input processing [4].

For output operations, the construction of a solution goes through the selection of many bundles. After selecting a bundle the state of the system must be updated to correctly select the next one. However, the solution cannot be considered as feasible until all the necessary bundles have been retrieved. For this reason, before starting the construction of the solution the state of the system is saved, and, in case the solution resulted as non-feasible, the state

of the system would be restored. The construction process goes on until the quantity retrieved does not respect the quantity required (i.e. constraints expressed in Eq.(6.1) and Eq.(6.2) are respected) and there are racks with feasible bundles in stock. If the quantity constraints are not respected and there are no more feasible bundles, the process is interrupted, the state of the system is restored, and a non-feasible solution is returned. Likewise, if the construction process ends respecting the quantity constraints (i.e. Eq.(6.1) and Eq.(6.2)), but the quality level required is not respected (i.e. Eq.(6.3)), again the state of the system is restored and a non-feasible solution is returned. During each step of the construction process, possible racks are sorted prioritizing those whose crane is available first to equally spread the workload across all the cranes. Then a rack is chosen via biased randomisation using the quasi-geometric distribution. The feasible bundles stored in that rack are listed. For a bundle to be feasible, the sum of its weight with the weight of the solution in construction must not exceed the upper bound: in other words, the constraint expressed in Eq.(6.2) must be respected. The bundles enter with previous input operations are considered only if the input operation is already concluded. If in the chosen rack there are no feasible bundles, the rack is removed from the list of possible racks and the process is repeated. Conversely, if there are some feasible bundles, they are sorted at first for decreasing weight (i.e. heaviest bundles are prioritised), and then for the increasing difference between their quality and the quality level required (i.e. bundles with a level of quality closer to that required are prioritised). Once the bundles have been sorted, one of them is selected according to the geometric distribution, please note the β the parameter used here might be different from the parameter used for racks. Then the weight of the solution, the total quality, and the number of retrieved items is updated; the state of the system is updated, and, finally, the list of possible racks to consider is restored. Then the process is repeated.

The pseudocode for the processing of output operations is reported in Figure 6.16.

```
procedure processOutput(nextR, solution, racks, beta)
01 sSolution <- copy(solution) %% save current state of the solution
02 sol <- NULL %% solution to nextR (in this case an array of operations)
03 pRacks <- copy(racks) %% list of possible racks
04 w <- 0 %% weight of solution to nextR
05 q <- 0 %% total quality of solution to nextR
06 n <- 0 %% number of bundles in the solution to nextR
07 while w < weightReq(nextR) - accError(nextR) and length(pRacks) > 0:
```

```
     %% repeat until the weight lower bound is respected
08   pRacks <- sort(pRacks, key:craneAvailability)%% sort racks prioritizing
those whose crane is available first
09   rack <- findBR(pRacks, beta) %% select a rack with geometric dist.
10   pBundles <- feasibleBundles(rack, w) %% feasible bundles
11   if length(pBundles) > 0: %% a bundle can be retrieved
12      pBundles <- sort(pBundles, key:increasingWeight)%% sort by weight
13      pBundles <- sort(pBundles, key:deltaQuality)%% sort by quality
14      bundle <- findBR(pBundles, beta) %% select a bundle with geometric
distribution
15      w <- w + weight(bundle) %% update the weight
16      q <- q + quality(bundle) %% update the total quality
17      n <- n + 1 %% update the number of retrieved bundles
18      op <- new OutputOperation(nextE, rack, place)%% instance a retrieve
19      scheduleOperation(solution, op) %% schedule the operation
20      sol <- add(sol, op) %% add the retrieve to the solution
21      pRacks <- copy(racks) %% restore the list of possible racks
22   else:
23      pRacks <- remove(pRacks, rack) %% remove the selected rack from the
list of possible racks
24   end if
25 end loop
26 if w>=weightReq(nextR)-possibleError(nextR) and q/n>=qualityReq(nextR):
     %% if weight and quality constraints are respected...
27   return sol %% returns feasible solution
28 else:
29   solution <- sSolution %% restore the solution as saved before
30   return sol %% returns NOT feasible solution
31 end if
end procedure
```

Figure 6.16. Pseudocode for output processing [4].

6.5.4. Incorporating the BR-DEH into a metaheuristic framework

In order to gain a further improvement, the BR-DEH is then integrated in a metaheuristic framework. The biased randomisation introduced in the constructive BR-DEH ensures that every time the procedure is computed, it returns a different solution, introducing slight variations to the geedy one. For this reason, by incorporating the BR-DEH in a metaheuristic, a wider spectrum of solutions can be explored.

The metaheuristic proposed might be associated with an Iterated Local Search (ILT). The procedure starts by generating a purely greedy solution: this can be made using the BR-DEH with *beta* values (i.e. parameters of the geometric distributions) very close to 1. In this way, the algorithm starts from an already reasonably good solution. Then, until the maximum computational time *maxTime* is not elapsed, the *beta* values are reiterated, a new solution is generated, and eventually the best solution found so far is updated. At the end of

the procedure, the algorithm returns the best solution found. For evaluating a solution the analytical model presented in section 6.2 is used. The pseudocode for the metaheuristic is represented in Figure 6.17.

```
procedure main (requests, racks)
00 beta <- setHighBetas() %% beta parameters of the geometric close to 1
00 newSol <- BR-DEH(requests, racks, beta) %% greedy solution
01 bestSol <- newSol %% best solution found
02 while elapsed < maxTime: %% iter until available time is not elapsed
03   beta <- reiterateBetas() %% reiterate betas with a new random seed
04   newSol <- BR-DEH(events, racks, beta) %% A new solution
05   if cost(newSol) < cost(bestSol):
06      bestSol <- newSol
07   end if
08 end loop
09 return bestSol %% return the best solution found
end procedure
```

Figure 6.17. Pseudocode of the metaheuristic framework [4].

6.5.5. Computational experiments

To validate the proposed solution a comparison with 3 different benchmark solutions is presented. The selected benchmark are the following:

1. A pseudo-random solution.
2. A greedy algorithm.
3. The simulated annealing described in the previous section.

The pseudo-random solution (1) is obtained implementing the proposed BR-DEH and replacing all the selections made according to a geometric distribution with a selection made using a uniform distribution. This can be easily obtained running the BR-DEH with *beta* values (i.e. parameters of the geometric distributions) very close to 0. However, using very low values of beta parameters, the algorithm is not able to find a feasible solution. Because of this, a pseudo-uniform distribution is used instead, setting all values of beta parameters equal to 0.3. Helped by this expedient, the algorithm is able to find a feasible solution in most iterations, but, on the other hand, the uniform selection is partially respected. Conversely, the greedy algorithm (2) is obtained implementing the proposed BR-DEH with beta values very close to 1. Finally, the last benchmark is the algorithm described in the previous section and published in Bertolini et al. (2019).

The SLC-AS/RS used for tests is made of 3 racks (with 500 storage locations per each), 2 input points, and 2 output points. The experiments consist in the comparison of the total makespan (expressed in minutes) needed to complete all the operations of 12 different

requests lists with different complexity (i.e. 30/60/90/150 requests), and each algorithm is tested 3 times on each requests' list in order to control its reliability.

The parameters of proposed algorithm, which are essentially the betas of the geometric distribution used in the selection of racks (i.e. β_r) and bundles (i.e. β_b), are reiterated in each iteration according to trimmed gaussian distributions as suggested by Juan et al. (2010). The averages of these gaussian distributions (i.e. μ_r = 0.7 and μ_b = 0.9) are defined through a series of empirical experiments trying all the combinations for $\mu_r \in (0, 1)$ and $\mu_b \in (0, 1)$ with a step of 0.1. The standard deviation was set equal to 0.025 for both, but the tests do not show it to be that relevant. Conversely, concerning the simulated annealing, the set of parameters already used by Bertolini et al. (2019) in their case study is used.

The results are reported in Table 6.3. The BR-DEH incorporated in the metaheuristic framework always overtakes the greedy and the pseudo-random procedures. This is not a really relevant aspect and it is exactly what the author was expecting to see, but, more important, both the greedy and the pseudo-random procedures are not always able to find a feasible solution, while the proposed algorithm does. The greedy solution could be associated with the solution an expert manager could find by hand, and the proposed procedure is always outperforming it.

On the other hand, observing the simulated annealing, it is possible to state that the simulated annealing always provides a better solution in case of long requests' lists (e.g. 150 requests), while it is always outperformed by the proposed solution in case of short requests' lists (e.g. 30/60 requests). This is mostly due to the fact that the simulated annealing collects a batch of requests before processing them all together, while the proposed algorithm simply processes the requests according to a FIFO logic as soon as they arrive. In case of medium-size requests' lists (e.g. 90 requests) it is possible to see an exact intermediate situation in which the annealing returns on average a better solution, but, sometimes, observing the single tests, it is outperformed by the proposed solution. Another interesting aspect is that the simulated annealing has a greater variability if compared to the proposed solution, which is more reliable and always returns similar solutions. This happens because the annealing is not driven by a strong greedy behaviour and it is wasting a lot of iterations exploring not relevant solutions. On the other hand, the proposed procedure maintains a greedy approach introducing only slight variations, and this aspect makes it very reliable.

Table 6.3. Computational experiments [4].

Request list	Number of requests	Test	Single-test makespan [min]				Average makespan [min]			
			BR-DEH	Greedy	Random	Simulated Annealing	BR-DEH	Greedy	Random	Simulated Annealing
1	30 (9 input/ 21 output)	1	34.47	37.24	47.43	42.34	**34.57**	37.24	48.36	40.38
		2	34.85		50.97	40.08				
		3	34.38		46.69	38.72				
2	30 (12 input/ 18 output)	1	31.12	35.34	38.86	37.05	**31.03**	35.34	39.41	34.19
		2	31,03		39.73	32.33				
		3	30.68		39.64	33.19				
3	30 (12 input/ 18 output)	1	37.32	40.53	45.5	39.84	**37.32**	40.53	49.47	40.83
		2	37.17		54.91	39.65				
		3	37.47		48.01	42.99				
4	60 (9 input/ 51 output)	1	51.4	55.46	None	54.98	**51.51**	55.46	71.95	57.52
		2	51.87		71.95	56.3				
		3	51.25		None	61.27				
5	60 (12 input/ 48 output)	1	46.95	52.2	60.87	55.87	**46.86**	52.2	69.00	54.25
		2	47.43		75.46	52.2				
		3	46.2		70.67	54.69				
6	60 (26 input/ 34 output)	1	49.95	52.24	72.4	50.01	**48.67**	52.24	72.04	50.25
		2	47.73		69.37	50.78				
		3	48.33		74.33	49.98				
7	90 (42 input/ 48 output)	1	105.03	112.00	129.04	91.32	106.05	112.00	132.43	**99.5**
		2	106.55		131.79	93.78				
		3	106.58		136.47	113.39				
8	90 (28 input/ 62 output)	1	87.73	89.29	108.52	85.9	87.29	89.29	115.24	**86.69**
		2	88.25		118.72	85.9				
		3	85.9		118.48	88.27				
9	90 (35 input/ 55 output)	1	104.98	108.36	131.06	97.43	104.53	108.36	132.14	**98.78**
		2	104.65		132.73	98.88				
		3	103.97		132.62	100.03				
10	150 (38 input/ 112 output)	1	241.72	None	None	210.12	238.97	None	None	**220.81**
		2	242.9		None	235.08				
		3	233.3		None	217.23				
11	150 (73 input/ 77 output)	1	226.8	230.19	254.84	214.10	227.94	None	255.67	**214.55**
		2	229.67		260.21	214.10				
		3	227.35		251.97	215.45				
12	150 (59 input/ 91 output)	1	197.87	None	234.78	182.90	195.75	None	232.51	**182.95**
		2	194.5		235.65	183.05				
		3	194.88		227.11	182.90				

Concerning the computational time, the algorithm was implemented in Python 3.7, compiled using the CPython interpreter, and tested on a standard personal computer, Intel QuadCore i7 CPU at 2.4GHz with 8Gb RAM and Ubuntu 18.04 OS. Being CPython an interpreter, the Python program do not execute as fast as other compiled programs, such as those written in C or C++, although the execution of the algorithm is reasonably fast, even without a JIT compiler (Pypy - https://www.pypy.org/) or a multiprocessing optimization. The execution takes on average 0.03 seconds per iteration, thus, to explore for example 500 solutions it

takes on average 15 seconds, which is a reasonable computational time even for an implementation in production.

6.6. Introducing a dynamic behaviour

6.6.1. Introduction

To the author's best knowledge, most of the solutions proposed for throughput improvement in AS/RS are designed to boost it in a specific operating scenario, which is assumed to be the heaviest or, at least, the most frequent one. Hence, once implemented one of these solutions, the AS/RS follows a single operating scheme, that is kept unaltered even if the current operating conditions significantly deviate from the supposed ones. This is a crucial limit, as it leads to a static behaviour of the system, independent of changes in the context in which it operates. This drawback could be avoided by implementing a flexible operating policy, namely Dynamic Operating Framework (DOF).

The SLC-AS/RS, as well as any other automated warehouse, is subjected to two dominant flows: an input and an output flow. The input flow is related to the entering bundles that, after weight and shape controls, wait on an input point until they are taken and stored at the assigned location. The output flow, instead, concerns the outgoing bundles, which are taken from a storage location and gathered on an output point until the shipment.

Since the SLC-AS/RS operates as an interface connecting production and shipping areas (also known as yard), the input flow depends, mostly, on production rate, working hours, and efficiency of the production lines. Similarly, the output flow mainly depends on the frequency (of arrival) and size of transport trucks and on the customers' demand over time. Hence, the state of the system undergoes considerable changes, determined by the combination of several factors changing over time.

The DOF, tries to identify these changes so that the system can dynamically readjust its operating behaviour accordingly. The main idea is to iteratively monitor the state of both input and output queues and to infer, from their length, the current situation (or state) in which the system is.

6.6.2. Possible states of the system

A large number of operating states could be certainly defined, but, to the author's opinion, a four-level classification is ideal to assure ease of use and a smooth operating functioning; using too many states, instead, could generate system's nervousness and instability, due to

frequent and sudden changes in the adopted operating policy. The states in which a SLC-AS/RS can be found could be therefore classified as follows:

1. Lazy state - Input and output requests come in slowly and the system has enough capacity to process them smoothly. Hence, both the input and output queues are low or even null.
2. Production pushed state - Production rate is intensive and growing fast, but the number of withdrawal requests is low. Hence only the input queue is high and critical, whereas the output one remains low.
3. Sales pulled state – In this condition, due to the production push state, the number of withdrawal requests is high, the output queue is critical and rapidly growing. Conversely, the number of input requests is contained, and the input queue remains low.
4. Busy state - The system is subjected to intensive input and output flow, at a rate close or even higher of its nominal capability. All operations must be completed as soon as possible and both queues rapidly grow.

In order to identify the current system's state, a threshold level (or criticality value) must be defined both for the input and output queues. A general rule cannot be defined, as these levels strictly depend on the system under analysis. For instance, the output queue of a system that prepares the customers' orders one or even two days before their shipments will be necessarily longer than that of a system operating on a just in time basis (i.e., where orders are collected and shipped as soon as they arrive). Hence, in these two contexts, the critical length of the queue will be very different.

Anyhow, in this work, the following strategy is used. Concerning inputs, physical queue of the bundles accumulated in the input bays or in their nearby is used, and criticality is related to the maximum storage capacity of those stocking areas. Conversely, for the outputs, the number of pending customers' requests is used. Hence, not a physical but an information queue is used, and, for this reason, criticality is not related to space constraints, but to the expected time needed to fulfil all the request.

6.6.3. Dynamic Operative Framework

A well-designed system should take advantage of the above-mentioned situations, modifying its behaviour accordingly. Indeed, in each state, the needs of the SLC-AS/RS change, as explained below:

- Lazy state - The system can take advantage of this low-peak state to properly reorganize the stock. The system, in fact, is under stressed and has more handling

capacity than required. So, rather than remaining idle, the system can use part of its extra capacity to carry out additional movements, to reallocate stocked items in more suitable locations.

- Production pushed state - The system should prioritize input operations, to keep the pace imposed by the production department and to reduce the input queues as much as possible avoiding possible congestion.
- Sales pulled state - The system should prioritize the output operations, to comply with customer's demand, to avoid delays and to lower the output queues as much as possible.
- Busy state – Performance maximization and the minimization of the throughput time become the main goals of the system. All input/output operations are equally important and must be completed as soon as possible.

In each state the needs of the system change, and so specific *'allocation'* and *'interleaving'* policies should be considered. Allocation policy is the selection of the storage location where to store or to retrieve a bundle. The interleaving policy, instead, defines how input and output operations will alternate. The allocation and interleaving policies implemented in the DOF are listed in Table 6.4. Each one of them has been conceived to be dynamic, easy and quick to implement and, at the same time, effective in terms of system performance. They are also generic enough to be effectively applied to any layout; the only exception concerns the allocation policy used for the busy case, which is ideal when input and output bays are located at opposite sides and at opposite ends of the rack they serve. The proposed allocation policy, however, could be generalized to other layouts with minor modifications.

Table 6.4. Storage allocation and interleaving policies for each state considered in the DOF

STATE	ALLOCATION POLICY	INTERLEAVING POLICY
Lazy	Close to opposite Input operations are assigned to the locations closest to the output bays; output operations, instead, are assigned to the locations closest to the input bays. The aim is to empty all the locations close to the entrance, moving all stocked items in the proximity of the exit. This is akin to a reorganization of the stock, aimed to speed up the operations that will be made in the other three states.	Alternate Input and output operations are alternated. e.g. {IN, OUT, IN, OUT, …}.
Production Pushed	Close to bay Each operation is assigned to the location closest to the interchange point between the crane and the bay from which the bundle is coming from (or it is going). In case of input, all the empty locations (with enough space to store the entering bundles) are considered, and the one closest to the input bay is selected.	Priority to inputs Higher priority is given to input requests that are more critical. Until the system remains in this state, output requests are postponed. e.g. {IN, IN, …, IN, OUT, OUT, …, OUT}.

	In case of output, all locations containing a bundle like the requested one are considered, and the one closest to the output bay is selected.	
Sales Pulled	Close to bay The allocation policy does not change (i.e., same one used in the production pushed state). What changes is the interleaving policies as shown next.	Priority to outputs Higher priority is given to the outputs. e.g. {OUT, OUT, …, OUT, IN, IN, …, IN}.
Busy	Dual command cycle If possible, one input and one output operation are always performed in pairs. For this reason, the 'alternate' interleaving policy is used. Locations are assigned to minimise the travelling time. To this aim, since input and output bays are located at opposite ends of the rack: - the input operation is assigned to a location randomly chosen among the empty ones with enough space to store the entering bundle, - the subsequent output operation is assigned to the downstream location closest to the one assigned to the input. In this way a dual command cycle with a short 'internal displacement' (i.e., movement between input and output location) is performed. Should the input or output operations be finished, the 'Close to bay' policy would be used instead.	Alternate Input and output operations are alternated. e.g. {IN, OUT, IN, OUT, …}.

To summarize, in the busy (most critical) state, with the adopted allocation and interleaving policy the system operates at its maximum capacity, to reduce as much as possible the throughput time of both input and output operations. Conversely, in the lazy (less critical) state, the system uses parts of its handling ability to move the stocked items in proximity of the exit points and, at the same time, it frees the locations that are close to the input point. In this way a sort of reallocation is brought about. Finally, in the intermediate states (i.e., production pushed, and sales pulled) the system speeds up the execution of input or output operations, depending on the current needs.

As anticipated above, the DOF infers the current state of the system, from the direct observation of the length of the input and output queues and, based on that, it selects a suitable operating policy. The selection is made using a set of '*if … then*' rules based on three threshold levels: (i) a low threshold h, (ii) a high threshold H, and (iii) a critical threshold C, where $h < H < C$. Letting i be the number of entering bundles and letting o be the number of pending retrieving requests, the following set of four rules is therefore used to figure out the current state:

1. **If** $i \leq h$ **and** $o \leq h$, **then** the system is in the lazy state;

2. **If** $i > H$ **and** $o > H$, **then** the system is in the busy state;
3. **If** $i > C$ **or** $o > C$, **then** the system is in the busy state;
4. **If** the previous conditions are all false **and** ($i > h$ **or** $o > h$), **then** the system is in an intermediate state and the longest queue determines whether it is production pushed or sales pulled. Hence, if $i > o$ the system is production pushed, otherwise it is sales pulled.

A last element that must be defined concerns the frequency with which these rules should be evaluated. Ideally, the queues could be continuously monitored and the adopted policy could be changed any time a new rule triggered. Yet, this would presumably generate instability and nervousness, as the system would be forced to change its operating functioning too many times and too often. To mitigate this potentially detrimental phenomenon, a discrete-time control is used. It is based on two time-intervals: a long time interval (i.e. Δ) and a short time interval (i.e. δ). Given *t* be the time when the last state transition took place and the operating policy was changed. The system preserves the same operating policy at least for Δ units of times when, at time (*t* + Δ), the queues are checked, and the rules are re-evaluated. If a transition takes place, the system stays in the new operating state at list for other Δ units of times, conversely if no change in the system's state is observed, the state is recontrolled every δ time units until the state does not change. As Δ assures a minimum interval of stability avoiding the nervousness of the system, the aim of δ is exactly the opposite one avoiding excessive stillness.

6.6.4. Case study design

To test the DOF, an extensive numerical campaign, based on discrete event simulation was made. To this aim the DOF was implemented in Python 3, compiled using the CPython interpreter, and tested on a standard personal computer. It was then integrated in a discrete event simulation model of the system written in the same programming language.

Concerning the warehouse and the layout selection, the decision is based on the attractiveness of the proposed solution. The objective of this section is not to define an optimal policy that maximizes performance in a specific operating environment and for a specific layout. Conversely, the aim is to evaluate, in a more general sense, the effectiveness and the benefits of a dynamic approach, which updates the adopted policy based on the observed system state. Hence, the use of a complex layout with more racks may involve further complications, such as the rack selection policy, and could even

generate bottleneck shiftiness, between the shuttle and the crane, as the shuttles would have to serve more than one rack. These aspects may affect the results, making it difficult to understand the actual impact of the proposed DOF. For all these reasons, the SLC-AS/RS used to validate the DOF has a single rack. For the same reason, also the position of the input/output points has been based on the most common industrial applications. Specifically, we placed the input and output bays at two opposite ends of the rack, and on opposite sides, as clearly shown in the planar and front view of Figure 6.17. This solution is indeed very common when the warehouse is placed in between the production and the shipping area. Loading of transport trucks can take place close to the output bay, without interfering in any way with the forklifts and the workers that are uploading the entering bundles on the input bay.

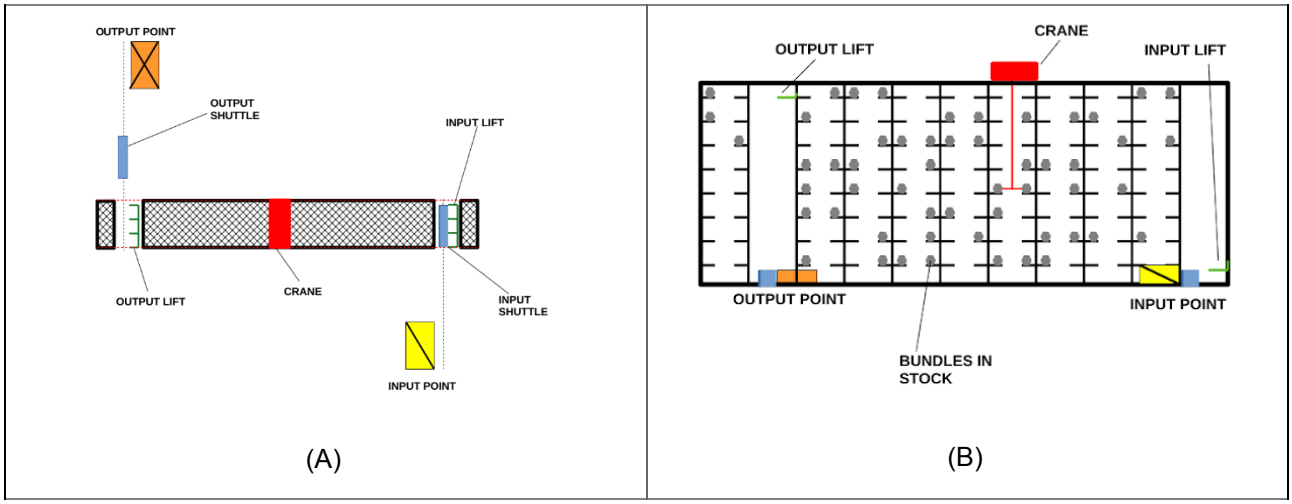


Figure 6.17. Planar (A) and frontal (B) views of the SLC-AS/RS considered in the case study [5].

In all tests, three different classes of codes A, B and C, in proportion 7 ÷ 2 ÷ 1, were used. So, every time a request is generated, it is assigned to a class according to following percentages: *(i)* 70% of operations refer to a class A code, *(ii)* 20% of operations refer to a class B code and *(iii)* 10% of operations refer to a class C code. Inside each class, just one code is used. Each code has a different length, expressed in number of occupied/required shelves: the code of class A occupies eight shelves, *(ii)* the code of class B occupies five shelves, and *(iii)* the code of class C occupies four shelves. We considered storage locations (inside the rack) with twelve shelves each.

Requests were generated in a way that accurately reproduces a dynamic environment, with interarrival times that periodically changes accordingly to a predefined stochastic law. The generation of requests is based on an exponential distribution where the shape parameters

of input and output inter arrival times, namely λ_i and λ_o, at discrete time intervals, can take either a low (λ_L) or a high value (λ_H). In other words, for exponentially distributed interarrival times, the expected interarrival time is given by $E[\tau] = \left(\frac{1}{\lambda}\right)$, λ_L and λ_H will be used to generate the lazy and the busy state, respectively. More precisely, letting μ [operations/hours] be the average handling capacity of the AS/RS, it is possible to combine the different values of λ_i and λ_o, to set the desired average utilisation levels of the system $u = \frac{(\lambda_i+\lambda_o)}{\mu}$. This is shown in Table 6.5 where, for each possible system's state, the combination of the lambdas and the utilisation level that will be used in the simulations are reported.

Table 6.5. System's states and corresponding lambda values [5].

System State	Input λ_i	Output λ_o	Util. Level u
Lazy	λ_L	λ_L	80%
Production pushed	λ_H	λ_L	92.5%
Sales pulled	λ_L	λ_H	92.5%
Busy	λ_H	λ_H	105%

Hence, to test the DOF in a challenging situation, the system is 'over saturated' during busy periods (with a utilisation rate higher than one). This condition cannot be sustained for an indefinite period and, to restore its standard operation functioning, the system has to take advantage of the other low peak states. Hence, the ability to face such a high peak period will provide an additional indication about the robustness of the DOF.

Concerning state transitions, we assumed the same transition probability $p_{ij} = 25\%$ for each possible pair of states *i* and *j*; as clearly shown in the state transition diagram of Figure 6.18.

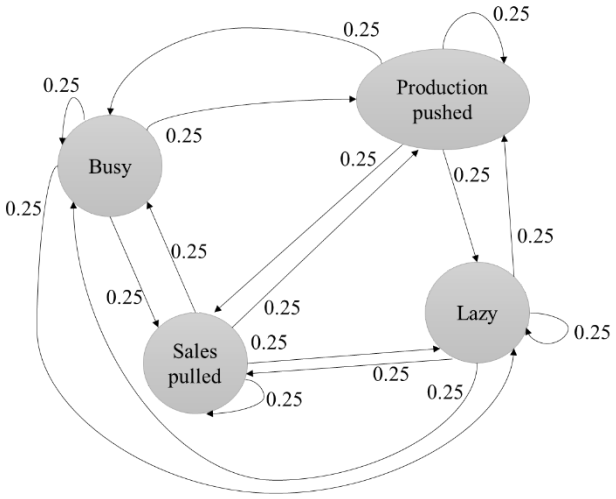


Figure 6.18. State transition diagram [5].

Also, state transitions occur every *T* time units, that is the system remains in a certain state for *T* time units and after that time it has an equal probability to remain in the same state or to switch to any one of the other three states. Operatively, every *T* units of simulated time, a random number *r* is extracted, and the state remains unaltered if $r \leq 0.25$, otherwise it changes, and input and output lambdas are modified accordingly.

6.6.5. Tests design

Eight alternative configurations of the SLC-AS/RS were considered and a total of forty simulation runs were executed, i.e., five repetitions for each configuration. Each configuration corresponds to a different combination of three design parameters, each one characterized by a low and a high value. The parameters are the following ones:

- The rack length *L (low: 60 meters; high: 120 meters)*.
- The shape factor b introduced by Bozer and White (1984) (low: 0.13; high: 0.25).
- The minimum time *T* spent in a state before a transition occurs (low: 4 hours; high: 8 hours).

The values defined for *L* and *b* are a good estimate of the minimum and a maximum value that can be found in common industrial application. Concerning *T*, this time was fixed to 4 hours (half of a standard working shift) for each state, except for the busy one. In fact, for the busy condition, we also tested a minimum dwell time of 8 hours, with the goal to see how well it performs in this hypercritical situation too.

Lastly, the lambdas, needed to assure the desired utilisation levels of Table 6.2, were empirically defined performing a specific simulation for each considered configurations. As

noted before, in fact, anytime the dimensions of the rack are modified, the average throughput time and, consequently, the corresponding handling capacity, inevitably change. Specifically, for each configuration (i.e., rack's length and shape factor) a simulation with unlimited input and output requests was run and the total number N_{TOT} or performed input/output operations was recorded. This allowed the estimation of the maximal handling capacity of the system, obtained as $\mu_{max} = \left(\frac{N_{Tot}}{S_{time}}\right)$, where S_{time} is the length of the simulation run. Next, the lambdas were simply calculated as follows: $2\lambda_L = 0.8\mu_{max}$ and $2\lambda_H = 1.05\mu_{max}$. All the values used in each simulation run are summarized in Table 6.6.

Table 6.6. Configurations used in the current case study [5].

Config.	Length L [m]	Shape factor b [-]	Dwell Time T [h]	Lamb. High λ_H [s^{-1}]	Lamb. Low λ_L [s^{-1}]
1	120	0.13	4	$\frac{1}{341}$	$\frac{1}{450}$
2	120	0.25	4	$\frac{1}{513}$	$\frac{1}{675}$
3	60	0.13	4	$\frac{1}{219}$	$\frac{1}{287}$
4	60	0.25	4	$\frac{1}{265}$	$\frac{1}{350}$
5	120	0.13	8	$\frac{1}{341}$	$\frac{1}{450}$
6	120	0.25	8	$\frac{1}{513}$	$\frac{1}{675}$
7	60	0.13	8	$\frac{1}{219}$	$\frac{1}{287}$
8	60	0.25	8	$\frac{1}{265}$	$\frac{1}{350}$

Concerning the threshold levels the following values {*h* = 2; *H* = 4; *C* = 7} were used both for the input and output queue, and for each one of the eight considered configurations. The critical level (*C*) corresponds, approximately, to the 75-th percentile of both queues when the system operates in dual command cycle and $\lambda_o = \lambda_i = \lambda_H$ (i.e., busy state). Similarly, the values of the low (*h*) and high (*H*) thresholds correspond, approximately, to the mode and to the 75-th percentile of both queues when the system operates according to a 'close to opposite' policy and $\lambda_o = \lambda_i = \lambda_L$ (i.e., lazy state).

Finally, for each test, 6 weeks are simulated, with 8 working hours per day and 5 days per week. Also, at the beginning of each test, a set up procedure is carried out to prefill the warehouse till the 60% of its storage capacity.

6.6.6. Accuracy tests

Before testing the proposed DOF, in the author's view, it was important to verify if and how the DOF is able to understand, in real time, the current state of the system. To this aim, for each simulation run, the percentage of time the system was in a certain state (*real state*) and the percentage of time the system operated assuming to be in that state (*supposed state*) were recorded. Obtained results are shown in Table 6.7, and the estimations made by the DOF are fairly accurate. Moreover, the biggest errors only occur in the configurations where the largest dimensions of the rack increase the variance of the travel time for the crane; or, otherwise, in the last four configurations, characterized by a longer permanence in the busy state.

Table 6.7. Comparison between real permanence and operative behaviour of the system in each state [5].

			Tests												
			Test 1		Test 2		Test 3		Test 4		Test 5		Average		Delta
			Real	Supposed	Real	Supposed	Real	Supposed	Real	Supposed	Real	Supposed	Real	Supposed	
Configurations	1	lazy	40%	27%	30%	31%	40%	48%	30%	35%	40%	40%	36%	36%	0%
		sales pulled	20%	18%	10%	29%	10%	17%	30%	25%	30%	23%	20%	22%	2%
		production pushed	10%	20%	20%	15%	50%	28%	0%	23%	10%	20%	18%	21%	3%
		busy	30%	35%	40%	25%	0%	7%	40%	17%	20%	17%	26%	20%	6%
	2	lazy	20%	44%	40%	37%	10%	14%	10%	14%	20%	29%	20%	28%	8%
		sales pulled	30%	18%	20%	30%	20%	10%	30%	7%	40%	20%	28%	17%	11%
		production pushed	20%	30%	40%	30%	30%	5%	20%	8%	20%	20%	26%	19%	7%
		busy	30%	8%	0%	3%	40%	71%	40%	70%	20%	31%	26%	37%	11%
	3	lazy	20%	10%	20%	19%	30%	25%	40%	21%	20%	19%	26%	19%	7%
		sales pulled	20%	20%	0%	20%	30%	18%	20%	23%	10%	20%	16%	20%	4%
		production pushed	30%	15%	20%	18%	20%	17%	20%	28%	30%	21%	24%	20%	4%
		busy	30%	55%	60%	43%	20%	40%	20%	28%	40%	40%	34%	41%	7%
	4	lazy	50%	27%	50%	41%	20%	28%	20%	31%	20%	13%	32%	28%	4%
		sales pulled	20%	33%	0%	23%	30%	25%	30%	28%	0%	15%	16%	25%	9%
		production pushed	10%	20%	40%	31%	30%	12%	30%	24%	40%	5%	30%	18%	12%
		busy	20%	20%	10%	6%	20%	34%	20%	17%	40%	67%	22%	29%	7%
	5	lazy	22%	23%	18%	18%	32%	32%	20%	27%	13%	10%	21%	22%	1%
		sales pulled	17%	22%	18%	15%	13%	30%	20%	23%	23%	10%	18%	20%	2%
		production pushed	17%	25%	20%	17%	22%	24%	17%	20%	13%	11%	18%	19%	2%
		busy	45%	30%	43%	50%	33%	14%	43%	30%	50%	68%	43%	38%	4%
	6	lazy	18%	7%	15%	4%	22%	21%	25%	12%	23%	11%	21%	11%	10%
		sales pulled	12%	2%	28%	3%	25%	25%	15%	15%	18%	12%	20%	11%	8%
		production pushed	17%	5%	23%	3%	27%	23%	25%	16%	22%	11%	23%	12%	11%
		busy	53%	86%	33%	89%	27%	31%	35%	56%	37%	66%	37%	66%	29%
	7	lazy	25%	15%	20%	13%	28%	2%	18%	1%	20%	3%	22%	7%	15%
		sales pulled	22%	16%	27%	17%	18%	4%	18%	1%	27%	4%	22%	8%	14%
		production pushed	20%	14%	17%	17%	17%	4%	17%	2%	10%	7%	16%	9%	7%
		busy	33%	55%	37%	53%	37%	90%	47%	96%	43%	86%	39%	76%	37%
	8	lazy	32%	13%	22%	3%	27%	17%	22%	21%	22%	2%	25%	11%	14%
		sales pulled	18%	11%	12%	3%	15%	16%	27%	19%	18%	3%	18%	10%	8%
		production pushed	17%	12%	13%	4%	25%	17%	22%	18%	12%	3%	18%	11%	7%
		busy	33%	64%	53%	91%	33%	49%	30%	42%	48%	92%	39%	68%	28%

6.6.7. Comparison with static policies

As stated above, the goal of this validation process is to evaluate the effectiveness of a dynamic operative framework, which can autonomously react to changes of the system's state. The author is aware that the policies are sub-optimal and could be improved, leading additional benefits and performance gains. However, the real objective is to demonstrate that, given a set of reasonable allocation policies, the system where the DOF is implemented outperforms a system characterised by static behaviour independent of boundary conditions.

For this reason, the DOF is not compared with completely different algorithms presented in literature, but instead the comparison is made between a SLC-AS/RS controlled by the DOF, with the same system operating accordingly to a static random allocation policy, or to a static dual command cycle with 'alternate' interleaving (i.e., the same policy used by the DOF in busy states). Also, both benchmarks are coupled with a class-based reorganization of the stock, executed daily, during non-working shifts, for a total duration of eight hours. The comparison is based on two aspects: (i) the average cycle time and (ii) the overall distance covered the cranes that, as discussed, is the element accounting for most of maintenance and energy costs. Concerning the average cycle time, this performance indicator is computed both for the input and output operations executed exclusively during real busy states. As explained before, only during busy states the system really needs to be fast; in all the other states (especially in the lazy one), part of the maximal handling capacity can be used to reorganize the stock. It is thus obvious, and indeed wanted, that in sub-critical states the average cycle time could deteriorate slightly.

Obtained results are presented in Table 6.8, where the cycle time is expressed in seconds and corresponds to the average made for the five 5 tests executed for each configuration. As the table clearly demonstrates, the DOF ensures very similar performance, even if there is no reorganization of the stock during non-working shifts.

Table 6.8. Performance comparison of the proposed DOF with the other two policies [5].

Config.	DOF				Random with reorganization of stock				Busy policy with reorganization of stock			
	Input		Output		Input		Output		Input		Output	
	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.
1	193	6	151	8	199	2	153	3	195	2	128	4
2	314	11	243	13	341	6	262	1	341	4	231	5
3	140	6	91	4	139	2	106	2	138	2	88	3
4	162	5	113	5	194	7	123	4	196	8	112	2
5	207	13	142	7	199	2	158	2	198	2	131	1
6	337	10	235	3	346	3	257	2	346	1	233	2
7	149	4	89	7	142	1	109	3	142	1	88	2
8	180	5	108	4	194	2	126	4	197	3	112	2

Concerning distances (Table 6.9), as the DOF is employed, the crane runs a longer distance during the working shift. This fact was expected because the stock is reorganized during the working shifts, taking advantage of the lazy moments. The increase in 'day-time' distance is, however, of modest entity and it is largely offset by the absence of 'night-time' distance. After all, as table 6 shows, the DOF allows an average reduction of the total covered distance of 29% and 22%, relatively to the random and to the static busy policy. To the author's best knowledge, this might be considered as an aspect of paramount importance, with relevant benefits in terms of energy saving and reduced maintenance costs.

Table 6.9. Distance run by the crane in each configuration, for each of the analysed policies [5].

Config.	DOF		Random with stock reorganization		Busy policy with stock reorganization	
	Working shift distance [km]	Non-working shift distance [km]	Working shift distance [km]	Non-working shift distance [km]	Working shift distance [km]	Non-working shift distance [km]
1	83	0	72	40	61	36
2	48	0	45	38	40	37
3	66	0	67	11	58	12
4	52	0	53	25	45	26
5	519	0	458	297	396	259
6	292	0	280	280	249	277
7	427	0	428	50	360	89
8	342	0	336	129	289	145

References

Arcus, A. L. (1965). A computer method of sequencing operations for assembly lines. *International Journal of Production Research*, 4(4), 259-277.

Belloso, J., Juan, A. A., Faulin, J., & Serrano, A. (2015). Using multi-start biased randomization of heuristics to solve the vehicle routing problem with clustered backhauls. *In Proc. ICAOR* (p. 16).

Bertolini, M., Mezzogori, D., & Zammori, F. (2019). Comparison of new metaheuristics, for the solution of an integrated jobs-maintenance scheduling problem. Expert Systems with Applications, 122, 118-136.

Bozer, Y.A., and White, J.A. (1984). Travel-time models for automated storage/retrieval systems. *IIE Transactions,* 16(4): 329-338.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1), 53-66.

Fikar, C., Juan, A. A., Martinez, E., & Hirsch, P. (2016). A discrete-event driven metaheuristic for dynamic home service routing with synchronised trip sharing. *European Journal of Industrial Engineering*, *10*(3), 323-340.

Glover, F. (1989). Tabu search—part I. *ORSA Journal on computing*, 1(3), 190-206.

Glover, F. (1990). Tabu search—part II. *ORSA Journal on computing*, 2(1), 4-32.

Grasas, A., Juan, A. A., Faulin, J., de Armas, J., & Ramalhinho, H. (2017). Biased randomization of heuristics using skewed probability distributions: a survey and some applications. *Computers & Industrial Engineering*, 110, 216-228.

Kirkpatrick, Gelatt, and Vecchi (1983). Optimization by simulated annealing. *Science*. 220, 671–680.

Juan, A. A., Faulin, J., Ruiz, R., Barrios, B., & Caballé, S. (2010). The SR-GCWS hybrid algorithm for solving the capacitated vehicle routing problem. *Applied Soft Computing*, 10(1), 215-224.

Juan, A. A., Pascual, I., Guimarans, D., & Barrios, B. (2015). Combining biased randomization with iterated local search for solving the multi-depot vehicle routing problem. International *Transactions in Operational Research*, 22(4), 647-667.

Lai K. K. and J. W. M. Chan (1997). Developing a simulated annealing algorithm for the cutting stock problem. *Computers and Industrial Engineering* 32, 115–127.

Nadir, H., Zaki, B., and S. Latéfa (2012). Metaheuristic based control of a flow rack automated storage retrieval system. *Journal of Intelligent Manufacturing*. 23, 1157–1166.

Bian, Dai, Cao, and Chang (2018). Research on path planning of stacker based on improved simulated annealing algorithm. *Pap. Asia*, 1, 106–110.

Yang, Miao, and Qin (2013). Job scheduling optimization in multi-shuttle automated storage and retrieval system. *Jisuanji Jicheng Zhizao Xitong/Computer Integr. Manuf. Syst. CIMS*, 19, 1626–1623.

Roodbergen, K. J., & Vis, I. F. (2009). A survey of literature on automated storage and retrieval systems. *European journal of operational research*, *194*(2), 343-362.

Tonge, F. M. (1965). Assembly line balancing using probabilistic combinations of heuristics. *Management Science*, 11(7), 727-735.

7. Managerial Decisions

The solutions presented in this work are mainly addressed to the AS/RS users, i.e. companies that operate in the steel industry and want to improve their logistics through algorithms and procedures, or through the installation of an automated warehouse. However, to the author's view, it is important to provide support to the manufacturers of AS/RS too, supporting the managerial decisions which concern the warehouse design and realisation. This would ensure a reduced lead time in supplying a new warehouse, a better organisation of work, an improved relationship between the seller and the warehouse user. This chapter is therefore dedicated to provide practical tools to the managers involved in the realisation of a new AS/RS.

7.1. Overview of managerial problems in the steel sector

In the steel sector, automated warehouses are generally designed to store heavy and bulky products such as metal dies, long tubes, billets, or long metal bars' bundles up to twelve meters long and from 1 to 5 tons heavy; for these reasons, their size can be 3 or 4 times bigger than classic AS/RS. Moreover, the pallets are rarely used and the handled unit loads are not standard, and this makes each warehouse unique and different from the others. There are also additional complications due to the plant configuration where the AS/RS is going to be installed, the structural needs, the marketing decisions, and so on. For all these reasons, the realisation of an AS/RS is typically characterised by long lead time, extreme variability, several drawbacks, a high number of tasks, and big investment costs, which are exactly the characteristics of any Engineering-To-Order (ETO) project. The realisation of an AS/RS, especially if designed for the steel sector, is therefore to consider an ETO project, and, as such, the responsibility for its achievement is in the hands of a Project Manager (PM). Consequently, the practice of project management assumes a key role (Vanhoucke, 2012). The knowledge a PM is required to have might be summarised in the following "ten ingredients for a successful project" already mentioned by Bertolini, Neroni, and Zammori (2020):

1. Knowledge of the scope. The scope of the project needs to be clear and the PM needs to have full control over decisions, which must be focused on goal-reaching.
2. Knowledge of the team. The PM needs to know the abilities and weaknesses of each member of the team. This allows a proper attribution of responsibilities and it is essential to generate cohesion, consensus and motivation.
3. Quick response to changes. The PM needs to be flexible and reactive; as the project progresses, even the main goal could change depending on the will of the stakeholders and the evolution of the boundary conditions.
4. Division of risk inside the team. Nobody can deal with all the problems that may occur during the project lifetime; hence, the PM needs to be able to separate risks and

responsibilities. A proper division of risks responsibilities helps team members to grow and improve their managerial skills.

5. Creation of a baseline. The project's baseline (defined as the original scope, cost and schedule of the project) must be completely defined before the project execution starts. To support control and performance analysis, the baseline should include a Gantt Chart (or timeline) and a detailed schedule.
6. Promotion of experience growth. People, their skills and the collective knowledge of a company are the primary elements of success. Because of this, continuous training, job rotation, and training on the field should be promoted through the creation of a dynamic and multi skilled team. Particular emphasis should be given to juniors' development, whose training should be seen as a "project inside the project".
7. Hold frequent meetings. Quick and frequent meetings are important to keep the project on track, to collect feedback from the stakeholders and to share opinions/information within the team. Every member is more motivated when he/she perfectly knows which are the goals and what he/she is required to do and why.
8. Quality over quantity. Defending the project's scope and its quality are generally more important than completing the job in time or under budget. To this aim it is essential to have a well-experienced team, with multidisciplinary competences, where everybody has specific tasks and responsibilities.
9. Continuous detection. Monitoring and measuring progress, wastes, and losses is essential for improvement: it is not possible to improve what cannot be measured.
10. Continuous planning. As the project proceeds, many unexpected problems can occur and flexibility of the PM in rearranging the plan can make the difference between success or failure.

The aim of this chapter is to provide the managers with practical tools to help them achieve the above results. In particular, the provided tools focus on points 3, 4, 5, 6, 8, and 9. At first, the theme of resources allocation, also known as project staffing, is concerned, because, according to Certa et al. (2009), the choice and allocation of renewable resources to tasks is one of the most relevant aspects of PM, which involves the division of risk inside the team (i.e. point 4), the creation of a project baseline (i.e. point 5), and the promotion of skills groth (i.e. point 6). It also requires a quick response to changes and a good flexibility to (i.e. point 3). Good choices in the allocation of resources to the project's tasks may do the difference between failure and success of the project. Indeed, only if tasks are assigned to the right resources and workloads are evenly distributed, the project can be completed without running over-time or over-budget (Gollenbeck-Sunke and Schultmann, 2010).

Then, a new lean tool called Project Time Deployment (PTD) is presented, whose objectives are to classify, identify, and eliminate losses in ETO projects. Topics such as quality (i.e. point 8) and continuous monitoring (i.e. point 9) are therefore considered. The PTD is a lean tool presented for the first time in this work which consists in an integrated version of the Manufacturing Critical-path Time (MCT) mapping approach originally proposed by Suri (1998), with the Manufacturing Cost Deployment (MCD) to quantify the inefficiencies proposed by Yamashina and Kubo (2002).

Both solutions have been validated in a real industrial environment comparing the provided results with those obtained by the current project managers employed in the respective companies. Furthermore, both solutions constitute valid tools in all ETO environments, not only in those related to the realisation of an AS/RS for the steel sector.

7.2. Project Staffing

7.2.1. Introduction

The project staffing, or resource allocation, has always attracted much interest in the field of operational research, where it is generally referred as a Resource-Constrained Project Scheduling Problem (RCPSP). In scientific literature, the problem is typically formulated as a mixed-integer programming model, with the objective of makespan minimization, and it has been tackled with several algorithms (Herroelen, 1972), including: integer programming (Pritsker et al., 1969), dynamic programming (Carruthers and Battersby, 1966), implicit (Christofides et al., 1987) and bounded enumeration (Patterson and Huber, 1974). The above-mentioned techniques are still in use but, starting from the ‘90s, they have been progressively substituted with modern heuristics and metaheuristics (Zamani, 2017), as shown in the comprehensive literature reviews by Viana and Pinho De Sousa (2000), Hartmann and Briskorn (2010) and Liao et al., (2011). Many additional constraints have been considered so far, both for deterministic (Berthaut et al., 2014; Montoya et al., 2014; Maghsoudlou et al., 2016), and stochastic environments (Elmaghraby and Ramachandra., 2012). Recently, some authors have also considered tasks that, to be executed, require resources with specific skills and/or expertise (Vanhoucke, 2014; Myszkowski et al., 2018). This problem, known as the Skilled Resources Project Scheduling Problem (SRPSP), is dealt with in different ways, depending on the way in which the mismatches (or gaps) between resources’ skills and tasks’ requirements are considered. Most of the authors, as for Liu and Wang (2012) or Heimerl and Kolisch, (2010), assume that the higher the mismatch, the longer the duration will be. Conversely, other authors classify team members in terms of skills, just to make sure that the most skilled and experienced resources work, as supervisors, together with the less skilled ones (Joshi et al., 2018). Anyhow, despite the numerous models and different approaches found in the scientific literature, almost none of them have ever found a real practical application. Although these approaches lead to the optimal solution, very few (if any) are currently implemented in modern project management softwares, so the allocation of renewable resources to tasks is done

directly by the PM without any significant support (Kastor and Sirakoulis, 2009). This is probably due to the complex and rigid mathematical formulation nature of the proposed approaches, that pose a very tight bond on the precision of the input data, and contrast with the key features of project management. In fact, because of the project's dynamism and uniqueness, the availability of accurate and reliable data is almost utopic. This does not mean that algorithmic and mathematical approaches should be abandoned, but it is imperative not to rob the PM of his/her flexibility and decision-making capabilities. The PM needs an easy, reconfigurable and rapid scheduling tool, offering valuable aid in the formulation of a feasible solution, even when data are uncertain and partial. In order to do this, the tool proposed below defines an allocation schedule for multi-skilled resources by adopting a set of rules which are completely customizable by the PM. These rules have been defined observing some of the ingredients for a successful project described above, and the shortcomings of the algorithms proposed in the literature. Thus, this paper abandons the over-optimistic hope to find a global optimum, by conjointly considering tasks' scheduling and resources assignments. Conversely, in line with the Dynamic-Scheduling-Approach (Vanhoucke, 2013), our approach starts from a predefined project's timeline and, next, it generates the resource allocation following a set of logical rules reproducing the reasoning of a human expert.

The rational elements of this choice for the proposed approach are the following: *(i)* solutions generated by a rule-based heuristic are robust and less sensitive to input data uncertainty, *(ii)* the execution time becomes extremely short, *(iii)* the PM can generate alternative schedules by simply modifying the objectives functions and/or the allocation criteria on which the rules are based, *(iv)* the generated schedule is rational, clear and easily interpretable.

The last point is particularly important because PMs are generally reluctant, legitimately, to accept the solution generated by a "*black box*". Conversely, solutions obtained using an algorithm that mimics the logical reasoning of an expert, should be more easily accepted as a standard practice of project management.

7.2.2. Detailed description of the problem

As every project management practitioner knows, one of the main assignments of the PM is to plan the project execution, in full compliance with the requirements and general objectives defined by the project's sponsor, in the business case. More precisely, as indicated by the PMI Standard (2017), this process can be split in three main steps:

- The objective must be clearly defined, especially in terms of quality and cost Pollack et al., 2018).

- The Work Breakdown Structure (WBS) is built and the total effort of the project is split into smaller work packages, and then split again till tasks are obtained. At this time, the PM defines the tasks duration, workload, and constraints, obtaining in this way the "*Uncapacitated Project Plan*".
- Finally, the resource allocation is carried out and it is usually performed in concomitance with tasks scheduling, since the possibility an activity is carried out is strictly related to the availability of the resources it requires. Additionally, overallocation and extra time should be avoided or minimized.

Hence, in a context like the one described, which generically recurs in every project, where can an automated procedure for resources allocation generate added value? Of course, at step three. Indeed, once tasks' requirements have been defined, the highly demanding and time-consuming resource assignment process, turns into a purely quantitative problem that could be easily automatized, with great benefits for the PM. Conversely, the first two steps are extremely dependent on the capacity of the PM, whose expertise can make the difference between failure and success. For this reason, both steps, in the author's view should be left to the discretion and to the judgment of the PM; trying to automate them both with an automatic procedure would be a nonsense, as the creation of the WBS, the definition of task as well as the conceptualization of the ideal teams depends, mostly, on subjective issues such as experience, risk aversion, company' policy and expectations.

As explained, the problem of resources allocation is the third phase of a long planning procedure: the PM, in this phase, takes as input a list of renewable multi-skilled resources, a list of pre-scheduled tasks (typically organized in a Gantt chart) and has the objective to generate a viable and feasible allocation. Here multi-skilled is used to refer to endowing multiple skills into a single worker; this implies promoting a wide range of skills and knowledge in workers to make them flexible and efficient on a variety of different tasks (Marzouk, 2009). For a proper assignment, workers should be classified and qualified in terms of *"skill levels"*, values representing the ability and experience of the resource in a specific field, either technical (e.g. coding and programming, data analysis, planning, design, etc.) or behavioral (e.g. communication ability, team-working, conflict resolution, etc.). Furthermore, as in the works by Pawiński and Sapiecha (2016) and by Zheng et al. (2017), resources can also be classified as "*senior*" and "*junior*", depending on their experience and years of employment. Typically, the hourly rate of a senior will be higher than that of a junior and the skill levels of a senior will be equal or greater than that of a junior in all, or most areas of expertise.

Concerning feasibility, the allocation should respect cost, quality and time requirements (Zid et al., 2020). Concerning the cost, we refer to the direct cost of the renewable resources, which should be kept at a low level; this indirectly means avoiding, if possible, resources over allocation and/or overtime. Concerning the quality of a task (or of the whole project), it is not that easy to measure,

although, it is possible to state that it depends mostly on the experience of resources who worked on it (Belout, 1998). According to this hypothesis, given a couple of resource-task where the resource is the person who completed the task, we can consider the quality level respected when the gap in skills between level required by the task and level owned by the resource is under a specific threshold. More in detail, depending on the skills gap, and relatively to a generic task, a resource can be classified as:

- *Ideal*, if the skill gap equals zero.
- *Over-skilled*, if his/her skill level is higher than that one required by the task (positive gap).
- *Under-skilled*, if his/her skill rate is lower than that one required by the task (negative gap).

Under-skilled and over-skilled resources, if the gap does not exceed a certain threshold, can be assigned to the task, and this concept is formalized as *productivity,* a percentage which implicates how easily a resource can deal with a task (Brusco and Johns, 2007). As we will see later on, productivity can be defined as a function of the skills gaps or, otherwise, it can be directly decided by the PM. Anyhow, relatively to a generic task, the following constraints should be fully respected:

- Productivity must equal one, for ideal resources.
- Productivity must be greater than one, for over skilled resources.
- Productivity must be lower than one, for under under-skilled but assignable resource.
- Productivity is zero, if the resource is unassignable.

Concerning the time, the original schedule is considered "*frozen*" and cannot be altered in any way. Tasks duration, starting and ending times are fixed and, consequently, tasks cannot be shifted either ahead or backwards in time. This assumption, which may seem overly restrictive, is typical for design firms and, more in general, for Engineering-To-Order (ETO) companies. Indeed, when order is won, the proposal, submitted in response to a call for tender, becomes binding and deadlines, milestones and deliverables must be fulfilled. If the quote is accepted and the project is commissioned, the plan can no longer be changed: payments and/or penalties are linked to dead-lines compliance and the respect of time becomes paramount.

The objective of the proposed solution is not only to allocate resources so to keep costs down and to find a good matching among resources' skills and tasks requirements, but it is also to foster job motivation and to get a harmonious and continuous growth of team members (Brenner, 2007).

7.2.3. Effort, productivity, and improvement

Productivity has a direct impact on the Planned Total Work (PTW) or effort, a value that quantifies the cumulated number of working hours needed to complete a certain task. It is evident, in fact, that the same task could be completed with less effort by over-skilled resources and with more effort by under-skilled ones. More precisely, according to the Project Scheduling Formula (given in Eq. (7.1)), PWT can be obtained by multiplying the task duration for the assignment levels (or allocation percentages) of all resources assigned to the task.

(7.1)

$$PTW_t = d_t \bullet \sum a^*_{r,t}$$

Where:

- d_t is the fixed duration of task *t*, as defined in the uncapacitated plan;
- $a^*_{r,t}$ is the percentage allocation (or assignment level) of resource *r* to task *t*;
- R^*_t is the ideal resource team the PM supposed to use, when he/she developed the uncapacitated plan;
- The asterisk (*) is used to denote an ideal resource or an ideal assignment.

Concerning $a_{r,t}$, this percentage quantifies the level of usage of resource *r* on task *t*, relative to the maximum amount (or capacity) available for resource *r*. Hence, the percentage allocation equals one in case of full-time engagement, it is lower than one for part-time engagement and it is higher than one in case of overtime. We also note that in Eq. (7.1) resources are considered ideal, and so their skill gap is null. Consequently, productivity equals one and so it does not appear explicitly in Eq. (7.1).

However, when the PM assigns real nominal resources, unless a perfect match is found between task's requirement and resources skills, the original assignment levels must be rescaled, using resources' productivity as a scaling factor. This is shown in Eq. (7.2):

$$d_t \bullet \sum_{r \in R^*_t} a^*_{r,t} = d_t \bullet \sum_{r \in R} \left(a_{r,t} \bullet p_{r,t} \right) \tag{7.2}$$

Where the left and right hand-side represent, the total planned work for task *t*, as defined in the uncapacitated and in the capacitate project's plan, respectively. Since the duration must remain unchanged, Eq. (7.2) allows one to compute the true percentage assignment $a_{r,t}$, required to complete task *t* in time, depending on the allocated resources.

A numeric example should make this concept clearer. Let us consider a task that must be completed in 10 days (i.e., $d_t = 10$). Let us also suppose that, when the PM developed the uncapacitated plan, he/she establishes that to respect this duration an ideal team composed of a partially engaged Senior (S), working four hours a day, and of two fully engaged juniors (J), working eight hours a day. In this case $a^*_{S,t} = 50\%$ and $a^*_{J_1,t} = a^*_{J_2,t} = 100\%$ and so, from Eq. (7.1),

the planned total work equals 25 man days or, equivalently, (25·8) = 200 man hours. Of these, 40 hours are assigned to the senior and the remaining 160 are assigned to the juniors, as shown below:

$$PTW_t = d_t \bullet \sum_{r \in R_t^*} a_{r,t}^* = 10 \cdot a_{S,t}^* + 10 \cdot \left(a_{J_1,t}^* + a_{J_2,t}^* \right) = 10 \cdot 0.5 + 10 \cdot (1 + 1) = 25$$

Please note that the assumption that all resources are ideal, and that their productivity equals one, does not mean that the senior and the juniors have the same skills, but rather that they will be used to carry out different activities (within the same task *t*), for which their skills are suitable. For instance, the senior could do the hardest stuff, with the juniors doing the supporting work. Or, alternatively, in case of a simpler task, the senior could be used just as supervisor.

Moving on with the example, let's now assume that, during the project execution, the only available resources are one senior with productivity $p_{S,t} = 1.25$ and three juniors with productivity $p_{J_1,t} = p_{J_2,t} = p_{J_3,t} = 0.8$. Hence, in this case, a perfect matching cannot be found, because seniors and juniors are more and less experienced than required, respectively. Nonetheless a solution can be found and indeed, splitting the work content of seniors and juniors, from Eq. (7.2) the following assignment is easily found:

$$PTW_{t,S} = 5 = \left(d_t \bullet a_{S,t} \bullet p_{S,t} \right) = (10 \cdot 1.25) \bullet a_{S,t}$$

$$PTW_{t,J} = 20 = d_t \bullet \sum_{i=1}^{2} \left(a_{J_i,t} \bullet p_{J_i,t} \right) = (10 \cdot 0.8) \bullet \sum_{i=1}^{2} a_{J_i,t} = 2a_{J,t} \bullet (10 \cdot 0.8)$$

Where we supposed to keep unaltered the number of people of the team (i.e., one senior and two juniors) and to use the same assignment level for the two juniors (i.e., $a_{J_1,t} = a_{J_2,t} = a_{J,t}$). So, solving for the assignment level, we get $a_{S,t} = 40\%$ and $a_{J,t} = 125\%$, which correspond to a workload of (8·0.4) = 3.2 [h/day] and (8·1.25) = 10 [h/day], respectively. Accordingly, the junior must perform two hours of overtime, while the senior can distribute his/her 3.2 hours through the day, at his/her will. For instance, a good solution could be to supervise the juniors for a couple of

hours in the morning and the rest at the end of the day. Also note that, due to a different mix of resource skills, although the duration has remained the same, the actual total work has increased, moving from 200 to 10·(3.2 + 10 + 10) = 232 man hours.

Moreover, to get a harmonious improvement of the human capital and to enhance learning on the field, the PM might want to maximise or ensure a certain improvement of his/her resources. By doing so, under-skilled workers will be valorised, and a positive synergy will be obtained among the team's members. To incorporate this feature in the model, we introduced the *"possible skill improvement"* per unit of time. This value is a number which expresses how much a resource can improve his/her skill level if assigned to a specific task, for one unit of time. For instance, if the possible skill improvement of resource *r* on task *t* equals 0.01 [skill level units/day], and if *r* is assigned at level $a_{r,t}$ = 100% to task *t* for a duration of 10 working days, at the end of the taks the skill level of resource *r* will rise of (10·0.01) = 0.1 points. In line with the work by Pawiński and Sapiecha (2016), these values should depend on the skills gaps. If the gap is too high (either in positive or negative sense) improvements by learning on the field are impossible or almost null. Conversely, if the gap is small (preferably if it is slightly negative) the change of improvements is high. In other words, barely under skilled resources are the most suitable to be assigned to challenging tasks, under the supervision of experienced seniors. According to these improvement rates per unit of time, each allocation will provide a different improvement of resources skills. This improvement can be considered as obtained at the end of the project, and it is an important output in order to support the job of the PM.

7.2.4. Sorting criteria, notation, and outcomes.

The allocation procedure makes use of a list of tasks L_T and a list of resources L_R. Before the allocation starts, L_T is filled with all the tasks of the project sorted according to some specific criteria previously defined by the PM, while L_R is empty. Next, tasks are considered one at a time, proceeding from the first to the last one, and, for each selected task *t*, $L_R(t)$ is filled with all resources that are eligible for *t* sorted according to some criteria decided by the PM. Finally, a subset of resources in $L_R(t)$ will be selected and used to saturate the workload required by task *t*, possibly without generating any over-allocation.

A task is said to be saturated if it has received the minimum number of seniors and the assigned resources (and their assignment level) are enough to satisfy its Total Planned Work. Conversely, a resource *r* is said to be eligible for a task *t,* if its productivity $p_{r,t}$ is greater than zero and if it has a residual availability for the whole duration of task *t*. Suppose that resource *r* have been assigned to task *t*, from day d_X to day d_Y, at a constant assignment level equal to $a_{r,t}$. Hence the residual

availability of *r*, say δ_r, from d_X to d_Y is $\delta_r = (1 - a_{r,t})$; if this value is higher than the minimum allowable assignment level defined by the PM (say 10 or 20%), resource *r* is eligible to be assigned to other tasks scheduled between d_X and d_Y, otherwise it is not. Note that, a maximum availability of a resource on the whole project is not considered, otherwise, at every time *τ* we compute the residual availability as in Eq. (7.3):

$$\delta_r(\tau) = 1 - \sum_{t \in T_\tau} a_{rt} = 1 - \alpha_r(\tau) \tag{7.3}$$

Where:

- T_τ is the set of all tasks that are in progress (or scheduled) at time *τ*,
- $\alpha_r(\tau)$ is the cumulated assignment level of resource *r*, at time *τ*.

As explained above, tasks are sorted according to a specific criteria, and those in the top positions are "favored" over those in lower positions. Let us consider the subset of tasks $t \in T_\tau$ that are in progress at time *τ*. As these tasks are saturated, the number of resources with a non-negligible residual availability $\delta_r(\tau)$ will be less and less. In other words, top ranked tasks will have the chance to draw resources from a set larger than that available for tasks in lower positions. Owing to this issue, it may be wise to sort tasks in order of importance, to offer the widest choice to the most important ones. Possible sorting criteria are listed in Table 7.1.

Table 7.1. Tasks' sorting criteria (Bertolini, Neroni, and Zammori, 2020).

Sorting criterion	**Explanation and motivation**
Slack Time	Priority is given to the tasks belonging to the critical path (i.e., with a null slack).
Start Time	Sorting tasks in terms of start time, other criteria being equal, the allocation is performed moving from the start to the end of the project.
End Time	As before, but in reverse order.
Number of Successors	Together with Slack-Time, this criterion identifies the most critical tasks, that, in case of over-allocation could heavily impact on the project.
Duration d_t	The longer the duration of a task, the longer the time in which resources allocated to it may be unavailable for other assignments. So, it could be wise to start from the longest tasks.
Planned Total Work	To have a wider choice of eligible resources, it could be wise to start from the tasks with the highest planned total work. This criterion is like the previous one, but duration is substituted by effort.
Residual Total Work	As before using the residual rather than the planned total work. Note that the residual total work is the remaining effort (or work) needed to saturate a task.
Number of Seniors	Generally, there are fewer seniors than juniors. So, it may be convenient to start the allocation from the tasks requiring the highest number of seniors.

A similar approach is used to select the resources too. More specifically, any time a new task is considered, the resources list $L_R(t)$ is regenerated and filled with all resources that are eligible for *t*. Next, the first resource *r* in $L_R(t)$ is taken and it is assigned to *t*. If *r* is enough to saturate *t*: (i) $L_R(t)$ is emptied, (ii) task *t* is removed from L_T, (iii) the next task in L_T, say (*t* + 1) is selected and (iv) $L_R(t + 1)$ is refilled with all resources eligible for task (*t* + 1). Conversely, if *r* is not enough, the next resource, say (*r* + 1), in $L_R(t)$ is selected and assigned to *t*. The process is iterated until task *t* is finally saturated.

It is thus clear that also resources should be sorted in order of priority, using one or more criteria related to the objectives pursued by the PM. Possible resources' sorting criteria are listed in Table 7.2, from which we can clearly see how they have been linked to the ten ingredients for a successful project.

Table 7.2. Resources' sorting criteria (Bertolini, Neroni, and Zammori, 2020).

Sorting criterion	Explanation and motivation
Productivity	To maximize project's quality, the most qualified resources should be assigned first
Productivity to Cost ratio	To minimize project's costs, resources should be ordered in terms of their productivity to cost ratio.
Residual Availability	To avoid work fragmentation (i.e., resources assigned to many tasks at low assignment levels), priority should be given to the resources with higher residual availability, defined as the number of working hours still available at a certain time *t*.
Average improvement	To promote the growth of the team, priority should be given to the resources with the highest potentiality for improvement, if assigned to a task.
Specific improvement	To promote improvement on a specific skill, priority should be given to the resources with the highest potentiality for improvement on that skill.
Senior or junior	To assure project quality, in case of critical tasks, priority could be given to senior resources. Conversely, in case of sub-critical tasks, to foster the improvement of the less skilled resources, priority should be given to juniors.

The inputs required by our tool are essentially:

- The capacitated project plan, possibly represented by a Gantt chart;
- Tasks' requirements in terms of number of resources and their associated skills;
- The list of the available resources, with specified skills levels and availability (i.e., resources' calendar).

To operate the algorithm, it is also necessary to define the resources' productivity and their skills improvement rates, as well as, an additional set of constraints such as the minimum and maximum allocation level, the maximum admissible over-time and others. These parameters and the notation adopted are described in Table 7.3.

Table 7.3. Notation used in the remainder of this work (Bertolini, Neroni, and Zammori, 2020).

Type	Symbol	Name	Description
Basic Fixed Inputs	$k \in \{1, \ldots, K\}$	Skill	One of the K skills used to qualify the resources.
	$r \in \{1, \ldots R\}$	Resource	One of the R resources available to do the work.
	$t \in \{1, \ldots T\}$	Task	One of the T tasks of the project.
	$\tau \in \{1, \ldots, \Gamma\}$	Time instant	A generic day, from the fist to the last one Γ.
	d_t	Task Duration	Task duration as defined in the uncapacitated plan.
	PTW_t	Planned Tot. Work	The planned effort (number of working hours) of task t.
	h_r	Hourly rate	Cost of resource r per unit of time.
	ε_r	Extra cost rate	Extra cost of resource r per unit of extra time.
	$c_{r,k}$	Res. skills levels	Skill level of resource r on field k.
	$s_{t,k}$	Task requirement	Task requirements in terms of quality (i.e., skill levels).
	$N_{S,t}$	Numb. of seniors	The minimum number of seniors required by task t.
	$N_t \geq N_{S,t}$	Numb. of resour.	The minimum number of resources required by task t.
Editable Inputs	a_{min}	Min. Assign. level	The minimum assignment of a resource on a task.
	α_{Max}	Max. Assign. level	The maximum admissible assignment. If overallocation is not admitted $\alpha_{Max} = 100\%$.
	$p_{r,t}$	Productivity	Productivity of resource r on task t.
	$u_{r,t,k}$	Skill upgrading rate	Rate of improvement of resource r on skill k, if assigned to task t for one unit of time.
Outputs	$a_{r,t}$	Assignment level	The assignment level of resource r on task t. For instance, $a_{r,t} = 100\%$ corresponds to 8 hours a day.
	$\delta_r(\tau)$	Residual availab.	Remaining capacity of resource r at time τ.
	RTW_t	Residual Total Work	Effort still needed to saturate task t. At the end of the assignment $RTW_t = 0$.

	$\alpha_r(\tau)$	Cumul. assign. level	Cumulated assignment level of resource *r* at time τ.
	$I_{r,k}$	Total improvement	Total improvement of resource *r* on skill *k.*
	TC	Total Cost	Total cost of the project (extra rate for overtime included).

The outputs of the algorithm are the assignment matrix $A\left[a_{r,t}\right]$, which gives the assignment levels $a_{r,t}$ of each resource *r* to each task *t*, and the cumulative assignment rate matrix $\mathrm{A}\left[\alpha_{r,\tau}\right]$ which gives the cumulative assignment levels $\alpha_{r,\tau}$ of each resource *r* at each time *τ*. Additionally, the Total Cost and the overall skill improvements $I_{r,k}$ obtained by resource *r* on skill *k,* are also returned as output.

These quantities are computed as in Eq. (7.4) and Eq. (7.5):

$$TC = \sum^{\Gamma} \sum\left(\alpha_{r,t} \bullet h_r\right) + \sum^{\Gamma} \sum\left(max(0, \alpha_{r,t} - 1)\cdot\varepsilon_r\right) \tag{7.4}$$

$$I_{r,k} = \sum\left(a_{r,t} \bullet d_t \bullet u_{r,t,k}\right) \tag{7.5}$$

Where Γ is the duration of the project, $u_{r,t,k}$ is the skill upgrading rate, and ε_r is the extra charge for overtime.

7.2.5. The procedure of the algorithm.

The full allocation procedure is composed of three main *allocation cycles*.

The goal of the first allocation cycle is to assign to each task the required number of seniors. Therefore, the selection is limited to seniors, who are assigned to tasks at the minimum admissible allocation level. This choice has a twofold purpose as it: *(i)* makes seniors eligible for assignment as many times as possible and *(ii)* reduces work fragmentation and multi-tasking. For instance, using a minimum level $a_{min} = 50\%$ no more than two parallel tasks can be assigned to the same senior. At this step the user has to define the minimum allowable assignment level a_{min} and the

sorting criteria both for tasks and for senior resources. Since seniors are expensive, and they do not have to improve, a wise choice could be to sort seniors in descending orders of the "*productivity to cost*". This should contain costs, without jeopardizing the project's quality. The sorted criteria used for the tasks is not that relevant, at least at this step. Indeed, since seniors are allocated using a minimum assignment level, also the bottom ranked tasks will have a wide choice in terms of resources. Thus, as a rule of thumb, tasks can be simply sorted using "slack-time" in ascending order and "planned total work" in descending order. In this way, priority will be given to the most onerous (in terms of effort) tasks of the critical path. Before starting the allocation, the PM has also to define the maximum allowable assignment level $\alpha_{Max} \geq 1$. Typically, this value will be used only during the third allocation cycle, but although rarely it could be useful also in the first cycle, as explained next. The allocation is performed through the following steps:

(1) The tasks list L_T is instantiated with all tasks requiring at least one senior, and it is sorted.

(2) For each task *t* in L_T the following sub-steps are performed:

a. A resource list $L_R(t)$ is generated, and it is filled with the seniors that are eligible for *t*, depending on their productivity and residual availability. At this step, overallocation is not admissible, and so the residual availability is computed as: $\delta_r(\tau) = (1 - \alpha_r(\tau))$.

b. The first $N_{S,t}$ resources in $L_R(t)$, equal to the number of seniors required by task *t* are selected.

c. The Residual Total Work RTW_t of t is updated as in Eq. (7.6) and (7.7). Where RTW_t^+ and RTW_t^- indicate the Remaining Total Work before and after the addition of resource *r*. Clearly, before the first resource is assigned to task *t*, its remaining total work coincides with the planned total work.

d. The allocation matrix $A\left[a_{r,t}\right]$ and the corresponding cumulated allocation matrix $A\left[\alpha_{r,\tau}\right]$ are updated.

e. If task *t* is saturated (i.e., $PTW_t \geq PTW_t$), it is removed from L_T, otherwise it will be re-considered in the following allocation cycles.

$$RTW_t^0 = PTW_t \tag{7.6}$$

$$RTW_t^+ = RTW_t^- - \left(a_{r,t} \bullet p_{r,t} \bullet d_t\right) \tag{7.7}$$

To conclude, we note that, extremely rarely, the number of eligible resources in $L_R(t)$ could be smaller than the number of seniors required by task *t*. This condition should never occur, as it indicates an evident shortage of staff; however, this critical condition can be tackled as explained next. First of all the residual availability of the seniors is updated using the maximum allocation level α_{Max}, rather than the standard value of 100%, i.e. $\delta_r(\tau) = (\alpha_{Max} - \alpha_r(\tau))$. This makes eligible

and additional number of senior resources that are added to $L_R(t)$. Next, to minimize over-allocations, resources are sorted in descending order of their residual availability $\delta_r(\tau)$ and the allocation procedure restarts from point 2.b.

If the cardinality of $L_R(t)$ remains smaller than $N_{S,t}$, the problem is declared unfeasible, unless the PM decide to further increase α_{Max} and/or to reduce a_{min}.

Then, the second cycle is computed, whose objective is to saturate all tasks, using junior resources too. Also in this step neither overtime nor overallocation are admitted; should some tasks remain unsaturated, they will be considered during the third and last cycle. The allocation starts with the update of L_T, to which all tasks that were not considered during the first cycle are added. In practice, L_T now contains all not saturated tasks (i.e., task having $RTW_t \neq 0$). The basic assignment logic does not change, but rather than using the fixed value a_{min}, the algorithm tries to allocate resources using the assignment level $a_{r,t}$ defined by Eq. (7.8), which corresponds to the minimum level needed to saturate a task:

$$a_{r,t} = \left[\frac{RTW_t}{(p_{r,t} \bullet d_t)} \right] \tag{7.8}$$

This value is used if it is lower than both a_{min} and $\delta_r(\tau)$; otherwise $min\{\delta_r(\tau), a_{min}\}$ is used.

We conclude this section noting that the PM is free to modify the sorting criteria established for the first allocation cycle. A natural choice is to sort tasks in terms of “slack-time” first, and in terms of “remaining total work” next. By doing so, tasks that may remain unsaturated and that will require overallocation (in the third cycle) are the ones that do not belong to the critical path and require a small number of resources. This should simplify project levelling and should keep delays low. Relatively to resources, it is now convenient to give priority to juniors with the highest potential improvement, as this should enhance the secondary objective to progressively improve the skills of the project’s team. Certainly, to contain costs, as a third and last ordering criteria the cost to productivity ratio could be used too.

If at the end of the second cycle some tasks remain unsaturated, the third and last one is activated. At this point, the only way to complete the assignment is to admit a certain degree of overallocation and so the overallocation constraint is relaxed. Hence, the residual availability is

updated using the maximal allowed assignment α_{Max}, and the eligible resources list is updated accordingly. Also, priority is given to the resource that has already been assigned and, as second criteria, to the ones with the highest residual availability. This limits work's fragmentation and the creation of oversized teams. If the cycle ends up leaving some tasks unsaturated the problem is declared unfeasible.

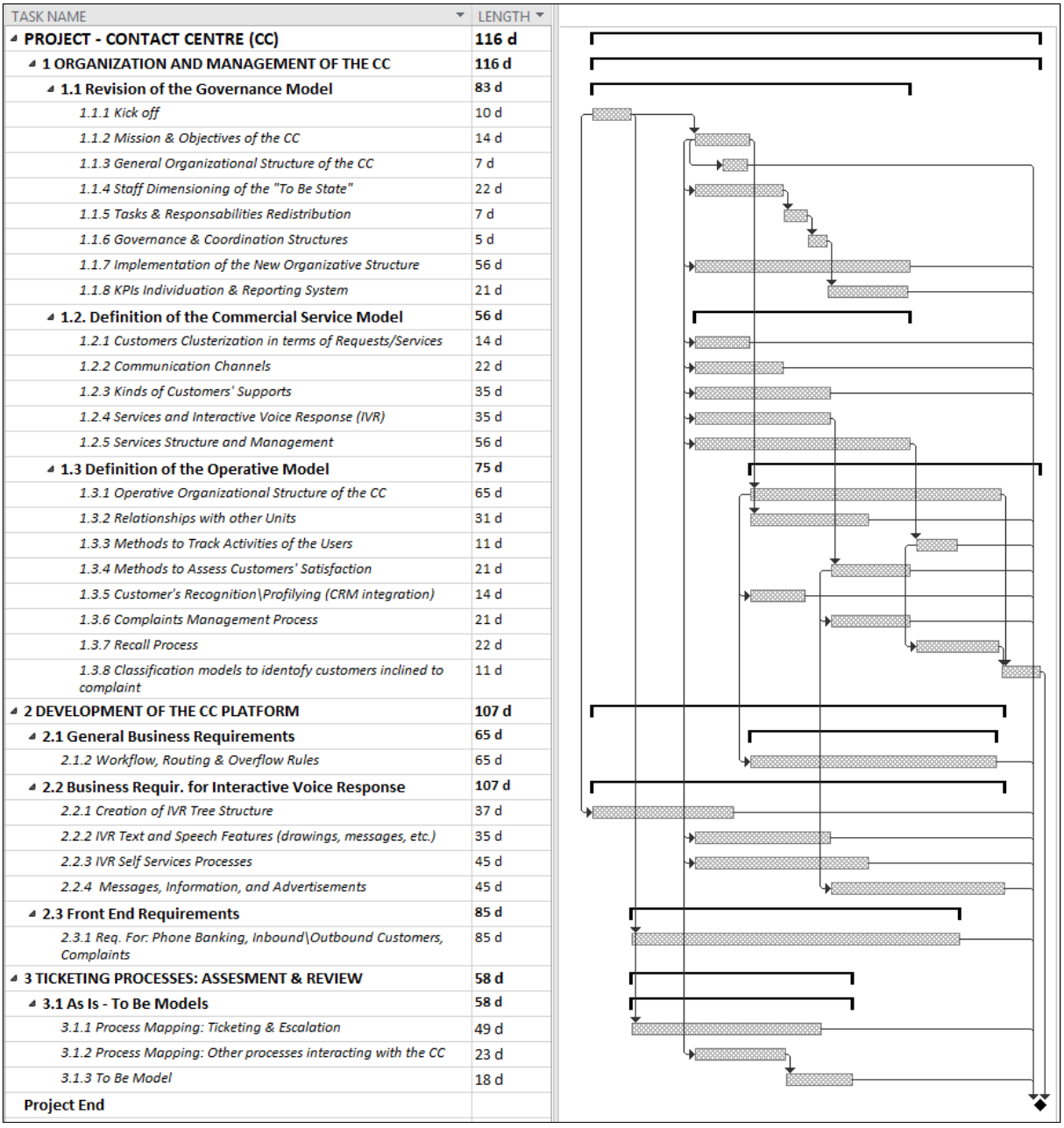
7.2.6. Description of the case study

To test the proposed tool, twenty projects recently completed by an important Italian Consulting Company were used. The identity of the company must remain screened and, from here on, it will be referred to as ICC.

The tool, implemented in Visual Basic.net as a standalone application for an easy integration with Microsoft Project © and Microsoft Excel ©, was used to automatically solve the staffing problem of the twenty investigated projects. Next the provided solutions were compared with the original allocations made by the PMs of ICC. We anticipate that the obtained results do not differ much from one project to another and so, due to space constraints, the discussion will be limited to the most complex one.

The considered project concerned the development of a Contact-Centre for a leading Italian company. Specifically, ICC had to take care of the set-up activities and of the design of the IT tools needed to support the operating activities of the company.

The uncapacitated project plan, made of thirty tasks, for an overall length of 116 working days, is shown in Figure 7.1.



TASK NAME	LENGTH
PROJECT - CONTACT CENTRE (CC)	**116 d**
1 ORGANIZATION AND MANAGEMENT OF THE CC	**116 d**
1.1 Revision of the Governance Model	**83 d**
1.1.1 Kick off	10 d
1.1.2 Mission & Objectives of the CC	14 d
1.1.3 General Organizational Structure of the CC	7 d
1.1.4 Staff Dimensioning of the "To Be State"	22 d
1.1.5 Tasks & Responsabilities Redistribution	7 d
1.1.6 Governance & Coordination Structures	5 d
1.1.7 Implementation of the New Organizative Structure	56 d
1.1.8 KPIs Individuation & Reporting System	21 d
1.2. Definition of the Commercial Service Model	**56 d**
1.2.1 Customers Clusterization in terms of Requests/Services	14 d
1.2.2 Communication Channels	22 d
1.2.3 Kinds of Customers' Supports	35 d
1.2.4 Services and Interactive Voice Response (IVR)	35 d
1.2.5 Services Structure and Management	56 d
1.3 Definition of the Operative Model	**75 d**
1.3.1 Operative Organizational Structure of the CC	65 d
1.3.2 Relationships with other Units	31 d
1.3.3 Methods to Track Activities of the Users	11 d
1.3.4 Methods to Assess Customers' Satisfaction	21 d
1.3.5 Customer's Recognition\Profilying (CRM integration)	14 d
1.3.6 Complaints Management Process	21 d
1.3.7 Recall Process	22 d
1.3.8 Classification models to identofy customers inclined to complaint	11 d
2 DEVELOPMENT OF THE CC PLATFORM	**107 d**
2.1 General Business Requirements	**65 d**
2.1.2 Workflow, Routing & Overflow Rules	65 d
2.2 Business Requir. for Interactive Voice Response	**107 d**
2.2.1 Creation of IVR Tree Structure	37 d
2.2.2 IVR Text and Speech Features (drawings, messages, etc.)	35 d
2.2.3 IVR Self Services Processes	45 d
2.2.4 Messages, Information, and Advertisements	45 d
2.3 Front End Requirements	**85 d**
2.3.1 Req. For: Phone Banking, Inbound\Outbound Customers, Complaints	85 d
3 TICKETING PROCESSES: ASSESMENT & REVIEW	**58 d**
3.1 As Is - To Be Models	**58 d**
3.1.1 Process Mapping: Ticketing & Escalation	49 d
3.1.2 Process Mapping: Other processes interacting with the CC	23 d
3.1.3 To Be Model	18 d
Project End	

Figure 7.1. The Uncapacitated Project Chart (Bertolini, Neroni, and Zammori, 2020).

The objective was to improve the junior resources (through learning on the field), assuring an almost perfect matching between tasks and resources, in terms of skills. Keeping direct costs low was considered as a secondary objective.

As an additional constraint, ICC also required the assignment of one or more seniors to each task. Although some tasks could have been carried out entirely by juniors, ICC believes that the supervision of a senior improves the project's quality and, most of all, it enhances corporate reputation, as the customer perceives it as a distinctive element of reliability and professionalism.

Tasks' durations and their Planned Total Work were defined by the PM assuming the use of ideal resources with skill levels (relative to four skills S1 to S4) as shown in Table 7.4.

Table 7.4. Tasks requirements (duration, planned total work and skill levels) (Bertolini, Neroni, and Zammori, 2020).

ID	Task	Duration [days]	Planned Tot. Work [man days]	Average Assign. Level	S1	S2	S3	S4
1	T1	10	4.00	40%	5	2	2	4
2	T2	14	1.40	10%	3	0	1	3
3	T3	7	2.10	30%	4	0	1	4
4	T4	22	2.20	10%	4	1	3	4
5	T5	7	3.50	50%	4	1	3	4
6	T6	5	3.00	60%	3	0	1	4
7	T7	56	5.60	10%	3	0	1	3
8	T8	21	4.20	20%	4	2	4	3
9	T9	14	7.0	50%	3	3	4	3
10	T10	22	8.80	40%	2	3	3	2
11	T11	35	21.0	60%	3	2	5	4
12	T12	35	7.00	20%	2	3	3	3
13	T13	56	28.0	50%	4	2	5	4
14	T14	65	19.5	30%	3	1	3	4
15	T15	31	12.4	40%	4	3	2	3
16	T16	11	13.2	120%	2	4	4	2
17	T17	21	16.8	80%	2	4	5	2
18	T18	14	5.60	40%	2	4	3	1
19	T19	21	16.8	80%	3	2	2	4
20	T20	22	13.2	60%	3	2	2	2
21	T21	11	8.80	80%	3	2	4	3
22	T22	65	84.5	130%	2	4	2	5
23	T23	37	7.40	20%	1	4	2	4
24	T24	35	14.0	40%	2	4	2	5
25	T25	45	9.00	20%	1	4	1	5
26	T26	45	18.0	40%	2	4	4	4
27	T27	85	51.0	60%	1	4	1	5
28	T28	49	19.6	40%	2	4	2	4
29	T29	23	11.5	50%	3	2	2	2
30	T30	18	14.4	80%	3	3	2	5

In the table, skills are quantified using a continuous scale ranging from 1 (scarce) to 5 (excellent), and have the following meaning:

- ✓ *S1 - Experience in the field* - A thorough and cross-sectional background, acquired through the involvement in similar projects.
- ✓ *S2 - IT instruments knowledge* - Knowledge of Front-End instruments and design capabilities for IT architectures for a contact center.
- ✓ *S3 - Sales and marketing experience* - Knowledge of different sales and marketing strategies.
- ✓ *S4 - Explicative and communicative skill* - Ability to interact and to communicate, both with a technical and non-technical audience.

Also, note that the planned total work is expressed as the number of days it would take to an ideal resource to complete the job, if assigned at the average level indicated in column 5. A value of the average assignment higher than 100% indicates the need of overtime and/or the use of additional resources.

To complete the project, a pool of six resources was made available; their hourly rates and skill levels are shown in Table 7.5, where senior resources (R1 to R3) are underlined.

Table 7.5. Resources' skills (Bertolini, Neroni, and Zammori, 2020).

ID	Resource	h_r [€/h]	S1	S2	S3	S4
1	R1	5.0	3.5	5.0	5.0	5.0
2	R2	4.0	4.0	3.0	4.5	4.0
3	R3	3.5	3.0	4.0	4.0	3.5.
4	R4	2.0	2.0	2.0	3.0	2.0
5	R5	2.0	3.0	2.0	3.5	2.0
6	R6	3.0	2.0	2.0	3.0	3.0

It is interesting to note that the total workload of the project sums up to 433.5 [man days]. So, given a total duration of 116 days, the average allocation percentage (for each one of the six resources) equals 62.5%. Nonetheless, due to the high level of parallelism among tasks (see Figure 7.1), the average level of daily assignment (per resource) strongly fluctuates and it exceeds 100% in the middle of the project. Therefore, due to the "eligibility" constraints and to the need to assign at least one senior to each task, multi-tasking overtime will be unavoidable. This is clearly shown in Figure 7.2, where bars in black highlight the days where an extra workload is required.

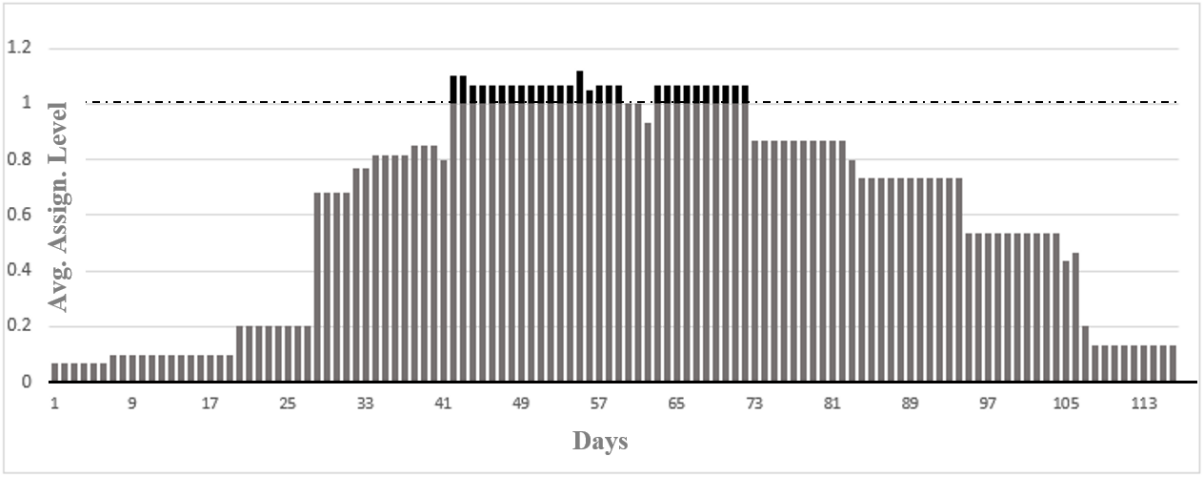


Figure 7.2. Average daily assignment level per resource (Bertolini, Neroni, and Zammori, 2020)

Productivities should be defined by the PM, in the function of the skills gaps among tasks and resources. To this aim, according to the PM, the sigmoid function of Eq. (7.9) was used.

(7.9)

$$p_{r,t} = \frac{P}{(P-1)+\exp(\quad)}$$

Where:

- ✓ *P* is the maximum value of productivity;
- ✓ γ is the shape parameter, defining the slope of the curve;
- ✓ $G_{r,t}$ is the average skill gap of resource *r*, relatively to task *t*;
- ✓ (*P* - 1) is the location parameter, that assures productivity equal to one when the average skill gap $G_{r,t}$ is null.

The rational among this choice can be motivated as follows. It is this clear that productivity $p_{r,t}$ must be an increasing function of the skill gaps, and that the higher is the positive gap the higher the productivity rate and vice versa (i.e., more experienced and highly skilled resources are faster). The use of a sigmoid fully complies with these requirements as it increases from zero to the maximum value *P* in a non-linear and asymptotic way. Specifically, setting *P* and γ respectively to 2 and 1.1, the productivity increases asymptotically from 0 to 2 (almost reached for a skill gap of -5 and 5, respectively), and equals 1 in case of perfect matching between task's requirement and resource's skills, as clearly shown in Figure 7.3. This behavior was considered reasonable by the PMs of ICC.

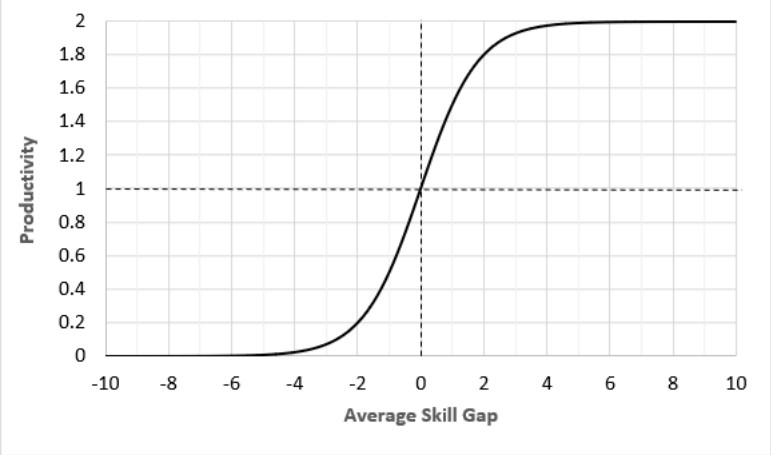


Figure 7.3. Productivity as a function of the average skill gap (Bertolini, Neroni, and Zammori, 2020).

Concerning $G_{r,t}$, the average skill gap of resource *r* on task *t*, it was computed as a weighted average of the skill gaps, as in Eq. (7.10):

(7.10)

$$G_{r,t} = \sum_{k}^{K} \left(g_{r,t,k} \cdot w_{t.k} \right) = \frac{\sum_{k} [s_{t,k} \cdot (c_{r,k} - s_{t,k})]}{\_}$$

Where:

- ✓ $g_{r,t,k} = (c_{r,k} - s_{t,k})$ is the skill gap between resource skill level $c_{r,k}$ and task requirement $s_{t,k}$;

- ✓ $w_{t,k} = \frac{s_{t,k}}{\sum s_{t,k}}$ is the weight factor (or importance of skill *k* relative to task *t*), equals to the ideal skill levels for task *t* normalized to one.

Applying Eq. (7.9) to the skills level and tasks requirement of Table 7.4, the productivities shown in Table 7.6 were finally obtained.

Table 7.6. Productivity of each resource (column) on each task (row) (Bertolini, Neroni, and Zammori, 2020).

Task ID	R1	R2	R3	R4	R5	R6
1	1.5	1.1	0.9	**0.0**	0.4	0.5
2	1.9	1.6	1.5	0.8	1.0	1.1
3	1.6	1.2	1.1	0.4	0.5	0.6
4	1.6	1.2	1.1	0.4	0.5	0.6
5	1.6	1.2	1.1	0.4	0.5	0.6
6	1.7	1.4	1.3	0.6	0.7	0.8
7	1.9	1.6	1.5	0.8	1.0	1.1
8	1.6	1.2	1.1	0.4	0.5	0.5
9	1.6	1.3	1.2	0.5	0.6	0.6
10	1.8	1.6	1.5	0.8	1.0	0.9
11	1.5	1.0	0.9	**0.0**	**0.0**	**0.0**
12	1.8	1.5	1.4	0.7	0.9	0.8
13	1.4	0.9	0.8	**0.0**	**0.0**	**0.0**
14	1.7	1.4	1.3	0.6	0.7	0.7
15	1.7	1.4	1.2	0.5	0.7	0.7

Task ID	R1	R2	R3	R4	R5	R6
16	1.6	1.2	1.1	0.4	0.6	0.5
17	1.4	1.0	0.9	**0.0**	0.4	**0.0**
18	1.6	1.4	1.3	0.5	0.8	0.6
19	1.7	1.5	1.4	0.7	0.9	0.8
20	1.9	1.7	1.6	0.9	1.1	1.1
21	1.7	1.3	1.3	0.5	0.7	0.6
22	1.6	1.4	1.1	0.5	0.7	0.6
23	1.5	1.3	1.1	0.5	0.8	0.5
24	1.4	1.1	0.9	**0.0**	0.5	0.4
25	1.3	1.1	0.8	**0.0**	0.6	**0.0**
26	1.4	1.1	1.0	**0.0**	0.5	**0.0**
27	1.9	1.8	1.7	1.2	1.4	1.2
28	1.7	1.5	1.4	0.6	0.9	0.8
29	1.9	1.7	1.6	0.9	1.1	1.1
30	1.5	1.2	1.0	0.4	0.6	0.5

Note that a productivity lower than 0.4 was considered unacceptable and substituted with a null value to clearly indicate the incompatibility between task and resource.

Concerning the improvements, even in this case, a function to automatically compute the skill upgrading rates $u_{r,t,k}$ was proposed to the PM. In this case, the function is the quadratic curve of Eq. (7.11):

$$u_{r,t,k} = max\{0; \frac{\left(-\beta_1 \cdot (g_{r,t,k})^2 - \beta_2 \cdot g_{r,t,k} + \beta_3\right)}{\varphi}\} \quad (7.11)$$

Where φ is a normalization factor (expressed in time units) and β_i are dimensionless shape factors.

The improvement should be high in case of slightly negative skill gaps and should be low in case of slightly positive skill gaps. Also, the upgrading rate per unit of time should be null when the absolute value of the gap $|g_{r,t,k}|$ is high i.e., a resource that is already excellent cannot improve, whereas a resource that is too weak cannot learn on the field. Specifically, shape parameters $\beta_1 = 0.4$, $\beta_2 = 0.4$ and $\beta_3 = 0.1$ and normalisation factor φ = 40 [days] were chosen. In this way, resources that can improve have skill gaps in the range [-1.5; 1], with a maximum for a gap equals to -0.5. The operating meaning is that an under-skilled resource with a negative skill gap of 0.5 could achieve an improvement of 0.2 points, if fully employed for 40 working days. For the sake of clarity, the obtained shape shown in Figure 7.4.

Figure 7.4. Improvement rates (per day) as a function of the skill gap (Bertolini, Neroni, and Zammori, 2020).

Tasks were sorted (both in the first and second allocation cycle) in ascending order of start time and, in case of a tie, in descending order of duration and of remaining workload. In this way, the

higher priority is given to the initial tasks that were considered essential for the achievement of the project's goals. The other two criteria have the purpose to keep loads levelled, as they give higher priority to the longest and most demanding tasks that keep resources busy for a long time.
Relatively to resources, different sorting criteria were used during the first and second allocation cycle. In the first one, to keep direct costs low, seniors were sorted in terms of their productivity to cost ratio. In the second cycle, in line with the overall objective to improve juniors' competencies, priority was given to juniors, using as a second criterion their potential improvement. To keep costs low, the productivity to cost ratio was chosen as the third and last criterion.
Additional constraints were also included in the allocation procedure, as detailed in Table 7.7.

Table 7.7. Additional constraints (Bertolini, Neroni, and Zammori, 2020).

Parameter	Value	Explanatory Notes
Minimum allowed productivity	0.40	The minimum productivity level that makes a resource eligible
Seniors assignment level during first all. cycle	0.05	This assignment level implies a minimum supervision of around 25-30 minutes per day.
Minimum residual availability for senior	0.05	A senior cannot be considered eligible for a task schedule at time τ, if its residual availability $\delta_r(\tau)$ is lower than 5%.
Minimum residual availability for junior	0.20	Same as before, but for junior resources.
Maximum number of resources per task	3.00	To avoid an excessive work fragmentation, no more than three resources, including the supervisor, can work on the same task.
Maximum admitted allocation level	1.40	If over allocation is needed to complete a task in time, no more than (0.4·8) = 3.2 hours of overtime can be used.

7.2.7. Results

The transpose of the allocation matrix $A_H^T\left[a_{r,t}\right]$ obtained with the proposed tool, and of the allocation matrix $A_P^T\left[a_{r,t}\right]$ independently developed by the PMs of ICC are shown in Table 7.8.

Table 7.8. Comparison of the automatic and expert made allocation matrices (Bertolini, Neroni, and Zammori, 2020).

Heuristic Allocation Matrix $A^T[\alpha]$

Task	R1	R2	R3	R4	R5	R6
T1		5%			80%	
T2		10%				
T3		20%	5%			
T4			5%			10%
T5			10%		20%	50%
T6			5%		20%	50%
T7		5%				5%
T8			5%	35%		
T9		15%	5%			45%
T10		5%				40%
T11	40%		5%			
T12		5%	0%		15%	
T13		5%	60%			
T14	15%	5%				
T15	25%		5%			
T16			65%		30%	55%
T17	45%		5%		30%	
T18	20%	5%				
T19			5%	20%		75%
T20		5%		55%		
T21		5%		100%		35%
T22		75%	5%			45%
T23			5%		20%	
T24			5%		65%	
T25	15%		5%			
T26			5%		70%	
T27				45%		
T28			5%	55%		
T29		30%				
T30		35%	40%			

PM's Allocation Matrix $A^T[\alpha]$

Task	R1	R2	R3	R4	R5	R6
T1	15%	15%				
T2			5%	5%		
T3	10%	10%				
T4		5%	5%			
T5	15%		25%			
T6		15%		65%		
T7			5%	5%		
T8	5%		10%			
T9	15%	15%				20%
T10	10%			30%		
T11	20%	35%				
T12	5%			15%		
T13	5%		55%			
T14	5%			20%		20%
T15	10%		20%			
T16	35%	35%				45%
T17	35%		20%		40%	
T18		10%			20%	20%
T19		20%		45%	25%	
T20			20%	30%		
T21		30%	30%			
T22	40%	55%				
T23		5%			20%	
T24		15%			25%	25%
T25	5%	5%			20%	
T26	10%	15%			20%	
T27		5%				45%
T28		15%			25%	
T29			10%	40%		
T30	25%				45%	30%

Note that the reported ones are the transpose of the allocation matrices; so, tasks are on the rows and resources are on the columns. For instance, the first row indicates that, according to the automatic assignment, task T1 is performed by resources R2 and R5, whereas the same task is performed by resources R1 and R2 in the assignment made by the PM. Similarly, the first column says that resource R1 is assigned to tasks {T11, T17, T18, T25} and {T1, T3, T5, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T22, T25, T26, T30}, in the automatic and expert made assignment, respectively.

The key performance indicators (KPIs) of the two alternative assignments are shown in Table 7.9, where HR denotes "heuristic solution" and PM "Project manager's solution". Specifically, for each resource, the following metrics are shown: *(i)* his/her maximum accumulated assignment level $\alpha_r(\tau)$, which is greater than one if the resource is overallocated, at least for one day, and need some hours of overtime, *(ii)* the total hours of overtime that he/she has to do (null if $\alpha_r(\tau) \leq 100\%$), *(iii)* the total number of task assigned to him/her, *(iv)* the maximum number of parallel tasks

assigned to him/her, which should be low to minimize multitasking and work fragmentation and *(v)* the average percentage improvement (on all skills) he/she should obtain at the end of the project.

Table 7.9. Comparison of KPIs (Bertolini, Neroni, and Zammori, 2020).

	Max. Assign. Level		Total hours of overtime		Total Assigned Task		Max number of parallel tasks		Percentage Improvement	
	HR	PM	HR	PM	HR	PM	HR	PM	HR	PM
R1	115%	140%	17	51	6	17	5	9	0.9%	1.4%
R2	140%	150%	87	81	16	17	9	9	1.0%	1.3%
R3	140%	115%	50	15	19	11	9	5	2.7%	2.1%
R4	100%	110%	0	9	6	9	3	5	2.9%	2.2%
R5	100%	150%	0	53	9	9	3	5	3.6%	2.6%
R6	125%	140%	42	34	10	7	4	5	2.9%	1.9%

Concerning seniors, the proposed tool favored the employment of R3, while the PM favored the use of R1. In line with the stated goal to promote skills' upgrading, the proposed solution seems preferable, as R3 is the less skilled senior. Also, and most important, the allocation percentages of R1 are rather low, and his/her maximum cumulated assignment equals 115% (as the first row of Table 7.9 shows), a typical overallocation for a consulting project. The resulting allocation plan ensures that this valuable resource can be easily reassigned, should unforeseen problems take place. On the other hand, (as clearly shown in the columns of Tables 7.8) it seems that, at least unconsciously, the reasoning of the PM is biased, as he/she always tends to reuse the same and best resources.

The proposed solution seems preferable for juniors too. Indeed, as Table 7.9 shows, the less skilled junior (R6) is the most employed one , whereas the most skilled junior (R4) is the less used one. These considerations are also supported by the average skill improvements in the last column of Table 7.9. In fact, except for R1 and R2, two highly skilled seniors that do not need to improve, our procedure generated the highest improvements for all resources.

Even in terms of daily workloads (i.e., the cumulated allocation level) and overallocation, the proposed solution seems superior. The respect of the due date is an issue of vital importance and so over allocation and overtime are levers of action that the PM can use at his/her will. However, the solution generated by the PM made reckless use of overtime, as the daily workload of R2 and R5 have peaks of 150%. Note that this value is even higher than the maximum threshold limit (of 140%) suggested by the PM as an additional constraint. A similar event does never take place in a solution generated by the automatic tool; also, and most important, R4 and R5 are never overallocated and the maximum daily workload of R6 (the most used junior) is just equal to 125%. This creates a better working environment, by avoiding too much stress and too many expectations on younger and less experienced resources. Similar considerations hold also concerning multi-tasking and job fragmentation, as the number of parallel tasks assigned to the same resource is never too high. This makes sure that juniors can focus on a few activities,

leaving to the seniors the supervision of several tasks and the overall vision of the progress of the project.

The quality of our allocation is graphically shown in Figure 7.5, which displays the values of

cumulated assignment matrix $A\left[\alpha_{r,\tau}\right]$, obtained as output of the allocation procedure. In other words, every bar of the chart corresponds to the workload (or cumulated assignment level) of a resource on a specific day. For instance, the first bar of the graph one on row two, shows that the

cumulated assignment level of resource R4 on day 11 equals 50% (i.e., $\alpha_{R4}(11) = 50\%)$. On the same chart, the cumulated assignment relative to the allocation plan made by the PM is shown in light grey. It is thus evident how the workloads generated by our tool is much more levelled and with fewer peaks.

On the other hand, the solution proposed by the PM is a little less costly, with a saving of 1%, which is probably due to the frequent use of more expensive but also more performing resources with an excellent cost to productivity ratio. Yet, this small saving is to the detriment of skills' improvements, as juniors are under-utilized.

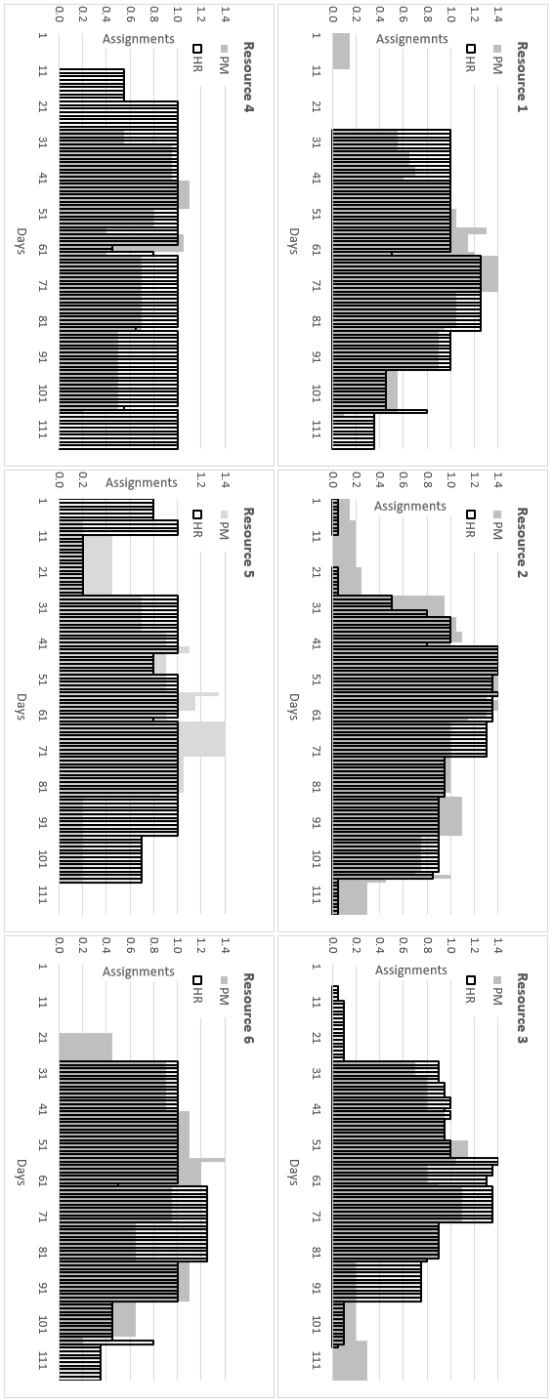


Figure 7.5. Comparison of the daily workloads generated by the heuristic (HR) and by the Project Manager (PM) (Bertolini, Neroni, and Zammori, 2020).

7.3. Project Time Deployment

7.3.1. Introduction

Lean manufacturing represents a successful paradigm in industrial and management contexts. Nowadays, there is unanimous agreement that its implementation leads to design, performance, and management benefits (Womack and Jones, 2003). In the last years, in recognition of the success of lean principles in manufacturing sectors characterised by high-volume low-variability productions (e.g. Vlachos, 2015), the amount of papers implementing lean outside the repetitive production environment has increased (Meng, 2019).

However, for Engineer-to-Order (ETO) companies, traditional lean tools and methods usually do not fit (Romero and Chavez, 2011) or have to be limited in use for lean improvements in simple processes (Al-Sudairi, 2007). ETO systems are typically characterized by low production volumes, extreme variability, long production lead times, high number of tasks involved, high cycle times, and fixed place manual assembly. Because of these substantial differences between ETO and repetitive production environments, the lean principles still apply, while tools and methods must be readapted depending on implementation context (e.g. Babalola, Ibem, and Ezema, 2019). Only with these new tools and methods it is possible to think to overcome one of the major constraints to unfold the full potential of lean in non-repetitive production environments.

For the above reasons, the development of new tools and methods able to support the implementation of lean principles in ETO production environments represents an important potential field of research, which could lead to important benefits in the realisation of automated warehouses too.

In particular, focusing the attention to the conventional initial step of lean implementation, i.e. the analysis of wastes and losses of a process, Value Stream Mapping (VSM) is the tool that has received the most attention in terms of evolution and adaptation for the use in ETO environments. VSM is a comprehensive analysis and visualisation tool used to illustrate the main processes and their operations, together with the lead times, buffers, and information flows (Rother and Shook, 2003). A typical shortcoming of applying the traditional VSM in an ETO environment is that it fails to map multiple products with different routings (Braglia, Carmignani, and Zammori, 2006). Matt (2014) reviewed the most relevant literature in the field of VSM and discusses its limitations regarding the application in an ETO environment. Then, a set of guidelines was developed based on an industrial case study. In the context of ETO construction industries, some researchers report about value stream macro-mapping (Fontanini and Picchi, 2004) or value network mapping (Khaswala and Irani, 2001) approaches especially focusing on the supply chain of a specific ETO company in the construction industry. Note that all tools cited above are very useful for understanding and improving manufacturing and managerial performances, although they are too

specific and limited to quantify the criticalities in ETO production environments, or excessively simplified to guide the selection and prioritization of improvement actions to reduce or eliminate the causes of losses. Probably, VSM is definitely not the most suitable tool to quantify the overall inefficiencies in ETO production environments.

On the other hand, Suri (1998) developed Manufacturing Critical-path Time (MCT). MCT is a time-based metric that defines lead time in a precise way so that it properly quantifies losses indicating the opportunities for improvements. Based on the aims and scope of Quick Response Manufacturing (QRM), the MCT is 'the typical amount of time from when a customer submits an order, through the critical-path, until the first end-item of that order that is delivered to the customer.' However, MCT is missing a crucial aspect to ensure the correct interventions to reduce losses: it quantifies the inefficiencies on a time scale, but it does not explicitly provide any information concerning the specific impact of any single loss. In this way, it is hard to find the most effective improvement actions to reduce or eliminate the losses.

To fill this gap, a new tool, named Project Time Deployment (PTD), is presented. The PTD combines and integrates the MCT with the Manufacturing Cost Deployment (MCD) (Yamashina and Kubo, 2002), which is the fundamental pillar of World Class Manufacturing (WCM). Note that, although PTD shares a similar matrix-based structure and an analogous logic with MCD, it is a (lead) time-based approach. Specifically, it quantifies each loss in terms of time lost. In other words, according to Suri (2011), the PTD allows the identification of inefficiencies and the comparison of several different losses synthesizing them in a single and easy-to-read quantitative representation: the time. For these reasons, the integration of the MCT with the MCD can represent a winning solution *(i)* to compensate the poor literature concerning the identification and quantification of inefficiencies in ETO production environments, and *(ii)* to identify and prioritize the improvement actions to reduce or eliminate the causes of losses.

7.3.2. Digression on the Manufacturing Critical-path Time (MCT)

The MCT is a time-based metric published by Suri (1998) in the context of supply chain management. Ericksen et al. (2005) evinced the need for shifting to time-based supply management as well as the importance of using MCT as the key metric in this effort. Then, Ericksen et al. (2007) justified the use of MCT as a robust, unifying, and enterprise-wide metric to be used to support order fulfilment and drive continuous-improvement projects. Recently, Suri (2011) focused on detailed rules about calculating MCT in practice and Chong and Ching (2014) conducted an MTC study in a job-shop environment. Thanks to its accuracy in estimating the time required to fulfil an order and quantifying the longest critical-path duration of order-fulfilment activities, the MCT is nowadays the main performance metric in the implementation of QRM.

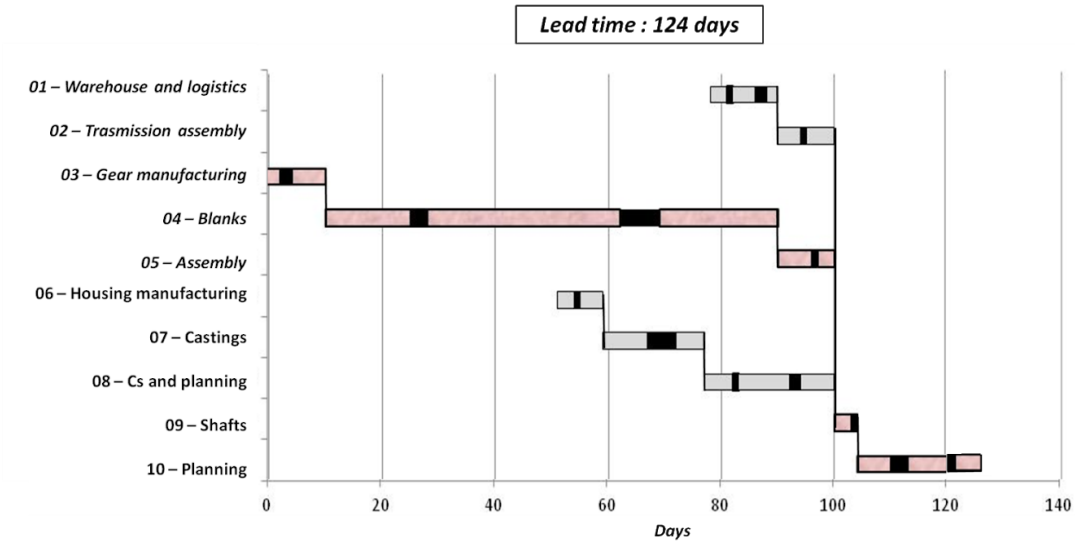


Figure 7.6. An MCT map for gears manufacturing process [3].

The initial goal is to produce an MCT map that graphically represents the flow of an order and to calculate the corresponding lead time (i.e. the MCT value). If there are multiple paths involved together the MCT is the value for the longest path from start to finish. The flow moves from left to right and its graphic representation is given by a coloured bar alternatively of grey and white. The grey spaces mark the total time when a resource is physically busy on the order. The white space, which is usually placed before the grey space of the same task, illustrates the remaining time spent when non-value-added activities are taking place. Examples of these activities include inventories, expediting of late jobs, rescheduling and, in general, all the activities the customer is not willing to pay for.

Specifically, Figure 7.6 illustrates a recurring situation, where the grey space is only a minimal part of the MCT. It is important to highlight here that only for a small part of the lead time the product is used in processing value-added activities. Consequently, the biggest opportunities for improvement are represented by the activities that are sketched in the white spaces (Suri 2011). Therefore, the MCT does not need to be data-intensive and it might be relatively easy-to-use and apply. Thus, by comparing ‘before’ and ‘after’ values, MCT provides a simple yet powerful metric with which to suggest improvements.

7.3.3. Digression on the Manufacturing Cost Deployment (MCD)

The objective of the MCD is to establish a systematic cost-reduction program (Yamashina and Kubo, 2002). In order to reach its fundamental objective, MCD uses a precise and well-structured procedure, which is rigorously supported using five matrices:

- A-Matrix is used to classify losses within the production system;

- B-Matrix clarifies cause-and-effect relationships among losses;
- C-Matrix presents the conversion of losses into manufacturing costs;
- D-matrix suggests possible improvement actions for losses;
- E-matrix presents the investment efficiency associated with each improvement action.

Later, MCD has been constantly refined through benchmarking with several automotive companies. At first, it was implemented at Volvo (Garbe and Olausson, 2014). Then, it has been integrated into the WCM model, at Fiat Group Automobiles Production Systems (FAPS), as a systematic way to sustain manufacturing cost reduction (Silva et al., 2013). It is important to highlight how the MCD, due to its structured step-by-step features, has been recently integrated with other tools, and adapted in production contexts different from the automotive one. For instance, Carmignani (2017) proposed a framework to approach the supply scrap management process (SSMP) and Abisourour (2019) developed an integrated management system combining costs deployment with VSM. Furthermore, Braglia et al. (2019) presented a modified MCD tool conceived to deal with the inefficiencies of ETO manual assembly tasks and Braglia et al. (2020) developed Energy Cost Deployment, a structured approach to tackling energy losses.

The MCD is an operative tool able to visualize in a structured and immediate way all the inefficiencies that affect the manufacturing process, and it sets the focus on areas where the greatest casual losses are placed, providing opportunities for greater efficiency and effectiveness in reducing and eliminating them. Finally, it also facilitates the selection of improvement activities to be activated to mitigate/remove the root causes of such losses.

7.3.4. Losses and processes classification

The PTD is an iterative well-structured procedure specifically conceived for ETO production environments. The PTD, by combining and integrating the MCT with some peculiarities of MCD, allows the identification of losses and facilitates the planning of interventions for their reduction.

Before describing the PTD, a possible classification of losses is presented, and a definition of the main business processes to investigate in which these losses occur is needed.

In general, within all ETO companies, a *'loss'* is represented by the amount of time that is lost in not-value adding activities. Therefore, the gap between the *actual productive time*, in which a task is processed under optimal operating conditions, and the *planned working time* can be viewed as the consequence of multiple causes of inefficiency. Due to planned and unplanned stops, only a portion of time is effectively used for value-adding activities.

As shown in Figure 7.7, the losses that can occur during an ETO project are divided into five main categories that specify the nature of each considered loss. In this systematic way, it is possible to evaluate where a loss occurs and subsequently to estimate the portion of time lost. The

classification of losses has been conceived enriching losses presented in the literature (Braglia et al., 2019; Aziz and Hafez, 2013; Lapinski, Horman, and Riley, 2006) with typical criticalities of ETO companies collected during several interviews, recording testimonies of project managers, logistics and manufacturing employers, and responsible personnel from the design office.

Owing to these considerations, losses might be classified into two main categories: *external* and *internal losses*. External losses are due to inefficiencies that are external to the ETO project under evaluation, while internal losses are due to inefficiencies directly ascribable to the ETO project. External losses are then subdivided into Unpredictable Losses (UL) and External Cross-project Losses (ECL). UL are due to unpredictable events, they cannot be predicted and, consequently, no improvement actions can be adopted (e.g. earthquakes, exceptional events, black-outs, outbreaks, strikes, etc.). ECL refers to inefficiencies inside the company, but not directly imputable to the analysed ETO project. These kinds of losses might be avoided by adopting improvement techniques, but their planning and definition are not the competence of managers involved in the project under exam (e.g. machines/equipment failures, maintenance operations, safety controls, lack of personnel involved in other projects, lack of materials, machines/equipment used in other projects, etc.).

Similarly, internal losses can be further divided into losses that are internal to the project and external to a specific task (Project Level Losses, PLL) (e.g. lack of personnel/materials/machines/equipment shared with other tasks, cross-areas communication problems, correction of defects, wrong documentation coming from previous tasks, etc.), and those that can be directly ascribable to a specific task. Losses directly imputable to a specific task can also be subdivided into Task Level Performance Losses (TLPL) (e.g. inadequate design, wrong scheduling, documentation mistake, wrong tool equipment, delays due to under-skilled resources, etc.) and Task Level Quality Losses (TLQL) (e.g. non-compliance with internal standards, non-compliance with customers' requirements, non-compliance with legal requirements, etc.). TLPL includes losses specific for each task, directly ascribable to it. TLQL includes failures to meet quality requirements. Their impact in terms of time is not directly observable, although it might be quantified by looking at rework activities they involve.

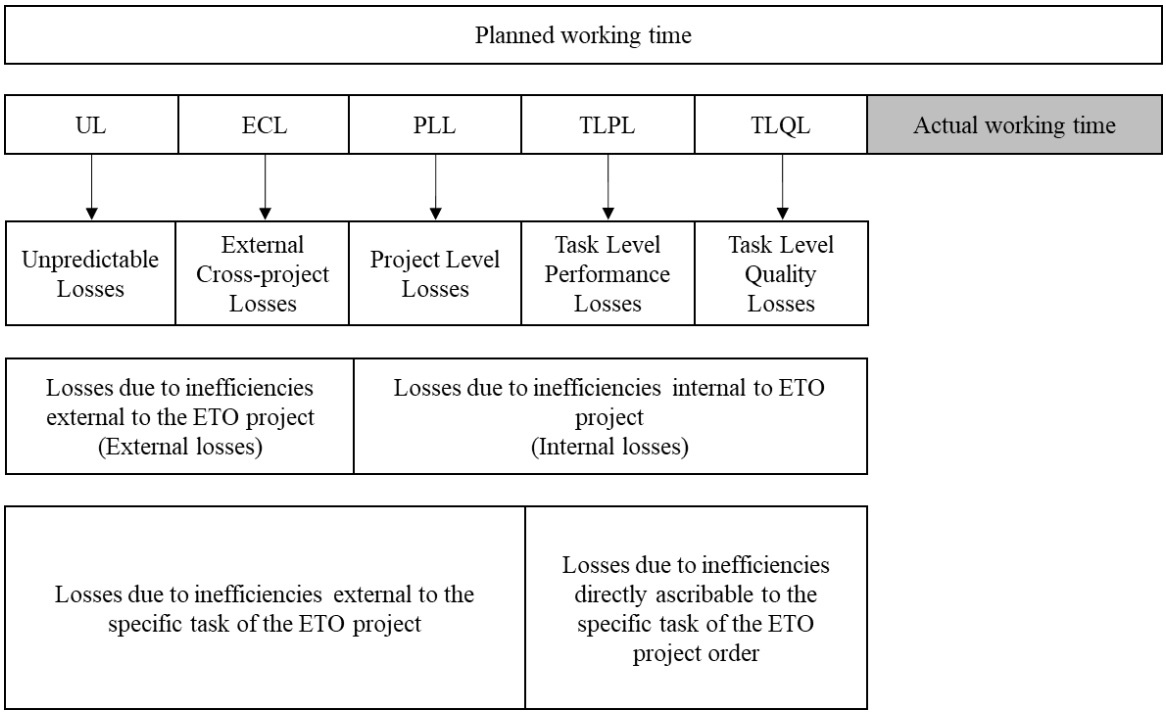


Figure 7.7. The PTD losses classification structure [3].

Concerning the classification of business process involved, that one already proposed by Strandhagen et al. (2018) is suggested:

- *Sales.* This category includes reception of requests from customers, contract negotiation and development, as well as preliminary choices concerning the product selection or configuration.
- *Engineering.* It includes software, electrical, structural, and mechanical design activities, as well as the development of documents and instructions needed in production, assembly and testing.
- *Procurement*. It includes the purchasing of raw materials and components based on the Bill of Material (BOM).
- *Production*. This category includes production, assembly, final shipment to customer, pre-testing, and final testing.
- *Management.* It includes managerial and workforce planning activities from the confirmation of sale until the end of final testing.

This is to consider as a first-level decomposition. It is noteworthy that aggregating tasks in business processes dramatically increases the planning process, consequently, a further level of decomposition would be useful to better locate the project inefficiencies. However, a general

definition of a second decomposition level is difficult to define, since it strictly depends on the manufactured product (i.e. automated warehouse).

7.3.5. The proposed approach

The PTD is developed in five steps:

- Step 1. Mapping the current state of the value chain via MCT
- Step 2. The A-Matrix: identifying losses in all tasks
- Step 3. The B-Matrix: defining cause-effect relationships between losses
- Step 4. The C-Matrix: quantifying the overall time lost
- Step 5. The D-Matrix: identifying and prioritizing improvement techniques to tackle losses

In the first step of the PTD, the MCT is mapped. The MCT map is roughly drawn to scale, so the magnitude of the various elements is evident. Thanks to its accuracy in estimating the time to fulfil an order, the MCT quantifies the longest critical-path duration of order-fulfilment activities. In this way, the map provides a high-level picture of opportunities for improvement and provides a benchmark to gauge subsequent improvements.

Note that, the MCT makes it possible to evaluate the losses that should be primarily addressed by improvement activities, i.e. the losses on the critical-path. However, the MCT highlights the overall time spent when non-value-added activities take place (i.e. *whites spaces*) without providing any information concerning the specific impact of any single loss.

Subsequently, the A-Matrix is built (Figure 7.8). This matrix classifies which loss occurs in which business process. The A-Matrix reports each single loss type in the columns, while the locations in which losses can occur are reported in rows. Figure 7.8 shows a two-level decomposition where the main business processes (i.e. first-level decomposition) are decomposed in tasks (i.e. T_i). The A-Matrix also highlights the critical tasks, previously highlighted with the MCT, by using a mark. If a loss arises in a given location, then the corresponding cell in the A-Matrix will be not-empty and a qualitative evaluation, concerning the parameter chosen, is provided. A three-colour representation can thus be used: red for very important losses, yellow for important losses, and green for minimal losses. Analysts can decide to adopt any criteria to assign colours or symbols and can also adopt four or five classes.

		Sales				Engineering				Procurement				Production				Management			
	Critical Task	x	x						x				x		x			x			
Loss category	**Loss type**	T1	T2	…	TS	T1	T2	…	TE	T1	T2	…	TU	T1	T2	…	TD	T1	T2	…	TM
Unpredictable Losses (UL)	Loss 1																				
	Loss 2																				
	…																				
	Loss M																				
External Cross-project Losses (ECL)	Loss 1																				
	Loss 2																				
	…																				
	Loss N																				
Project Level Losses (PLL)	Loss 1																				
	Loss 2																				
	…																				
	Loss O																				
Task Level Performance Losses (TLPL)	Loss 1																				
	Loss 2																				
	…																				
	Loss P																				
Task Level Quality Losses (TLQL)	Loss 1																				
	Loss 2																				
	…																				
	Loss Q																				

Legend:

- No losses
- Minimal losses
- Important losses
- Very important losses

Figure 7.8. The A-Matrix [3].

When a loss occurs in a generic task, it is plausible that other losses will occur elsewhere because of that loss. For example, wrong technical documentation may originate an operator mistake during an assembly task. Hence, the cause-effect relationship between losses should not be neglected, as no effective improvement can be carried out if this aspect is ignored. In this regard, a loss, in a given location, is defined as *resultant* if it originates from one or more losses, otherwise, it is *causal*. The B-Matrix (Figure 7.9) includes causal losses (and the task they concern) in the rows, and the possible resultant losses in columns. A mark is used to indicate which causal loss relates to which resultant loss.

It is important to note that only losses belonging to critical-path are reported in columns, because they are the only ones whose impact is to be considered significant for the lead time reduction. Conversely, all causal losses (i.e. critical and not) are reported in rows, since non-critical losses may have a noteworthy impact on critical ones, and in this case, it is important to intervene on them to eliminate the loss at the root. Besides, more than one resultant loss may be related to a single causal loss. Consequently, a loss in a specific location cannot be both causal and resultant at the same time.

				Resultant losses															
	Where			Sales								Management							
				Task 1			…	Task S			…	Task 1			…	Task M			
				Loss r_1	…	Loss r_R	…	Loss r_1	…	Loss r_R	…	Loss r_1	…	Loss r_R	…	Loss r_1	…	Loss r_R
Causal losses	Sales	Task 1	Loss c_1	•								•		•				
			…															
			Loss c_C	•														
		…																
		Task S	Loss c_1			•		•		•								
			…															
			Loss c_C													•		
	…																	
	Management	Task 1	Loss c_1	•														•
			…															
			Loss c_C	•														
		…																
		Task M	Loss c_1			•				•								
			…															
			Loss c_C							•						•		

Figure 7.9. The B-Matrix [3].

The C-Matrix (Figure 7.10) is then used to quantify each loss in terms of time lost. For each causal loss (reported in row) the C-Matrix shows: *(i)* the time lost in the task where that loss occurs and affecting the critical-path (i.e. direct critical time lost, *DCTL*); and *(ii)* the indirect time lost due to cause-and-effect relationships and affecting the critical-path (i.e. indirect critical time lost, *ICTL*). Note again that any impact not affecting the critical-path is not to consider, since its reduction would not provide any tangible improvement.

The notation used in Figure 7.10 is the following. $C = (c_1, \ldots, c_h, \ldots, c_C)$ and $R = (r_1, \ldots, r_k, \ldots, r_R)$ are, respectively, the set of causal and resultant losses introduced in the previous step. Then, $i = 1, \ldots, N$ refers to tasks affected by the causal losses, while the index $j = 1, \ldots, M$ refers to the tasks affected by resultant losses. According to this, $t(c_h, i)$ refers to the amount of time directly ascribable to the causal loss c_h in task *i*. Also, $t_{(c_h,i)}(r_k, j)$ refers to the amount of time that is lost because of a resultant loss r_k occurring in task *j*, which is caused by causal loss c_h occurring in task *i*. Then, since a distinction between critical and non-critical tasks is due, the coefficient γ is introduced, where γ is equal to 1 if a generic task is critical, 0 otherwise.

Once $t(c_h, i)$ and $t_{(c_h,i)}(r_k, j)$ are available, it is possible to calculate the $DCTL_{(c_h,i)}$ and the $ICTL_{(c_h,i)}$ ascribable to the causal loss c_h occurring in task *i*. Finally, the corresponding total time lost (*TTL*) is:

(7.12)

$$TTL_{(c_h,i)} = DCTL_{(c_h,i)} + ICTL_{(c_h,i)} = t(c_h, i) \bullet \gamma_i + \sum_j \gamma_j \bullet t_{(c_h,i)}(r_k, j)$$

			Resultant losses							Direct critical time lost (DCTL)	Indirect critical time lost (ICTL)	Total time lost (TTL)
			Task 1			…	Task M					
			Loss r_1	…	Loss r_R		Loss r_1	…	Loss r_R			
Causal losses	Task 1	Loss c_1	$t_{(c_1,1)}(r_1, 1)$		$t_{(c_1,1)}(r_R, 1)$		$t_{(c_1,1)}(r_1, M)$		$t_{(c_1,1)}(r_R, M)$	$t(c_1, 1)\cdot\gamma_1$	$\sum\gamma_j \bullet t_{(c_1,1)}(r_k, j)$	$DCTL_{(c_1,1)} + ICTL_{((c_1,1)}$
		…										
		Loss c_h	$t_{(c_h,1)}(r_1, 1)$		$t_{(c_h,1)}(r_R, 1)$		$t_{(c_h,1)}(r_1, M)$		$t_{(c_h,1)}(r_R, M)$	$t(c_h, 1)\cdot\gamma_1$	$\sum\gamma_j \bullet t_{(c_h,1)}(r_k, j)$	$DCTL_{(c_h,1)} + ICTL_{(c_h,1)}$
		…										
		Loss c_C	$t_{(c_C,1)}(r_1, 1)$		$t_{(c_C,1)}(r_R, 1)$		$t_{(c_C,1)}(r_1, M)$		$t_{(c_C,1)}(r_R, M)$	$t(c_C, 1)\cdot\gamma_1$	$\sum\gamma_j \bullet t_{(c_C,1)}(r_k, j)$	$DCTL_{(c_C,1)} + ICTL_{(c_C,1)}$
	…	…										
	Task N	Loss c_1	$t_{(c_1,N)}(r_1, 1)$		$t_{(c_1,N)}(r_R, 1)$		$t_{(c_1,N)}(r_1, M)$		$t_{(c_1,N)}(r_R, M)$	$t(c_1, N)\cdot\gamma_N$	$\sum\gamma_j \bullet t_{(c_1,N)}(r_k, j)$	$DCTL_{(c_1,N)} + ICTL_{(c_1,N)}$
		…										
		Loss c_h	$t_{(c_h,N)}(r_1, 1)$		$t_{(c_h,N)}(r_R, 1)$		$t_{(c_h,N)}(r_1, M)$		$t_{(c_h,N)}(r_R, M)$	$t(c_h, N)\cdot\gamma_N$	$\sum\gamma_j \bullet t_{(c_h,N)}(r_k, j)$	$DCTL_{(c_h,N)} + ICTL_{(c_h,N)}$
		…										
		Loss c_C	$t_{(c_C,N)}(r_1, 1)$		$t_{(c_C,N)}(r_R, 1)$		$t_{(c_C,N)}(r_1, M)$		$t_{(c_C,N)}(r_R, M)$	$t(c_C, N)\cdot\gamma_N$	$\sum\gamma_j \bullet t_{(c_C,N)}(r_k, j)$	$DCTL_{(c_C,N)} + ICTL_{(c_C,N)}$

Figure 7.10. The C-Matrix [3].

Once causal losses have been quantified using the C-Matrix in terms of time lost, it is necessary to evaluate which improvement techniques and corrective actions are appropriate for tackling each loss previously identified and quantified. The D-Matrix presents the causal losses in the rows, and the admissible improvement techniques in columns (Figure 7.11). The improvement techniques are classified into two main categories: (i) the specific lean management tools and techniques (a clear explanation of the various available lean improvement tools and techniques is presented in Zahraee (2016)), and (ii) the additional improvement actions (e.g. engineering activities that involve some project technical modifications or the implementation of a training program).

Then, the D-Matrix prioritises the improvement interventions. Specifically, each possible combination tool-loss is ranked using the TCE index. The TCE is the acronym of Time impact, Cost and Easiness and represents a qualitative estimation of the intervention. Given a causal loss c_h occurring in task *i*, and given an applicable improvement technique, the corresponding $TCE_{c_h,i}$, is:

$$TCE_{c_h,i} = T_{c_h,i} \bullet C_{c_h,i} \bullet E_{c_h,i} \tag{7.13}$$

where:

- Time impact factor ($T_{c_h,i}$) expresses qualitatively, on a 1 to 5 scale, the time impact of the loss c_h occurring in task *i* on the critical-path. T = 1 means low impact, while T = 5 means high impact.
- Cost factor ($C_{c_h,i}$) expresses, on a 1 to 5 scale, the economic weight of costs that should be sustained to improve the system by removing or reducing the causal loss c_h occurring in task *i*. C = 1 means high cost, while C = 5 means low cost.
- Easiness factor ($E_{c_h,i}$) represents, on the same scale, the simplicity, in terms of resources and time, of the actions that are necessary to reduce/eliminate the causal loss c_h occurring in task *i*. E = 1 means low ease, while E = 5 means high ease.

Thus, the TCE index qualitatively expresses the degree at which the loss may be attacked, on a scale ranging from 1 to 125. The higher the value, the greater the capacity of the technique to tackle causal loss c_h occurring in task *i*.

Since the proposed approach is strictly focused on time, it is evident that, when losses owning to critical-path are reduced, the critical-path can change. Hence, before making any further improvement, the whole project must be reanalysed to find out the new critical losses. MCT should be recalculated after the corrections to see whether the losses have gone down, and to quantify the novel critical-path. This makes the PTD an effective tool only if iteratively implemented according to the philosophy of *continuous improvement*.

	Improvement techniques									
	Lean management techniques							**Additional improvement actions**		
	TPM	Standardized work	…	Poka-Yoke	Visual management	The 5S method	Six Sigma	Design modification of the equipment	…	Internal training

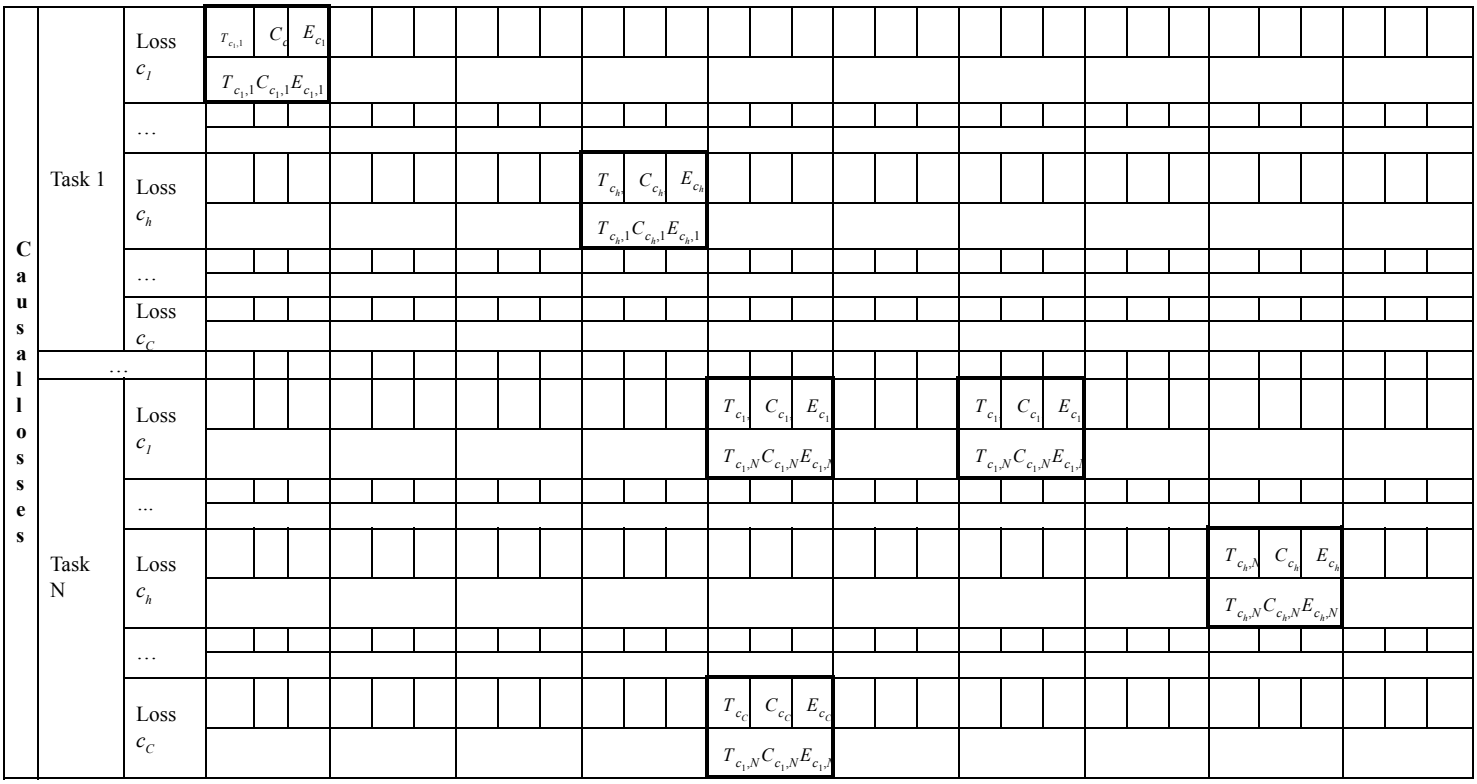


Figure 7.11. The D-Matrix [3].

7.3.6. The case study

In order to validate it, the PTD is applied to an Italian EPC company located in Northern Italy and operating in the steel sector. The core business of the company consists in the design and installation of AS/RSs for the steel sector. Operating in a very competitive scenario, the company needs to consider some of the highly innovative features to reduce the lead time and to increase the overall quality. To this extent, the PTD has been recognised by the company as a valid tool to identify the hidden losses, to quantify the wastes and to identify the possible improvement actions. Being this the first implementation of an experimental tool such as PTD, it was decided to take into consideration a consolidated product. A Vertical Lift Module (VLM) designed for the steel sector like that presented in chapter 5.

The tasks and losses identified in the realisation of the VLM are reported in Table 7.10 and 7.11, where, to each of them is associated a univoque code to briefly identify it.

Table 7.10. Breakdown of tasks involved [3].

Business processes	Tasks	Code
Sales	Negotiation with customer	S1
Engineering	Software design	EN1
	Electronic design	EN2

	Mechanical design	EN3
	Structural design	EN4
Procurement	Suppliers selection	PU1
	Negotiation with suppliers	PU2
Production	Manufacturing	PO1
	Assembly in-house	PO2
	Shipment	PO3
	Assembly in-place	PO4
	Start up and testing	PO5
Management	Schedule organization	M1
	Kaizen meetings	M2

Table 7.11. Loss types considered[3].

Loss category	**Loss type**	**Code**
Unpredictable Losses	Strike	UL1
External Cross-project Losses	Machine failure	ECL1
	Lack of personnel involved in other projects	ECL2
	Lack of materials used in other projects	ECL3
	Lack of machines/equipment used in other projects	ECL4
Project Level Losses	Lack of personnel shared with other tasks	PLL1
	Lack of materials shared with other tasks	PLL2
	Lack of machines/equipment shared with other tasks	PLL3
	Correction of defects due to previous operations	PLL4
Task Level Performance Losses	Manufacturing mistake	TLP1
	Inadequate design	TLP2
	Documentation mistake	TLP3
	Assembly mistake	TLP4
	Under-skilled resource assigned to the task	TLP5
Task Level Quality Losses	Quality mismatch with internal standards	TLQ1
	Quality mismatch with customer requirements	TLQ2
	Quality mismatch with legal requirements	TLQ3

The MCT is reported in Figure 7.12. It shows the complexity of the project, characterised by a high level of parallelism at the beginning, and a long set of critical tasks at the end. The resulting lead time was about 127 days.

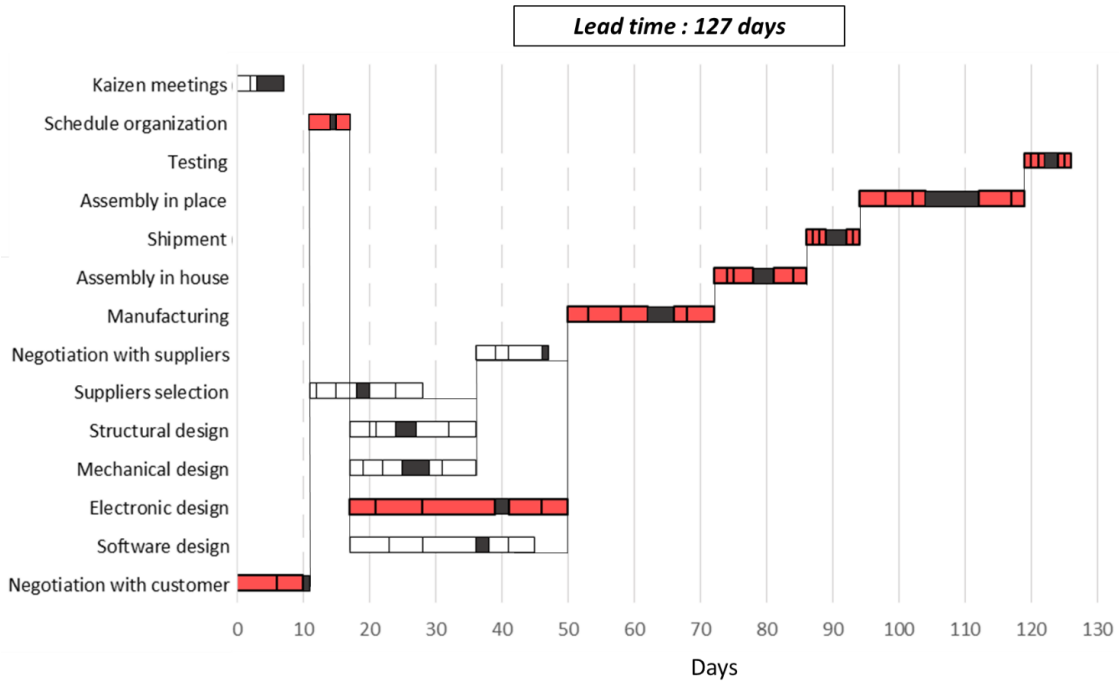


Figure 7.12. MCT representation for the case study [3].

The A-Matrix is reported in Figure 7.13. Losses are classified according to three different levels of criticality, where red cells correspond to very important losses, yellow cells to important losses, and green cells to minimal losses. This first screening criterion considers the amount of time lost that is qualitatively estimated by the working team. The A-Matrix also visualizes the critical tasks highlighted with the MCT by using a mark.

		Tasks													
Critical Task		X		X					X	X	X	X	X	X	
Loss category	**Code**	S1	EN1	EN2	EN3	EN4	PU1	PU2	PO1	PO2	PO3	PO4	PO5	M1	M2
Unpredictable Losses	UL1														
External Cross-project Losses	ECL1														
	ECL2														
	ECL3														
	ECL4														
Project Level Losses	PLL1														
	PLL2														
	PLL3														
	PLL4														
Task Level Performance Losses	TLP1														
	TLP2														
	TLP3														
	TLP4														
	TLP5														
Task Level Quality Losses	TLQ1														
	TLQ2														
	TLQ3														

Figure 7.13. A-Matrix for the case study [3].

Cause and effect relationships between losses are reported in the B-Matrix (Figure 7.14) [3].

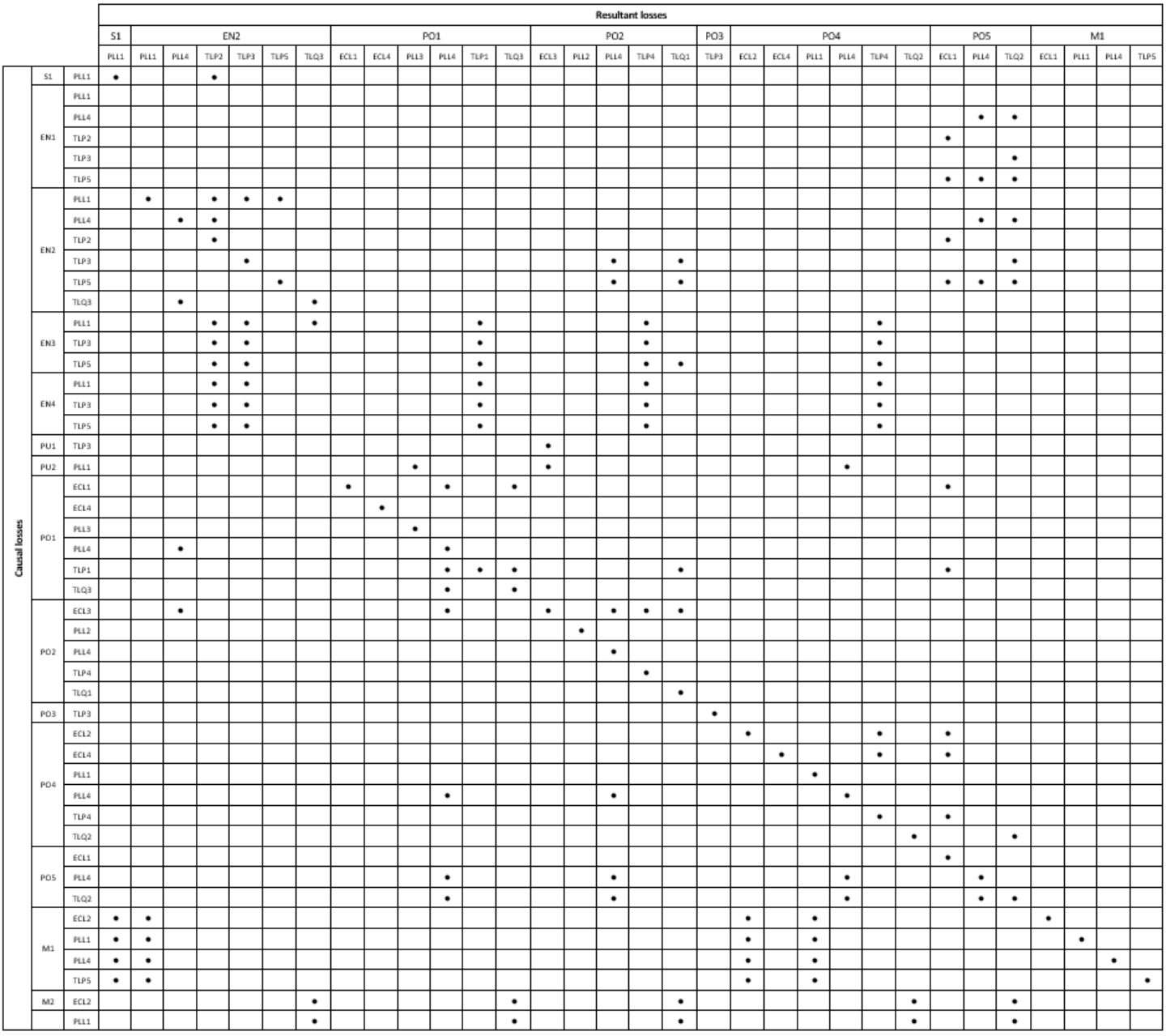


Figure 7.14. The B-Matrix for the case study [3].

The C-Matrix is then used to quantify each loss in terms of time lost (Figure 7.15). Time is expressed in working days, and, to simplify the readability of the matrix, only the DCTL and the ICTL columns are reported.

	Tasks	Loss types	Direct critical time lost (DCTL)	Indirect critical time lost (ICTL)	Total time lost (TTL)
Causal Losses	S1	PLL1	7	1	8
	EN1	PLL1	0	0	0
		PLL4	0	0.5	0.5
		TLP2	0	2	2
		TLP3	0	0.5	0.5
		TLP5	0	0.5	0.5
	EN2	PLL1	3	0.5	3.5
		PLL4	0.5	1	1.5
		TLP2	2	1	3
		TLP3	0.5	1	1.5
		TLP5	0.5	1	1.5
		TLQ3	2	0.5	2.5
	EN3	PLL1	0	1	1
		TLP3	0	1	1
		TLP5	0	1	1
	EN4	PLL1	0	1	1
		TLP3	0	1	1
		TLP5	0	1	1
	PU1	TLP3	0	1	1
	PU2	PLL1	0	1	1
	PO1	ECL1	0.5	0.5	1
		ECL4	1	0	1
		PLL3	1	0	1
		PLL4	3	1	4
		TLP1	2	1	3
		TLQ3	1	0.5	1.5
	PO2	ECL3	1	1	2
		PLL2	0.5	0	0.5
		PLL4	3	0	3
		TLP4	0.5	0	0.5
		TLQ1	1	0	1
	PO3	TLP3	2	0	2
	PO4	ECL2	1	1	2
		ECL4	0.5	1	1.5
		PLL1	1	0	1
		PLL4	3	3	6
		TLP4	2	0.5	2.5
		TLQ2	0.5	0.5	1
	PO5	ECL1	0.5	0	0.5
		PLL4	0.5	0.5	1
		TLQ2	0.5	0.5	1
	M1	ECL2	0.5	0.5	1
		PLL1	0.5	0.5	1
		PLL4	0.5	0.5	1
		TLP5	0.5	0.5	1
	M2	ECL2	0	1	1
		PLL1	0	2	2

Figure 7.15. C-Matrix for the case study [3].

Once causal losses have been quantified using the C-Matrix, it is necessary to evaluate which improvement techniques to start first. Consequently, the D-Matrix is compiled (Figure 7.16) and each improvement is ranked using the TCE index. Note that, only the lines with at least a TCE value equal to or greater than 25 are reported.

			Improvement techniques																				
			Lean management techniques												**Additional improvement actions**								
	Tasks	**Loss types**	Process analysis and VSM			Poka-Yoke			Visual Management			Preventive Maintenance			Internal training			Internal communication			External communication		
Causal Losses	S1	PLL1	100						25									45					
			5	4	5				5	1	5							5	3	3			
	EN2	PLL1																25					
																		1	5	5			
		TLP2																32					
																		2	4	4			
		TLQ3														25							
															1	5	5						
	P01	PLL4										36				16							
												4	4	2	4	2	2						
		TLP1										45											
												3	5	3									
		TLQ3										20											
												2	2	5									
	PO2	PLL4				45						27			18								
						3	5	3				3	3	3	3	3	2						
	PO5	PLL4													24								
															3	2	4						
		TLQ2													24								
															3	2	4						
	M1	ECL2	44															30			30		
			3	4	4													3	5	2	3	5	2
		PLL1	36															30			30		
			3	4	3													3	5	2	3	5	2
		PLL4													24								
															3	2	4						
		TLP5													24								
															3	4	2						
	M2	ECL2	36															30			30		
			3	4	3													3	5	2	3	5	2
		PLL1							50									125					
									5	5	2							5	5	5			

Figure 7.16. D-Matrix for the case study [3].

Concerning the Time impact factor, the conversion was made by using Eq.(7.14). Given $Max(TTL)$ the highest TTL registered in the C-Matrix, $s = 1, \ldots, S$ the lines of the C-Matrix, and TLL_s the TTL corresponding to causal loss in line s, for each improvement technique affecting the loss was assigned a Time impact factor (T_s) calculated as:

$$T_s = \lceil \frac{TTL_s \cdot 5}{Max(TTL)} \rceil \tag{7.14}$$

The Cost factor was defined looking at the working hours needed to implement the improvement action, while the Easiness factor was defined looking at the time needed to collect all necessary data to implement it. Table 7.12 reports the basis for the linguistic judgement scales used to estimate the Cost and the Easiness factors.

Cost factor	**Score**
The working hours needed to implement the improvement are ≥ 400	1
The working hours needed to implement the improvement are ≥ 300	2
The working hours needed to implement the improvement are ≥ 150	3
The working hours needed to implement the improvement are ≥ 80	4
The working hours needed to implement the improvement are ≥ 40	5
Easiness factor	**Score**
The estimated days needed to collect the required data are ≥ 360	1
The estimated days needed to collect the required data are ≥ 180	2
The estimated days needed to collect the required data are ≥ 90	3
The estimated days needed to collect the required data are ≥ 60	4
The estimated days needed to collect the required data are ≥ 30	5

Table 7.12. Conversion table for Cost and Easiness factors [3].

Then, regarding the scores of TCEs, it is possible to choose which improvement/corrective action to take first. According to this, the company decided to start the implementation of corrective actions following the TCE ranking. In particular, *(i)* a process analysis via Business Process Modelling Notation (BPMN) was implemented to reduce wastes in the negotiation with customers (S1). Then, *(ii)* the same BPMN analysis was made to reduce inefficiencies due to wrong scheduling and project staffing tasks (M1). In parallel with already mentioned techniques, *(iii)* a general improvement of internal communication was made to improve the already scheduled kaizen meetings (M2). Finally, *(iv)* a mathematical model for preventive maintenance in production

(PO1) was designed and tested via simulation. The latter improvement will lead to greater improvements in the long term, although, according to PTD, it was important to start with other improvements quicker to implement and easier to manage. After implementing the highlighted improvements, the lead time was reduced to 97 days with a reduction of about 24%. This result is the proof of the effectiveness of the proposed tool. Furthermore, PTD is not data intensive and does not need to store a huge amount of historical data.

References

Abisourour, J., M. Hachkar, B. Mounir, and A. Farchi. 2019. “Methodology for integrated management system improvement: combining costs deployment and value stream mapping.” International Journal of Production Research.

Akintoye, A. S., and M. J. MacLeod. 1997. “Risk analysis and management in construction.” International Journal of Project Management 15 (1): 31-38.

Alam, M., Gale, A., Brown, M. and Khan, A. 2010. The importance of human skills in project management professional development, Managing Project in Business, 3 (3), 495 – 516.

Al-Sudairi, A. A. 2007. “Evaluating the Effect of Construction Process Characteristics to the Applicability of Lean Principles.” Construction Innovation 7 (1): 99–121.

Aziz, R. F., and S. M. Hafez. 2013. “Apply lean thinking in construction and performance improvement.” Alexandria Engineering Journal 52: 679-695.

Babalola, O., E. O. Ibem, and I. C. Ezema. 2019. “Implementation of lean practices in the construction industry: A systematic review.” Building and Environment 148: 34-43.

Berthaut, F., Pellerin, R., Perrier, N. and Hajji, A, 2014. Time-cost trade-offs in resource-constraint project scheduling problems with overlapping modes, International Journal of Project Organisation and Management, 6 (3), 215 – 236.

Bertolini, M., Neroni, M., Zammori, F. (2020). A flexible operating tool to provide an efficient project’s staffing and resource allocation. *International Journal of Project Organisation and Management*.

Belout, A. 1998. Effect of human resources management on project effectiveness and success: toward a new conceptual framework, International Journal of Project Management, 16 (1), 21 - 26.

Braglia, M., D. Castellano, R. Gabbrielli, and L. Marrazzini. 2020. “Energy Cost Deployment (ECD): A novel lean approach to tackling energy losses.” Journal of Cleaner Production 246.

Braglia, M., G. Carmignani, and F. Zammori. 2006. “A New Value Stream Mapping Approach for Complex Production Systems.” International Journal of Production Research 44 (18/19): 3929–3952.

Braglia, M., M. Frosolini, M. Gallo, and L. Marrazzini. 2019. “Lean manufacturing tool in engineering-to-order environment: Project cost deployment.” International Journal of Production Research, 57(6): 1825-1839.

Brenner, D.A. 2007, Achieving a Successful Project by Motivating the Project Team, Cost Engineering, 49 (5), 16-20.

Brusco and Johns 2007. Staffing a Multiskilled Workforce with Varying Levels of Productivity: An Analysis of Cross-training Policies, Decision Sciences, 29 (2), 499 – 515.

Carmignani, G. 2017. “Scrap value stream mapping (S-VSM): a new approach to improve the supply scrap management process.” International Journal of Production Research, 55 (12): 3559-3576.

Carruthers J.A. and Battersby A. 1966. Advances in Critical Path Methods, Journal of the Operational Research Society, 17 (4), 359 - 380.

Certa, A., Enea, M., Galante, G., and La Fata, C.M. 2009. Multi-objective human resources allocation in R&D projects planning, International Journal of Production Research, 47 (13), 3503–3523.

Chong, K. E., and H. W. Ching. 2014. “Tool for mapping manufacturing critical-path time in job shop environment.” International Symposium on Research in Innovation and Sustainability, Malacca, Malaysia, 15-16 October, pp. 1585-1589.

Christofides N., Alvarez-Valdes R. and Tamarit J.M. 1987. Project scheduling with resource constraints: A branch and bound approach, European Journal of Operational Research. 29 (3), 262-273.

Elmaghraby, S.E. and Ramachandra, G. 2012, Optimal resource allocation in activity networks - Stochastic environment, International Journal of Project Organisation and Management,4 (4), 322 – 338.

Ericksen, P. D., R. Suri, B. El-Jawhari, and A. J. Armstrong. 2005. “Filling the Gap: Rethinking Supply Management in the Age of Global Sourcing and Lean.” APICS–The Performance Advantage 15 (2): 27–31.

Ericksen, P., N. Stoflet, and R. Suri. 2007. “Manufacturing Critical-path Time (MCT): the QRM metric for lead time.” Technical Report, University of Wisconsin-Madison, Center for QRM.

Fontanini, P. S., and F. A. Picchi. 2004. “Value stream macro mapping – a case study of aluminium windows for construction supply chain.” Proceedings of the 12th Annual Conference of the International Group for Lean Construction (IGLC-12), Elsinore, Denmark, 3-5 August.

Garbe, M, and G. Olausson. 2014. “Evaluation of Manufacturing Cost Models - An evaluation of manufacturing cost models for an assembly line and an evaluation of a data collection method.” Dissertation from Department of Industrial Production, Lund University.

Gollenbeck-Sunke, N. and Schultmann, F. 2010. Resource classification and allocation in project-based production environments, International Journal of Project Organisation and Management, 2 (2), 122 - 134.

Hartmann, S., and Briskorn, D. 2010, A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research, 207 (1), 1 - 14.

Heimerl, C. and Kolisch, R. 2010. Scheduling and staffing multiple projects with a multi-skilled workforce, OR Spectrum, 32 (2), 343 - 368.

Herroelen, W. 1972. Resource-constrained Project Scheduling - The State of the Art, Journal of the Operational Research Society, 23 (3), 261 - 275.

Joshi, D., Mittal, M.L. and Kumar, M. 2018. A Teaching–Learning-Based Optimization Algorithm for the Resource-Constrained Project Scheduling Problem, In Harmony Search and Nature Inspired Optimization Algorithms, 1101–1109.

Kastor, A. and Sirakoulis, K. 2009. The effectiveness of resource levelling tools for Resource Constraint Project Scheduling Problem, International Journal of Project Management, 27 (5), 493 - 500.

Khaswala, Z. N., and S. A. Irani. 2001. “Value Network Mapping (VNM): Visualization and Analysis of Multiple Flows in Value Stream Maps.” Proceedings of the Lean Management Solutions Conference, St. Louis, September 10–11.

Lapinski, A. R., Horman, M. J., and D. R. Riley. 2006. “Lean Processes for Sustainable Project Delivery.” Journal of Construction Engineering and Management 132 (10).

Liao, T.W., Egbelu, P.J., Sarker, B.R., Leu, S.S. 2011. Metaheuristics for project and construction management - A state-of-the-art review. Automation in Construction, 20 (5), 491-505.

Liu, S.S., & Wang, C.J. 2012. Optimizing linear project scheduling with multi-skilled crews. Automation in Construction, 24, 16 - 23.

Marzouk, M. 2009. Assessment of construction workforce skills needs in Egypt, International Journal of Project Organisation and Management, 1 (4), 398 - 407.

Maghsoudlou, H., Afshar-Nadjafi, B. and Niaki, S.T.A. 2016. A multi-objective invasive weeds optimization algorithm for solving multi-skill multi-mode resource constrained project scheduling problem. Computers and Chemical Engineering, 88, 157 - 169.

Matt, D. T. 2014. “Adaptation of the Value Stream Mapping Approach to the Design of Lean Engineer-to-order Production Systems: A Case Study.” Journal of Manufacturing Technology Management 25 (3): 334–350.

Meng, X. 2019. “Lean management in the context of construction supply chains.” International Journal of Production Research 57 (11): 3784-3798.

Montoya, C., Bellenguez-Morineau, O., Pinson, E., & Rivreau, D. 2014. Integrated Column Generation and Lagrangian Relaxation Approach for the Multi-Skill Project Scheduling Problem. In Handbook on Project Management and Scheduling, 565 - 586.

Myszkowski, P.B., Olech, Ł.P., Laszczyk, M., and Skowroński, M.E. 2018. Hybrid Differential Evolution and Greedy Algorithm (DEGR) for solving Multi-Skill Resource-Constrained Project Scheduling Problem, Applied Soft Computing Journal, 62, 1 - 14.

Patterson J.H. and Huber W.D. 1974. A Horizon-Varying, Zero-One Approach to Project Scheduling, Management Science, 20 (6), 990-998.

Pawiński, G., and Sapiecha, K. 2016. Speeding up global optimization with the help of intelligent supervisors, Applied Intelligence, 45 (3), 777 - 792.

PMI Global Standard, 2017. A guide to the project management body of knowledge PMBOK sixth edition, Project Management Institute, ISBN-10: 9781628251845.

Pritsker A.A.B, Waiters L.J. and Wolfe P.M., 1969. Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. Management Science, 16 (1), 93-108.

Romero, D., and Z. Chavez. 2011. “Use of Value Mapping Tools for Manufacturing Systems Redesign.” Proceedings of the World Congress on Engineering, London, July 6–8.

Rother, M., and J. Shook. 2003. Learning to see: Value Stream Mapping to Create Value and Eliminate Muda. Cambridge: Lean Enterprise Institute.

Silva, L. C. S., J. L. Kovaleski, S. Gaia, M. Garcia, and P. P. Andrade Júnior. 2013. “Cost Deployment Tool for Technological Innovation of World Class Manufacturing.” Journal of Transportation Technologies 3: 17–23.

Strandhagen, J. W., L. R. Vallandingham, E. Alfnes, and J. O. Stranhagen. 2018. “Operationalizing lean principles for lead time reduction in engineering-to-order (ETO) operations: A case study.” IFAC PapersOnLine 51(11): 128-133.

Suri, R. 1998. Quick Response Manufacturing: A Companywide Approach to Reducing Lead Times. New York: Productivity Press.

Suri, R. 2011. It's About Time. The Competitive Advantage of Quick Response Manufacturing. New York: Productivity Press.

Vanhoucke, M. 2012. Project Management with Dynamic Scheduling. Springer, Berlin, Heidelberg. Print ISBN: 978-3-642-25174-0

Vanhoucke, M. 2013. Project baseline scheduling: An overview of past experiences, Journal of Modern Project Management, 1 (2), 18 - 27.

Vanhoucke, M. 2014. Blended learning in project management an overview of operations research & scheduling group, Journal of Modern Project Management, 1 (3), 106 - 119.

Viana, A. and Pinho De Sousa, J. 2000. Using metaheuristics in multiobjective resource constrained project scheduling. European Journal of Operational Research, 120 (2), 359-374.

Vlachos, I. 2015. “Applying lean thinking in the food supply chains: a case study.” Production Planning & Control 26(16): 1351–1367.

Womack, J., and D. Jones. 2003. Lean thinking: banish waste and create wealth in your corporation. London: Free press.

Yamashina, H., and T. Kubo. 2002. “Manufacturing Cost Deployment.” International Journal of Production Research 40 (16): 4077–4091.

Zahraee, S. M. 2016. “A Survey on Lean Manufacturing Implementation in a Selected Manufacturing Industry in Iran.” International Journal of Lean Six Sigma 7 (2): 136–148.

Zamani, R. 2017. An evolutionary implicit enumeration procedure for solving the resource-constrained project scheduling problem, International Transactions in Operational Research, 24 (6), 1525 - 1547.

Zheng, H., Wang, L. and Zheng, X. 2017. Teaching–learning-based optimization algorithm for multi-skill resource constrained project scheduling problem. Soft Computing, 21 (6), 1537–1548.

8. Conclusions

In this work several algorithms, methodologies, frameworks, and operative policies for improving the logistics are presented. In particular, the work is focused on those automated storage and retrieval solutions that, even if used in industrial sectors of crucial importance such as the steel industry and the agricultural automotive, have been neglected for years by the scientific community.

The overall work is structured in 3 main parts and in each of them a different aspect of logistics is concerned. The first part is dedicated to the logistics of small products, the second one is dedicated to the logistics of big bulky products, and, finally, in the third part the managerial issues related to automated storage solutions discussed in the previous two sections are discussed. In each section several solutions are presented and validated in real industrial cases or, when it is possible, comparing them to the solutions already proposed in literature. Given the variety of the covered topics, for a more accurate overview of the results, the author invites the readers to refer to the case studies and computational analysis presented in each individual section.