Ph. D. Thesis

Decision Support Systems for Task Scheduling: Applications in Manufacturing and Healthcare

Manuel Alejandro Dios Rubio

supervised by
Prof. Jose Manuel Framiñán Torres

University of Seville

To my parents, Manuel and Laura, that with their effort have driven me to where I never thought I could arrive.

To my brothers, Carlos Alberto and Daniel, that have offered me their support from the beginning to the end of these long journey.

To Violeta, that has suffered this journey the same or more than me, but has been all the time encouranging me to continue.

Acknowledgements

I want to first acknowledge the person that has guided me during this long process, my advisor Jose M. Framinan. Thanks to him I started in the exciting world of research, that was a field that never attracted me before but now I do not want leave. He has served me as a reference of what being a good researcher means. I will be always grateful for all the time he has spent on me and all the help he has given me. And apart from the professional perspective, I would also like to thank him for his personal treatment and his closeness that have also served to make this difficult process much more bearable.

I cannot forget other persons that have helped me a lot and have made me be that comfortable in the academic world. First, I would like to make a special mention to Erwin Hans, that hosted me during my first research stay and showed me what means loving what you do. I still do not know what likes me more from him, the way he works or the way he is (probably the second). I would also like to make another special mention to Rainer Leisten, that hosted me in my second research stay and made me feel like if I were at home.

I also want to be thankful with the great support of my colleagues in the Department of Industrial Organisation and Business Management of the University of Seville. Although each of them has been really important to carry out this Thesis, I want to be specially grateful to the encourage, and the research and methodological help of Victor Fernandez, Roberto Dominguez, Jose M. Molina, Jose L. Andrade, Paz Perez and Salvatore Cannella.

Finally, I would like to write something in spanish for those persons that has helped me in my personal life to finish this Thesis. Me gustaria agradecer sobre todo a mi familia, a mis padres, Manuel y Laura, que nunca sabran todo lo que les agradezco lo que me han dado, a mis hermanos, Carlos Alberto y Daniel, que siempre están ahí para darme su apoyo en los buenos y en los malos momentos, y a Violeta, que seguramente se alegre hasta más que yo de que esto llegue a su fin, ha sido un largo camino que he superado en parte gracias a ti. También me gustaría hacer una mención especial a mi tía Carmen (o Mari) que fue la primera persona en la familia que se intereso por el mundo académico y de la que también he aprendido a valorar la ciencia,gracias tita! Por último, no querría que se me olvidara nadie pero gracias a todos los que me habéis aguantado y animado en este duro camino.

Contents

List of Figures

List of Tables

Part I

PRELIMINARIES AND INTRODUCTION

Chapter 1

Introduction and Objectives of the Thesis

This thesis focuses on the problem of *task scheduling*. Although slightly different definitions of task scheduling can be found in the literature, here it is defined as the allocation of a number of tasks – single actions that must be performed to complete a specific process–, to a set of resources, at specific moments in time. Examples of task scheduling can be found in many settings, as for example, the order in which the different parts of a car have to be manufactured in a set of machines, the allocation of operating rooms and surgeons to the surgical interventions in a hospital, or the order in which the customers of a restaurant should be served. Clearly, task scheduling is a core activity of many companies, both in manufacturing and in services, as it is essential for the coordination of the work between the different involved actors, such as departments, resources (human and physical) or external entities.

In most settings, task scheduling involves treating large amounts of data related to the process and properly handling the set of constraints controlling this process. As a consequence, task scheduling is usually carried out with the help of computer tools that offer some type of support to the decision maker. In this regard, the rising of Information Technologies (ITs) in the last decades has helped enormously to develop computer systems providing support for decision making – i.e. Decision Support Systems (DSSs)– for many decisions, including task scheduling. At the same time, there has been a notable increase in computer capacity that has made possible facing task scheduling problems that were considered unsolvable some years ago.

Despite these advances, an important gap between theory and practice has been found when translating these new conditions into practice, as it can be proven by the relatively short number of documented

systems that have been correctly implemented and accepted by users. The working hypothesis in this Thesis is that, in order to reduce this gap between theory and practice, these tools should consider a number of aspects that have been studied in the literature but that have not been taken into account in practice during the implementation process, such as the role of the decision makers in these tools, the organisational context where scheduling decisions take place, or the consideration of scheduling as a dynamic process: typically, every time an organisation requires to implement of a DSS for task scheduling – DSSTS in the following–, it faces two different options: either acquiring an *off-the-shelf* solution, or designing and developing an *in-house* tool. If the former option is chosen, the acquired solution may not fit perfectly into the activities of the organisation, and, since task scheduling is *company-specific*, this approach may result in a situation widely documented in the literature where there exist limited implementations that needs information systems working in parallel to deal with the specificities of the organisation. On the contrary, the second option usually derives in large implementation times with poor results, as the development team may not take into account errors or successes from former implementations, such as the functionalities that the system should include or the profiles required for the decision makers among others. As a summary, the design and implementation of DSSTS suffer a number of problems which constitute a root cause for the existing gap between the scheduling theory and its implementation into practice.

In order to improve the activity of designing and developing DSSTS, a common **framework**, i.e. a common set of functionalities and rules for the development of a task scheduling system, would be extremely helpful to provide developers with guidelines to fulfill the requirements stressed above. Such framework should be as general as possible to cover a wide range of applications of task scheduling –in the manner as the *off-the-shelf* software– and, at the same time, allow for a high degree of customisation –as in an *in-house* system–. Although some work on architectures for designing manufacturing scheduling tools can be found in literature, to the best of our knowledge, a common framework to help the design and development of this type of DSSs does not exist. This is the aim of this Thesis.

1.1 Objectives the Thesis

As discussed in the previous section, the goal of this thesis is to **propose a common framework for the development of DSSTS. In order to ensure the validity and range of application of this framework, its feasibility is analysed within two specific fields of applications, and two implementation case studies are conducted within these fields**. A by-side product of the implementations is the development of several state-of-art methods and algorithms to conduct a successful implementation of the DSSTS in the case studies.

Characteristic	Manufacturing	Operating Rooms
Frequency	1 day - 1week	1 week - 2 weeks
Average Task Duration	Short (minutes) - Long (days)	Medium (hours)
Production Quantity	Unit - Lot	Unit
Criticality of Due Dates	Medium	High
Last Minute Changes / Cancellations	Low	High
Uncertainty	Medium	High
Human Intervention	Low / Medium	High
Expertise of Schedulers	Medium	Low
Common Use	Manual - Basic Calendar Applications	Manual

Table 1.1: Main characteristics of manufacturing and operating room scheduling

Regarding the specific sectors where the validation and evaluation of the framework will be carried out, in this thesis we focus on manufacturing and healthcare. Manufacturing can be seen as one of the most important causes of economic growth and it is nowadays characterized by a fierce competition among companies trying to satisfy a changing demand of an increasing number of highly customized products. Under these circumstances, it is clear that the ability of a company to efficiently perform scheduling decisions has a great impact on its capacity to respond to customers in a fast and reliable manner. Within the healthcare sector, we focus on operating room scheduling, which is a key decision in hospitals since there is an increasing social and economic pressure to provide their services with maximum quality and minimum costs. Finally, an additional reason to pick up these two sectors is the fact that a relevant number of differences exist among them, which makes this choice more suitable to test the degree of generalisation of the proposed framework. In Table 1.1 we briefly show some differential characteristics.

A set of objectives related to the framework and to its validation on each sector of application has been devised in order to achieve the general goal mentioned above. These are the following:

- **O1**. To propose a framework for the design and development of DSSTS. This goal will be addressed in Chapter 3 in Part II of the Thesis.
- **O2**. To analyse existing implementations of DSSTS in order to check the alignment of the framework proposed with the task scheduling systems implemented in the two sectors chosen for the evaluation of the framework. To do so, a systematic review of existing DSSTS within manufacturing scheduling is carried out in Chapter 4, and in Chapter 5 for operating room scheduling. This goal will be addressed in Part III of the Thesis.
- **O3**. To conduct the design and implementation of two DSSTS according to the proposed framework in order to demonstrate its applicability. This is carried out in Chapter 6 for the manufacturing sector and in Chapter 7 for the healthcare sector. This goal will be addressed in Part IV of the Thesis.

1.2 Outline of the Thesis

This Thesis has been structured in five parts:

1. PART I. It is composed of two chapters. In Chapter 1, the motivation of this thesis is presented together with its main objectives and the outline of the document. In Chapter 2, an analysis of the problems when transfering task scheduling techniques and techniques to practice is carried out. The finding would serve to provide a number of guidelines for the framework that will be proposed.

2. PART II. It is composed of Chapter 3, where the common framework for a DSSTS is proposed. This framework is described by means of different perspectives, including its functions, modules, user roles, IT paradigms and deployment architectures.

3. PART III. This part is composed of two chapters. In Chapter 4 we present a systematic review of DSSTS in manufacturing. In Chapter 5 we show a review of DSSTS for operating room scheduling. These two reviews serve to check the alignment of the proposed framework with the existing DSSTS in these two fields.

4. PART IV. This part is divided in two chapters. In Chapter 6, we present the implementation of a DSSTS following the poposed framework for a real manufacturing case. In Chapter 7 we present the implementation of a DSSTS following the poposed framework for a real healthcare case. The implementation of these two DSSTS in real contexts serves to validate the proposed framework.

5. PART V. Finally, in this part (Chapter 8) we summarize the main results and conclusions of the Thesis and highlight future research lines that remain open for further work.

Chapter 2

Task Scheduling: Introduction and Main Implementation Problems

Since the main objective of this Thesis is to propose a framework for DSSTS, we first analyse the existing literature on the nature of task scheduling. After this analysis, we argue that the failures behind the implementations of some DSSTS can be classified into four causes. The explicit consideration of these causes will be instrumental to develop the framework to be proposed in Chapter 3.

There are different definitions of task scheduling that can be found in literature. In the Cambridge Dictionary scheduling is defined as *"A list of planned activities or things to be done showing the times or dates when they are intended to happen or be done"*. We also find similar definitions in Herrmann (2006) where it is defined as "the actual assignement of starting and/or completion dates to operations or group of operations to show when these must be done if the manufacturing order is to be completed on time", or in McCarthy and Liu (1993) or Wiers (1997), who introduce the concept of resources by defining scheduling as the allocation of resources over time to perform tasks.

This classical definition of task scheduling, although widely accepted, is being subject to discussion by several authors. The roots of this discussion lie in some issues that have been found when trying to translate the classical concepts and approaches for task scheduling (i.e. when conducting the implementation of a DSSTS) from theory to practice. These issues can be mainly classified in four causes:

- Task scheduling models are oversimplistic.
- Task scheduling is assumed to be static.
- The organizational context of task scheduling is neglected.

- The role of the decision maker (*scheduler*) is not taken into account.

We analysed these four issues and are discussed in the next subsections.

2.1 Oversimplification in the Task Scheduling Models

For decades, authors from the Operations Research community have focused on obtaining the best possible solutions for different scheduling problems, and different approaches have been traditionally used to solve them (Framinan et al., 2014). These approaches have been found to be extremely complex when problems from practice are tackled (Ruiz et al., 2008), so in order to make them feasible, most authors make a set of assumptions that, in many cases, are questionable from a realistic point of view. Typical assumptions in taskl scheduling models are (McCarthy and Liu, 1993):

- Complete availability of resources. Resources are typically considered to be always available and never stop working.
- Limitations in processing of jobs. A resource can process at most one task at a time and a task can be processed only by one resource at a time.
- No release time. Tasks are considered to be available when processing starts.
- No preemption allowed. Once a task starts, it cannot stop processing until it is completed.
- Independent setup times. Setup times are the same no matter the processing order and they are typically considered together with processing times.
- Every constraint is deterministic. They are supposed to be known in advance and do not ever change.

Even with these assumptions in a relatively simple environment, as it is for instance the case of the single machine problem in manufacturing, i.e. we have a number of jobs to be scheduled in one machine, the problem quickly becomes NP-hard (e.g. Garey and Johnson, 1979 for the objective of makespan minimization in a flowshop or Lawler, 1983 for the objective of minimizing the number of late jobs in a single machine). In order to be able to deal with complexity, approximate methods such as heuristics (see e.g. NEH heuristic Nawaz et al., 1983) or metaheuristics (see e.g. Armentano and Ronconi, 1999) have become the most preferable approach within this field. This situation, coupled with the oversimplistic assumptions already mentioned, may cause that the solutions to the models provided by these approaches may not be translated into practice in an straightforward manner, as on the one hand, a different problem

is being solved in the models and on the other hand, the solution to these models may not even be of a high quality.

2.2 Task Scheduling is not Static

Many authors focus on obtaining better results for the mathematical combinatorial problem mentioned in Section 2.1, i.e. they assume the scheduling problem as just solving this mathematical problem, but omit the rest of the scheduling process, such as gathering all the information required, updating schedules or controlling their execution, to name a few. This approach implicitly considers scheduling as a *static* problem – where the characteristics do not change over time – rather than the dynamic problem that is tackled in real situations (MacCarthy et al., 2001). McPherson and White Jr. (2006) emphasizes the need for considering this dynamic view of scheduling as they assume that a number of reactions are initiated every time a change in the environment appears, i.e. one reaction upstream related to goal achievement and one downstream related to planning. For example, if a new rush order enters the system and the main objective is meeting due dates, the upstream reaction would be related to this objective, e.g. if it is possible to negotiate due dates with other customers, or what is the penalty for not satisfying the due dates, and the downstream reaction would be associated to how the available resources can be used to continue fulfilling the due dates. Therefore, we can conclude that scheduling is much more than a simple combinatorial mathematical problem (Romero-Silva et al., 2015).

In order to consider task scheduling as a dynamic problem, many authors describe the possibility of rescheduling tasks whenever an event appears (for a detailed review see Vieira et al., 2003), i.e. the best possible schedule is proposed at the beginning of the planning period and its execution is monitored, thus obtaining a new schedule each time an event occurs. Indeed, there are authors as Li et al. (2000) in manufacturing or Stuart and Kozan (2012) in healthcare, that assume that rescheduling is more important than scheduling, so they propose obtaining an initial schedule using a simple and quick technique, such as a dispatching rule, and then use more complex approaches to continuously reschedule the plan everytime an event arises. On the other hand, other authors (Herrmann, 2006) state that rescheduling itself presents new problems as it can involve new costs and wastes in the planning horizon if enough degrees of freedom are not offered, i.e. the best – optimal – solution we can obtain every time a reschedule is executed is a local optimal solution for the new working conditions.

From the discussion above, it becomes clear that, in order to properly address task scheduling, the complete process of scheduling as carried out in practice has to be analysed. In order to do this, we follow the description in McKay and Buzacott (2000) and McKay and Wiers (2003), where the scheduling pro-

cess is dividided into seven different steps. Although these authors describe the process for manufacturing, it can be easily translated to any other context. These seven steps are:

- Step 1. Situation assesment. At the beginning of the scheduling process it is important to check the current situation, i.e. how much the situation differs from what it was expected and why it differs. It is necessary to check the number of tasks that took place, if they agreed with the plan and in the case they did not, check what was the reason for that, e.g. cancellations, delays, problems with availabilities of materials and so on.

- Step 2. Crisis identification. The second step is the identification of *hot tasks*, i.e. those tasks that have been delayed more than they should, due to problems with the availability of materials, problems with resources, etc.

- Step 3. Immediate resequencing and task reallocation. According to available resources and taking into account the current list of tasks to do, in this step the decision maker must try to schedule *hot tasks* found in Step 2 as soon as possible. In order to achieve this, it is possible to carry out actions such as delaying other non urgent tasks, planning overtime in some resources, etc.

- Step 4. Complete scenario update. The next step will be to restablish the whole scenario, i.e. update the status of every task that should be updated, after Step 3 has been executed. This will be the initial point to introduce new tasks from the list of tasks to do into the schedule.

- Step 5. Future problem identification. In this step, the decision maker looks at the whole planning horizon and to all tasks in the list of task to do. Within this list of tasks, he/she has to identify those that could present complications, e.g. their due date is close, or that have special characteristics, such as they require special resources or materials, etc..

- Step 6. Constraint relaxation and future problem resolution. Considering those tasks detected in Step 5, the decision maker has to propose a new schedule including them and at the same time he/she has to try to identify possible risks and try to obtain a schedule that minimize or avoid the impacts of their inclusion.

- Step 7. Scheduling by rote. The last step is the easiest one as it consists on mechanically including the rest of tasks, i.e. *normal tasks*, into the schedule. This step will follow some objective/s that will depend on the specific organization where it takes place.

Several authors (McKay et al., 1995; Jackson et al., 2004; Snoo et al., 2011) propose a different perspective to analyse the task scheduling process. Instead of analysing the time dimension, i.e. the set of

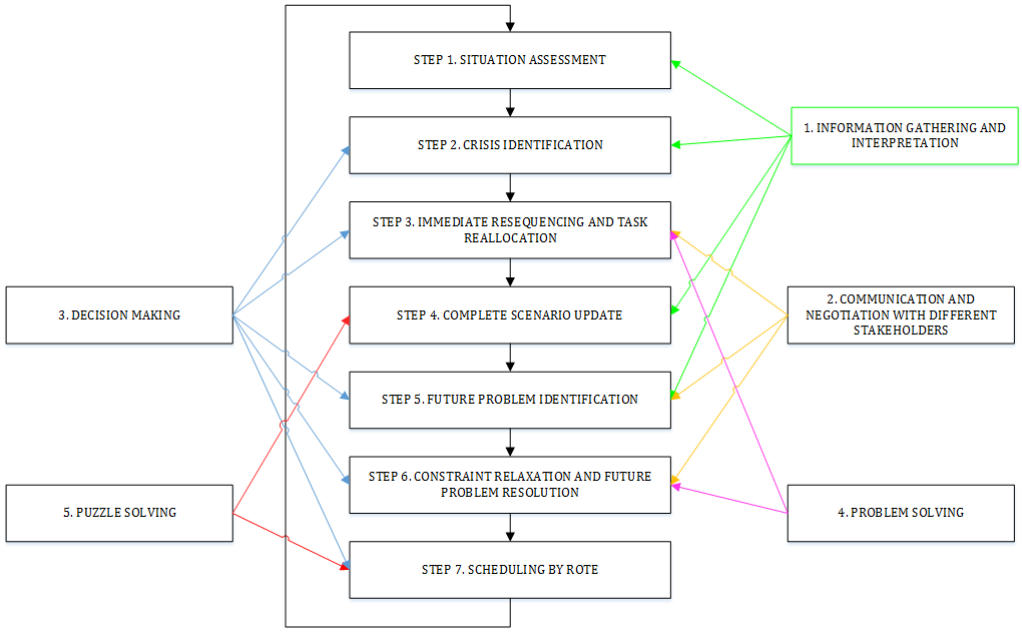

Figure 2.1: Relationship between scheduling process and its main activities. Based on McKay et al. (1995), McKay and Buzacott (2000), McKay and Wiers (2003), Jackson et al. (2004) and Snoo et al. (2011)

stages that must be accomplished as described above, they propose a set of activities that have to be performed. In order to bring together these two perspectives of the task scheduling process, in Figure 2.1 we show the relation between these two perspectives. Next we detail these activities and their relation with the steps.

- Information gathering and interpretation. Although information is necessary in every step, gathering information and interpretation is specially present in Step 1, where the decision maker needs all information related to scheduling, from the list of tasks to the resources where they are performed, including people that performed the tasks, times, possible incidences, and so on. This activity is also important in Step 2 where a big amount of information is required to obtain a whole view of the system and to be able to anticipate problems. Additionally, to establish the whole scenario picture after making changes (Step 4) again a lot of information is required. Finally, and similarly to Step 5, information is crucial to anticipate problems.

- Communication and negotiation with different stakeholders. This activity is highly linked to the organisational context of scheduling that will be discussed in the next section. A good communication and negotiation between the different departments, units, ... is specially important as if, for example, a change must be done in the scheduling, it is important to know if people involved in carrying out an specific task is able to do it in the new scenario. There are three steps specially concerned with communication and negotiation, namely step 3, 5 and 6. In all of them, the decision

maker must be sure that the changes he/she proposes can be correctly executed.

- Decision making. The next activity is decision making. It may seem similar to problem solving, but we assume that decision making does not need any complex solving, it is only related with selecting among a number of possibilities. In this sense, decision making is present in almost every step, except from steps 1 and 4, where a collection of data is to be gathered without any decision involved.

- Problem solving. Regarding problem solving, we assume that some more complex decision must be done, i.e. steps requiring this activity are usually accomplished by using some kind of software, such as a solver for MILP models or simply an Excel spreadsheet to help in obtaining a solution. This activity is related to decision making, as the aim of using such software is to obtain aids for selecting a solution. Problem solving is considered in steps 3 and 6.

- Puzzle solving. This last activity can be seen as a mechanical activity, in the sense that once the decision of how different situations are going to be managed it just follows these rules. We see how it is used only in the last step, where once a rule has been selected to schedule tasks (decision making), they have to be located where the rule says they have to be located.

As it can be seen, the mathematical/model solving steps are a (relatively minor) part of the scheduling process, therefore a DSSTS should encompass the rest of activities and steps analysed in this section.

2.3 The Organizational Context of Task Scheduling is Neglected

A number of authors (Stoop and Wiers, 1996, Wiers, 1997, McKay and Buzacott, 2000 or Berglund and Karltun, 2007) highlight that, when addressing the task scheduling problem, it is necessary to include the organisational context where it takes place. To analyse this context, it is important to locate scheduling properly into the management structure of the organization. Throughout this Thesis we assume the classical hierarchical approach for decision making as seen in Figure 2.2. The idea is that decisions in upper levels serve as constraints to lower levels, so there must exist a synchronisation between decision levels (Vogel et al., 2016). In this figure we see the adaptation of this approach to two different environments, healthcare (operating theatre) and manufacturing, that will be analysed in more detail in Parts III and IV.

It is also possible to find other decompositions of managerial levels such as the one found in Shobrys and White (2000), where authors distinguish between forecasting, strategic planning, tactical planning, scheduling and control, but usually, these decompositions can be easily translated to the generic approach shown before. For example in the decomposition by Shobrys and White (2000) forecasting and strategic plannig

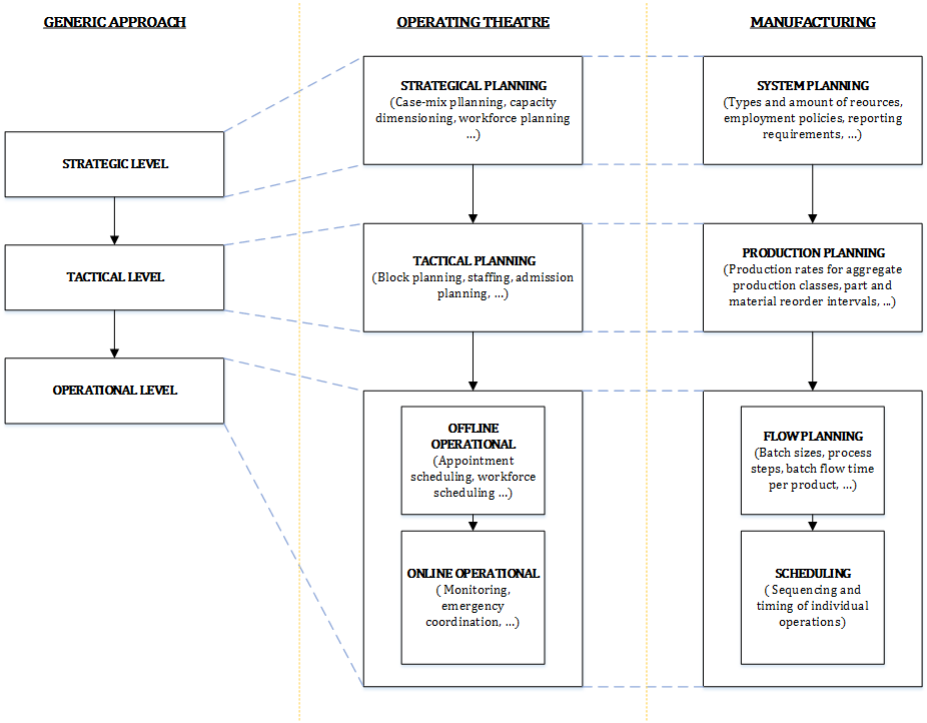

Figure 2.2: Hierarchical Approach for decision making in different environments: Generic, Operating Theatre (Hans et al., 2011) and Manufacturing (McPherson and White Jr., 2006).

correspond to the strategical level, tactical planning corresponds to the tactical level, and scheduling and control are within the operational level.

Although the management structure in Figure 2.2 is commonly assumed as a good approach to deal with managerial decisions, its implementation has also received critics from a number of authors (Herrmann, 2006), as it has been proven to fail when taken into practice. However, these failures are typically due to the fact that its appropriateness is not properly studied before its implementation. The organisational structures of nowadays organizations are quite rigid and inflexible, so as it is required to briefly adapt these structures to make them fit into this paradigm, it becomes hard to apply, resulting in poor versions that usually work even worse than the monolithic scheme.

Another important issue that should be highlighted within this subsection is that, although we have seen that this classical approach can be translated to many different situations it is also necessary to take into account that scheduling is highly context dependent (MacCarthy et al., 2001), i.e. it highly depends on the specifities of the sector, and on the concrete organization where it takes place.

2.4 The Role of the Decision Maker is not Taken into Account.

Scheduling theory has accepted traditionally a set of assumptions for every problem, such as data is defined and known in advance, it does not change through time or it is determinisitic. These assumptions model a problem that can be solved automatically by a computer without any human intervention, which has been proven unrealistic in cases, as it is widely documented that task scheduling is carried out with human intervention (McKay and Wiers, 1999). To analyse this problem, we first analyse whether a DSS can replace human decision making or, in contrast, it should only provide tools to help (human) decision makers. According to Herrmann (2006), automation is only possible in well-defined (structured) problems, while in the cases where ill-defined (unstructured or semi-structured) problems are present this is not possible. As a consequence, the most common problems that must be addresed by humans are characterised by:

- **Missing information**. In the case where information is not available, where some information is missing or even in those cases where information is known to be inadequate, human decision makers can use their implicit knowledge to fill the gaps or correct the information.
- **Dynamic knowledge required**. Human decision makers are also necessary to define the degree of importance of constraints in those cases where they can be violated, also known as soft constraints. Something similar happens with defining the importance of different objectives.
- **Level of uncertainty**. The higher the level of uncertainty in the problem, the more difficult is to use a DSS, and the more often rescheduling will be necessary. Nevertheless, there exist different techniques to deal with uncertainty, although the best possible approach is to try to complement human tasks by providing more functionalities, such as the possibility of analysing scenarios considering changes in the variables where uncertainty is more important, or offering robust alternatives that minimize the impact of uncertainty.
- **Transparency**. Related to the previous point, we have to take into account the need for transparency that decision makers usually exhibit. The more uncertainty presents the problem, the more necessary is transparency. Decision makers requires to be in direct control of what happens when a disruption appear.
- **Complexity**. The degree of complexity of the system can also help us in deciding how much support a DSS can offer. For those cases where the problem is structured, if the complexity is high, the use of a DSS becomes advisable as, although implicit knowledge is not required, it is difficult for

humans to deal with too many constraints. In contrast, computers can deal with complex structured problems.

- **Additional activities related to scheduling**. As discussed in Section 2.2, scheduling is often considered as simply a mathematical problem to solve. This part of scheduling is related to what Jackson et al. (2004) call the decisional role of the scheduler. In addition, there are two other roles commonly neglected in literature and are hardly driven by a DSS, namely the informational role and the interpersonal role. The former is related with the fact that the scheduler is in charge of centralizing the communication between the different actors of the process, and the second is related with the personal relations of the scheduler with the other persons involved in the process.

Once we have studied the type of problems that decision makers have to deal with, we analyse the process of human decision making. In order to do that, we show in Figure 2.3, the adaptation of the model of human decision behavior of Reason (2003), made in McKay and Wiers (1999) and Herrmann (2006). This model is a modification of the *Decision Ladder Model* by Rasmusen et al. (1994). A further discussion on this model and other cognitive models can be found in Fransoo et al. (2010). In this model, different levels of attention and routine in human reasoning, are assumed. The more often a task is repeated, the more routinely it is and the less attention it needs. According to human information processing (Herrmann, 2006), different levels can be distinguished as it is shown in the figure:

- Skill-based Level. Tasks are done almost automatically, i.e. the person is so accustomed to carry out the task that almost no reasoning is needed to perform it. The performance of the activities is supervised regularly to check if they are correctly executed. In the case that a problem is recognized, the control goes to the rule-based level.

- Rule-based Level. Two options are possible within this level, the case where some pattern can be recognized in the problem and the case where this is not possible. In the former, some if-then rules are checked and the set that is satisfied is applied. In the case where no pattern is recognized, the control is passed to knowledge-based level.

- Knowledge-based Level. In this level no formal solving methods are available, so the problem has to be identified, analysed and solved by trying to combine novel and existing knowledge.

This solving process is iterative, i.e. different alternatives are analysed when trying to solve the problem and finally the best one is selected. The knowledge that can be extracted from this process can serve to add new if-then rules to the rule base utilized in the *Rule-based level*.

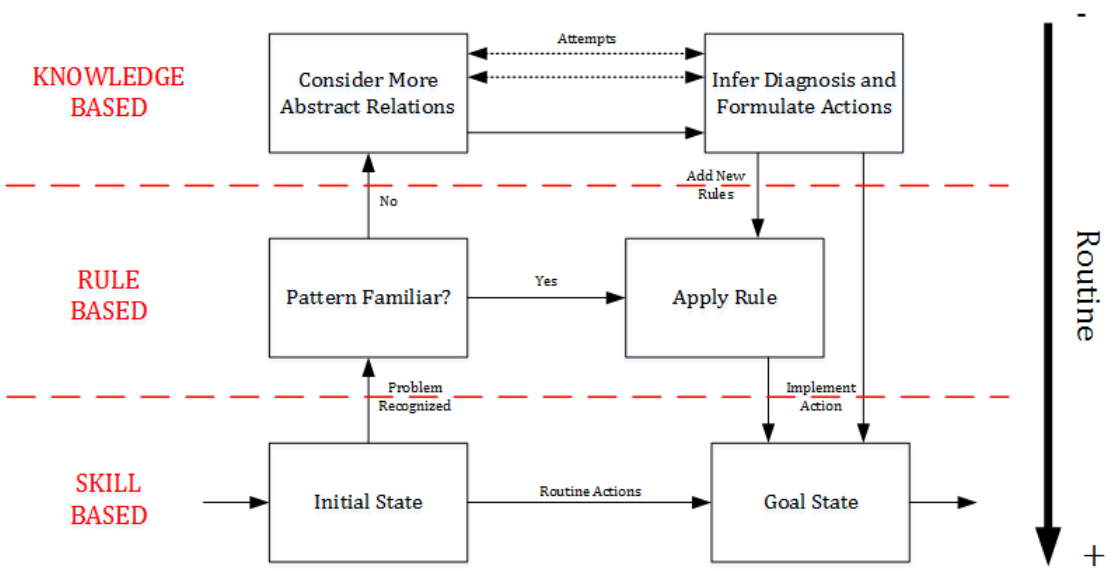

Figure 2.3: Model of human decision behavior by Herrmann (2006)

There exist a clear difference in this process depending on the expertise of the decision maker. As stated in Guerin et al. (2012), experts' ability of anticipation is greater than novices', and they are able to consider a more abstract representation of the context, i.e. novices are usually guided only by simple features of the problem while experts can make a deeper analysis of it. This difference has to be taken into account when designing a DSS (Herrmann, 2006) for task scheduling, as it has to be useful both for novices and experts. An example of this consideration, is the flexibility that was highlighted in the previous section. Due to the ability of anticipation of experts, they will be able to modify manually schedules to prevent them from further problems Therefore, the possibility of manually modifying the schedules must be present in the system. On the contrary, a good engine that provides good alternative schedules is required for more novice schedulers. It would also be a good practice to capture the knowledge applied by experts in these situations. This is in line with the affirmation (McKay et al., 1995) that schedulers usually spend around 80%-90% of their time on identifying constraints, i.e. gathering information. Therefore, if the information they gather every time they make a schedule is captured, it could serve to improve the performance of the scheduling activity in the future.

2.5 Conclusions

In this chapter we made a brief introduction on task scheduling and we introduced the main problems found when implementing DSSTS into practice. Each of these problems has different causes and different

remedies that will be applied to the framework for a DSSTS proposed in the next chapter. Among them we have:

- Different problems are found referring to how task scheduling models are used in practice. In the literature there is a huge amount of contributions related to task scheduling but most of them has resulted in failures each time they have been tried to be implement in practice, due to the difficulties in modelling practice and to the required balance between detailed models and computational effort in solving them. To deal with this problem of task scheduling, the framework proposed must offer, in a properly structured way, a set of possible models and solving approaches in order to face different situations. It is also important to give the scheduler the necessary tools to decide until which extent he/she is more concerned about the quality of the solution or about the speed in solving the problem. Additionally, in order to cope with those constraints that are not or cannot be considered when solving the scheduling, it is crucial to provide the scheduler a way of manually manipulating the obtained solutions.

- Traditionally, when dealing with task scheduling the literature has focused on just solving a mathematical combinatorial problem, without considering the rest of the task scheduling process. In this section, we have detailed the main stages and activities that must be made when performing task scheduling. To overcome this problem, the framework to be proposed has to consider these stages and give support to the scheduler on addresing them.

- Apart from considering more activities, it is important to consider more actors in the process to properly address task scheduling. The framework to be proposed will have to take this into account and guarantee the correct synchronisation between the different decision levels, i.e. it must consider the relation between planning, scheduling and control. Moreover, taking into account that scheduling is context-dependent, it is also required that the framework is flexible enough to fit different environments.

- Scheduling theory has not traditionally considered the human scheduler as relevant. We have discussed which are the main problems that cannot be solved by a DSS and have to be solved by the human scheduler. Moreover, the decision making problem has been analysed and also the main roles that schedulers use. In order to deal with this problem, our framework needs to consider the human scheduler and offer him/her with a number of functionalities to support those activities that cannot be performed automatically.

From the previous conclusions a number of guidelines for the framework we will propose in the next chapter can be extracted. More specifically:

1. The framework must provide a number of different decision models and solutions algorithms that allow the scheduler to deal with a number of different situations.
2. The framework must allow the scheduler to manually modify the resulting schedules in order to include constraints that were not include in the decision models.
3. The framework must consider the different stages in the scheduling process and support them, e.g. offering the possibility of gathering data from different information systems.
4. The framework must provide means to consider planning (higher level) and control (lower level) together with scheduling.
5. The framework must be adaptable to different environments. To do this a flexible implementation of the framework is required.
6. The framework must support those activities that have been found to be necessarily implemented by humans in the scheduling process, i.e. cope with missing information, consider uncertainty, ...

Part II

FRAMEWORK PROPOSAL

Chapter 3

A Framework for Decision Support Systems for Task Scheduling

In this chapter we propose a generic framework of a DSSTS. To describe it we follow the proposal by Zachman (1987) where the author present a framework for information systems architecture and discuss the required perspectives that a framework must include: data, function and network. Additionally, we consider the work by Sowa and Zachman (1992) where authors extend the previous framework by adding three new perspectives: organization, schedule and strategy. In accordance with these works, the framework for a DSSTS is described through the following perspectives:

- Model-View-Controller (MVC). We use this paradigm to give a general description of the framework.
- Roles perspective. We detail which are the roles of the users of the resulting DSSTS and define which are their main functions.
- Functional Description. We describe the functional features of the framework by detailing the three modules that are commonly used in the literature to describe a DSS: database management module, model management module and user dialogue management module.
- Deployment Perspective. We detail different computer architectures that can be used to deploy the resulting DSSTS.

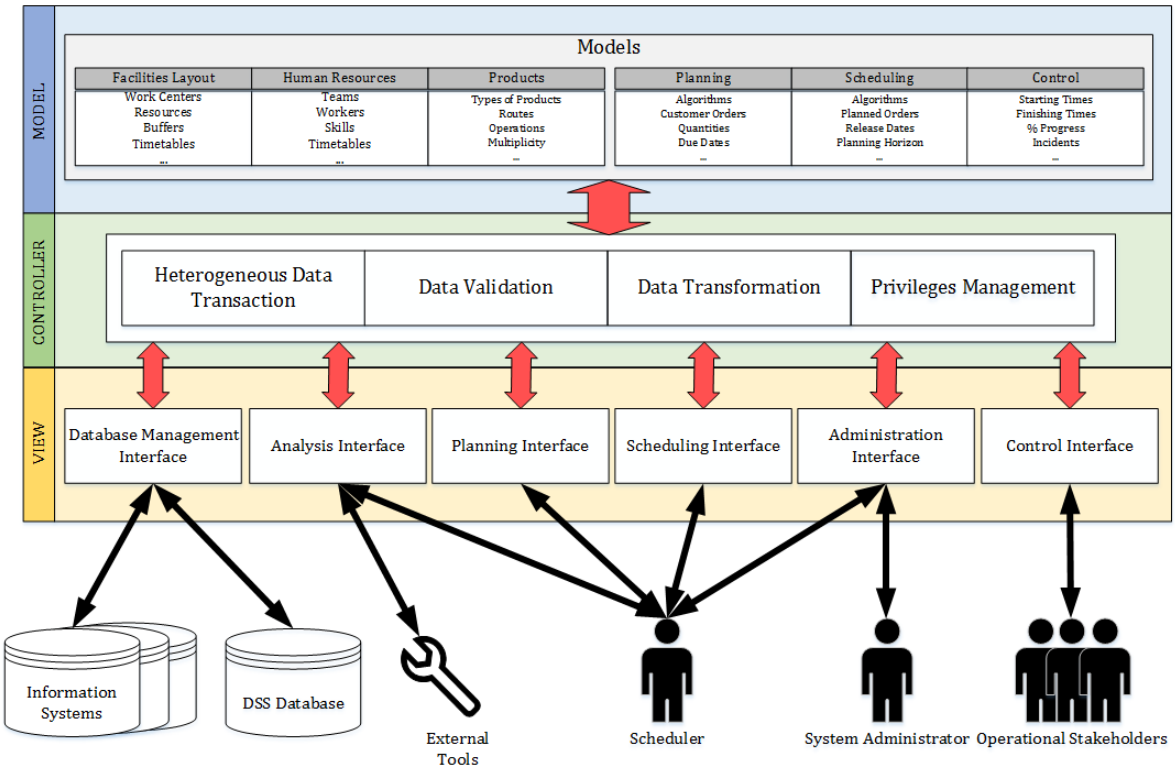

Figure 3.1: Generic MVC framework of a DSS for task scheduling

3.1 Model-View-Controller Description

To show a general description of the framework we describe it using the MVC paradigm. In this perspective we show the main components of the framework and how the different roles of the system interacts with them. The main properties of the MVC paradigm are flexibility and modularity (Krasner and Pope, 1988), in the sense that any of the modules can be modified, totally or partially, without affecting the rest. The idea is to separate the DSSTS into a component holding the core functionality and the data, a component in charge of the input/output interaction and another component in charge of handling communication between the previous two (Sauer and G., 1999). A brief description of these three parts is given below (Krasner and Pope, 1988,Avgeriou and Zdun, 2005,Shams and Zamanifar, 2014) and a general schema is shown in Figure 3.1.

- **Model**. This component encapsulates context-application data and the logic to manipulate it, i.e. this component concerns the conceptual definition of the database and the different techniques and methods used to address the different decision models tackled by the DSSTS. Thus, the model is responsible of the storage and management of all data. Therefore, it is supported by a Database Management System(DBMS). This component is specially context-dependent as it maps the model

data of the scheduling process and the organizational structure where it takes place. Because of this and in order to maintain the reusability of the system, it is mandatory to maintain this component absolutely independent from interfaces.

Within this component we see two main components:

- Data Model. This component contains all the business logic. It contains all the information about the context where the DSSTS is to be implemented and about the scheduling process, i.e. between different DSSTS implementations this component changes substantially.
- Decision Models. This component is highly interconnected with the previous one and contains all the algorithms and procedures that the DSSTS uses.

- **View**. This component is in charge of presenting data to users. It requests data to the *Model* component and displays them to the users. Each of these views is associated exactly to one model, but models can be associated to any number of views, e.g. certain application can display data in multiples forms, but how an specific information is shown is strictly associated to that application. This component, although it can also be considered as context-dependent, is more suitable for standardisation with small modifications depending on the specific case where the DSSTS is to be applied. It is composed of the following components:

  - Database Management Interface. This component handles all the communication between the DSSTS and the different databases and information systems in the organization. It must be as transparent to users as possible.
  - Analysis Interface. This interface supports calculations on the data of the DSSTS. Apart from offering this functionality to the scheduler, it serves as a gateway to external tools, e.g. if a statistical analysis is required from an external tool, this component serves to normalize the received data.
  - Planning Interface. This is the interface intended to manage planning. As commented in the conclusions of the previous chapter, the framework must allow the scheduler to deal with the different managerial levels, so this interface must offer the possibility of performing planning or a form of incorporating planning from an external source.
  - Scheduling Interface. This interface offers the scheduler the functionalities associated to scheduling.
  - Control Interface. In a similar manner to the *Planning Interface* component, the framework

must provide the functionality of control, or a form of exporting the scheduling in order to make a external connection to lower levels.

- Administration Interface. This last component serves to configure and maintain different facets of the DSSTS, including these *system oriented* such as user management, connections to database and so on, and *functionality oriented*, as the upload of new decision models and solving procedures, data model update, etc.

- **Controller**. The request of data made by the views are managed by controllers, i.e. a controller receives user input through a view, and translates it into a request for the model. They act as an interface between views and models. Their mission is also to coordinate the interactions with other view-controller pairs. These pairs, can be conceived as user interfaces. Controllers depend more on views and models than on the specific context where the system is used, so we can assume that this component can be also standardised.

  This component is composed of the following components:

  - Heterogeneous Data Transaction. The main objective of this module is to normalize data and to allow the communication between the different components of the DSSTS in a seamless way.
  - Data Validation. This component is in charge of maintaining data intergity in the DSSTS, i.e. the data is in the correct format and there is no missing data. Every data going into the DSSTS must be checked by this component before accessing the Model.
  - Data Transformation. This component transforms data each time it is required, e.g. if an external tool requires a specific input format or if a specific format is required to feed the scheduling interface.
  - Permission Management. This component is related with user management. Not every stakeholder of the system have access to all functionalities, so this component must ensure that actors access only to these functionalities for which they are authorized.

3.1.1 Roles of the System

As shown in Figure 3.1, there are a number of roles interacting with the DSSTS. In this subsection we detail the roles perspective of the framework where the organisational context of scheduling is explicitly taken into account, i.e. the different actors involved in scheduling are described. In addition, this perspective considers briefly the process of scheduling, as the flow of the scheduling process can be drawn from the actors involved in it.

- **System Administrator.** This role is in charge of the correct overall performance of the system. The person/s with this role are supposed to have knowledge in both, informatics and the field where task scheduling take place. Between their main functions we find:

  - Users Management. He is responsible of assigning the correct access permission to the different users according to their roles and their position in the organization.
  - System Configuration. This role has to deal with all aspects related to the configuration of the system, e.g. time unit considered in scheduling, feed the system with calendars containing festivities, ...
  - Decision Models Maintenance. The maintenance of the Decision Models is also part of the duties of this role. This role has to update models when necessary or include new ones, following the guidances given by the scheduler, in order to avoid the system to become obsolete.
  - System Maintenance. The system administrator also looks for maintaining the correct technical performance of the system, so any technical problem that could appear during the use of the system is also his/her responsability.

- **Scheduler.** As discussed in Section 2.3 when we went into detailing the organizational context of scheduling, planning and scheduling can be considered as a continuum of activities, so it is very difficult for a system to consider them separately. This discussion is also made by Framinan et al. (2014). In that work, authors argue that in some cases, planning is not required as they do not have enough scope to carry it out, e.g. the case of a small shop where there are not many customer orders. To take this possibility into account, in our framework proposal we include a part dedicated to planning, that can be ommited if necessary. Within these conditions this role can be splitted into two different roles when necessary, namely planner and scheduler role. The first one is in charge of carrying out the long term plan, i.e. this role define a non detailed long term plan where it is checked if there are enough available resources to carry out the tasks that have to be executed. This plan serves to make an estimation on when the tasks should be released, taking into consideration both, the available time of resources and other tasks that are already being executed. From the results of this plan, the role of the scheduler must generate the detailed short term schedule. These two roles make use of the algorithm's libraries, according to their necessities. Apart from the pure scheduling or planning activities, this role must offer the possibility to obtain/save data from/in the database.

- **Operational Stakeholders.** The main function of this role is to update the data related to the

ongoing work in the database. Every time a task is started or is completed, it must be registered. Additionally, it is also recommended to have a list of predefined events, normally defined by the System Administrator, that could be registered in the system, e.g. problems with resources, not availability of materials, etc. By doing so, the DSSTS is always precisely informed about how the schedule performs, allowing the scheduler to make the corresponding changes in the schedule if required. It is also important that users with this role report if there is any incident during the execution of the schedule.

- **Information Systems.** This role is assigned to the already existing systems in the organization when the DSSTS is to be deployed. It will be in charge of the inter-operation between the DSSTS and the rest of the systems. It is important within this role to define correctly the transformations and connections between the databases of the systems, in order to make them work together properly.

- **External Tools.** In some cases there exists the need of using external application for data analysis. This role will manage the gathering of data from the DSSTS and the storage of new data generated by these external tools. This role differs from the *Information Systems Role* in the sense that it is not required to maintain a complete mapping between systems to assure data completeness. This role is only used to carry out single activities where no further actions are required, as for example complex statistical analyses based on data taken from the DSSTS.

In Table 3.1 we show a summary with the roles of the system.

3.2 Functional Description

In this section we show a functional description of the proposed framework. In order to analyse the main functionalities of the DSSTS we illustrate in Figure 3.2 the UML component diagram of the framework. It is constructed upon the architectures by Pinedo (2012) and Framinan et al. (2014), but with a brief difference when defining the modules of the system. More specifically, we use the traditional decomposition of a DSS (Sprague, 1980), i.e. database management module, user dialogue management module and model management module, but including what Framinan et al. (2014) consider as *Business Logic Unit/Data Abstraction Management Module*, within the other three. The components shown in the diagram represents independent entities in charge of a number of different functionalities. Therefore, the modification or replacement of some of these components should not affect to the rest, which stresses the idea of modularity commented in Section 3.1.

System Administrator	
Profile:	Informatics and Operations Management knowledge
Main functions:	- Users Administration
	- Maintenance of Decision Models
	- System Configuration
Planner/Scheduler	
Profile:	Operations Management knowledge
Main functions:	- Long-term/Short-term planning
	- Use of the decision models
	- Interaction with the database
Operational Stakeholders	
Profile:	No specific knowledge required
Main functions:	- Update schedule information
	- Notify schedule incidents
Information Systems	
Profile:	Existing systems from the organization
Main functions:	- Interaction with the DSSTS database
External Tools	
Profile:	Any external tool required
Main functions:	- Information analysis
	- Statistical analysis
	- ...

Table 3.1: Roles of the DSSTS

In addition to addressing the guidelines described in 2.5, with the proposed framework we also face some issues of the architecture in Framinan et al. (2014) that have not been correctly addressed before, namely:

- Interoperability with the rest of the business information system.
- Interaction between the dialogue management, the database management and the model management modules.
- Inclusion of the algorithm library into the database.
- Necessity of including part of the model management module into the dialogue management module.
- Consideration of the whole scheduling process.
- Rescheduling functionalities.

In the following subsections we analyse each component of the framework.

3.2.1 Database Management Module of the DSSTS

This module is in charge of handling the data, i.e. consulting, updating and deleting functions. The components of this module are:

1. **DSSTS Database**. This component contains the data model of the DSSTS. It can works in standalone manner or in collaboration with other existing information systems using the *Integration with other IS Management* component. The data in this database can be classified in two groups, as stated by Pinedo (2012):

   - Static Data. It corresponds to all data that rarely change over time, i.e. data related to the layout – such as grouping of resources, resources, etc. –, to human resources – such as teams, workers, etc. – or to the product catalogue of the organization – such as type of products, routes, etc –.
   - Dynamic Data. These data will be changing constantly, which include start times or finishing times of tasks, assigments to specific resources, data related to control, etc.

   These two types of data must coexist in the same database. It is also possible to separate them into two different databases, what will have some pros and cons. For example, an advantage of separating this database into two is that it is easier to handle scenarios as, an scenario is composed only by dynamic data. However, an important drawback of this approach is that, the communication between databases would increase the complexity of the system and the time required to obtain the data. According to this last drawback and to obtain a smoother approach, it is advisable to have them together.

2. **Model Database**. As suggested in Framinan et al. (2014), we include the model database in this module to give more flexibility to the DSSTS. This approach is in line with the guidelines described in Section 2.5, where we discussed the necessity of offering different decision models to schedulers in order to have a flexible DSSTS that could be adapted to different contexts. Data relevant to the decision models, such as configuration parameters, preferred algorithms, etc. are stored in this database. As in the previous component, this component can be implemented as a standalone database or integrated with the other databases of the organization. Components in the *Model Management* module will access this database to obtain the required data regarding decision models to apply the solving approaches for planning, scheduling and rescheduling. This is in line with the requirement of managerial levels synchronisation discussed in the guidelines.

3. **Data Filtering**. This component is in charge of retrieving filtered information from the database. In order to face the decision problems, most components need data from different databases. These data are not usually raw data but require some processing that is done by this component, e.g. if information from certain worker is required, this component is in charge of filtering data from workers

and retrieving the data for the specific worker. We consider this component as associated to both databases. This module could also be considered as included into the *DSSTS Database* component, but we consider it separately in order to explicitly allow other components to connect directly to the database without requiring the filtering of data, as we will see in the last two components of this module.

4. **Data Analysis**. In some cases, apart from filtering data, some data analysis is required. For example, if the scheduler needs to know the number of tasks that has been executed during certain day, depending on how the database is implemented, some data filtering (performed by the previous component) is required and some calculations need to be done in order to obtain the final result. This calculations are done by the *Data Analysis* component, which also serves to support any external tool that needs some previous data analysis to perform its activity.

5. **Scenarios Management**. A basic functionality of a DSSTS is the possibility of carrying out *what-if analysis* (McKay and Wiers, 2003), i.e. simulate what would happen if the tasks must be executed under certain conditions. This functionality supports some aspects that needs to be taken into account by the scheduler as commented in the guidelines (see Section 2.5). It allows the scheduler to deal with uncertainty by modifying the conditions of the scheduling environment in order to be able to have a wider view on the possible situations that could appear if this conditions change through time. To consider this functionality correctly, we assume that it is essential to have an specific component dedicated to the management of *scenarios*, i.e. these specific conditions that we want to assume. We make this assumption as *scenarios* are in the core of the system, in the sense that, every plan that is finally approved to be executed starts as a tentative scenario. This component is in charge of managing the process of creating, simulating, discarding, validating, etc. the different scenarios that are used by the scheduler. We assume that this component needs complete access to the database, so that is why it is directly connected to it without the need of the *Data Filtering* component.

6. **Integration with other Information Systems**. This last component is intended to tackle the problem that usually appears when a new system is to be installed in an organization where other information systems already exist. If we do not take into account the fact of integration, some problems arise:

    - Waste of time in filling already existing data. Someone must introduce data, most of the times repeated, in every single system in the organization, what typically supposes a waste of time

and problems with the workers in charge of doing this activity.

- Integrity of data. Same data in different systems could be different, i.e. as every system is fed with data independently, each system could have different data because of human failures, coordination problems, etc. For example, if the production control system is different to the scheduling system, data from the control system are introduced on real time while the scheduling system is updated some times a day, resulting in the fact that if changes appear while the scheduling is being updated, there would be a lack of coordination between both systems.
- Disruption awareness. This problem is related with the previous one and with possible disruptions that could appear. If the update of a system is made some times a day and a disruption appear, the system is not aware of the problem until it is updated, so it would be very difficult to perform a correct monitoring of the execution process.

Because of these problems, we assume that our framework must incorporate a component dedicated to the interconnection of the database/s in our DSSTS with the rest of the systems in the organization. It is clear that, in the case where there are no more information systems in the organization, this component would be useless and could be omitted. Similar to the previous component, this component also needs complete access to the data in the database, so it should also include its own procedures for data storing and retrieving. Note that this component also follows the guidelines to build the framework as it supports the consideration of the organizational context and the decision process of scheduling by offering the possibility of considering data related to other actors or activities related to task scheduling. It also helps with the interpersonal role of the scheduler commented in Chapter 2.4, allowing the communication of the different actors in the organization through already existing information systems.

3.2.2 User Dialogue Management Module of the DSSTS

This module includes the interface between the DSSTS and the users. The interface to every funcionality of the system must be considered within this module. Below, we briefly detail every component that must be included.

1. **System Configuration**. This component is in charge of the general configuration of the system. The main funcionalities supported are:

   - User account management, i.e. all information related to accesibility of users to the different modules of the DSSTS

- Configuration of the execution of tasks, i.e. if workers are considered as resources to be assigned to tasks or not, if buffers should be allowed between stages in the production process, etc.
- Tieme configuration, i.e. the granularity of time when scheduling, the different working hours according to seasons, calendars containing holidays, etc.
- Information about products or raw materials, i.e. measuring units, maximum availability, etc
- Notation, i.e. to change the naming of the different aspects involved in scheduling to fit into the common jargon of the organization. To make the framework as general as possible, it is interesting to maintain the notation used in the organization as similar as possible to the notation they traditionallly use in their daily work, e.g. it is possible to find many different names for the work that must be performed in a production plant: purchase orders, customer orders, work orders, tasks, activities, etc. An interesting way of facing this topic is to include a dictionaryin the data model, mapping those terms commonly used in each context where the DSSTS is implemented to a set of generic definitions. This allows for a fast deployment of the DSSTS without requiring to customize it in every case.

2. **Analysis Tools**. This component includes the interface to all those functionalities that require an analysis of the data in the database. These functionalities usually look for improving the decision making capability of the scheduler by applying some mathematical analysis to data. This component is independent to the solution approaches managed by the *Model Management Module* (see Section 3.2.3).

3. **Reporting Tools**. This component includes interfaces to the two types of data present in the *DSSTS database* component, namely dynamic and static data. Functionalities offered by this component are commonly omitted in literature, but in practice, they give the largest support to decision makers, both for making the schedule and for communicating it to the different actors in the scheduling process. This component follows the guidelines commented in Section 2.5 regarding the additional activities that the scheduler must do, specifically the informational role discussed in Section 2.4.

4. **Decision Problems Handling**. To make the best possible use of the solution approaches included in the DSSTS, it is important to offer the scheduler easy-to-use and friendly interfaces, to configure and select the algorithms that would be applied to obtain a schedule. Depending on how the solution approaches are to be used, i.e. how the solving approach is selected: automatic, by user ..., these interfaces can vary. It is also interesting to note that this component include all those activities that

make use of solving approaches, such as planning, scheduling or rescheduling. Typically, planning interfaces will be independent to scheduling and rescheduling interfaces as the approaches differ. However, scheduling and rescheduling can share interfaces to ease the work of the scheduler, i.e. the simpler the better. This component must also consider the representation of the solutions obtained. This representation of solutions can differ depending on the specific context where the DSSTS is to be used. A detailed discussion on solution representations can be found in Framinan et al. (2014).

5. **Schedule Control**. In order to monitor how the scheduling is being performed, the DSSTS must provide interfaces to ease the update of incoming data. Interfaces in this component have especial characteristics that must be taken into account, such as:

   - Expertise of Operational Stakeholders. Users making use of these interfaces typically do not have any computer skills, so they should be as simple as possible.
   - Robust Devices. The facilities where these interfaces are used can vary importantly, unlike scheduling interfaces that are commonly used in an office. Therefore, registering data should be as quick and simple as possible.
   - Feedback. These interfaces should give feedback to users regarding the work they are doing, e.g. if a worker is registering the time required to perform an activity, it is interesting to let him/her know his/her required time in his previous register, the average duration of other workers, or what task he/she has to carry out next.
   - Register Events. Due to the number of possible problems that can arise during the execution of tasks, this component should include means to notify possible incidents to the scheduler.

   This component also follows the guidelines to build the framework from Section 2.4 in the sense that it helps the decision maker to consider the scheduling process and it offers the scheduler the possibility to communicate with other actors, i.e. to take into account the organisational context.

6. **Scenarios Handling**. This component is complementary to *Scenarios Management* component in the *Database Management Module*. Due to the importance of properly using *scenarios*, this component helps the scheduler in managing them. It is the basis for the *what-if analysis*, so it should offer the possibility of creating, modifying or restoring scenarios to the scheduler. It must be also possible to interact with various scenarios at the same time, i.e. try different conditions for the same planning horizon and compare the results. This component support the same issues of the guidelines as the *Scenarios Management* component did.

7. **Input/Output Interface**. It is important for decision makers to feel comfortable when working with the DSSTS. In order to achieve this, it must offer a common and homgeneous interface from which decision makers can access to all functionalities. The same is valid both, for entering and receiving data. The rest of the components must be integrated with this common interface, maintaining the same aspect, the same distribution of controls, etc. This component can be omitted if these principles are taken into account when developing the rest of the components in this module.

3.2.3 Model Management Module of the DSSTS

This module work in three different activities, namely planning, scheduling and rescheduling. In some manner, the *Model Database* component could be considered as being part of this module but, as commented in Section 3.2.1, we separate it to allow its integration with the *DSSTS database* component. This module interacts with the *Database Management Module* in order to automatically offer schedules, considering decision maker's preferences, that can be adjusted to obtain the best possible performance. This module is adapted from the *Module for Generating Schedules* by Pinedo (2012), and the *Schedule Generator Module* by Framinan et al. (2014). We can see the main components of this module below.

1. **Preprocessor**. The first component is in charge of creating the instance that will be solved by the appropriate solution approach. To do so, this component will access both the DSSTS database and the model database to obtain the required data. Depending on how the solution approach is selected, this component will use different information. It is also responsible of analysing the instance and obtaining which are the possible approaches that can be applied in two different ways, first which is the managerial level that wants to be faced, i.e. planning, scheduling or rescheduling, and second, which approaches are applicable for that managerial level. If the selection is made automatically, some fitness measure must be computed for each possible candidate, and the one with a higher value is finally selected. In the case where the decision maker selects the approach, only those that can be applied shoud be shown to the decision maker. It is also possible to assign a fitness measure in this last option to guide the decision maker, although the final choice is his/hers.

2. **Solution Approaches Library**. The actual solution approaches are managed by this component. There are different strategies to develop these solution approaches. To maintain modularity, it is preferable not to have any built-in solving approach. Therefore, one interesting way of implementing this component is to have a repository of solving approaches executables that are managed from the *Model Database* component, i.e. their location, their main features, where they are applicable and so on, are stored in the database, while the solving approaches are stored in some directory.

An alternative to implement this component is, as commented in Framinan et al. (2014), to store the approaches in the database in the format of an interpreted language, and load and execute them when necessary. This is how SAP R/3 deals with this component. From our point view, this approach increase unnecesarily the complexity of the system and also the workload of the database.

It is also interesting to distinguish between two types of solution approaches:

- General Purpose Approaches. These are common approaches that can be applied to any problem. In scheduling we could find dispatching rules, some metaheuristics, e.g. tabu search, genetic algorithms, etc. or any other approach that could be applied without considering the problem under study. Typically, this approaches will obtain feasible but not optimal solutions, so they will be recommended both, when the problem under study is very complex or when speed is the main requirement.
- Specific Solution Approaches. When we need optimal solutions for specific problems, new approaches taking advantages from the structure of these problems must be developed. These type of approaches are what we assume as specific approaches. They are valid only for those problems which they are developed for. They typically obtain better performances at the cost of higher developing and computational times. We must highlight that this component needs to retrieve data from the *Model Database* component in order to know which are the models that can be solved. Within this type of solution approaches we could consider the possibility of having approaches that are developed for a specific type of problem, but that can also serve to solve a number of related problems. This typically happens when we have particularization of problems. Their solutions approaches are valid for them and for any kind of generalization of the problem. As an example, if we need to solve a hybrid flowshop problem with missing operations (for more information see Part IV), the specific solution approach for that problem could serve for solving the hybrid flowshop scheduling problem. It is important to note that, in general, the performance of the approach will worsen as compared to the performance of the approach for the problem for which it was developed. To use this type of approach it must be perfectly defined in the *Model Database* component which is the hierarchy of problems that can be solved with each solution approach, so that a solution procedure made for a problem in the hierarchy can solve this problem and all the levels below.

Solution approaches can be also classified according to an adaptation of the classification made by Framinan et al. (2014):

- Quick Methods for Feasibility. These type of approaches are those where optimality is not important at all, and the only aim is to find feasibility. In general, these approaches will not provide good solutions, in the sense of robust or adjusted plans or schedules. This type of approaches are interesting in those cases where the scheduler needs a quick feasible solution to modify it according to his/her experience.
- Exact Methods. The aim of these approaches is to look for optimality. According to Framinan et al. (2014) there are two types of exact algorithms, namely constructive and enumerative algorithms. The first type is only possible for a number of problems, typically simple. These approaches try to exploit the specific properties of the problem to construct a solution that is guaranteed to be optimal. Within the second type, we distinguish Integer Programming, Branch & Bound and Dynamic algotithms. As we already stated, these approaches looks for the optimal solution by enumerating all possible solutions. As expected, depending on the complexity of the problem these approaches usually become extremely slow and, because of that, are difficult to use in real practice.
- Approximate Methods. Among approximate methods we find heuristics and metaheuristics. Heuristics are commonly developed for specific problems. They usually take two different forms, constructive or iterative. The first one tries to construct good solutions from scratch following some procedure based on the characteristics of the problem, while the second type try to reduce the space of solutions according to some criterion. On the other hand, metaheuristics are general approaches that can be applied to a number of different optimization problems. For a wide view of metaheuristics in production we refer the reader to Zäpfel et al. (2009).

3. **Postprocessor**. The objective of this component is to refine the solutions obtained by the solution approaches. If the scheduler considers it necessary, this component offers the capability of carrying out an improvement of the solution by means of some procedure. This component is not model-dependent, but depends on the constraints of the problem, and, as such, it computes the schedule according to the DSSTS constraints, and not with respect to any specific scheduling model. An interesting application of this component is related to *what-if analysis*, as it allows the user to look for different solutions in different computational times so, if speed is not a concern, the scheduler could try to improve the solutions by using this posprocessor with different computational time limits.

4. **Dispatcher**.The last component is responsible of translating the solution obtained from the different approaches, and in some case from the postprocessor, to a generic format and communicating it to the

corresponding compnent of the user dialogue management module. As commented in Framinan et al. (2014), it could be also interesting to offer the possibility of including some additional constraints at this point, in order to give the decision maker aditional means of reflecting his/her experience in the plan/schedule.

The four components of this module are used sequentially, although depending on the case some of them can be ommited, e.g. if we focus on feasibility, the *Postprocessor* component can be skipped.

3.3 DSSTS Deployment

The framework can be implemented according to different computer/network architectures, including:

- Standalone Architecture. The simplest case is to deploy the DSSTS as a standalone system. In this architecture every module is contained in the same computer. Planning, scheduling and control are managed from the same computer, so it does not allow for a real time updating of the work in the plant unless the system is close to the plant and the plant operators inform the scheduler everytime they start or finish a task. This architecture could be useful for small companies if the facilities are not separated from the offices, a real time control of the tasks is not required, the size of the scheduling data (resources, tasks, ...) is low and where no other information systems are present in the organization. We can see an example of the implementation of this architecture in Figure 3.3.

- Client-Server Architecture. Within this architecture, part of the system is located in a server while the rest is located in a number of different clients that access that server. They share common database management and model management modules, while each client has its own user dialogue management module. This could allow to modify this last module in order to have a better fit with the profile of the users of each terminal. This architecture could serve for a small or medium organization, with a number of stages in the business process not too high. It would offer the possibility of a real-time control of the tasks executed in the facilities and it would also be possible to interconnect it with the rest of the information systems in the organization. An example of this architecture is shown in Figure 3.4.

- Service Oriented Architecture (SOA). This last architecture is oriented to medium and big organizations. It is the most flexible one, allowing for an easy access to the DSSTS in the case that the organization has different buildings (or offices) placed in different geographical locations. It does not matter where the services provider is located. On the contrary, it is the most complex architecture

Issues	*Database Management Module*	*User Dialogue Management Module*	*Model Management Module*
Address different decision problems	X		X
Manipulate schedules manually		X	
Consider the different stages of the scheduling process	X		
Synchronisation between managerial levels	X		X
Adapt to different contexts	X	X	X
Support activities done by human decision makers	X	X	

Table 3.2: Fulfillment of problems in DSSTS implementations.

according to technical requirements. In this case, all the funcionalities offered by the DSSTS are encapsulated into services that are consumed by actors of the scheduling process at each point. This architecture is similar to the previous one but it avoids the necessity of modifying anything in users' devices everytime something from the DSSTS is updated. The modification of the services is completely transparent to users. Figure 3.5 shows an example of this architecture.

3.4 Conclusions

In this section we have presented the framework that will be used and validated in the next sections of this Thesis. The framework has been analysed from different perspectives. First, to analyse the main directives of the framework and how the different actors interact with the DSSTS resulting from its implementation we provided a MVC description of the framework. The proposed framework intedns to have a high degree of flexibility and modularity that ease the implementation of the resulting DSSTS. Second,we detailed the main roles of the actors in the scheduling process, together with the common requirements for each of them. Next, we provided a detail analysis on the different modules composing the framework together with the main funcionalities a DSSTS should offer. Additionally, we propose three different implementations of the framework depending on the chracteristics of the organization where the DSSTS is to be deployed.

To conclude this chapter we show in Table 3.2 a summary of how the functionalities of the modules of the proposed framework fullfil all requirements extracted in Chapter 2.5 from the common problems implementing DSSTS that were found in the literature. The next chapters will serve as an evaluation and validation of the framework in two very different fields: manufacturing and healthcare.

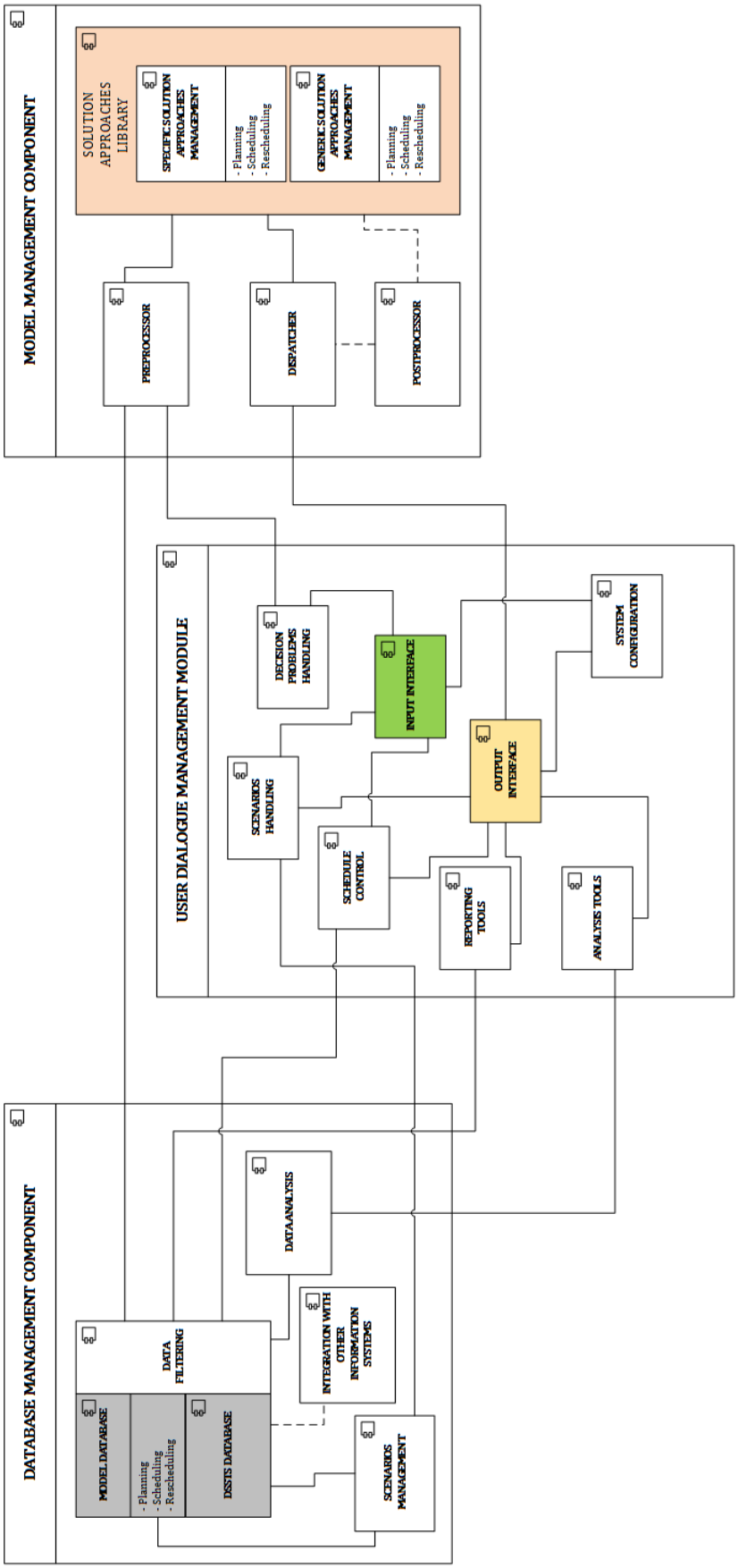

Figure 3.2: Generic UML Component Diagram of a DSSTS.

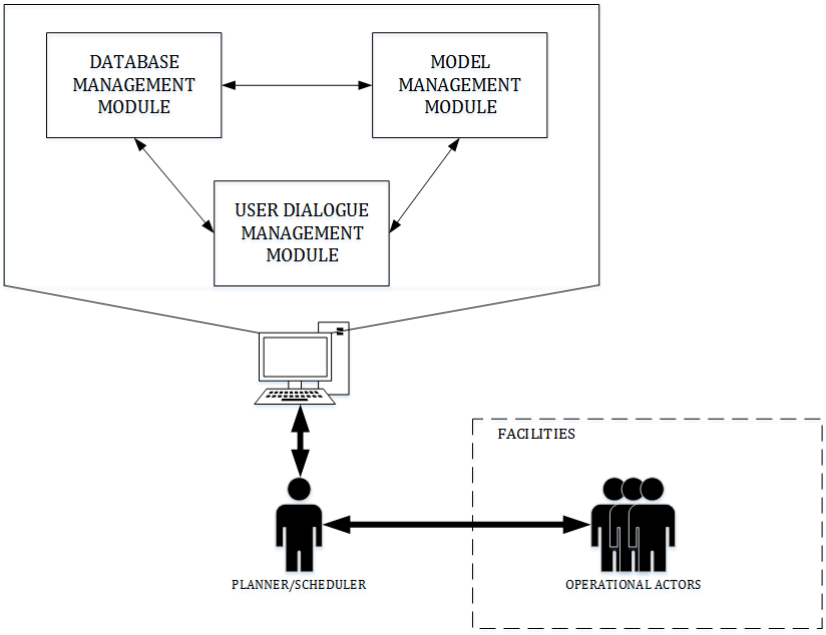

Figure 3.3: Deployment of the DSSTS as a standalone system

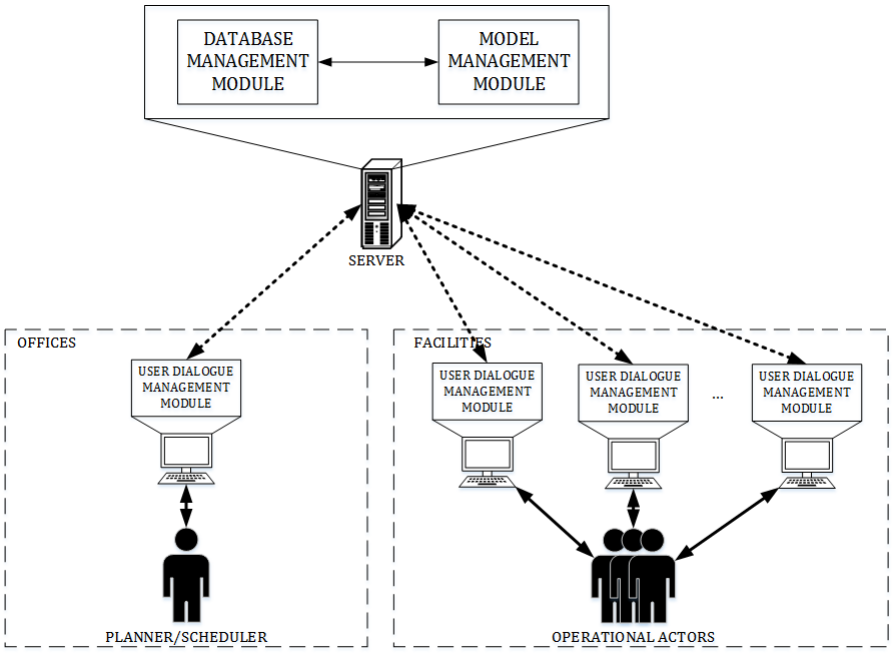

Figure 3.4: Deployment of the DSSTS in Client-Server architecture

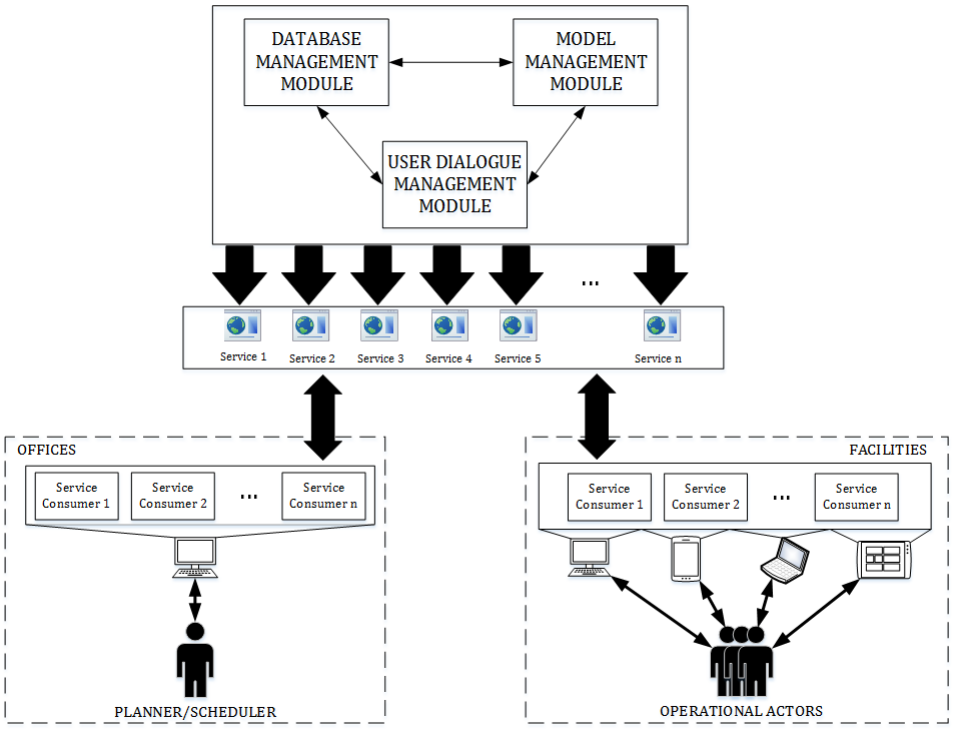

Figure 3.5: Deployment of the DSSTS in SOA

Part III

REVIEW ON DECISION SUPPORT SYSTEMS

Chapter 4

Manufacturing DSSTS: Background and Literature Review

To check the alignment of the framework proposed in the previous chapter with the existing DSSTS in the literature of manufacturing scheduling, in this chapter we carry out a systematic review of the literature on manufacturing DSSTS and provide a scheme for their classification. We also make an attempt to review commercial manufacturing DSSTS, but according to the large amount of cases in the literature, to the existence of reviews on this topic (see Pinedo, 2012), and to the difficulty in gathering data from companies due to confidentiality issues, we focus only on those systems described in the academic literature.

Although a number of case studies and descriptions of implementation of manufacturing DSSTS is available in the literature, there is a great variation regarding the functionalities of this software, ranging from relatively simple applications focused on a specific problem, to sophisticated information systems capable of supporting a wide range of scheduling decisions. This variability, coupled with the specific nature of scheduling, makes difficult to have a coherent picture of the developments in the area, which in turn hides both specific topics not yet addressed and issues already solved in a satisfactory manner.

In order to analyse the validity of the proposed framework, in our systematic review we focus on what we call the **structure** of the manufacturing DSSTS, i.e. which are their functionalities and how these functionalities are organised. In this way, we investigate what these DSSTS are made for rather than focusing on how these functions are achieved. The review shows a field in which great advances have been accomplished, but also where some mismatches between research and practice are revealed (such as the implementation problems already discussed in Chapter 2). This review is organized in three parts. First, the methodology used to select the contributions to review is detailed. Next, in Section 4.2, we

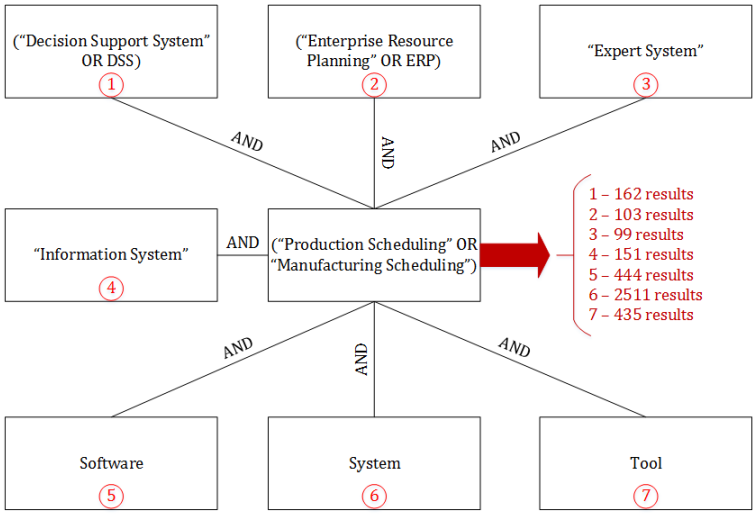

Figure 4.1: Review queries and summary of the results

briefly discuss the main issues regarding manufacturing DSSTS and their structure in order to present the schema to classify existing contributions on the topic. Section 4.3 shows the results of the classification of the manufacturing DSSTS and finally, in Section 4.4 we present the main findings of the review.

4.1 Review Methodology

The procedure adopted for this review consists of two stages. In a first stage, a systematic review was developed for papers published from 2000 to 2016. Given the processing and graphical capabilities of computers prior to that date, we first focus on that period. We used the SCOPUS search engine by Elsevier, given that the majority of relevant journals and conference proceedings are indexed in this database. Different queries were performed to take into account as many systems as possible. To do so, we also used different definitions of systems commonly employed in scheduling practice, such as Decision Support Systems, or Expert Systems. The queries used for the review are shown in Fig. 4.1 together with the number of results obtained.

Due to the heterogeneity and ambiguity of the results of the first stage, some of them were not suitable for our study. Therefore, we adopted a three-step procedure to filter the results:

- Title. First, we rejected those works whose title was not relevant for our study.
- Abstract. The abstracts of works that seemed to be relevant were carefully read and those that did not focused on the topic were excluded.
- Full Document. Those works still remaining were analysed in full-depth in order to obtain the final

set of contributions for the review.

In a second stage we extended the number of contributions by selecting all relevant references cited by the works in the first stage, including references prior to 2000. To filter theese new contributions, we adopted the same three-step procedure as in the first stage. Moreover, we include book chapters that were not considered in previous stage, but that were listed in the references selected in the second stage.

4.2 The Structure of Manufacturing DSSTS

Since manufacturing DSSTS constitute a special type of business information systems (Framinan et al., 2014), they can be described along a list of *functionalities* or pieces of business functions that the system is capable to support. In other words, the functionalities describe which manufacturing decisions are supported by the system. These functionalities and the way in which they are organised is what it is called in the remainder of the chapter *structure* of the system and will serve to distinguish the different features found in the systems described in the literature.

Clearly, as many systems described are company-specific, it is necessary to distinguish between specific functionalities (unique for each software application) and those which are common to most systems and that constitute the *architecture* of the system. The work by Framinan and Ruiz (2010) presents a classification of the generic (i.e. high-level) functionalities of a manufacturing scheduling system. Therefore, we can use this classification as a starting point although, given its abstract nature, a modification and extension of the classification is required. In addition, since our review is based on actual descriptions of manufacturing scheduling systems, some categories present in the architecture are not found in practice.

As a result of the analysis of the final set of references commented in the previous section, a number of types of functionalities were identified. These types constitute the schema for classification (see Figure 4.2), and are discussed in the following subsections.

4.2.1 Problem Modelling

This type of functionality refers to the ability of the system to capture in an autonomous or semi-autonomous manner different parameters of the corresponding shop floor. The following functionalities within this type are considered:

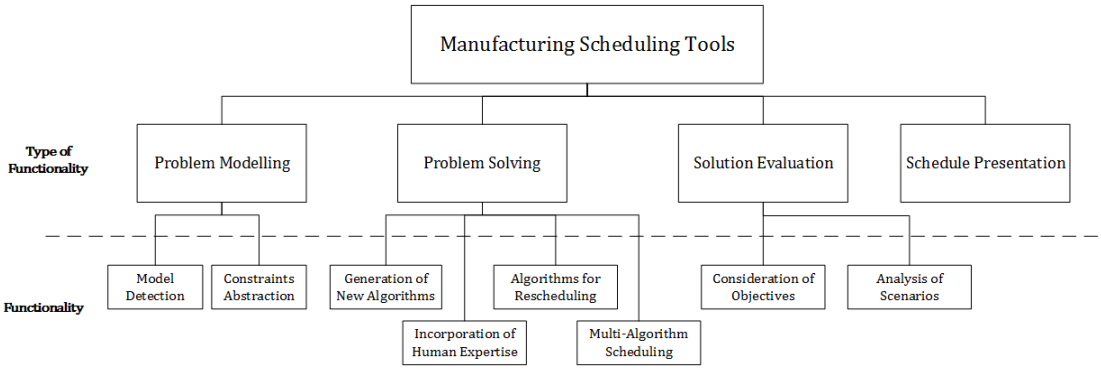

Figure 4.2: Classification schema

Model Detection (MD)

This functionality refers to the ability of the system to determine the most suitable (abstract or theoretical) scheduling model from the raw instance data provided to the system. Since some theoretical scheduling models can be seen as simplification of a real-life setting, model detection might be seen as a type of constraint abstraction. However, since it focuses on a specific type of relaxation (i.e. that to reach to specific scheduling models so solution procedures taken from these models could be applied), we keep it apart. Model detection is achieved by the systems reviewed using different approaches:

- **Reduction Trees (RT)**. Reduction trees constitute a useful taxonomy for scheduling problems, as they establish the interdependences for well-known scheduling problems together with their corresponding scheduling algorithms. Therefore, if the user enters the type of scheduling problem to be solved, it is possible for the system to look for the stored algorithms which are closest to this problem according to the reduction tree.
- **Decision Trees (DT)**. In this approach, the user has to iteratively respond to a number of questions related to the constraints and objectives in the scheduling decision so the system finds the model (among those entered in the system) that better fits with the models in the system.
- **Raw Data Detection (RDD)**. In this approach, the system is able to detect the corresponding layout from raw instance data.

Constraints Abstraction (CA)

Constraints abstraction refers to the ability of the system to work/detect a simplified subset of constraints, so those with little impact in the evaluation of the solution may be ignored or omitted for a further consideration, if possible. This functionality has been implemented in several ways:

- **Inclusion of Constraints into the Objective Function (COF)**. The constraints are included in the objective function, usually via penalization.

- **Hard Constraints vs. Soft Constraints (HSC)**. The system allows the user to distinguish between *hard* constraints and *soft* constraints (so-called preferences). The goal is to find a schedule satisfying all *hard* constraints and that relaxes as few as possible of the *soft* constraints. This approach can be further refined by providing the system with the importance (or weight) of each constraint, so a hierarchy of constraints is defined to evaluate the degree of satisfaction of the soft constraints.

- **Constraint Violation Warning (CVW)**. In some systems, only general constraints relative to the usage of resources (machine conflict resolution and machine selection) are considered. For the rest of constraints, the system launches warnings to the user so he/she can manually try to comply with these constraints.

- **Black-box Constraints Abstraction (BCA)**. The relaxation of certain constraints is performed internally by the system to augment the search space. Once a solution has been found, the relaxed constraints are enforced iteratively. The relaxation of these constrains may refer to either temporal and due-date related constraints are relaxed, or to the consideration of partial schedules.

4.2.2 Problem Solving

The functionalities grouped under this type refer to how the system generates schedules (i.e. solutions to the problem). Based on the classification by Framinan and Ruiz (2010), a number functionalities are identified. These are discussed in the next subsections.

Generation of New Algorithms (GNA)

This refer to the ability of the system to incorporate new algorithms to solve scheduling models. Here we can distinguish between the incorporation of new algorithms at **design level**, or at a **user level**, depending in whether a re-compilation of the system is needed. While for the first case, the challenge is to accommodate the design of the system to ease the incorporation of new algorithms without altering its structure, the generation of new algorithms allows the user to incorporate new algorithms by one of the following approaches:

- **External Generation (EG)**. The algorithms are incorporated into the system as executable files that capture the relevant data and export the results via a well-defined input/output interface.

- **Language-Based Generation (LBG)**. The algorithms can be introduced by the user in a relatively straightforward manner using a sort of language to produce algorithms, although in the experiences found in the literature, these algorithms are confined to simple sequencing rules.
- **Generation by Combination of Existing Algorithms (CEA)**. In this approach, the system stores chunks or blocks of different selection rules (mostly dispatching rules) that can be combined into different *scheduling strategies* (i.e. order-based scheduling, resource-based scheduling, and operation-based scheduling), so different *composite* algorithms can be incorporated into the system.
- **Rule-based generation (RIT)**. In this approach the user can generate a number of rules of the type $IF \ldots THEN$ to determine which job is to be selected to be scheduled.

Incorporation of Human Expertise (HE)

.

This functionality refers to the ability of the system to incorporate the knowledge of the Decision Maker. Several approaches to implement this feature have been adopted in the systems reviewed:

- **Ad-hoc Solution Procedures (AHSP)**. The system allows to introduce ad-hoc solution procedures that reflect the expertise of the scheduler.
- **Working Conditions (WC)**. Instead of introducing ad-hoc solution procedures, these are obtained as the result of a set of working conditions established by the expert. Moreover, these working conditions are in some cases entered in an interactive manner.
- ***Excuses* (E)**. Some authors propose a functionality called *excuses* that allows the user to see why an order is late. This functionality makes easier the development of new ad-hoc heuristics.
- **Ad-hoc Constraints (AHC)**. It is possible to introduce specific process constraints, such as grouping certain types of jobs.

Algorithms for Rescheduling (AR)

This functionality indicates the capability of the system to offer support for events that may impact the current schedule, such as machine breakdowns, rush orders, etc. In this functionality, we include systems which do not completely rebuild the existing schedules (perhaps with the addition of new incoming jobs), but perform a reschedule in which a subset of jobs change their schedule. The construction of the new schedule is achieved using different approaches:

- **Forward Rescheduling (FR)**, i.e. pushing ahead the starting time of the operations affected by the disruptive event.
- **Backward Rescheduling (BR)**, i.e. pushing backwards the starting times of the operations affected by the disruptive event, whenever this results in a feasible schedule.
- **Exclusion of the Affected Operations (EAO)**, so the system allows the user to reject a scheduled operation. A particular case of this option can be found in Fox (1994), where a rather sophisticated method is proposed: the jobs affected by these activities were *unscheduled* and their former schedule (i.e. the reservation of the time on each operation) was transformed into a *soft* constraint. Then, the jobs affected were rescheduled taken into account the new *soft* constraints.
- **Breaking Down the Tasks (BDT)** into smaller pieces so they can be executed and/or rescheduled (entirely or in part) in the idle times of the schedule resulting from the disruptive event. This option is labelled opportunistic insertion by Chan and Zeng (2003).
- **Merging Tasks (MT)**. This case is the opposite to the previous one, and the rescheduling is performed by trying to merge different operations into a single one.
- **Changing the Weight/Priority of Affected Operations (CW)**. The weight of the jobs affected by the unexpected event is altered, so they have a higher priority for being rescheduled.
- **Searching for Alternative Resources (SAR)** for scheduling the affected operations, perhaps including the possibility of outsourcing production to other operators is also considered.
- **Modify Constraints between Tasks (MCT)**. In some systems, the schedule is not explicit, but implicitly described using some rules and/or constraints among operations. In those systems, rescheduling is done by modifying these rules.
- **Case Based Reasoning (CBR)**. A database with information about how similar problems were addressed in the past is used to modify the existing schedule.

Multi-Algorithm Scheduling (MA)

This functionality indicates that the system supports a logical separation of models and the corresponding solution procedures, so different solution procedures could be applied to a given model, and one solution procedure could be employed for different models. If such separation exists, several aspects have to be defined, i.e. the **algorithm library** (which types of algorithms are included in the system to be selected), the **selection mechanism** (how a single algorithm is picked from the algorithm library), and

the **selection mode** (in which time period the selection is performed). Regarding the algorithm library, different types of algorithms can be found in the systems reviewed:

- **Generic Algorithms (G)**, that may be based on bottleneck/machine workload considerations, or are simply based in sequencing methods/dispatching rules at the different resources.
- **Local Search and Metaheuristics (LSM)**
- **Specific algorithms (SA)**.

Regarding the selection mechanism, the following approaches have been encountered:

- **Selection by User (U)**. Using this approach, a set of algorithms is available for the user, who chooses one from the library.
- **Guided Selection (G)**. In some systems the selection is done by the user, but the system offers some interactive aid that helps him/her.
- **Autonomous Selection (A)**. The system is capable to perform the selection, and the approach is more similar to a *black box* so the user is provided with the best solution found among part/all algorithms in the library but he/she cannot trace back which one provided the best solution.

Finally, regarding the selection mode, the following options have been implemented:

- **On-line selection (ON)**, where the algorithms are tested in real-time (or quasi real-time) with respect to the solution obtained in the problem instance that it is to be scheduled.
- **Off-line selection (OFF)**, where a benchmark on existing testbeds in the system is performed, so the efficiency of the algorithms is tested before solving the instance.

4.2.3 Solution Evaluation

This type of functionality refers to how the system evaluates the schedules obtained. Two aspects –discussed in the next subsections– were identified within this type of functionalities.

Consideration of objectives

With respect to how objectives are considered by the system, several options are present:

- **Constrained Approach (CA)**, where objectives are embedded into constraints.

- **Weighted Combination (WC)**. The system employs a single objective function composed of a weighted sum of different objectives.
- **Lexicographic Approach (LA)**. The decision maker can establish a primary objective, and then a set of secondary objectives.
- **User Selection / Single-objective (US)**, The system offers the decision makers the list of available objectives and allows him/her to select one of them.
- **User Selection / Several Objectives (USO)**. The system offers a set of solutions (presumably found by different solution procedures, each one seeking one objective), and evaluates them with respect to the set of objectives available in the system.
- **User Selection / Multi-criteria (USM)**. The system offers the Decision Maker a set of solutions obtained from one/several multicriteria algorithms (pseudo-Pareto set).
- **Stochastic evaluation (SE)**. In addition to the deterministic evaluation of objectives, the system is able to evaluate the schedules on an stochastic (e.g. simulation) basis, so that the potential impact of unexpected events, or deviations from the (deterministic) data introduced in the system can be assessed.

Analysis of Scenarios (AS)

This functionality refers to the ability of the system to manage different potential schedules, so the user is able to handle different schedules at a time. Fargher et al. (1994) distinguishes two types of analysis, i.e.:

- **Implicit analysis (IA)**, which means that a new schedule can be evaluated so its impact can be assessed. Typically, different schedules are produced by the system and their results are compared and offered to the decision maker.
- **Explicit analysis (EA)**, which is used to check the impact of modifications of the working conditions of the shop floor in the current schedule. This is usually referred as a what-if analysis.

4.2.4 Schedule Presentation

This type of functionalities refers to how the system presents the information to the Decision Maker and how it interacts with him/her. This type of functionality is adopted in the reviewed systems using different means:

- **Text (T)**, as the simplest form of presenting data.
- **Gantt Charts (GC)**. Classical chart where horizontal bars, representing the time required to execute a task, are located into a diagram where the vertical axis represents the resource. Although it typically serves to present information, in some cases it is possible some interaction, such as e.g. drag and drop to manually modify the schedule.
- **Job Screens (JS)**. This functionality presents the jobs in the schedule together with their main attributes to the Decision Maker.
- **Other Charts (OC)**. Since there is a large variety in the forms of presenting the data other than the charts described above, we will use this option to group them and will discuss them separately.

4.2.5 Interaction with other Decisions

Finally, this category refers to the type of manufacturing decisions supported by the system in addition to those specifically referring to scheduling. In the reviewed systems, the following decisions have been found:

- **Planning**, i.e. time- and resource- aggregated allocation of jobs to resources.
- **Control**, i.e. real-time tracking of the schedule's execution.
- **Transportation**, i.e. movement of goods before/after their manufacture.

4.3 Analysis of Manufacturing DSSTS

Equipped with the classification for the structure of the systems for manufacturing scheduling discussed in Section 4.2, we have reviewed the contributions in the literature. A total of 114 cases have been found, and their classification is summarized in Table 4.1. These references represents a total of 99 systems (in some works different contributions describe the same system) over a timespan of 30 years, which speaks for the existence of a sizeable repository of cases from which lessons can be learnt. Note that most contributions (72) refer to journal papers, which shows the quality and importance of the contributions reviewed.

Regarding the distribution by year of publication, Figure 4.3 shows a relatively uniform distribution of them, which seems to indicate that the interest of the topic has remained constant through time.

Figure 4.4 shows the degree of maturity of the systems. We have differentiated the following degrees according to the information provided at the time of publication:

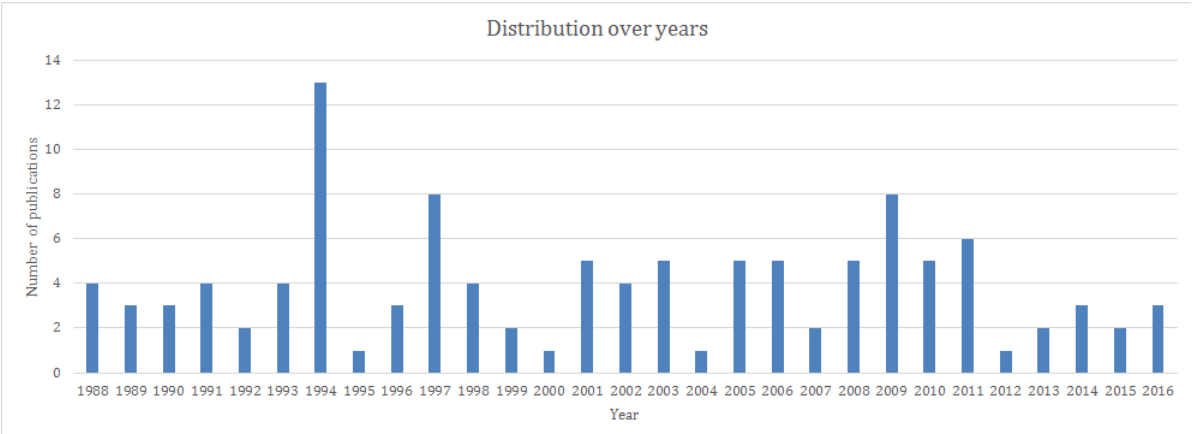

Figure 4.3: References by date

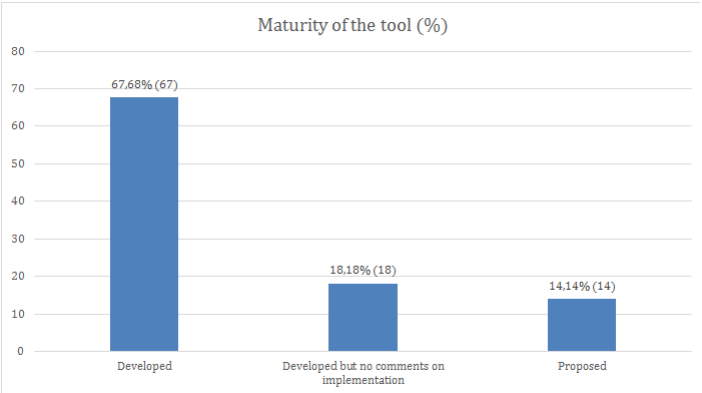

Figure 4.4: Maturity of the systems

- The system has been fully developed (D),
- The system may have been developed, but no screenshots or results on their implementation has been provided (E)
- The system has been proposed, but not implemented (P).

As it can be seen from Figure 4.4, most of the reviewed works present systems already developed, but there is also a number of works where the details are not provided, due to work-in-progress, or to the privacy on the results.

When fitting the functionalities into the framework discussed in Section 4.2, we find that there is a wide diversity in the number and type of features considered (see Figure 4.5): while Schedule Presentation features are described in around 85% of the systems (84), Problem Modelling is present only in about 28% (28). Furthermore, less than 10% systems (8) cover all types of functionalities (see Table 4.1), which means that only few systems cover the whole scheduling process, from modelling to solution representation.

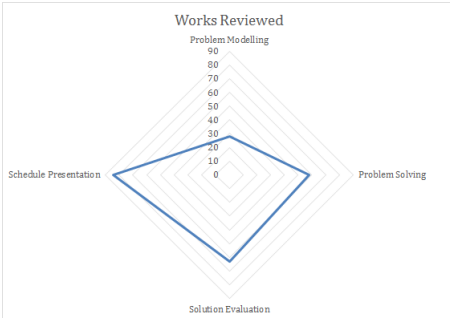

Figure 4.5: Relative Importance of Functionalities

Reference	Interaction with other Decisions			Problem Modelling		Problem Solving				Solution Evaluation		Scheduling Presentation			
	P	C	T	MD	CA	GNA	HE	AR	MA	CO	AS	T	GC	JS	OC
Collinot et al. (1988) Le Pape (1994)		X						X					X		
Kerr and Ebsary (1988)		X			X			X				X	X		
Kolen and Woerlee (1988) Speranza and Woerlee (1991)												X	X		X
Lamatsch et al. (1988)				X		X						X	X		
Numao and Morishita (1988) Numao and Morishita (1989) Numao and Morishita (1991) Numao (1994)					X										X
Savell et al. (1989)												X			
Bona et al. (1990)		X			X							X	X		
Hadavi et al. (1990) Hadavi et al. (1992)	X	X			X			X		X	X	X			
Niew et al. (1990)										X					X
Adelsberger and Kanet (1991)		X											X		
Burke and Prosser (1991)		X			X								X		
Boccalatte et al. (1992)											X	X	X		
Hsu et al. (1993) Prietula et al. (1994)		X			X		X				X	X	X		
Sadeh (1993) Sadeh (1994)		X								X		X	X		
Sauer (1993)		X				X	X	X	X	X	X		X		
Zong et al. (1993)						X	X				X	X			
Aerts et al. (1994)					X					X	X		X		
Boccalatte et al. (1994)					X								X		
Fargher et al. (1994)	X							X			X	X	X		
Flower and Cheselka (1994)		X					X				X		X		
Fox (1994)					X			X				X	X		
Kempf (1994)		X							X				X		
Marriott (1994)		X									X		X		X
O'Donoghue et al. (1994)		X					X					X			
Taunton and Ready (1994)						X	X						X		X
Weigl (1995)		X					X		X		X		X	X	X
Goldman et al. (1996) citeGol1997		X			X		X	X					X		
Karacapilidis and Pappis (1996)	X						X				X	X			
May and Vargas (1996)		X					X				X	X			
Artiba and Aghezzaf (1997)	X						X		X	X	X		X		
Esquirol et al. (1997)							X						X		X
Pinedo and Yen (1997) Yen (1997)									X	X		X	X	X	X
Sauer and Bruns (1997)		X			X	X		X		X		X	X		
Wang and Lin (1997)				X			X		X			X			
Weintraub et al. (1997)		X										X			X
Kuo and Hwang (1998)							X					X			
Patkai (1998)		X					X		X		X		X		
Sauer et al. (1998)	X	X				X						X	X		
Vaidyanathan et al. (1998)					X						X	X			
Marinho et al. (1999)									X	X	X		X		
Murthy et al. (1999)									X	X	X	X	X		X
Henning and J. (2000)		X						X		X	X	X	X	X	X
Blazewicz et al. (2001)		X						X	X	X		-	-	-	-

Reference	Interaction with other Decisions			Problem Modelling		Problem Solving				Solution Evaluation		Scheduling Presentation			
	P	C	T	MD	CA	GNA	HE	AR	MA	CO	AS	T	GC	JS	OC
Blazewicz et al. (2007)															
Everett (2001)			X				X								X
Gazmuri and Maturana (2001)						X					X	X			X
Kotak et al. (2001)											X	X			X
Musselman (2001)		X						X			X		X	X	
Gupta et al. (2002)		X						X		X	X	-	-	-	-
Keskinocak et al. (2002)					X				X	X	X				X
Metaxiotis et al. (2002)				X					X	X		X			
Saydan and Cooper (2002)	X										X	X			
Chan and Zeng (2003)		X						X		X	X	-	-	-	-
Concannon et al. (2003) Hindle and Duffin (2006)		X							X	X	X	X	X		
Feng et al. (2003)												-	-	-	-
McKay and Wiers (2003)	X	X									X	X	X		
Ozbayrak and Bell (2003)										X		X	X		
Bredstrom et al. (2004)	X		X		X							-	-	-	-
Appelqvist and Lehtonen (2005)		X								X			X		X
Cheeseman et al. (2005)		X						X					X		
Geiger (2005) Geiger (2011)									X	X			X		X
Munawar et al. (2005)		X			X								X		
T'kindt et al. (2005)				X		X			X	X	X		X	X	
Almeida and Marreiros (2006)									X	X		X			
Chan et al. (2006)										X		-	-	-	-
de Castro et al. (2006)										X		-	-	-	-
Kumar and Rajotia (2006)	X											-	-	-	-
Jacobi et al. (2007)	X	X					X			X	X	X	X		
Abreu et al. (2008)										X		X			
Bonfill et al. (2008)			X			X				X			X		
Gao and Tang (2008)							X			X	X	-	-	-	-
Józefowska and Zimniak (2008)	X									X	X	X	X		
Sotiris et al. (2008)							X	X	X		X	X	X		
de Ugarte et al. (2009)		X					X	X			X	X			X
Lopez and Villar (2009)		X										X			
Missbauer et al. (2009)	X				X					X		X			
Silva (2009)					X						X	X	X		
Stevenson et al. (2009)		X											X	X	
Tang et al. (2009) Zhang et al. (2010)							X				X				X
Varela et al. (2009)						X			X			X	X		
Yang and Lin (2009)			X	X					X			-	-	-	-
Ko and Wang (2010)										X		X			
Leung et al. (2010)		X						X				-	-	-	-
Maturana et al. (2010)					X							-	-	-	-
Sadi-Nezhad and Darian (2010)												X			X
Angelidis et al. (2011)						X					X				
Trojet et al. (2011)					X						X	X			X
Nieuwenhuyse et al. (2011)	X										X	X			
Zhao and Lin (2011)												-	-	-	-
Zhongyi et al. (2011)	X											X	X		
Barlatt et al. (2012)	X	X			X		X	X		X	X		X		
Korosec et al. (2013)	X									X	X	X	X		
Madureira et al. (2014b) Piairo et al. (2013)		X						X		X			X		
Madureira et al. (2014a)		X					X	X			X	X	X		
Upton and Quilligan (2014)							X				X	X	X	X	
Figueira et al. (2015)	X				X		X	X		X	X	X	X		X
Guo et al. (2015)	X	X			X					X		X			X
Fanti et al. (2016)		X								X	X	X	X		
Mourtzis et al. (2016)							X	X	X	X			X		
Zheng et al. (2016)				X			X								X

Table 4.1: Functional Features Review

Regarding the approaches adopted to implement the different features within the framework discussed in Section 4.2, these are discussed for each type of functionality in the next subsections.

4.3.1 Problem Modelling

Model Detection is implemented only in 6 of the systems described. The rest of the systems usually leave to the user the choice of the most suitable model (see Pinedo and Yen, 1997; Yen, 1997) or, in the simplest cases, only one specific model can be used (Ko and Wang, 2010), therefore making the system hardly usable if any change takes place in the scheduling process. Among the works implementing Model Detection, the approaches are summarised in Table 4.2, ranging from those using Reduction Trees (RT) (Lamatsch et al., 1988; Zheng et al., 2016), those using Decision Trees (DT) (Wang and Lin, 1997; Metaxiotis et al., 2002), and those that can detect the layout from raw data (T'kindt et al., 2005; Yang and Lin, 2009).

Regarding constraint abstraction, 22 of the systems implement this functionality using different approaches. COF is adopted in Bredstrom et al. (2004) where authors include constraints into the objective function with a penalty cost. The HSC implementation can be found in Kerr and Ebsary (1988); Bona et al. (1990); Hsu et al. (1993); Boccalatte et al. (1994); Prietula et al. (1994); Sauer and Bruns (1997); Munawar et al. (2005); Missbauer et al. (2009); Silva (2009); Figueira et al. (2015). An improvement of HSC can be found in Fox (1994), where the author brings together the existence of hard and soft constraints and the assignment of a penalty cost to soft constraints to gain some control when trying to satisfy them, i.e. which ones should be considered first. A more relaxed approach (CVW) is used by Numao and Morishita (1988, 1989, 1991); Numao (1994); Hadavi et al. (1990, 1992); Vaidyanathan et al. (1998). This approach could be seen as a special case of the previous one, where hard constraints are those related to resources and the process of satisfying soft constraints is left to the user. In the last approach detected (BCA), the system itself is in charge of relaxing constraints to try to obtain a solution. Once a solution is obtained, the system tries to force the constraints. In Burke and Prosser (1991); Goldman et al. (1996); Goldman and Boddy (1997); Maturana et al. (2010); Trojet et al. (2011), their systems relax temporal and due-date related constraints, whereas in Keskinocak et al. (2002); Barlatt et al. (2012); Guo et al. (2015), authors use the concept of partial schedules, i.e. schedules considering only those jobs with compatible characteristics, and try to reach the best possible solution by refining these schedules iteratively.

4.3.2 Problem Solving

Since classical scheduling focuses in solving scheduling problems in the most efficient manner, problem solving functionalities could be expected to be widely implemented in the reviewed systems. However, this only happens in a bit more than half of the systems.

As described in Section 4.2, 4 different functionalities within this type were considered. The first one is

Reference	Problem Modeling	
	Model Detection	Constraints Abstraction
Kerr and Ebsary (1988)		HSC
Lamatsch et al. (1988)	RT	
Numao and Morishita (1988) Numao and Morishita (1989) Numao and Morishita (1991) Numao (1994)		CVW
Bona et al. (1990)		HSC
Hadavi et al. (1990) Hadavi et al. (1992)		CVW
Burke and Prosser (1991)		BCA
Hsu et al. (1993) Prietula et al. (1994)		HSC
Aerts et al. (1994)		BCA
Boccalatte et al. (1994)		HSC
Fox (1994)		HSC
Goldman et al. (1996) Goldman and Boddy (1997)		BCA
Sauer and Bruns (1997)		HSC
Wang and Lin (1997)	DT	
Vaidyanathan et al. (1998)		CVW
Keskinocak et al. (2002)		BCA
Metaxiotis et al. (2002)	DT	
Bredstrom et al. (2004)		COF
Munawar et al. (2005)		HSC
T'kindt et al. (2005)	RDD	
Missbauer et al. (2009)		HSC
Silva (2009)		HSC
Yang and Lin (2009)	RDD	
Maturana et al. (2010)		BCA
Trojet et al. (2011)		BCA
Barlatt et al. (2012)		BCA
Figueira et al. (2015)		HSC
Guo et al. (2015)		BCA
Zheng et al. (2016)	RT	

Table 4.2: Classification of textitProblem Modeling.

the generation of new algorithms (GNA), which at the design level is implemented in the systems described in Zong et al. (1993); Gazmuri and Maturana (2001); Bonfill et al. (2008). The main shortcoming of this approach is that it is not valid for non-technical users.

The incorporation of algorithms at the user level has had more acceptance within authors, 8 systems (73%). Lamatsch et al. (1988); Angelidis et al. (2011) use specific languages to add new algorithms (LBG), while CEA is adopted by Sauer (1993); Sauer and Bruns (1997); Sauer et al. (1998). RIT can be found in Taunton and Ready (1994), where, instead of pieces of algorithms, users can generate rules that can be added or removed as required. Finally, EG is proposed in T'kindt et al. (2005), where authors developed a complete module to this end, and in Varela et al. (2009) where a web-based platform allowing for the incorporation of new solving methods is provided. Table 4.3 shows the classification.

The second functionality refers to systems able to acquire knowledge from users, which can be done in several ways according to our framework. AHSP is adopted by Sauer (1993); O'Donoghue et al. (1994); Goldman et al. (1996); Goldman and Boddy (1997); Artiba and Aghezzaf (1997). The most extended approach (15 systems out of 27) is via WC, which is implemented in Hsu et al. (1993); Zong et al. (1993); Flower and Cheselka (1994); Weigl (1995); Karacapilidis and Pappis (1996); Wang and Lin (1997); Kuo and Hwang (1998); Madureira et al. (2014a); Figueira et al. (2015); Zheng et al. (2016). A slight

modification of this approach by allowing their application in an interactive manner is adopted in Esquirol et al. (1997); Patkai (1998); de Ugarte et al. (2009). Moreover, Gao and Tang (2008); Barlatt et al. (2012) allow for both types, decision makers can introduce the working conditions before executing algorithms and, after a solution has been found, they can modify it interactively. The possibility of evaluating why orders are late to ease the development of new algorithms (E) can be found in Lamatsch et al. (1988); Tang et al. (2009); Zhang et al. (2010). Finally, allowing the expert to enter new constraints obtained from his/her experience or from his/her knowledge on the field (AHC) is implemented in May and Vargas (1996); Everett (2001); Jacobi et al. (2007); Sotiris et al. (2008); Upton and Quilligan (2014); Mourtzis et al. (2016).

The AR functionality refers to systems able to take into account possible uncertain events that could happen during the normal functioning of the shop. The classification is detailed in Table 4.3. Systems offering this functionality are able to deal with these events in a well-managed manner, i.e. we exclude systems allowing for rescheduling by just obtaining a completely new schedule from the scratch incorporating the changes arisen. For this functionality we classified the different approaches used in the literature as described in Section 4.2. The n.s. (*not specified*) label corresponds to contributions in which authors state that their system allow for rescheduling without giving further information. It is interesting to note that most systems apply just one approach for rescheduling, although in Henning and J. (2000); Chan and Zeng (2003) different approaches are combined.

The last functionality related to problem solving is Multi-Algorithm Scheduling (MA). Almost all systems implementing this functionality (18 out of 19) include generic algorithms. There are also many cases where the system offers different possibilities: this is the case for 7 systems combining generic models with local search and metaheuristics, and other 7 cases where generic systems are combined with specific algorithms. We found one system (T'kindt et al., 2005) with only metaheuristics and local search methods. The second criterion to classify this functionality is the selection mode. Here, we evaluate the degree of assistance of the system. The case where no help is available (i.e. user selection - U), is present in 6 systems (Weigl, 1995; Pinedo and Yen, 1997; Yen, 1997; Concannon et al., 2003; Hindle and Duffin, 2006; T'kindt et al., 2005; Sotiris et al., 2008; Varela et al., 2009). This is the simplest method for offering this functionality and the less useful for decision makers, as they are assumed to know which algorithm fits better for the problem under study. A guided selection (G) where the system gives some hints to decision makers to make a good decision is implemented by Sauer (1993); Wang and Lin (1997); Metaxiotis et al. (2002); Geiger (2005, 2011); T'kindt et al. (2005). The last mechanism is the autonomous selection of the algorithm (A). This mechanism is preferred for those cases where users does not have any technical knowledge. In these cases, the system decides the algorithm instead of supporting its selection. There are several approaches to implement this selection mechanism:

- Experimentation-based (cited in Sauer, 1993; Patkai, 1998; Marinho et al., 1999; Blazewicz et al., 2001; Mourtzis et al., 2016), where several/all algorithms are run so the system offers the solution found by the best one.

- Rule-based (Gupta et al., 2002; Artiba and Aghezzaf, 1997). A set of expert rules are used to analyse the scheduling problem with respect to the optimization criterion, constraints, and search space size, so it is able to select the appropriate algorithm or heuristic stored in the algorithm library.

- Agent-based. Perhaps the most ambitious generic algorithm library is described in Murthy et al. (1999); Keskinocak et al. (2002), where a set of algorithms is embedded by means of three types of agents: constructors (who use both deterministic and randomized generic algorithms to obtain solutions), improvers (who try to improve the current set of solutions modifying or combining existing solutions to create new solutions), and destroyers (who remove bad solutions from the population). In Almeida and Marreiros (2006), a set of different algorithms are also encapsulated in an agent-based architecture, although it is not clear whether they exchange information about their solutions.

Finally, regarding how the selection of the algorithm is made, we distinguish between on-line selection, and off-line selection. Eleven systems use the former, while eight systems use the latter approach, and one work (T'kindt et al., 2005) uses both selection modes. The detailed classification can be seen in Table 4.3.

4.3.3 Solution Evaluation

This type of functionalities are in charge of supporting the decision maker in the evaluation of the scheduling obtained by the system. First, we analyse how the systems consider scheduling objectives. The classification of the systems according to their consideration of objectives is detailed in Figure 4.6. As discussed in Section 4.2, these are classified with respect to 7 different criteria. The most common approach is using a single objective function with a number of weighted objectives (WC), which is found in 13 systems. Another two widely used approaches, with 8 systems found for each, are USM, where decision makers are offered a Pareto set of optimal solutions, thus letting him/her the decision of selecting the best one, and US, which is the most simple way of dealing with this functionality, i.e. the user selects his/her most preferred objective.

With respect to the objectives considered, it has to be noted that, in some systems, the objectives are expressed in terms of managerial objectives (such as maximizing resource usage, or minimizing work in process), whereas in other cases these are expressed in terms of scheduling criteria (such as makespan

Reference	Problem Solving						
	GNA		HE	AR	MA		
	Level	Type			Algorithm Library	Selection Mechanism	Selection Mode
Collinot et al. (1988) Le Pape (1994)				FR/EAO/BDT			
Kerr and Ebsary (1988)				FR			
Lamatsch et al. (1988)	User	LBG					
Hadavi et al. (1990) Hadavi et al. (1992)				CW/SAR			
Hsu et al. (1993) Prietula et al. (1994)			WC				
Sauer (1993)	User	CEA	AHSP	n.s.	G/SA	G/A	OFF
Zong et al. (1993)	Design	n.s.	WC				
Aerts et al. (1994)							
Fargher et al. (1994)				EAO			
Flower and Cheselka (1994)			WC				
Fox (1994)				EAO			
Kempf (1994)					G	A	OFF
O'Donoghue et al. (1994)			AHSP				
Taunton and Ready (1994)	User	RIT	E				
Weigl (1995)			WC		G	U	ON
Goldman et al. (1996) Goldman and Boddy (1997)			AHSP	MCT			
Karacapilidis and Pappis (1996)			WC				
May and Vargas (1996)			AHC				
Artiba and Aghezzaf (1997)			AHSP		G/SA	A	ON
Esquirol et al. (1997)			WC				
Pinedo and Yen (1997) Yen (1997)					G/LSM/SA	U	OFF
Sauer and Bruns (1997)	User	CEA		FR			
Wang and Lin (1997)			WC		G	G	OFF
Kuo and Hwang (1998)			WC				
Patkai (1998)			WC		G/LSM	A	ON
Sauer et al. (1998)	User	CEA					
Marinho et al. (1999)					G	A	ON
Murthy et al. (1999)					G/LSM/SA	A	ON
Henning and J. (2000)				FR/BR/EAO/BDT/MT			
Blazewicz et al. (2001)				EAO	G/SA	A	OFF
Everett (2001)			AHC				
Gazmuri and Maturana (2001)	Design	n.s.					
Musselman (2001)				SAR			
Gupta et al. (2002)				FR			
Keskinocak et al. (2002)					G/LSM/SA	A	ON
Metaxiotis et al. (2002)					G/LSM	G	ON
Chan and Zeng (2003)				FR/BR/BDT/CW/SAR			
Concannon et al. (2003) Hindle and Duffin (2006)					G	U	ON
Cheeseman et al. (2005)				EAO			
Geiger (2005) Geiger (2011)					G/LSM	G	OFF
T'kindt et al. (2005)	User	EG			LSM	U/G	ON/OFF
Almeida and Marreiros (2006)					G	A	OFF
Jacobi et al. (2007)			AHC				
Bonfill et al. (2008)	Design	n.s.					
Gao and Tang (2008)			WC				
Sotiris et al. (2008)			AHC	SAR	G	U	ON
de Ugarte et al. (2009)			WC	EAO			
Tang et al. (2009) Zhang et al. (2010)			E				
Varela et al. (2009)	User	EG			G	U	ON
Yang and Lin (2009)					G/SA	A	OFF
Leung et al. (2010)				CBR			
Angelidis et al. (2011)	User	LBG					
Barlatt et al. (2012)			WC	EAO			
Madureira et al. (2014b) Piairo et al. (2013)				SAR			
Madureira et al. (2014a)			WC	n.s.			
Upton and Quilligan (2014)			AHC				
Figueira et al. (2015)			WC	EAO			
Mourtzis et al. (2016)			AHC	CBR	G	A	ON
Zheng et al. (2016)			WC				

Table 4.3: Classification of *Problem Solving*.

Reference	Solution Evaluation	
	Consideration of Objectives	Analysis of Scenarios
Hadavi et al. (1990) Hadavi et al. (1992)	CA	EA
Niew et al. (1990)	CA	
Boccalatte et al. (1992)		IA
Hsu et al. (1993) Prietula et al. (1994)		IA
Sadeh (1993) Sadeh (1994)	WC	
Sauer (1993)	US	IA
Zong et al. (1993)		IA
Aerts et al. (1994)	WC	EA
Fargher et al. (1994)		EA/IA
Flower and Cheselka (1994)		EA
Marriott (1994)		IA
Weigl (1995)		EA/IA
Karacapilidis and Pappis (1996)		EA/IA
May and Vargas (1996)		EA
Artiba and Aghezzaf (1997)	USO/SE	EA
Pinedo and Yen (1997) Yen (1997)	WC	
Sauer and Bruns (1997)	US	
Patkai (1998)		EA
Vaidyanathan et al. (1998)		IA
Marinho et al. (1999)	LA	EA
Murthy et al. (1999)	USM	EA/IA
Henning and J. (2000)	CA	EA/IA
Blazewicz et al. (2001)	US/SE	
Gazmuri and Maturana (2001)		IA
Kotak et al. (2001)		IA
Musselman (2001)		EA
Gupta et al. (2002)	USO	IA
Keskinocak et al. (2002)	USM	EA/IA
Metaxiotis et al. (2002)	US	
Saydan and Cooper (2002)		IA
Chan and Zeng (2003)	WC	EA
Concannon et al. (2003) Hindle and Duffin (2006)	WC	EA
McKay and Wiers (2003)		EA/IA
Ozbayrak and Bell (2003)	US	
Appelqvist and Lehtonen (2005)	LA	
Geiger (2005) Geiger (2011)	USM	
T'kindt et al. (2005)	USM	IA
Almeida and Marreiros (2006)	WC	
Chan et al. (2006)	WC	
de Castro et al. (2006)	US	
Jacobi et al. (2007)	US	EA
Abreu et al. (2008)	WC	
Bonfill et al. (2008)	WC	
Gao and Tang (2008)	WC	EA
Józefowska and Zimniak (2008)	USM	IA
Sotiris et al. (2008)		EA
de Ugarte et al. (2009)		EA/IA
Missbauer et al. (2009)	WC	
Silva (2009)		EA
Tang et al. (2009) Zhang et al. (2010)		EA/IA
Ko and Wang (2010)	USM	
Angelidis et al. (2011)		EA
Trojet et al. (2011)		EA
Nieuwenhuyse et al. (2011)		EA
Barlatt et al. (2012)	CA	EA
Korosec et al. (2013)	USM	IA
Madureira et al. (2014b) Piairo et al. (2013)	CA	
Madureira et al. (2014a)		EA
Upton and Quilligan (2014)		EA
Figueira et al. (2015)	WC	EA
Guo et al. (2015)	USM/SE	
Fanti et al. (2016)	SE	EA
Mourtzis et al. (2016)	WC	

Table 4.4: Classification of *Solution Evaluation*.

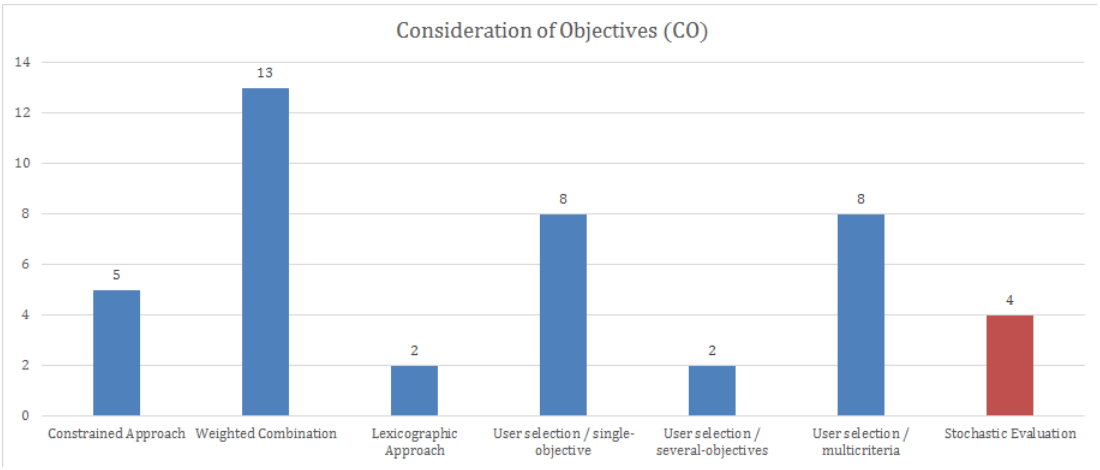

Figure 4.6: Consideration of objectives

minimization, or flowtime minimization). Whenever possible, we accommodate the latter criteria to managerial objectives, as this provides a more model-independent view of the goals pursued by the system. We also do not take into account the possible inter dependencies between objectives/criteria, i.e. it is known that maximizing machine utilization is equivalent, in some cases, to minimize makespan. The results of this classification are summarised in Table 4.5.

In some cases, decision makers prefer evaluating different schedules before deciding which one will be put into practice. For this reason, we analyse the functionality *Analysis of Scenarios* (AS). Around 44% of the systems (44 out of 99) provide this functionality. From the possibilities discussed in Section 4.2, we found that the explicit analysis of scenarios (EA) was the most widely used, being implemented in 22 systems. There were also some of them (9) adopting both types of analysis. The detailed classification is summarised in Table 4.4.

4.3.4 Schedule Presentation

This subsection discusses how the different systems –when this information is provided– show the schedules to decision makers. This classification is summarised in Table 4.6. Moreover, in Table 4.7 the contributions dealing with other presentation types are shown. From this classification we can conclude that even for the most modern systems, classical representations are preferred, i.e. textual representation is used in 53 systems (more than half of the reviewed systems) and Gantt Charts are used in 55. It is also interesting to note that, for many systems, information on the presentation of the schedules is not provided.

Reference	Schedule Presentation			
	Text	Gantt Chart	Job Screen	Other Charts
Collinot et al. (1988) Le Pape (1994)		X		
Kerr and Ebsary (1988)	X	X		

Reference	Schedule Presentation			
	Text	Gantt Chart	Job Screen	Other Charts
Kolen and Woerlee (1988) Speranza and Woerlee (1991)	X	X		X
Lamatsch et al. (1988)	X	X		
Numao and Morishita (1988) Numao and Morishita (1989) Numao and Morishita (1991) Numao (1994)				X
Savell et al. (1989)	X			
Bona et al. (1990)	X	X		
Hadavi et al. (1990) Hadavi et al. (1992)	X			
Niew et al. (1990)				X
Adelsberger and Kanet (1991)		X		
Burke and Prosser (1991)		X		
Boccalatte et al. (1992)	X	X		
Hsu et al. (1993) Prietula et al. (1994)	X	X		
Sadeh (1993) Sadeh (1994)	X	X		
Sauer (1993)		X		
Zong et al. (1993)	X			
Aerts et al. (1994)		X		
Boccalatte et al. (1994)		X		
Fargher et al. (1994)	X	X		
Flower and Cheselka (1994)		X		
Fox (1994)	X	X		
Kempf (1994)		X		
Marriott (1994)		X		X
O'Donoghue et al. (1994)	X			
Taunton and Ready (1994)		X		X
Weigl (1995)	X	X		X
Goldman et al. (1996) Goldman and Boddy (1997)		X		
Karacapilidis and Pappis (1996)	X			
May and Vargas (1996)	X			
Artiba and Aghezzaf (1997)		X		
Esquirol et al. (1997)	X	X		X
Pinedo and Yen (1997) Yen (1997)	X	X	X	X
Sauer and Bruns (1997)	X	X		
Wang and Lin (1997)	X			
Weintraub et al. (1997)	X			X
Kuo and Hwang (1998)	X			
Patkai (1998)		X		
Sauer et al. (1998)	X	X		
Vaidyanathan et al. (1998)	X			
Marinho et al. (1999)		X		
Murthy et al. (1999)	X	X	X	X
Henning and J. (2000)	X	X	X	X
Blazewicz et al. (2001) Blazewicz et al. (2007)	n.s.	n.s.	n.s.	n.s.
Everett (2001)				X
Gazmuri and Maturana (2001)	X			X
Kotak et al. (2001)	X			X
Musselman (2001)		X	X	
Gupta et al. (2002)	n.s.	n.s.	n.s.	n.s.
Keskinocak et al. (2002)				X
Metaxiotis et al. (2002)	X			
Saydan and Cooper (2002)	X			
Chan and Zeng (2003)	n.s.	n.s.	n.s.	n.s.
Concannon et al. (2003) Hindle and Duffin (2006)	X	X		
Feng et al. (2003)	n.s.	n.s.	n.s.	n.s.
McKay and Wiers (2003)	X	X		
Ozbayrak and Bell (2003)	X	X		
Bredstrom et al. (2004)	n.s.	n.s.	n.s.	n.s.
Appelqvist and Lehtonen (2005)		X		X
Cheeseman et al. (2005)		X		
Geiger (2005) Geiger (2011)		X		X
Munawar et al. (2005)		X		
T'kindt et al. (2005)		X	X	
Almeida and Marreiros (2006)	X			

Reference	Schedule Presentation			
	Text	Gantt Chart	Job Screen	Other Charts
Chan et al. (2006)	n.s.	n.s.	n.s.	n.s.
de Castro et al. (2006)	n.s.	n.s.	n.s.	n.s.
Kumar and Rajotia (2006)	n.s.	n.s.	n.s.	n.s.
Jacobi et al. (2007)	X	X		
Abreu et al. (2008)	X			
Bonfill et al. (2008)		X		
Gao and Tang (2008)	n.s.	n.s.	n.s.	n.s.
Józefowska and Zimniak (2008)	X	X		
Sotiris et al. (2008)	X	X		
de Ugarte et al. (2009)	X			X
Lopez and Villar (2009)	X			
Missbauer et al. (2009)	X			
Silva (2009)	X	X		
Stevenson et al. (2009)		X	X	
Tang et al. (2009) Zhang et al. (2010)				X
Varela et al. (2009)	X	X		
Yang and Lin (2009)	n.s.	n.s.	n.s.	n.s.
Ko and Wang (2010)	X			
Leung et al. (2010)	n.s.	n.s.	n.s.	n.s.
Maturana et al. (2010)	n.s.	n.s.	n.s.	n.s.
Sadi-Nezhad and Darian (2010)	X			X
Angelidis et al. (2011)				
Trojet et al. (2011)	X			X
Nieuwenhuyse et al. (2011)	X			
Zhao and Lin (2011)	n.s.	n.s.	n.s.	n.s.
Zhongyi et al. (2011)	X	X		
Barlatt et al. (2012)		X		
Korosec et al. (2013)	X	X		
Madureira et al. (2014b) Piairo et al. (2013)		X		
Madureira et al. (2014a)	X	X		
Upton and Quilligan (2014)	X	X	X	
Figueira et al. (2015)	X	X		X
Guo et al. (2015)		X	X	X
Fanti et al. (2016)	X	X		
Mourtzis et al. (2016)		X		
Zheng et al. (2016)				X

Table 4.6: Classification of *Schedule Presentation*.

4.4 Main Findings

To gain insights about what is available in literature regarding manufacturing DSSTS, we performed a systematic review of contributions on this topic and proposed a classification framework to classify their functionalities. A number of findings can be extracted from the review:

1. Regarding the integration of scheduling and related decisions, there seem to be a relatively high autonomy of scheduling decisions with respect to other issues in production management, save that of control, which is often seen as the way to *close the loop* of the manufacturing DSSTS.

2. Most systems are designed for a specific scheduling model. Whenever several scheduling models are available, the system offers some guide to the user to choose the specific model, usually in the form of family trees. In contrast, there are several works handling some form of constraint abstraction, particularly regarding the acknowledgement of *soft* and *hard* constraints. Since in

Reference	Feasibility	Resource Usage	Makespan	Inventory	Cycle Time	Meeting Deadlines /Due Dates	Set-up Time Reduction	Production Cost	Quality Related	Others (Problem Specific)
Collinot et al. (1988)	X	X				X				
Le Pape (1994)										
Kerr and Ebsary (1988)	X	X								
Kolen and Woerlee (1988)	X	X	X			X				
Speranza and Woerlee (1991)										
Lamatsch et al. (1988)	X									
Numao and Morishita (1988)										
Numao and Morishita (1989)	X	X					X			
Numao and Morishita (1991)										
Numao (1994)										
Hadavi et al. (1990)				X	X					
Hadavi et al. (1992)	X	X								
New et al. (1990)	X	X				X				X
Burke and Prosser (1991)	X									
Boccalatte et al. (1992)	X					X				
Hsu et al. (1993)				X					X	
Prietula et al. (1994)										
Sadeh (1993)				X						
Sadeh (1994)	X	X		X		X				
Sauer (1993)		X				X	X			
Zong et al. (1993)	X		X							
Aerts et al. (1994)	X									
Boccalatte et al. (1994)	X	X				X				X
Fargher et al. (1994)									X	X
Flower and Cheselka (1994)										
Fox (1994)				X		X		X	X	X
Marriott (1994)	X									
O'Donoghue et al. (1994)		X								X
Taunton and Ready (1994)				X		X				X
Weigl (1995)		X	X							
Goldman et al. (1996)	X									
Karacapilidis and Pappis (1996)		X				X	X			
May and Vargas (1996)				X		X				
Artiba and Aghezzaf (1997)			X			X	X			
Esquirol et al. (1997)	X									
Wang and Lin (1997)			X	X	X	X		X		
Weintraub et al. (1997)						X				
Kuo and Hwang (1998)						X	X			X
Patkai (1998)	X									
Vaidyanathan et al. (1998)						X				
Marinho et al. (1999)		X				X				
Murthy et al. (1999)						X				
Henning and J. (2000)	X									X
Everett (2001)										X
Musselman (2001)				X		X	X			
Gupta et al. (2002)	X						X			X
Keskinocak et al. (2002)						X				X
Saydam and Cooper (2002)		X		X						X
Chan and Zeng (2003)						X				
Concannon et al. (2003)				X		X	X			
Hindle and Duffin (2006)										
Feng et al. (2003)			X							
McKay and Wiers (2003)	X									
Ozbayrak and Bell (2003)			X	X	X					
Bredstrom et al. (2004)							X	X		X
Appelqvist and Lehtonen (2005)		X				X				X
Cheeseman et al. (2005)	X									X
Geiger (2005)			X		X	X				
Geiger (2011)										
Munawar et al. (2005)				X		X		X		X
Almeida and Marreiros (2006)			X		X	X				
Chan et al. (2006)		X			X	X				
de Castro et al. (2006)										X
Kumar and Rajotia (2006)	X	X								
Jacobi et al. (2007)	X				X					
Abreu et al. (2008)				X		X	X	X	X	X
Bonfill et al. (2008)				X	X	X				X
Gao and Tang (2008)						X	X	X		X
Józefowska and Zimniak (2008)		X		X			X	X		X
Sotiris et al. (2008)	X	X	X		X	X	X			X
de Ugarte et al. (2009)	X									X
Lopez and Villar (2009)	X									
Missbauer et al. (2009)			X							
Silva (2009)						X				X
Stevenson et al. (2009)	X					X				
Tang et al. (2009)		X		X			X			X
Zhang et al. (2010)	X									
Yang and Lin (2009)			X							
Ko and Wang (2010)			X	X		X				
Leung et al. (2010)			X							X
Maturana et al. (2010)				X						X
Sadr-Nezhad and Darian (2010)				X			X	X		X
Trojet et al. (2011)	X		X			X				
Nieuwenhuyse et al. (2011)					X					
Barlatt et al. (2012)	X					X		X		
Korosec et al. (2013)		X				X	X			
Madureira et al. (2014b)	X	X	X							X
Piairo et al. (2013)										
Figueira et al. (2015)						X		X		X
Guo et al. (2015)	X	X			X	X				
Fanti et al. (2016)	X		X							
Mourtzis et al. (2016)					X	X		X	X	
Zheng et al. (2016)	X									

Table 4.5: Objectives considered in the reviewed manufacturing DSSTS.

Reference	Chart	Description
Kolen and Woerlee (1988) Speranza and Woerlee (1991)	Flow Chart	It shows the problem instance.
Numao and Morishita (1988) Numao and Morishita (1989) Numao and Morishita (1991) Numao (1994)	Diagrammatic View of the Scheduling	It allows user to see the production flow and machine utilization at the same time.
Niew et al. (1990)	Spreadsheet-like form	It is similar to Excel. The user can manually edit the cells.
Marriott (1994)	Simulation Interaction	It allows the user to see the status of the different work stations and the manufacturing lots.
Taunton and Ready (1994)	Simulation Interaction	Schematic View of the Plant Layout for Simulation.
Weigl (1995)	Manning Level Charts	It shows the manning levels for a certain order sequence.
Esquirol et al. (1997)	Placement Charts	The system shows how the different metal sheets are filled (with parts).
Pinedo and Yen (1997) Yen (1997)	Capacity Bucket Interface	It shows the load of each machine.
Weintraub et al. (1997)	Priority List of Jobs for each machine (similar to Capacity Bucket) /Chart of the utilization of each machine within a cell	The statistics and model information are presented at three levels: machine (most detailed), management and factory.
Murthy et al. (1999)	Detailed shcedule of each machine (with the jobs and their priority in that machine)	It shows the load of each machine.
Henning and J. (2000)	Evolution of Storage Devices	The system displays the evolution of storage devices along the planning horizon through line plots.
Everett (2001)	Macro-driven Spreadsheets	Not described in the paper.
Gazmuri and Maturana (2001)	User defined reports (can be exported to spreadsheets)	The system provides means for the user to customize the reports he /she wants to obtain from the system.
Kotak et al. (2001)	Different interfaces for ripsaws and for the chop line	For the former the system includes loads to be used, their sequence and the priority of individual ripping. For the latter components to be produced, their kicker assignment and the production sequence.
Keskinocak et al. (2002)	GUI for view and modify solutions	Authors do not give any deeper detail about their user interface.
Appelqvist and Lehtonen (2005)	Other non detailed charts and reports	Some of them are utilization rate for each workstation, order backlog, task lists in each workstation or a summary report.
Geiger (2005) Geiger (2011)	Pareto frontier according to objectives	The system allows for defining certain aspiration levels for each objective and it represents a Pareto frontier with all these objectives highlighting those achieving the selected aspiration levels.
de Ugarte et al. (2009)	Chart for hot rolling	In this chart, the sequence and the width of each slab (height of rectangles proportional to the slab width) are shown together. It also allows the user for seeing where the schedule has problems.
Tang et al. (2009) Zhang et al. (2010)	Loading of resources	This chart shows graphically the loading of each furnace.
Sadi-Nezhad and Darian (2010)	Excel spreadsheet	The system uses Excel to connect to Lingo solver and retrieve the results. There are not much more details about the schedule presentation.
Trojet et al. (2011)	Cumulative curves for each resource	These charts show the evolution of resource consumption over time in each resource.
Figueira et al. (2015)	Excel spreadsheet	The system provides different information and charts about the performance of the schedule (e.g. evolution of the stock, paper campaigns, production rate, etc.).
Guo et al. (2015)	Production detail of orders in an order group	Apart from the job screens where the system details the operations in an order, it also presents the details of the order groups.
Zheng et al. (2016)	Similar to a dashboard to manage the schedule	The user has access to all data involved in scheduling (including data from past schedules). He/she can also receive information about the most crucial factors influencing the scheduling.

Table 4.7: Details about *Other Charts*.

practice, many constraints are soft, perhaps the interest lies in identifying those hard constraints and then provide the user with a system to handle the soft constraints. In addition, there would be what we could call *hidden* constraints (i.e. constraints that do not appear during the development of the manufacturing DSSTS, but are considered by users nevertheless). To handle these constraints, the user's manipulation of the schedules provided by the system (ideally via an interactive interface) is required.

More work is needed regarding to model detection abilities. We see this issue as critical since the vast majority of developments in classical scheduling assume the existence of a model (particularly a layout) upon which specific solution procedures are developed. The need of identifying the underlying models from the raw data becomes clear if all this enormous research effort has to be put into practice. Otherwise, developments in manufacturing DSSTS would be difficult to be translated from one implementation to another, thus leaving only generic methods to be applied in generic manufacturing DSSTS.

3. The incorporation of human expertise into the manufacturing DSSTS for manufacturing scheduling has been always controversial, and the limited number of systems reporting this functionality are mostly confined to a sort of *parametrization* of the system rather than to a pure knowledge-based system. Therefore, the possibility of incorporation of the scheduler knowledge should be done in a more subtle way by leaving the system more open and flexible to integrate constraints and objectives, i.e. as discussed before it is preferable to support the decision maker better than replacing him/her. More specifically, perhaps it would be interesting to discuss the argument the other way round, i.e. to make the suggestions of the system more transparent to the scheduler so he/she can learn from the system. In this regard, we miss systems in which the estimation of different key data that must be entered by the scheduler –such as e.g. processing times, set up times, etc.– are suggested/contrasted against the real data obtained from control, so the scheduler can produce better estimates in the future. This may be, undoubtedly, an area in which research results could be easily translated into practice.

4. Despite the fact that most effort has been devoted to algorithm development, it has been difficult to exploit this knowledge by integrating them in the manufacturing DSSTS under review. Most systems dealing with the integration of new algorithms perform this task at the design level. The few of them dealing with it at user level just integrate dispatching rules, which results in relatively poor- performance algorithm. However, this may not be as grave as it seems, given the fact that many systems suffice with finding feasible solutions and that, as mentioned before, the users may

be permitted to worsen the solutions provided by the system in order to cope with *hidden* constraints.

Part of the problems regarding the application of algorithms could be tackled by properly addressing model detection abilities, giving yet another reason for prioritizing this area of research. Another avenue could be the integration of more sophisticated, generic algorithms. Indeed, it is well-known that many local search approaches work well for different scheduling problems (see e.g. the exceptional performance of the iterated greedy algorithm Ruiz and Stützle, 2007 for rather diverse problems and objectives), so perhaps it may be possible to develop systems for designing this type of algorithms at user level.

5. The concept of a library of algorithms is not considered in most manufacturing DSSTS. Most classical literature on scheduling has focused on devising *champion* algorithms that outperforms existing ones. However, most of the times, the performance of these algorithms is instance-dependent, therefore the idea of a library of algorithms –particularly fast ones– may be of special interest. Up to now, the few works dealing with several algorithms in classical scheduling research simply consider running a set of algorithms for a problem instance, and then taking the best result. However, as Murthy et al. (1999) show, there may be high benefits in devising a cooperation among algorithms to find the best solutions.

6. Regarding the objectives usually considered in the manufacturing DSSTS, it is interesting to note that the most frequent (roughly half of the systems reviewed) are those related to due dates fulfilment. The objectives classification and their appearances can be seen in Figure 4.7. An important number of systems consider problem-specific objectives, and many of these systems simply seek feasible schedules. Makespan –by large the most employed objective in scheduling research– is not among the principal objectives considered in the manufacturing DSSTS reviewed. Such pre eminence of due date -oriented objectives is probably a reflection of the prevalence of a customer-oriented view in manufacturing rather than to a cost-oriented view. In this regard, it would be interesting to further explore the interface between scheduling and order fulfilment.

7. Very few systems consider the stochastic evaluation of deterministic solutions, which is in contrast with the common belief that stochasticity is present in most shop floors. Possibly the problems addressed by the systems are sufficiently complex so a deterministic setting can be considered a first-order approximation of the solution, but still is surprising that no further checking of these deterministic solutions is carried out. We also believe that another cause may be in the lack of

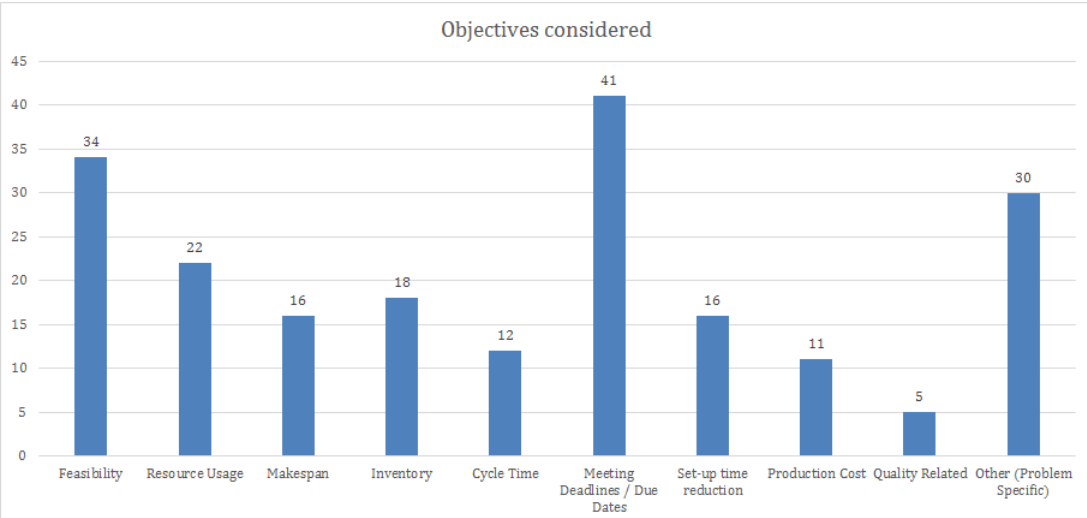

Figure 4.7: Objectives considered

reliable data to perform such checking, therefore a promising research avenue could be to investigate solution procedures robust or at least not very sensitive to variations in the deterministic inputs.

8. Rescheduling seems to be considered, in many cases, an integral part of a manufacturing DSSTS. However, most algorithms devised for scheduling are extremely simple (e.g. forward scheduling), or are driven by user judgement (e.g. exclusion of affected operations, or changes in the priority of the works). While useful, these options do not fully exploit the possibilities of rescheduling, particularly taking into account the advances in algorithms in this field.

9. The *user interface* of the manufacturing DSSTS reviewed does not have an homogeneous form for presenting and manipulating results. Although it does not seem to be the most important part of a manufacturing DSSTS, in many situations it is the factor determining if the system is going to be used or not. Therefore we think that making progresses in obtaining an homogeneous interactive and easy to use user interface constitutes another opportunity for research. Furthermore, devising sophisticated user interfaces may pay off as a way to allow Decision Makers to handle the aforementioned *hidden* constraints.

4.5 Conclusions

A number of interesting conclusions can be extracted from the findings of the previous section in order to analyse the alignment of the framework proposed in Chapter 3 with existing manufacturing DSSTS described in the literature. To discuss the conclusions we follow the same order as in the previous section:

1. Most of the systems reviewed deal with scheduling in isolation without considering upper or lower managerial levels. As already discussed in Chapter 2, the synchronisation between managerial levels was required in order to consider the organisational context of scheduling, therefore we can conclude that most existing manufacturing DSSTS do not consider this issue properly. The proposed framework considers this issue by taking into account planning, scheduling and control.

2. In line with the implementation problems discussed in Chapter 3, a DSSTS should offer a number of different decision models and solving approaches in order to deal with different situations. The reviewed systems usually consider only one specific scheduling model, therefore we can also conclude that current manufacturing DSSTS do not usually address this issue. The proposed framework considers this issue by offering the Model Database component that contains a set of different decision models and the *Solution Approaches Library* component that contains a number of solution approaches for the different decision problems.

3. The third finding is related with the incorporation of human expertise, as most manufacturing DSSTS do not typically consider this possibility, which makes difficult the support of activities different to scheduling, and also makes difficult the adaptation of the manufacturing DSSTS to different contexts, as the scheduler does not have any flexibility in the case the scheduling process change unexpectedly. In the proposed framework, this issue is address within the *Database Management Module* and the *Model Management Module*. The former allows the scheduler to manipulate the decision model and the data related to the scheduling process, and the latter allows him/her to manipulate the solution approaches.

4. Most manufacturing DSSTS reviewed do not allow the inclusion of new algorithms into the system, or allow it in a very basic form. The conclusion related to this finding is similar to the previous one, as if the possibility of adding new decision models or solution procedures is that poor, the system is eager to get obsolete in the short/medium term. The proposed framework allows for the inclusion of new algorithms via the *Solution Approaches Library* component.

5. The concept of library of algorithms serves to provide flexibility to the system when dealing with different situations. A vast majority of the systems found in the literature do not include this functionality, so they stick to a single solution procedure that does not allow the user any interaction with the system related to the generation of solutions. The proposed framework considers this library of algorithms in the *Solution Approaches Library* component.

6. Most common objectives used in practice differ from the most common objectives studied in litera-

ture, which emphasizes the aforementioned gap between theory and practice. This problem could be also solved with the main concept of the previous conclusions, i.e. flexibility. It should be possible to offer the scheduler with a number of objectives that he/she could apply according to his/her interests. This issue is addressed by the proposed framework in the *Model Database* component.

7. Uncertainty is a topic that is not properly addressed by the reviewed systems. Most of them consider data as known in advance and not changing through time. However, this topic is referred to one of the human activities that a DSSTS should support as discussed in Chapter 2.4. This issue can be tackled in the proposed framework in two ways: using *what-if scenarios* to analyse the scheduling under different conditions (using the *Scenarios Management* component and the *Scenarios Handling* component) or considering stochasticity in the decision models (using the *Data Analysis* component and *Analysis Tools* component).

8. Most systems reviewed do not consider rescheduling or consider it in a very limited way. As we discussed in Chapter 2, rescheduling, as part of the scheduling process, should be adequatedly supported when designing a DSSTS. This finding highlights the neglection of the scheduling process in existing manufacturing DSSTS already identified in Chapter 2. This issue is considered explicitly in the proposed framework in the *Solution Approaches Library* component.

9. The design of the user interface for a DSSTS should be as standard as possible and should be adapted to the requirements of the scheduler. The only way to achieve it is by designing it in a modular way that permits the update and interchangeability of the different parts of the user interface. The conclusion of this finding is that this usually does not happen in existing DSSTS. The proposed framework considers this issue in the *User Dialogue Management Module*.

As a summary, we can say that, in general, the systems reviewed in this chapter suffer from the same problems that were discussed in Chapter 2, which validates the set of guidelines commented at the end of that chapter. Therefore, since the framework proposed in Chapter 3 was designed in order to tackle these problems, we can conclude that implementing DSSTS for manufacturing following the proposed framework would help in succesfully implementing them.

Chapter 5

Operating Room DSSTS: a Review on Commercial Systems

As opposed to the case of manufacturing, the literature on DSSs for operating room (OR) scheduling is not so wide. We tried to carry out a similar review as in the previous chapter, but it was not possible due to the lack of contributions. Nevertheless, in order to check the alignment of the proposed framework with the operating room DSSTS used in practice, in this chapter we carry out a review on commercial operating room DSSTS. We found similar problems to those with the commercial manufacturing DSSTS. In addition, there are few commercial software systems supporting OR management, and these hardly ever go beyond the functionalities of multi-agenda management. While major attention has been given on helping doctors to make good diagnoses by using computer systems, (by e.g. Clinical Decision Support Systems or Physician Order Entry, see Garg et al., 2005; Johannsen, 1994; Kaushal et al., 2003), the adoption of such systems for the surgical planning process itself has not been so common. Nevertheless, some interesting insights could be extracted from the review. To show them, in Section 5.1 we first analyse the methodology that was followed to select the systems to review. Next, Section 5.2 describe the classification criteria we used for classifying the systems, that briefly differs from the criteria used in the previous chapter due mainly to the different contexts where the operating room DSSTS are applied, and the difference between the information given for commercial and literature operatingn room DSSTS. In Section 5.3 we present the classification of the systems together and, finally, in Section 5.4 the main findings of the review are discussed.

5.1 Review methodology

As commented in the previous section, most of the systems that can be found in the field of OR management are not inteded for solving the OR scheduling problem. However, some of them include modules that can help decision makers in this task. Therefore, in this review we focus both, on those systems that constitute independent systems whose main functionality is the scheduling of operating rooms, and on those who offer modules for supporting the operating room scheduling problem.

The process of identifying and selecting systems for the review is split into three different parts as follows:

1. Stage I. Reference search. In this stage we followed the search methodology proposed in Meneses et al. (2005), selecting automatized search engines as our aim is to obtain the largest accuracy in available contents. We used www.google.com to perform our search, as it is one of the best performing search engines for medical information as stated in Wang et al. (2012). Our search strategy was (("operating room" OR "surgical") AND "scheduling") AND "software", that it is opened enough to consider a number of different designations for the systems we are searching. From a total of 826000 results, we focused on the most relevant results (210). They were carefully inspected and filtered according to their title and their content summary. This first filtering offers a total of 32 systems. The entries for these systems were later analysed more in detail. This analysis took us to discard 9 more systems.

2. Stage II. Contact with developers. In this stage we tried to get in contact with the developers of the detected systems. In order to do this, we sent them an email explaining the aim of the study we were performing and requesting and evaluation version of the system or some extra information to look into the characteristics of the systems. A set of 17 requests were sent, as we already had information for the other systems. Only 6 replies were obtained, what allowed us to discard another three systems that did not offer the operating room scheduling capability. For those developers that did not respond, we classified the systems according to the available information.

3. Stage III. Detailed Analysis. With the information gathered from the previous stages, we make an in-depth analysis to determine which systems should be considered for the final review. In this stage, 3 more system were discarded as they were not aligned with the topic of the review. Therefore, we assume a total of 17 systems in our study.

In Figure 5.1 we show a summary of the whole process we follow to choose the systems to review.

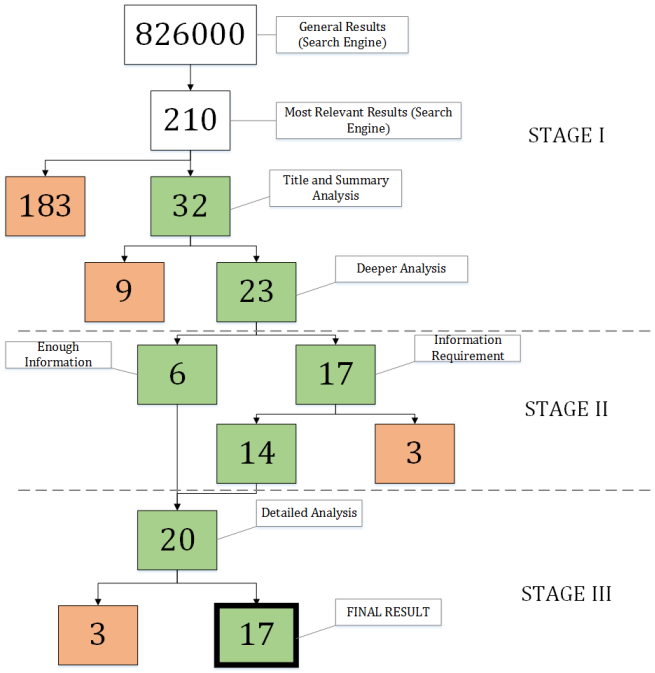

Figure 5.1: Search results for OR scheduling systems

Table 5.1 shows a summary of the reviewed systems.

5.2 Classification Criteria

In this section we will list and describe the criteria established for classifying the systems. These criteria are grouped according to the structure of a DSS followed to describe the functionalities of the framework proposed in Chapter 3, i.e. database management module, model management module and dialogue management module. We also include an *Additional Features* category.

	Company	System	Link
1	Healthworcs	ORWORCS	http://www.healthworcs.com
2	MCKESSON	Pathways Healthcare Scheduling	http://www.mckesson.com
3	M-Soft Inc.	O.R. Essentials	http://www.msoftcorp.com
4	PICIS	Intelligent Perioperative Suite	http://www.picis.com
5	Adjuvant	CALL SCHEDULER	http://www.call-scheduler.com
6	UMS	DIGISTAT Smart Scheduler	http://www.unitedms.com
7	Valen Computer Healthcare	GOWIN QUI	http://www.valen.es
8	MEDITECH	Surgical Services	https://ehr.meditech.com
9	GE Healthcare	Centricity Perioperative	http://www3.gehealthcare.com
10	Healthcare Control Systems	OR Control	http://www.hcs.us.com
11	SURGIMATE	Surgimate on Schedule	http://www.surgimate.com
12	Leonardo MD	Medical Scheduling Software	http://www.leonardomd.com
13	Meridian Health Informatics	OTIS	http://www.meridianhi.com
14	ePreop	Surgical Valet	https://www.epreop.com
15	MAX Systems Inc.	MAX GOLD	http://www.max-gold.com
16	TheraManager	Practice Management	http://www.theramanager.com
17	CosmetiSuite	Surgical Scheduler	http://www.cosmetisuite.com

Table 5.1: Reviewed Operating Room DSSTS.

5.2.1 Database Management Module

The first subject examines the aspects related to the database of the system, from its access to how each software system interacts with it. Data types supported by each system (interventions, resources, etc.) are also analysed. The following features are considered in this module:

- Remote Access. In this field we analyse if the system can be accessed remotely. This feature makes the system more flexible as it does no require users to be in a determined location. It allows, for example surgeons, to access the scheduling at any time to search for details of the patients they have to intervene (clinical information) or the interventions they have to perform (timetable, assigned operating room, etc.). On the contrary, for systems not offering this feature, once the planner has carried out the scheduling, he/she must distribute it to the different memebers of the personnel in a traditional way, such as email or operating room timetables.

- Profiles. In this field we analyse the different roles adopted by the healthcare stakeholders and the importance of DPA (Data Protection Act) in this domain. To this end, it is necessary to restrict user access to certain data. Thus, in this field we analyse how the systems provide access to the system using different profiles, granting appropriate rights to different users of the system.

- Database Integration. The level of intergration of the database with the Hospital Information Systems (HISs) is analysed. According to this, we find *proprietary* databases (P), i.e. those databases made specifically for the system and are hardly expandable to the rest of the HIS, and integrated databases (I), which adapt to the existing database of the HIS and in some cases extend it. In systems with a propietary database, changes made in data do not have any impact in any other system of the hospital, making difficult to keep the data up to date. Therefore, integration of the database with the HIS is a very desirable property since it allows for the interoperability of all the systems, granting data unicity. In some systems, we find a hybrid solution of these two approaches (P/I), i.e. they are adapted to the database of the HIS but also incorporate a new database for specific purposes related with the system.

- Patients Data. Availability of information about patients is considered in this field. Systems providing this information generally allow for the storage and modification of both demographic and clinical patient information, and also make possible the establishment of constraints and alerts once the scheduling have to be done. An example of demographic information utility is the possibility of avoiding the scheduling of patients that live far away from the hospital early in the morning. On the other hand, regarding clinical information, it is possible, for example, to consider allergies

of the patients to avoid their scheduling in operating rooms not correctly sterilised. Moreover, it is important to perfectly define each patient through an identification number and to specify the intervention each patient has to receive. This mapping between patients and interventions relates this field with the next one. We analyse if the reviewed systems consider this type of information (Yes) or not (No).

- Interventions Data. This field refers to the required data for scheduling interventions. This data includes information such as the intervention duration, the due date for the patient, the intervention code (e.g. ICE-9), etc. Additionally, information about material resources needed to carry out the intervention, or information about required human resources, is also considered. This allows the system to check if all the required resources are available before scheduling the intervention. We classify the systems according to the consideration (Yes) or not (No) of this type of data.

- Human Resources Data. In the previous field, information about the amount of resources, both human and physical, needed to carry out an intervention were considered. In this field, we take into account data about human resources, i.e. data about hospital staff. This data allows for analysing the different healthcare stakeholders separately. However, almost every system identified only considers surgeons, so to make a clearer classification, we grouped them together in a single field showing if the systems consider human resources (Yes) or not (No).

- Physical resources Data. This field is similar to the previous one, but considering physical resources instead of human resources. We detected that there is no common consideration of the resources, so for the classification we assume, operating rooms (OR) and postoperative rooms (PR), including the material and devices used in them, e.g. gauzes, surgical knives, bands, etc. The most important physical resource is the operating room, as it is where patient is intervened and is one of the most limited resources, becoming a bottleneck in many situations. In most cases, operating rooms are allocated to specialties depending on the interventions types that can be executed in them. Moreover, there are some devices that, due to their cost, are scarce and restrict the number of interventions of a certain type that can be done in parallel. It is easy to realize the importance of considering as many operating room characteristics as possible, to make the scheduling process as similar to the real process as possible. In addition to operating rooms, we take into account postoperative rooms. This resource has a great importance in scheduling as it can block the normal performance of the operating room, i.e. if there is no availability of these postoperative rooms, it is not possible to continue executing interventions until some of them are released. As we commented in operating rooms, it is also important to consider available devices and materials in each room as it is not

possible to treat every patient in every room. Finally, we highlight that preoperative rooms have not been considered because there is no system facing this problem. This is because in the systems, it is assumed that patient entry has been correctly managed in hospital strategic planning.

5.2.2 Model Management Module

We could consider this subject as the core of OR scheduling systems since this is the module in charge of obtaining the solutions that will be given to users. It is important to remark that, despite trying to contact system developers, there are some fields that could not been properly defined due to the opacity they showed in regard to the internal functioning of the systems. We consider the following features within this module:

- Manual scheduling. Systems offering this feature allow user to move interventions from one "operating room/day" to another just clicking and dragging them (technique commonly known as Drag & Drop). This type of scheduling does not make use of any optimisation strategy; it is just based on planner knowledge and experience. This field is closely related to "conflict management" field, in the sense that when an intervention is moved, it is possible that conflicts with surgeons, time, etc. appear (Framinan and Ruiz, 2010), so it is necessary to manage them in an automatic way or through visual inspection (which is inadvisable). We will see in Table 3 which systems offer this feature (Yes) and which not (No).

- Supported scheduling. Unlike previous field, systems providing this kind of scheduling, sort interventions automatically, acording to some criteria. Furthermore, certain systems offer the possibility to apply heuristics to reach an improved scheduling. This was the most difficult aspect to achieve from developers as we commented before. In the overall classification we will show systems offering this feature (Yes) and not (No). Further information on this topic can be found both in bibliography from "Block scheduling" field, and in Augusto et al. (2010); Fei et al. (2010); Liu et al. (2011); Roland et al. (2010) where an "Open scheduling" strategy is assumed.

- Block Scheduling. This item considers a typical OR scheduling property. This property is concerned with the possibility of limiting days and operating rooms to certain specialities, i.e. authorizing specialities to work only in specific operating rooms and days, e.g. paediatric surgery could be limited to operating rooms two and three on Mondays and Thursdays. As we said in previous field, the alternative to this property is "open scheduling" which basically consists on scheduling interventions in any operating room according to their time of arrival. Additional information about this topic

can be found in Belien et al. (2009); Chaabane et al. (2008); Fei et al. (2006); Kharraja et al. (2006); Santibanez et al. (2007); Testi et al. (2007).

- Conflicts management. In this field we analyse which systems provide interventions conflict management. According to that, several forms of achieving it could be identified, from the simplest one that allow the planner to move interventions without taking into account if there exists an overlap, to those which offer procedures to automatically detect problems when interventions location is modified. We classify the most common procedures: the highlight of conflicts (H) and the generation of conflicts lists (L).

- Rescheduling. In this review we define rescheduling as the generation of a new scheduling based on a previous conducted scheduling, and subject to some additional constraints, e.g. some interventions are fixed to an operating room and to a specific day, it is not possible to make use of a particular operating room, etc. Generally, rescheduling is employed when an irregular behaviour takes place or a disturbance arises, whether in the planning horizon (the patient cannot be intervened a specific day, etc.) or in human (surgeon illness, etc.) or physical resources (unavailability of certain device, etc.). As an example, we could consider a surgical block where a one week planning horizon scheduling has been carried out (from Monday to Friday). If a large amount of urgent patients arrives to hospital at the same time, it will cause that every available operating room has to be allocated to urgencies, making impossible to execute the plan previously scheduled. Assuming this context, the solution is rescheduling, i.e. starting from the schedule done before, first fix already executed interventions, then restrict operating rooms occupied by urgencies and finally generate a new plan using only the available resources.

5.2.3 Dialogue Management Module

The final component considered in Sprague (1980) deals with the interactivity of the system. It aims at offering information, both from the database and related to the obtained solutions, to users and to allow them to interact with the system, by entering new data or preferences. In this section we analyse how the systems offer these features. We also include two features identified from the studied systems that allow for the communication between actors of the scheduling process: “Notifications” and “White Board”. The next features are considered in this module:

- Textual representation. In some cases, apart from charts and other graphical artifacts, we found a system offering the possibility to obtain an operating room scheduling avoiding any kind of graphical

processing which involves an increase in solution processing speed. An advantage of this representation could be the use of the system on mobile devices, e.g. PDAs or smartphones, which normally boast limited processor capacity.

- Graphical interface. Assuming that every system offers a graphical interface (most of the times Gantt Charts or similar), in this field we make a remark about how they offer it. Here we show temporal granularity of the graphical interface of systems, i.e. we point out one or more of the three identified possibilities: daily (D), weekly (W) or monthly (M).

- Notifications. This field displays any kind of notification from system to users, and also, notifications between different stakeholders through the system. These notifications could be made inside the system, through messages shown to users in the system interface, or through mailing managed by the system. An example is the automatic mail delivery with the new scheduling each time a new schedule is performed and accepted. We only consider notifications among healthcare personnel registered in the system. Although it could be interesting, we do not take into account notifications with patients or other kind of role not registered in the system.

- White board. To promote collaboration among system users, some systems provide a "white board", i.e. an available virtual space where users can access and introduce information that becomes visible for the rest of the users or a limited set of them, depending on the preferences defined by the user who introduces the information. An example is the negotiation of surgeons' agendas, where each surgeon suggests his own agenda and after supervision by the person in charge, reach an agreement that allows an efficient usage of available resources. This feature could be considered as an improvement of the traditional way of communication used in the surgical theatre, where there generally exists a place where healthcare stakeholders can annotate information that could be useful for the rest of the staff. We can find some references about this type of communication in Bahlman and C. (2005); Iaconetti et al. (2004); Xiao et al. (2008).

- Reporting. This field comprises every issue related to obtaining documents and reports about any of the system abilities previously commented, from documents containing a detailed scheduling for certain planning horizon to reports describing patient associated costs. In this field we only consider the generation of exportable documents (pdf, doc, txt, etc.) leaving out information offered by the graphical interface.

- Statistics procurement. In this field we point out systems that are able to, from the database information and the scheduling results, obtain useful statistic data both for scheduling ORs in

a more precise way, e.g. finding out mean values of the duration of interventions to avoid the requirement of introducing each intervention duration, and for providing information about the efficiency of scheduling, e.g. percentage of OR time used after the execution of the scheduling.

5.2.4 Additional features

In addition to previous categories, we identified some more interesting features that could not be classified as any of them. These features are shown below.

- Costs. This field evaluates the ability of systems to manage intervention related costs. This feature could be considered in database category due to its data intensive orientation, but because of this data needs some processing, we decided to consider it separately. The classification of costs considered is the following:

  1. Activities cost (A). Costs related to care provided by healthcare personnel during surgical process, i.e. any care received by patient from his entry to his leaving.
  2. Consumables cost (C). Costs related to consumables used during surgical process.
  3. Stay cost (S). Costs related to patient stay in hospital, which depend on the specific specialty and the spent time.

- System integration. In this last field we look into how is the implementation of the system, and we also consider if it is possible to include new features to it. Mainly we found the two possibilities described below:

  1. Independently developed systems whose only aim is OR scheduling. Typically it is difficult to include new features to this type of systems.
  2. Systems developed as a module of a more general system, that includes modules with other functionalities, such as EMR (Electronic Medical Records), CSCW (Computer Supported Collaborative Work), etc.

5.3 Classification and Analysis of DSSTS

In this section we present the review and classification of the 17 systems selected in Section 5.1 according to the classification criteria from Section 5.2. Tables 5.2, 5.3 and 5.4 show the results that are discussed in the next subsections. Note that due to the difficulties in gathering data from the systems there are some fields that could not be classified. These fields appear in the classification as n.s. (not specified). It

is also important to remark that according to the same reason, the classification is made based on all the available data published on the internet, not only that by the official web site, thus in case of any doubt we refer the reader to check the data offered by developers (see Table 5.1).

5.3.1 Database Management Module

The first two fields of this module are quite related as they explain how is the access to the systems. If we focus on the first field (*Remote Access*) we discover that almost two thirds of the systems (58.8%) offers the possibility of accessing the scheduling remotely. This is specially important to reduce changeover times of the personnel working in the operating rooms and it is even more important in big hospitals where the operating rooms are not located close to each other. Despite this importance, there are also 6 systems (35.3%) that presents a more traditional way of accessing information of the schedules. One advantage of this approach is the protection of the patients' data as it is not required any extra security for the remote access to the system. Also related to the issue of data security, systems providing the second feature (*Profiles*) hide or provide relevant data according to the profile of the user of the system, e.g. a surgeon have access to different data than a nurse. This feauture is usually related one to one with remote access, i.e. if remote access is provided profiles are necessary, but if we have a look at the classification we find that there are 2 cases where this is not true. In those cases, we can assume that the remote access is only provided to certain type of users that will have access to the same kind of information.

The next feature is related to the storage of information. For this feature the systems do not commonly use the same approach, we see that 6 systems (35.3%) use an independent database, 9 systems (53%) use the existing database and there are also 2 systems (11.7%) using both approaches together, i.e. apart from the existing database they store information in a different database. We can think that the integrated approach is always preferable, but according to the heterogeneity of systems found in nowadays healthcare institutions, in some cases the non integrated approach can outperform the integrated approach, as it does not require any modification in the existing database and, therefore, ease further improvements of the scheduling system.

The following four features are related to the type of data stored by the system. In the first two (*Patients* and *Interventions* data) there exist a complete parallelism, as those systems providing the former, always provides the latter. We can see in the tables that almost all systems (14 out of 17) stores this data, i.e. the system is able to use and provide clinical data from the patients and specific information about their interventions. This is logical, as this is core information for creating a schedule. The use of *Human Resource* is different, as more than half of the systems (53.3%) do not take them into account.

	ORWORCS	PATHWAYS HEALTHCARE SCHEDULING	OR ESSENTIALS	INTELLIGENT PERIOPERATIVE SUITE	CALL SCHEDULER
DATABASE MANAGEMENT MODULE					
Remote Access	No	Yes	No	Yes	Yes
Profiles	No	Yes	No	Yes	Yes
Database	P	I	I	P/I	P
Patients	No	Yes	Yes	No	No
Interventions	No	No	Yes	Yes	No
Human Resources	Yes	No	No	Yes	No
Physical Resources	OR	PR	OR	OR	No
MODEL MANAGEMENT MODULE					
Manual Scheduling	Yes	No	Yes	Yes	Yes
Supported Scheduling	Yes	No	Yes	No	Yes
Block Scheduling	Yes	Yes	No	Yes	No
Conflicts Management	H	H	H	n.s.	No
Rescheduling	No	Yes	Yes	Yes	No
DIALOGUE MANAGEMENT MODULE					
Textual Rep.	No	No	Yes	No	No
Graphical Interface	D/M	n.s.	D	D/W/M	D
Notifications	No	Yes	No	Yes	Yes
White Board	No	No	No	No	Yes
Reporting	Yes	No	No	Yes	Yes
Statistics Procurement	Yes	No	No	No	No
ADDITIONAL FEATURES					
Costs	No	A/C/S	No	A/C/S	No
System Integration	Module	Module	Module	Module	Independent

Table 5.2: Commercial Operating Room DSSTS (Part I).

	DIGISTAT SMART SCHEDULER	GOWIN QUI	SURGICAL SERVICES	CENTRICIT PERIOPERATIVE	OR CONTROL	SURGIMATE ON SCHEDULE
DATABASE MANAGEMENT MODULE						
Remote Access	No	No	No	Yes	Yes	Yes
Profiles	No	No	Yes	Yes	No	n.s.
Database	P	I	P	P	P	I
Patients	Yes	Yes	Yes	Yes	Yes	Yes
Interventions	Yes	Yes	Yes	Yes	Yes	Yes
Human Resources	Yes	No	No	No	Yes	No
Physical Resources	OR	No	No	OR/PR	OR/PR	OR
MODEL MANAGEMENT MODULE						
Manual Scheduling	Yes	Yes	Yes	Yes	Yes	Yes
Supported Scheduling	Yes	No	No	No	No	No
Block Scheduling	No	No	Yes	No	Yes	No
Conflicts Management	No	No	H	H	L	H
Rescheduling	No	No	Yes	Yes	Yes	Yes
DIALOGUE MANAGEMENT MODULE						
Textual Rep.	No	No	No	No	No	No
Graphical Interface	D	W	n.s.	D/W	D	D/W/M
Notifications	No	No	No	Yes	Yes	No
White Board	No	No	Yes	Yes	No	No
Reporting	No	Yes	Yes	Yes	No	Yes
Statistics Procurement	No	No	Yes	No	No	Yes
ADDITIONAL FEATURES						
Costs	No	No	A/C/S	C/S	No	No
System Integration	Module	Module	Module	Module	Independent	Independent

Table 5.3: Commercial Operating Room DSSTS (Part II).

	MEDICAL SCHEDULING SOFTWARE	OTIS	SURGICAL VALET	MAX GOLD	PRACTICE MANAGEMENT	SURGICAL SCHEDULER
DATABASE MANAGEMENT MODULE						
Remote Access	Yes	n.s.	Yes	No	Yes	Yes
Profiles	n.s.	Yes	n.s.	Yes	No	n.s.
Database	P	I/P	I	P	P	I
Patients	Yes	Yes	Yes	Yes	Yes	Yes
Interventions	Yes	Yes	Yes	Yes	Yes	Yes
Human Resources	Yes	Yes	No	Yes	No	No
Physical Resources	OR	OR	OR	OR	No	OR
MODEL MANAGEMENT MODULE						
Manual Scheduling	Yes	Yes	Yes	No	Yes	Yes
Supported Scheduling	No	No	No	Yes	No	No
Block Scheduling	No	Yes	No	No	No	No
Conflicts Management	H	No	No	No	No	No
Rescheduling	No	No	Yes	Yes	No	Yes
DIALOGUE MANAGEMENT MODULE						
Textual Rep.	No	No	No	No	No	No
Graphical Interface	D/W/M	n.s.	D/W/M	D/W/M	D/W	D/W/M
Notifications	Yes	No	Yes	Yes	Yes	Yes
White Board	No	Yes	No	No	No	No
Reporting	Yes	Yes	Yes	Yes	Yes	Yes
Statistics Procurement	No	Yes	Yes	Yes	No	Yes
ADDITIONAL FEATURES						
Costs	No	No	n.s.	A/C/S	A/C/S	n.s.
System Integration	Module	Independent	Module	Independent	Independent	Module

Table 5.4: Commercial Operating Room DSSTS (Part III).

Systems not considering human resources usually perform the staff scheduling based on the results of patients scheduling, i.e. they consider that human resources will be always available when developing the schedule, and then they adapt it to fit with the available staff. The last feature os this module is *Physical Resources*. Most systems provide this feature but in different ways. As we saw in Section 5.2, we classify systems depending on which data they offer. We found that most of the systems provide information of the operating rooms (11 systems), but there are also some (3 systems) that includes information of the post operative resources. In our opinion, this last option is specially important as it can improve the efficiency of the scheduling by avoiding those bottlenecks that appear after the intervention has taken place, e.g. in those cases where there is no beds available in the PACU after a surgery, the operating room remains blocked until a bed is released.

5.3.2 Model Management Module

As we already discussed in section 3.2.3, this is the most important part of the scheduling system. It is in charge of executing the scheduling using the data from the previous module. There exists mainly two types of systems regarding this module. Those offering users the possibility of creating the schedule by themselves and those that support the creation of the schedule based on some methods or algorithms. To classify the systems regarding this characteristic we use the first two features, i.e. *Manual Scheduling* for the first type and *Supported Scheduling* for the second. According to the results, we see that a large majority (15 systems out of 17) use the first type, i.e. these systems provide means to facilitate the creation of the schedule by decision makers but without suggesting a schedule based on an objective. If we focus on the second type, we see how this number is reduced significantly (only 5 systems). Moreover, from these 5 systems, all except one (MAX GOLD) also offers the possibility of manually modifying the schedule. From our point of view, this is the best approach as an initial schedule is given to the decision maker and based on his/her experience he/she is able to modify it in order to fulfill all his/her expectations. On the contrary, the pure manual scheduling does not differ too much from the traditional way of performing schedules, i.e. totally based on the experience of the scheduler. Nevertheless, the use of a computer system for generating the schedule, even in this case, offers many advantages to the scheduling process that are related with the other features of this module. It is important to remark here that it was impossible for us to get more information on the methods and algorithms used for the supported scheduling. This is the most opaque information of the systems when asking developers, as here is usually were the power of the systems resides.

The next feature of this module is *Block Scheduling*. It is present in only a 35% (6) of the systems.

Considering the heterogeneity present in the operating theatre from different institutions, we believe that this feature should be present in more systems. The common approach within the systems is offering a blank timetable for the operating rooms where it is not possible to differentiate between specialties or surgeons when the interventions are going to be planned. In those cases where a block scheduling strategy is selected, this approach forces the scheduler to consider himself/herself the different operating rooms assigned to the different specialties.

The *Conflicts Management* feature also offers an important help to the scheduler as it releases him/her for checking all conflicts every time a new patient is scheduled. More than half of the systems (53%) provide this feature. Almost all of these systems (8 out of 9) use the highlight of the conflicts every time a new patient is inserted, but in the case of OR CONTROL, the system generates a list of possible problems every time a conflict is detected and informs all the actors involved in the case where the conflict appeared.

The last feature of this module is *Rescheduling*. This feature is specially important in healthcare institutions due to the implicit uncertainty that is always present. In other sectors as manufacturing, once a schedule is developed, the probability of a change is relatively low, unless some problem appears. On the contrary, in healthcare, the probability of change in a schedule is very high. A proof of this is that, the operating room schedule, in some hospitals, can be done daily in order to minimize the problem of uncertainty. According to this, in the classification we see that more than half of the systems (53%) offers the possibility of rescheduling if it is necessary. Note here that, the rescheduling is considered according to the data we gathered. However, in most of the cases the rescheduling capacity is based on manually modifying those interventions that could not be performed or that requires to be modified. Similarly to the first two features of this module, it was impossible to gather information about methods or algorithms used by the systems for rescheduling.

5.3.3 Dialogue Management Module

It is important to consider the features of this module properly, since at the end, this module is the interface between the system and the users. A number of features could be classified, but due to the lack of a common understanding on this type of systems, it is important to remark that they provide this module in quite a different form. The two first features are related to the display of the schedule to users. In general, we see how every system offers a graphical representation of the schedules, except 4 systems where we could not find any detailed information on their graphical interface. Apart from this, we also analysed the granularity offered by these interface in order to get insights on the visibility given to decision makers. Almost all of them (12 out of 13) provides a daily view of the schedule. Moreover,

8 of these 12 systems also offers the possibility of seeing a weekly or monthly view of them. We focus so much on this feature as it can be considered as one of the most important of the whole system, since a critical requirement for a good implementation, is its acceptance by the scheduler, that will deal eith graphical interface in a daily basis. Besides more sophisticated graphical interfaces, we also detected a system (O.R. ESSENTIALS) that offered the possibility of obtaining a simple textual representation of the schedule in order to overcome any type of technical problem, offering a very simple form of sharing the results of the schedule among the actors involved in it.

The feature of *Notifications* is also present in more than half of the reviewed systems (59%). It is also important a fluent communication between the users of the system (including the system itself). Here we also find different approaches for the different systems but the point in common among all of them is the importance given to communication. This makes totally sense according to the number of different actors involved in the scheduling process. Also in relation with communication, we find the next feature *White Board*. It is similar to the previous one but in a more structured way. We found 4 systems (24%) offering users a common space where they could share messages or comments related to the schedule, e.g. one of the systems offered this space were every actor could show his/her preferences in order to decide the working hours of the whole staff.

The last two features are related to the use the systems make of the processed data. In first place, we detected that the majority of the systems (76.5%) provides the possibility of generating documents from the results of the schedule (*Reporting*). This is obvious for those systems that do not offer other way of communicating the results of the schedule to the rest of the persons involved in the process, but it is also desirable for the other systems as it provides a greater flexibility to the decision maker. On the contrary, only 6 systems offers the possibility of processing the results of the schedules in order to obtain more information that could be useful for further analyses. Among the use of this *Statistics Procurement* feature, we could mention the use of the schedules results to forecast intervention durations, to predict the appearance of problems according to certain factors, etc.

5.3.4 Additional Features

Apart from the features related to each of the main modules of the systems, we added two more features that seem interesting when analysing this type of systems. First, the possibility of considering the costs related to each intervention. We found 6 systems (35.3%) offering this feature. Among them, 5 systems take into account all types of costs, i.e. activity costs, consumables costs and stay costs, and one (CENTRICIT PREIOPERATIVE) considers only the last two. Although this feature is out of the scope of

scheduling, considering the tight budgets of nowadays hospitals, it constitutes an important help for the management board of the hospital. The second featutre reviewed in this category is the integration of the system,i.e. if it is a standalone system or if it is part of a wider system. In this case, there are almost the same number of each approches: a 41.2% of the systems are standalone systems while a 58.8% are modules of more complete systems not focused only on scheduling, but on an overall management of the hospital.

5.4 Main Findings

In order to analyse the current available information on DSSs for OR scheduling and due to the lack of information in academic literature, we performed a systematic review on available commercial systems for OR scheduling. A set of conclusions could be extracted from our review:

1. The control of the access to the systems is specially important, as they should be accessed from different locations in order to provide actors of the scheduling process with up to date information. From the review we obtain that there are still a number of systems that do not take advantage of the possibilities IT offers nowadays, thus it is important to consider these opportunities when a new DSSTS for OR scheduling is to be implemented.

2. There is not an uniform approach when defining the database of these systems. It is interesting to make an effort to define a common approach to define standards that allow different HISs and DSSs to communicate to each other in a seamless way. These would bring advantages as it would allow every institution decide the system that fits its requirements the best avoiding problems related to communication. It would also open a field for developers to build systems that could be widely used in many different environments.

3. Regarding the data, it would be also interesting to unify the criteria in which data should be considered in these systems. This data should be wide enough to satisfy the requirements of the most complex operating theatres but also flexible enogh to allow more simple environments to work only with a subset of this data to make the scheduling easier.

4. From the results of the review and also based on our experience, it is mandatory to provide the possibility of modifying the schedules manually in order to give the scheduler means to make the final decision on the schedule. It would permit him/her to consider constratins based on experience that are impossible, or at least very difficult, to code in a computer system. Nevertheless, it is specially

recommendable to offer the scheduler a set of initial proposals for the schedule based on state-of-the-art methods, that allows them to focus only on specific cases instead of having to consider every single patient on the waiting list.

5. The *Rescheduling* feature should be improved. In most of the reviewed systems the rescheduling in only addressed by giving the scheduler the possibility to manually change patients from the schedule once a problem arises. We think that an approach similar to the base schedule should be also considered here, i.e. once a problem appears the scheduler should define a number of constraints and preferences, such as those patients that should remain scheduled in the same position they were, and after the system should offer a new proposal of schedule based on the base schedule and on the new constraints and preferences. This would help him/her in obtaining a new schedule faster and easier.

6. According to the results of the review, we saw that offering means of communication between the different actors of the scheduling process is worthwhile. The importance of this feature has mainly two causes. First, in the healthcare sector the reluctance to change is specially important, so it is necessary to offer a system that mimics as much as possible the traditional way of working of all actors. And second, the highly human interaction between all participants of the surgical process makes it necessary to improve the efficiency and reduce the time required to perform these interactions.

7. In relation with the graphical interface we could assume that a Gantt Chart representation of the schedule is pertinent in this type of systems. As compared to manufacturing systems, the duration and frequency of tasks (patients) in OR scheduling makes this representation valid in almost every situation. This chart offers an overview of the whole situation to the scheduler that allows him to make changes on it in the case of any disturbance. It is also important to remark that the possibility of modifying patients of the schedule by dragging and dropping them in other OR/day is highly recommendable.

As a general summary we could say that, although every system we reviewed could be highly valuable to help in supporting the OR scheduling problem, there is none of them offering all the features that a system of this type should offer. Moreover, each of them is made based on a different approach, they are able to perform slightly different actions and provide different functionalties to schedulers. For this reason, we consider that the assumption made at the beginning of this thesis about the necessity of a common framework for DSSs for task scheduling is reinforced and we argue that the framework proposed

in Chapter 3 could serve as a common framework for this type of systems.

5.5 Conclusions

In this last section we extract a number of conclusions from the findings of the previous section in order to analyse the alignment of the framework proposed in Chapter 3 with existing operating room DSSTS described in the literature. The following conclusions can be drawn:

1. Security is a sensitive issue when dealing with DSSTS in healthcare. From the literature we see that there are still some systems that do not consider user management properly, therefore we can conclude that, although the task of user management is not directly a scheduler activity, it must be considered mandatorily when implementing a DSSTS in healthcare. This issue is considered in the *System Configuration* component, where all related to access permissions is handled.

2. There exist an important heterogeneity in how data is stored in the different operating room DSSTS from the market, so it is crucial to consider the intergration of the different information systems in the organization when implementing a DSSTS. This issue is addressed by the proposed framework with the *Integration with other Information Systems* component, that is in charge of the communication with other systems from the organization.

3. The different systems reviewed deal with different data when carrying out the schedule and most of them are limited to a restricted set of data, e.g. in some of them it is not possible to consider postoperative information. In order to give flexibility to the user to deal with different situations (see the main problems when implementing DSSTS in Chapter 2), the system should offer the possibility of including data that it is not present in the system. So, we can conclude that existing operating room DSSTS in the market do not offer enough flexibility to address unexpected situations. The propose framework takes this issue into account via the *Model Database* and the *DSSTS Database* components, that are in charge of storing and maintaining the data of the system.

4. There are some systems that do not offer the possibility of manually manipulating the resulting schedules. As we commented in Section 2.5, this issue is important when implementing an operating room DSSTS as it allows the user to include implicitly constraints that are not present in the decision models. As a conclusion we can say that those operating room DSSTS from the market that do not include this capability are too rigid to be adapted to a so changing sector as helathcare. The framework proposed provide this functionality in the *Scheduling Interface* component.

5. In a similar way to what we discussed about manufacturing in Section 4.5, rescheduling is not usually considered ot it is considered through a very simple approach. As rescheduling is part of the scheduling process, we can conclude from this finding that this functionality is not properly covered by the systems reviewed. This issue is explicitly considered in the *Solution Approaches Library* of the proposed framework.

6. Regarding the importance of the communication between the different actors of the process, specially in the healthcare sector, can conclude that a number of systems do not address this issue properly, but in general it is taken into account. The proposed framework considers this functionality via the *Integration with other Information Systems* where communication capabilities from other systems of the organization could be included into the resulting DSSTS.

7. A standard user interface that can be understood by most actors of the scheduling process is required. We can conclude from this review that, in general, this requirement is properly fullfiled by existing operating room DSSTS. This is issue is considered in the *Input* and *Output* components of the proposed framework.

8. The conclusion of the last finding reveals that, although there are many different commercial DSSTS for operating room scheduling, there exist a huge variability in the way they face the scheduling problem, how they manage its data and how the system communicates with the actors. This conclusion shows the relevance that a common framework as the one proposed in Chapter 3, has for this type of DSS.

To sum up these conclusions, we can say that, although they offer slightly different problems to those shown for the manufacturing case (see Chapter 4), the systems reviewed in this chapter also show the same problems discussed in Chapter 2. This validates the set of guidelines commented at the end of that chapter for operating room scheduling. Therefore, considering that the framework proposed in Chapter 3 was designed in order to tackle these problems, we can conclude that implementing DSSTS for operating room scheduling following the proposed framework would help in their succesful implementation.

Part IV

REAL APPLICATIONS OF THE FRAMEWORK

Chapter 6

The Manufacturing Case

This chapter presents an application of the framework proposed in Chapter 3 to a real case study within the manufacturing sector. To this end, we describe the design and implementation of a DSSTS in manufacturing designed according to the framework proposed in this Thesis. In order to get a fully understanding of the case study, we first describe the problem addressed, i.e. the hybrid flowshop scheduling problem with missing operations. Then, we detail the context where the DSSTS was deployed and its main functionalities. Next, we carry out an analysis of the hardness of the problem and present a set of efficient heuristics to address it. Finally, the main results of the implementation of the DSSTS are discussed.

6.1 PROMIA: A DSSTS for Manufacturing

The company where the DSSTS is to be implemented belongs to a manufacturer in the plastic sector and whose main activity is the manufacturing of merchandising products. The layout of the manufacturing plant can be compared to a hybrid flowshop, i.e. products follow a number of stages sequentially, with the possibility of missing operations, i.e. although stages are followed sequentially, some of them can skipped. In Figure 6.1 we can see the main manufacturing process.

The process can be splitted in two parts. The first part is dedicated to the design of the product. Although this part is not purely manufacturing, it can be easily incorporated into the manufacturing process. The first stage is *Final Artwork*. This stage is necessary for those products that have some illustration, e.g. a product with a logo printed on it. It is related to the development of the final graphical design of the illustration. In some cases this is done by the company but in others, customer provides this graphical design. We assume a single resource for this stage. The second stage is *Screen Printing*. Once the final artwork is available, the illustration is printed into a screen that will be next printed

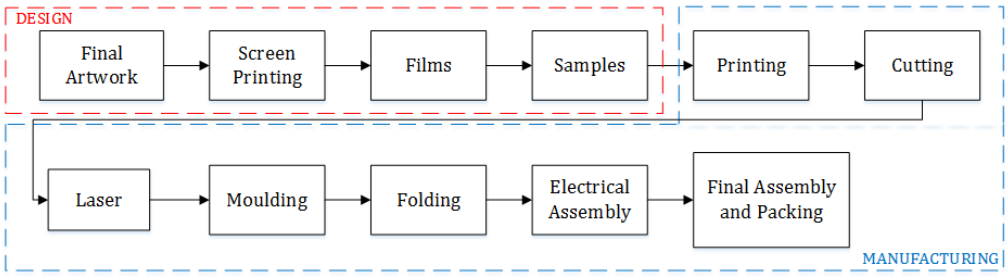

Figure 6.1: Manufacturing process of the case study

into the product. Two screen printers are available in this stage. For some special works, instead of screens, the illustration is printed into films. This operation is made in stage *Films*, and there is one resource available for it. For those more complex cases, the production of samples before obtaining the final products is required. With these samples, the main characteristics of the products can be tested and improved if necessary. We assume a single resource for obtaining these samples. The second part is dedicated exclusively to the manufacturing process. Once everything is checked, and the properties of the final products have been validated, the manufacturing process starts. The first stage of this part is *Printing*. In this stage the screens or films are printed into the products. Therefore, every product with an image on it must go through this stage. There are three printing machines available. The next stage is *Cutting*. In this stage, every needed cut is made on the product or on any part that is required later in its assembly. Two cutting machines are used within this stage. The next stage is *Laser* printing. This is required just by some of the products. To deal with this task one laser printer is available. The *Moulding* of the required parts is the next step and for this a moulding machine is available in the factory. It is also necessary some times to fold different parts. This is achieved in the *Folding* stage by means of two folding machines. Finally, for a number of products, *Electrical Assembly* needs to be done. For this stage we assume one available machine. And the last stage in the manufacturing process is the final assembly of all the parts that have been previously prepared and its packing. This stage is called here *Final Assembly and Packing* and is composed of three different stands or machines. According to this description, the layout of the factory we have just described is as shown in Figure 6.2. According to the product that is to be manufactured some of the previous stages can be ommited. Therefore, it is easy to assimilate this layout to the hybrid flowshop with missing operations described in the previous sections.

6.1.1 Hybrid Flowshop Scheduling Problem with Missing Operations

Flowshop scheduling problems have been largely studied in literature during the last 50 years (Pan et al., 2014). Due to a rising demand of products (Ribas et al., 2010), both in variety and quantity, it is

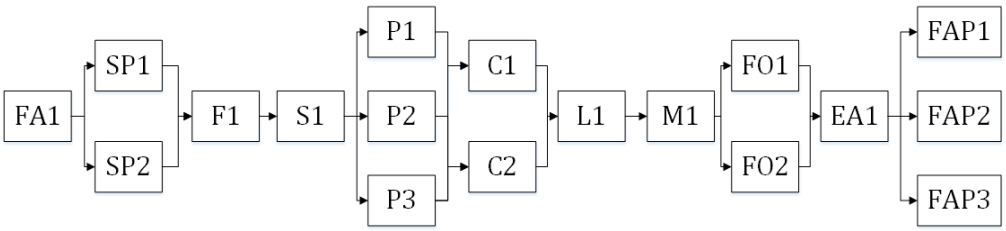

Figure 6.2: Layout of the manufacturing process

commonplace that companies increase their capacity by adding new resources (both physical and human) to some stages in the manufacturing process to avoid clear bottlenecks. As a result, some processing stages are formed by several machines, and a flowshop layout turns into a hybrid flowshop, a manufacturing setting that is gaining importance nowadays (Ruiz and Vazquez-Rodriguez, 2010).

Depending on the characteristics of the machines in every stage, a number of variations in the Hybrid Flowshop Scheduling (HFS) problem have been analysed in the literature: identical machines (the processing time for all machines in a stage is the same), uniform machines (each machine in a stage has a speed parameter associated that controls the processing time), and unrelated machines (the processing times of the different machines in a stage are independent). We focus on the HFS problem with identical parallel machines and with the possibility of missing operations, i.e. not every job has to go through all the stages.

Regarding the objective function of the HFS problem, the main objectives used in literature can be classified into those related with the completion time of the jobs, with the delay of jobs, with multiple criteria or those with an specific problem-dependent criterion (Ribas et al., 2010). Among all objectives, the makespan is clearly the most common one. We focus on this objective, which would enable us to compare and to discuss our results with existing contributions.

With regard to solution approaches, the HFS problem with makespan objective for 2 stages where at least one of them has more than one machine has been proven to be NP-hard (Gupta, 1988). Therefore, most researchers have focused on developing efficient heuristics and/or metaheuristics. We refer to the review by Ruiz and Vazquez-Rodriguez (2010) for a more detailed description and for an analysis of the algorithms used in the HFSP.

The problem of scheduling hybrid flowshops with missing operations has not been widely analysed in the literature. Most often, it has been addressed as an ordinary hybrid flowshop scheduling problem assuming that a missing operation is tantamount to an operation with processing time equal to zero. Nevertheless, since this assumption implies that every job has to be processed in every stage even with zero processing times, this approach results in an increase of the completion times of the

jobs (see e.g. Leisten and Kolbe, 1998 and Sridhar and Rajendran, 1993). More specifically, we refer to the problem under study as *Hybrid Flowshop Scheduling with Missing Operations* (HFSMO) problem. Note that this problem can also be found in literature as *Flexible Flow Line Scheduling*, see e.g. Leon and Ramamoorthy (1997) and Kurz and Askin (2003) or simply as *Hybrid Flowshop Problem*, see e.g. Ruiz et al. (2008). This problem with makepan objective is denoted as $FHm\big(PM^{(k)}\big)_{k=1}^{m}|skip|C_{\max}$ according to e.g. Vignier et al. (1999) and Ruiz and Vazquez-Rodriguez (2010).

The HFSMO problem with makespan objective can be stated as follows: We consider a set of n jobs, $N = \{1, ..., n\}$ that have to be processed in s different stages, $S = \{1, ..., s\}$. Each stage is composed of a set of s_i identical parallel machines, $S_i = \{1, ..., s_i\}$, i.e. the processing time of the job in all machines within a stage is the same. The processing time of a job in a specific stage is p_{ij}, being i the job, $i \in N$, and j the stage, $j \in S$. Every job has the same routing through the stages, but some of them can be skipped, i.e. there may be missing operations. Jobs are processed by exactly one machine at each stage. The objective is to find the sequence of jobs on each stage so the maximum completion time (makespan) is minimised.

The general HFS problem –from which the HFSMO problem is a particular case– has been widely analysed in the literature, and several reviews can be found. In Linn and Zhang (1999) authors briefly review the different types of hybrid flowshop problems according to the number of stages, i.e. 2-stage, 3-stage and k-stage. Ribas et al. (2010) presents a review focusing on the characteristics of the problem, i.e. the type of machines within each stage and the job constraints, reviewing the different solution approaches found in the literature. With a similar focus, Ruiz and Vazquez-Rodriguez (2010) review more than 200 works according to the classification and nomenclature of Vignier et al. (1999).

Among the approximate procedures for the HFS problem with makespan objective, four heuristics are worth to note. The first one is NEH, first proposed by Nawaz et al. (1983) for the flowshop problem. This heuristic starts by ordering jobs according to their longest processing time, i.e. those with the highest sum of processing times across all stages ($\sum_j p_{ij}$) and constructs a solution by selecting all jobs, one by one, and inserting them into the best possible position, i.e. the position providing the lowest (partial) makespan. The pseudocode of this heuristic is presented in Section 6.3.2. The NEH is first applied to hybrid flowshops by Brah and Loo (1999), and the authors found that it outperforms other usual flowshop heuristics such as CDS1, CDS2, PAM and HO.

Another heuristic is that by Rajendran (1993), a modification of the NEH that reduces its computational time at the expense of a reduction in the quality of the solution. In this case, instead of inserting all jobs, it selects half of them and search for their best possible position in the sequence. Its pseudocode is shown in Section 6.3.2.

Finally, the heuristics WT1_NEH(x) and WT2_NEH(x) by Kizilay et al. (2014) are based on the profile-fitting method by McCormick et al. (1989). In these heuristics, authors construct a number of sequences based on a parameter x, and looks for the one obtaining the minimum makespan by reducing as much as possible the waiting time between jobs in each stage. Once they obtain a solution, they apply the NEH heuristic, i.e. instead of using as initial solution the order obtained by computing the longest processing time of the jobs, they use the solution found by their procedure. Both heuristics are shown to perform better that the NEH for the HFS problem with makespan objective.

Among those works considering explicitly missing operations, Leisten and Kolbe (1998) address missing operations in the classical m-machine flowshop. In their work they analyse the suitability of using strict permutation schedules against the possibility of allowing for job passing, taking into account that the permutation constraint is not realistic at all, i.e. it assumes that the line will remain stacked although there are jobs that can be processed in another machine idle. Their conclusion is that advantages can be found when calculating makespan (or total flowtime) if job passing is allowed. In Tseng et al. (2008), the difference between permutation and non permutation schedules in a two-stage HFSMO problem is analysed, showing that non permutation schedules obtain better results without greatly increasing the computation times. They present a heuristic for generating a non permutation schedule from a permutation one, obtaining a reduction in the makespan. More recently, Saravanan et al. (2014) solve the k-stage HFSMO with makespan objective using two different metaheuristics (simulated annealing and particle swarm optimization), obtaining that the former outperforms the latter, both in computational time and in the quality of the solutions. Finally, Marichelvam and Prabaharan (2014) propose a new hybrid metaheuristic for the k-stage HFSMO problem with makespan objective named Improved Hybrid Genetic Scatter Search (IHGSS), where authors combine a Genetic Algorithm (GA) and a Scatter Search (SS) and compare its results against the isolated version of the GA and the SS, obtaining that the IHGSS outperforms both of them.

Apart from these contributions specifically devoted to missing operations, some others also take them into account in a more complex layout: In Kurz and Askin (2003), a set of heuristics are proposed for the k-stage hybrid flowshop with sequence-dependent setup times, assuming the possibility that the jobs skip some stage. The authors consider three levels of this skipping probability, 0%(low), 0.05%(medium) and 40%(high). In Naderi et al. (2010), a new dispatching rule and an iterated local search metaheuristic are proposed for the hybrid flexible flowshop, i.e. an hybrid flowshop considering missing operations and sequence-dependent setup times. Several heuristics from literature are also adapted in this work. In their case, two levels of skipping probability are considered, i.e. 10% and 40%.

Note that a sign of the lack of studies on missing operations is the heterogeneity in selecting values for

the skipping probability. It is also possible to find some comments about missing operations in problems slightly different to the classical hybrid flowshop problem, such as in e.g. Behnamian and Fatemi Ghomi (2011), where the authors consider the case with machine and resource dependent processing times. They also present a hybrid metaheuristic composed of a genetic algorithm and a variable neighborhood search. In this case they also consider the possibility that jobs can skip stages, and consider the same levels as in Kurz and Askin (2003).

In summary, there are many different heuristics for the HFS problem with makespan objective, and a few of them for the particular case HFSMO. Therefore, it would be interesting first to assess whether the additional feature of missing operations make these two problems substantially different. This analysis together with a set of heuristics for the HFS problem will be discussed in Section 6.3.

6.2 Main Use Cases of the DSSTS

After the analysis of the case study, a number of use cases –i.e. functionalities or blocks of functionalities– are identified by comparing the company's requirements with the framework proposed in Chapter 3. For the sake of clarity, we illustrate them by using a simple manufacturing scenario and following the normal procedure from the acceptance of the order to the final schedule of the tasks followed by their monitoring and control. We consider two different products and a total of 3 different customer orders. The first order is composed of 100 units of *Product 1* and 100 units of *Product 2*, the second one is composed of 500 units of *Product 1* and 600 of *Product 2*, and the last order is formed by 300 units of *Product 1* and 200 of *Product 2*. The products are manufactured according to the process shown in Figure 6.3, each one with a number of missing operations represented in dashed lines. It can be noted that both products share some resources, i.e. *Screen Printing*,*Printing* and *Final Assembly and Packing*, while they also use some resources in isolation, i.e. *Cutting* and *Moulding* for *Product 1* and *Final Artworks* and *Samples* for *Product 2*.

The following use cases have been identified according to the proposed framework:

- Alternatives Analysis, which corresponds to component *Analysis Tools* in the framework.
- Long term planning, Short Term Scheduling and Rescheduling, which corresponds to component *Decision Problems Handling* in the framework.
- Control, which corresponds to component *Schedule Control* in the framework.

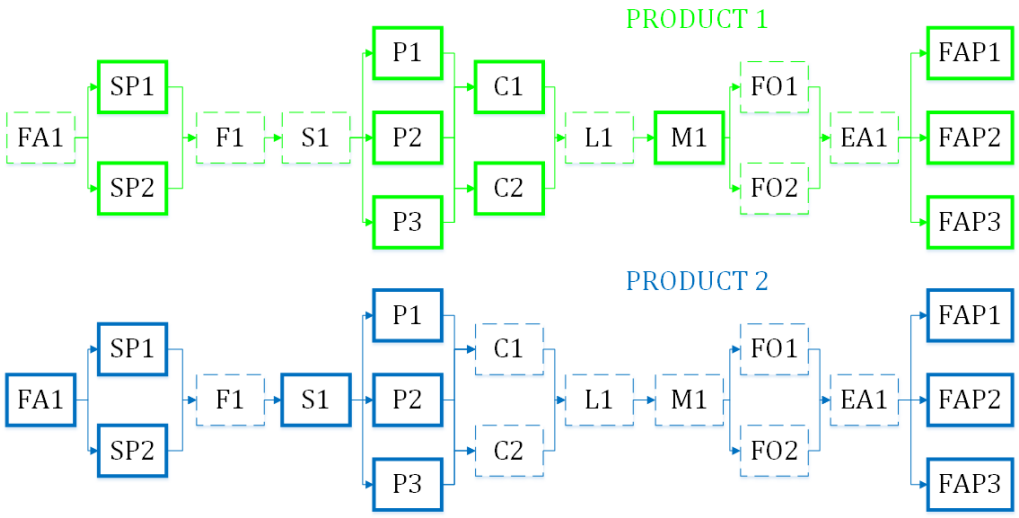

Figure 6.3: *Product 1* and *Product 2* manufacturing process

6.2.1 Alternatives Analysis

The first use case of the system offers the possibility of analysing the customers orders as soon as they are received, in order to be able to decide if the products can be made, and more important, if they can be produced within the due date demanded by the customer. It also can serve in those cases where we have to offer the customer an estimation of the required time to deliver the products. The system is able to analyse the production cadence of a specific product, i.e. how often it is able to produce a unit/lot of product/s. To do so, a goal cadence is introduced and the system analyse the manufacturing process of the product according to its route, i.e. the different resources it has to go through, and it responds if it is possible to accomplish the goal cadence, which is the most saturated resource and how much it is saturated. An example of this use case is shown in Figure 6.4. In this example we see how, for *Product 1*, a goal cadence of 4 hours/unit can be achieved, and for that case, the most saturated resource is the *Screen Printing* with a saturation of 75%.

This use case can be useful for both, the sales department or the planner to obtain a better idea of how the system reacts to the manufacturing of an specific product. Note that the results obtained from this use case are only an estimation, as they depend on the existing workload of the factory once the order is released.

6.2.2 Long Term Planning

This use case is directly related with what we have defined as planning throughout this thesis. Once a number of customer orders have been accepted, it is necessary to decide when these orders are to be

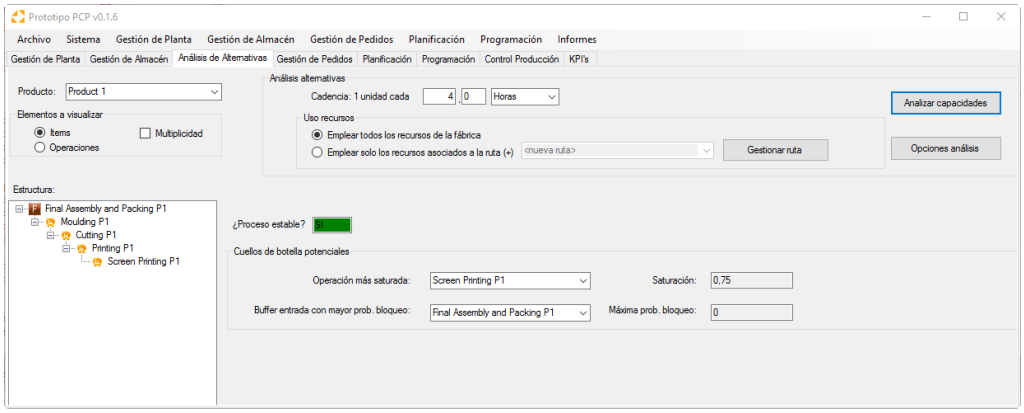

Figure 6.4: Alternative Analysis Screen

released to the factory. To this end, we need a more specific understanding of the time they need to be processed –considering the number of units per order, the type of products to be produced, etc.– and when they should be started to fulfill the deadlines imposed by the customers. Note that, at this point it is not necessary to have a detailed schedule of the different tasks that have to be done for each order. To deal with this necessity, the system offers the possibility of obtaining a long term plan with the information required. The long term plan is obtained in three steps, that the planner can use as he/she prefer. First, the planner decides which customer orders he wants to plan. In this first step it should be considered the importance of the customer –i.e. if it is a regular customer or if it is paying more for receiving the product as soon as possible, in this last case its orders should be planned before–, the trust on it –i.e. if the planner thinks that an order could be cancelled based on his/her experience, it should be planned as late as possible– or the urgency of the orders. Next, the planner decides if the plan should be obtained taking into account the orders that are already in the factory, or if it is preferable to obtain a plan considering an empty factory. According to this decision, a plan is obtained for the selected customer orders, giving information about when the different orders should be started or, in the case they could not be finished in time, which are the most overloaded resources. The two first steps should be used iteratively i.e. a number of orders are selected and planned, and in the case the results are not satisfactory some orders are taken out of the plan and some others are included. Once the planner is satisfied by the results, the third step is their release to the factory, where they will be scheduled in detail.

Within the system, these three steps are made through the screen shown in Figure 6.5. In the upper part, all the customer orders that have not been scheduled yet appear. Then they are moved to the middle part, where the selected orders are planned, and finally, they can be moved to the lower part of the screen where they are ready to be scheduled. In the figure we can see the three customer orders detailed before already prepared for scheduling.

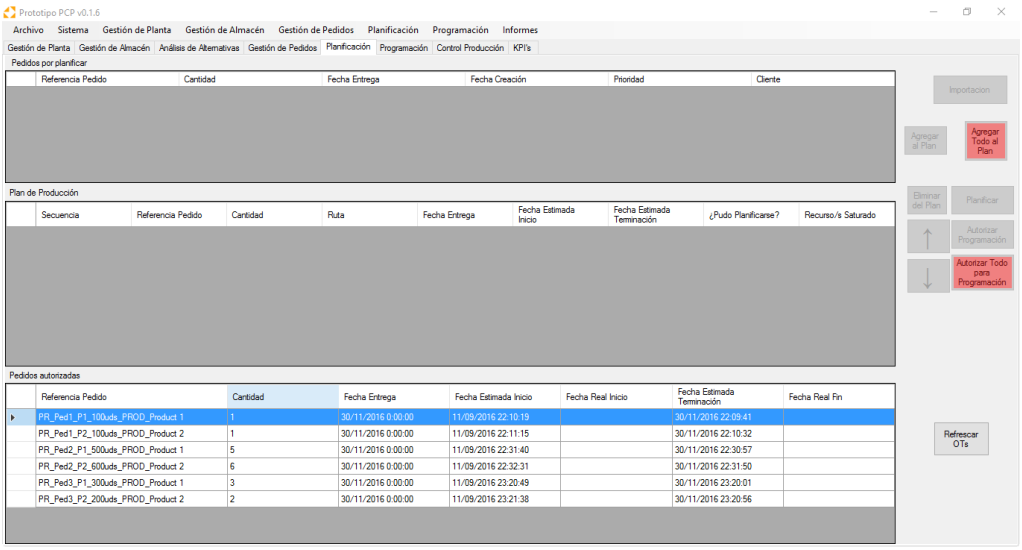

Figure 6.5: Planning Screen

6.2.3 Short Term Scheduling

The core use case of the system is the short term scheduling. Its aim is to, given a number of orders to be scheduled with its corresponding due dates, schedule them in the best possible way according to an objective, such as meeting due dates, reducing makespan, etc. taking into account the availability of the resources where they have to be manufactured. The output offered by this use case is an ordered set of tasks together with the time where they should start and the resource where they have to be processed. Depending on the features of the company, i.e. uncertainty in processing times and orders, probability of machine breakdowns,etc. the planning horizon can be up to two weeks.

There are also a number of steps to follow in this use case. First, the specific scenario must be configured. This scenario is composed of a planning horizon -i.e. the specific days to produce the schedule, considering holidays and weekends-, the availability of the resources for the planning horizon –i.e. the working hours of the different resources taking into acount possible maintenance tasks or scheduled stops– and finally the scheduling algorithms that the planner decide to apply to the specific case. In this part, the planner can decide among a number of possible algorithms, both heuristics or exact algorithms, according to the layout of the scheduling problem. The algorithms embedded in the DSSTS will be discussed in Section 6.3. Apart from the scheduling algorithm, it is also possible to apply a post processing to try to further improve the solutions. This post processing is based in trying to improve the objective of the schedule by reordering the tasks in every resource within a specified time period. As soon as the scheduling algorithm and the post processing -in the case it was selected- finish, two different views of the scheduling are provided to the planner, a Gantt Chart and an ordered list of tasks per resource together

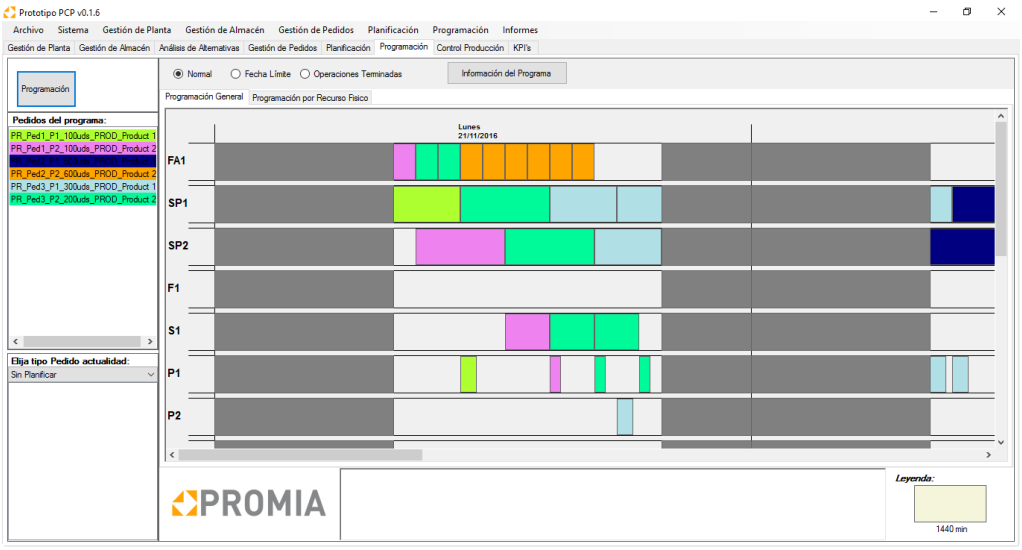

Figure 6.6: Gantt chart view of the example

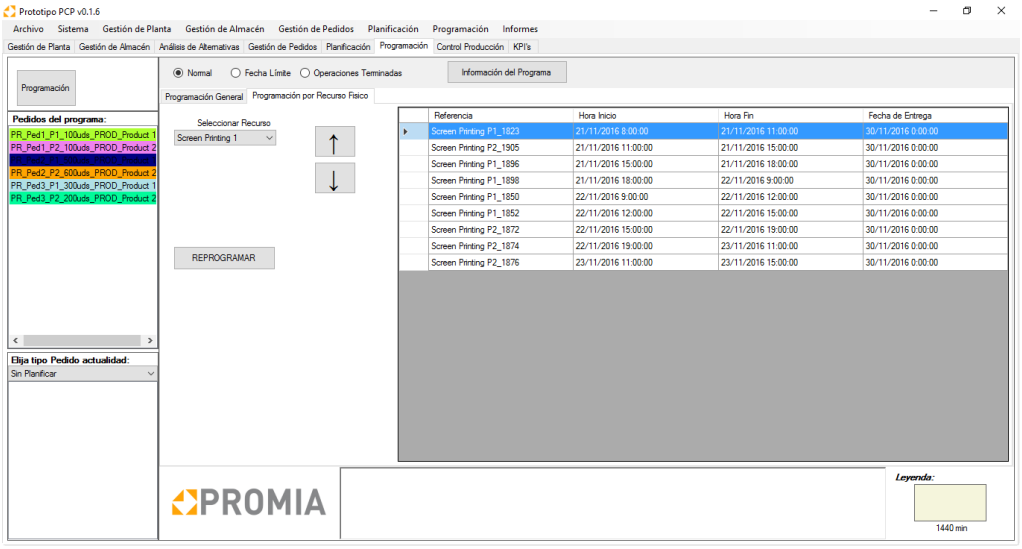

Figure 6.7: Tasks per resource view for *Screen Printing 1*

with their start and finishing times. Typically, the Gantt Chart is prefered for those cases where there are not many tasks and their processing times are quite large so the planner can get a good understanding of the solution having a look at the chart. For those cases where the number of tasks is high and their processing times are short, it is more useful the tasks per resource view, as in the Gantt Chart it is not possible to distinguish between tasks when the squares representing the tasks in the chart are too narrow or when the chart has to be too wide to represent all the tasks. In Figures 6.6 and 6.7, both views of the short term scheduling are shown.

6.2.4 Rescheduling

In relation with the previous use case, if some disrupting event appears during the planning horizon, it must be possible to reschedule those tasks affected. To do so, the system allows for modifying the order of the tasks in the scheduling in any of the resources and recalculate the starting and finishing times of the whole schedule. This is done in the *Tasks per resource view* (see Figure 6.7) by just moving up or down the tasks and clicking on *RESCHEDULE* (REPROGRAMAR). The resulting Gantt Chart and the new starting and finishing times of the tasks are automatically calculated and updated in the system.

6.2.5 Control

This last use case is used to maintain the data in the system up to date and to monitor and control the manufacturing process in real time. In an important number of companies, the schedule is released to factory as a paper -handmade or printed from any scheduling system- where the schedule is described. To ease and improve this process, our system offer the possibility of releasing the schedule by means of any electronic device/s containing a web browser. These devices are connected to the the system and provide the possibility to access the schedule from any point of the factory, so once a new schedule is released, it is automatically accesible for all the factory. Apart from this, the devices also serve as an input interface where workers (or people in charge of updating the data) input the tasks they are working on, their starting time and their finishing time. By doing so, the data in the system is maintained up to date in every moment and a real time view of the progress of the whole scheduling is available for the planner. This view helps the planner in anticipating possible disruptions or delays. In comparison with traditional procedures, it also removes the need for updating data related to which tasks have been done, where they were executed and when they started and finished, at the end of the working shift, thus reducing the possibility of generating errors when introducing this data.

In Figure 6.8 we can see the main page of the control application. On the left part of the screen the worker should select in which task he is going to work. For this, he/she should select in which customer order and in which resource he/she is working. After selecting the task, he/she should inform the system that the task is going to start. From this moment, the system assumes that the task continues until the worker informs that the task has been finished. It is also important that, before finishing the task or after each working shift, the worker informs about how long he has been working on it and what is the percentage accomplished of the total amount of work required for that task. By gathering and analysing these data, the system can *learn* to guide the planner for the next schedules. Finally, from this screen it is also possible for workers to inform the system if there has been any incidence while executing the

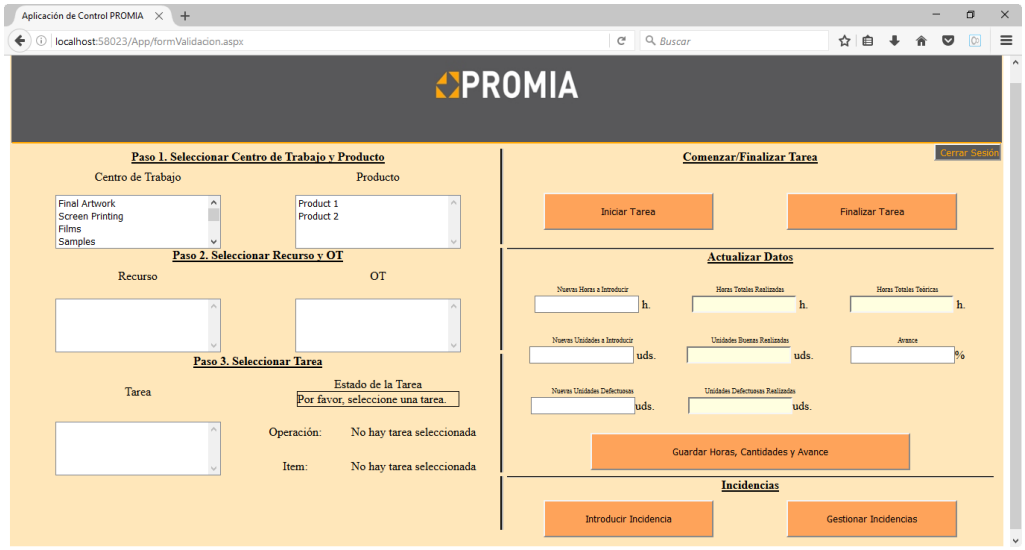

Figure 6.8: Main page of the control application

task. This also helps the planner to know in real time if there is some disruption in the factory and why it happened.

6.3 Solution Procedures for the Case Study

In this section we carry out an in-depth analysis of the HFSMO problem. First, a hardness analysis of the problem is made to analyse the convenience of developing new algorithms to handle it,and then, a set of efficiente solution algorithms are proposed.

6.3.1 Analysis of the problem

In this section we present an empirical analysis of the structure of solutions of the HFSMO problem as compared to that of the HFS. The idea is to analyse how easy is –in statistical terms– to obtain good solution for the problems under study. To do so, a complete enumeration of the space of solutions of the problem is carried out for a large number of instances, and the deviation with respect to the optimal solution for each of them is studied, so the empirical distribution of the deviation from the optimal makespan can be obtained. This type of analysis has been proved to be very useful for different scheduling problems (see e.g. Taillard, 1990; Armentano and Ronconi, 1999; Framinan et al., 2001, or Perez-Gonzalez and Framinan, 2015). A disadvantage is that this method can only be used for a small number of jobs/stages, given the NP-hard nature of the problem under study.

With respect to instances to be analysed, different parameters have to be determined: The processing

Table 6.1: Hardness Analysis Parameters

Parameter	Values
n	5,6,7,8
p_{ij}	$U\,[1,99]$
% Missing Operations	0%,20%,40%,60% / aditional (0%,20%,40%,50%)
s	3,5,10,15 / aditional (2,3,4,5)
S_i	1,2,$U\,[1,5]$,$U\,[2,4]$
Number of Instances	300

times are generated according to the [1,99] uniform distribution, which is the most common distribution used in literature, both for the permutation flowshop (Taillard, 1993) and for the HFS (Ruiz et al., 2008). Regarding the percentage of missing operations, authors typically use 3 levels, i.e. 0%,20%, and 40% (see e.g. Rajendran and Ziegler, 2001 and Tseng et al., 2008). We add the level 60% to extend the analysis. In order to set the missing operations for each job, we follow a procedure similar to Rajendran and Ziegler (2001), i.e. we generate all processing times and then make a x% of the total number of operations to be equal to zero. By doing so, we avoid the appearance of patterns in the distribution of missing operations, i.e. we avoid that the number and stage of missing operations in the jobs are always the same. We also limit the number of missing operations in a job to $(m-1)$, i.e. a job cannot have all its operations with processing times equal to zero. Regarding the number of stages, we make a first analysis for 3, 5, 10 and 15 stages, as it is not possible to consider just 2 stages for a 60% level of missing operations. Therefore we in addition conduct an analysis for 2, 3, 4 and 5 stages with a maximum of 50% of missing operations.

The number of machines per stage has also to be set for the instances. We consider two deterministic and two uniformly distributed cases. In the first two, we assume that there are one or two machines in every stage, respectively. For the next case we assume an uniform [1,5] distribution is employed to generate the number of machines in each stage function from 1 to 5 machines. Note that, by doing so, it is quite likely to have a bottleneck stage, i.e. a stage where jobs have to wait to be processed. Finally, the fourth case uses a [2,4] uniform distribution, which is expected to provide a more balanced shop. In order to increase the statistical significance of the analysis we use 300 instances of each problem size, thus processing a total of 76,800 instances for each type of analysis. In Table 6.1 a summary of the parameters employed is shown.

The results of the computational experiments (Figures 6.9- 6.16) are shown as an empirical probability distribution, with the distance to the optimal (in percentage) in the x-axis and the probability that a solution is in that distance (non accumulative) in the y-axis,e.g. in Figure 6.9, for the case of 8 jobs, if we randomly select a solution from all possible solutions, there will be a probability of around 15% of being

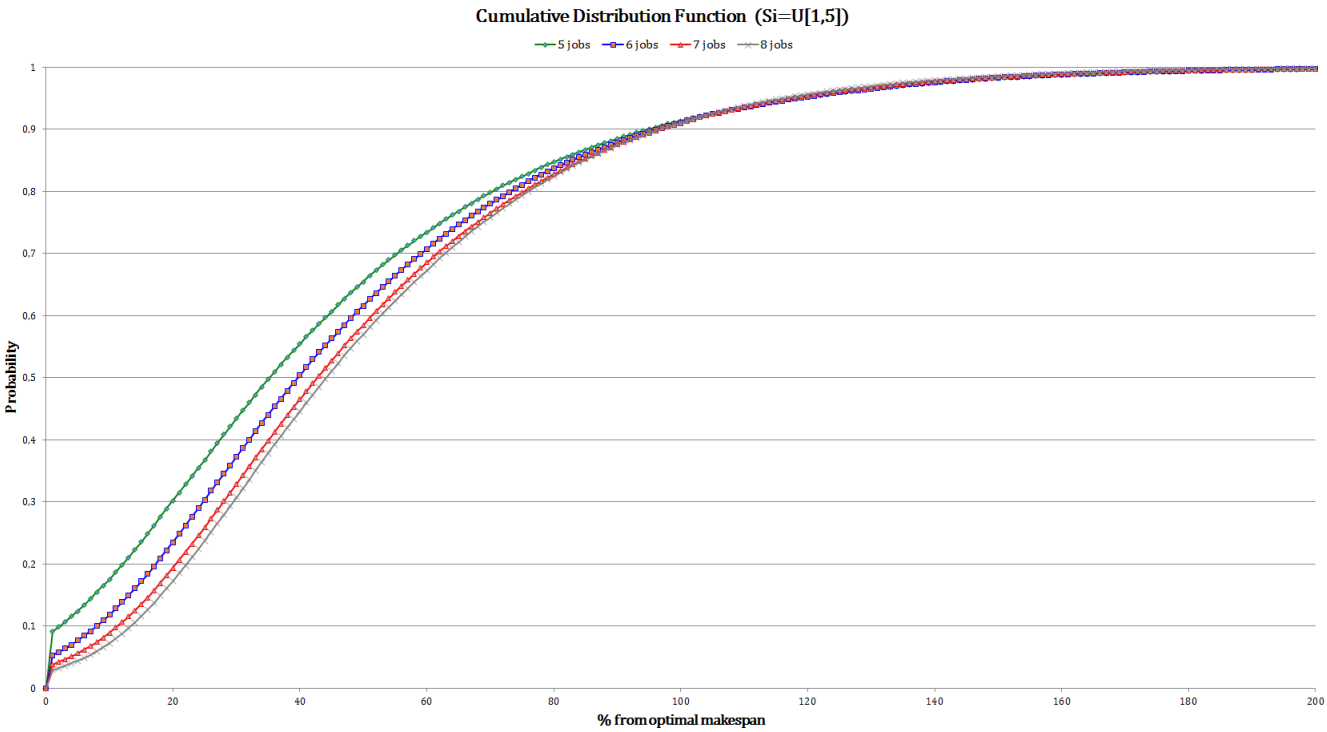

Figure 6.9: Hardness Analysis for $S_i = U[1,5]$ (jobs)

within a distance of 20% to the optimal solution, while in the case of 5 jobs this probability is increased to a probability of around 30%.

Depending on the number of machines per stage we are able to analyse different shop layouts. According to the results from the analysis the following conclusions can be made:

- Number of Jobs

Although we could not use a large number of jobs to carry out the analysis, it is possible to detect an increasing trend in hardness as the number of jobs increases. This behaviour is quite similar for any configuration of the layout, i.e. number of machines per stage, e.g. in Fig. 6.9 we see the case of an hybrid flowshop with a wide variation of machines in the stages ($S_i = U[1,5]$). In the additional analysis we did not find any different behavior.

- % Missing Operations

The analysis of the results according to the percentage of missing operations is not as direct as with the number of jobs. It can be seen that the pattern followed by the empirical hardness is similar for $S_i = 1$ and $S_i = U[1,5]$. In these cases, the greatest hardness is achieved for those problems without missing operations, and the tendency is that the higher the percentage of missing operations, the less harder the

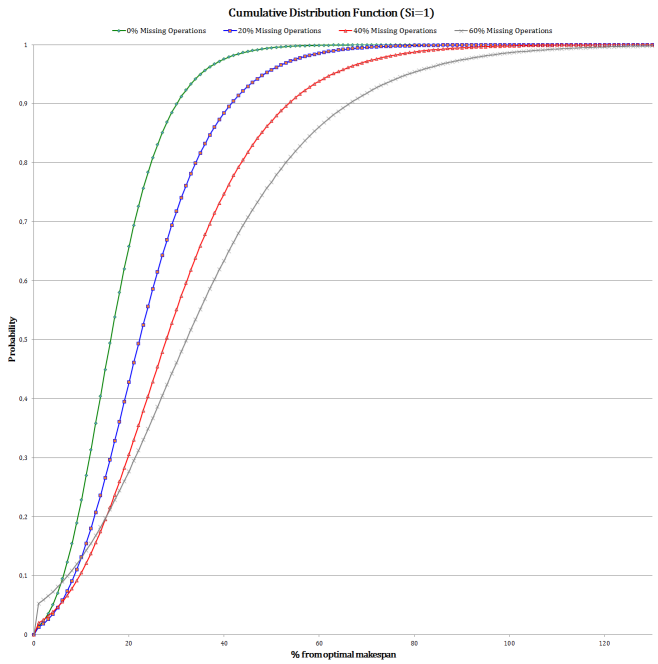

Figure 6.10: Hardness Analysis for $S_i = 1$ (missing operations)

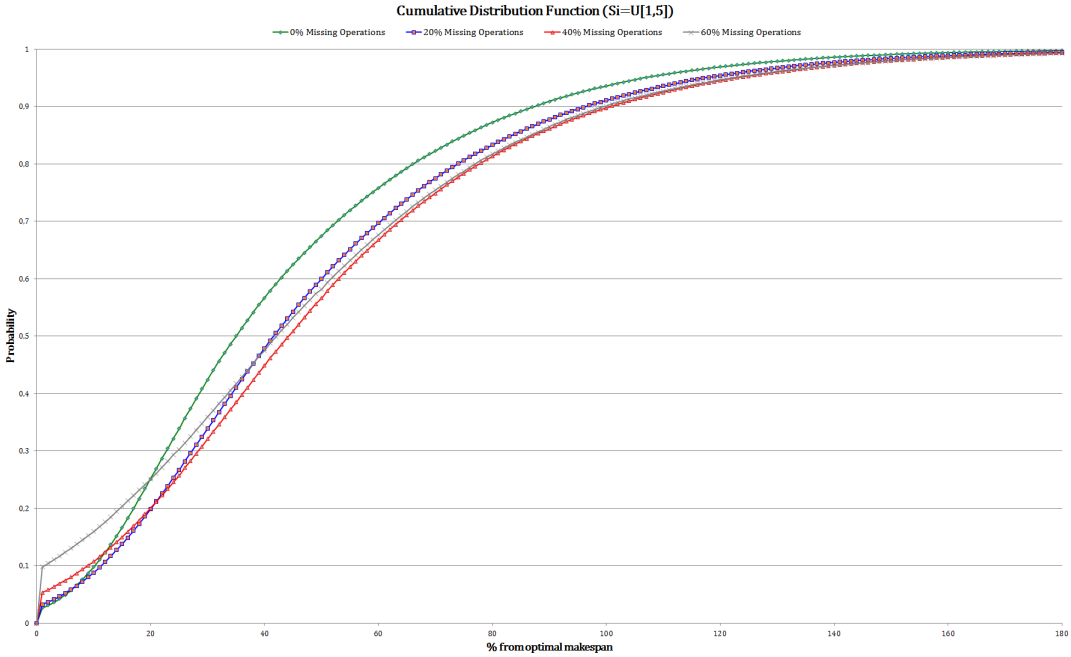

Figure 6.11: Hardness Analysis for $S_i = U[1,5]$ (missing operations)

problem is. Moreover, there is a clear difference in the hardness of $S_i = 1$ and $S_i = U[1,5]$, being the latter harder than the former (see Fig.6.10 and 6.11).

For $S_i = 2$ and $S_i = U[2,4]$, the tendency is the opposite (see Fig. 6.12 and 6.13), i.e. the higher the percentage of missing operations, the harder the problem is. However, for $S_i = 2$ and $S_i = U[2,4]$

the difference in hardness is not as pronounced as in the previous two. The most significant difference is that, for $S_i = 2$ there is a slightly smaller difference in hardness between problems with 0%, 20% and 40% of missing operations (see Fig. 6.12 and 6.13). A possible explanation of the different behaviours is that, in the first two cases, the possibility of finding stages very highly loaded in the shop, as processing times remain the same for all of them, is much higher than in the other two cases, i.e. in $S_i = 1$ there is a high probability that any of the single-machine stages becomes very loaded while for the case where the number of machines per stage varies between one and five, it is foreseeable that those stages with a single machine, or even those with two, become the most loaded stages. Therefore, the existence of missing operations is an opportunity to unload these stages. On the other hand, in the last two cases, the hardness increases with the number of missing operations because the appearance of missing operations can cause an *over-loading* in some stages, as these operations free the stage where they take place but with the cost of loading the following stages.

For the additional analysis the behavior is also similar. However, depending on the case, as we get closer to higher percentages of deviation from the optimal solution, the probability of finding solutions to problems without missing operations increases, e.g. for the case $S_i = 1$, we find that there is a higher probability of finding a solution with a distance to the optimum lower than 10%, but this probability is higher for finding solutions over 10%. In a similar manner, for the case $S_i = U[1, 5]$, we also find this change in hardness but it occurs at around 16%. The comparison between these two cases is shown in Figure 6.14.

- Number of Stages

For those cases with more than one machine per stage, i.e. $S_i = 2, U[2, 4]$ and $U[1, 5]$, a similar behaviour is detected, i.e. the more stages are considered in the shop, the harder the resolution of the problem is (e.g. see Fig. 6.15 for the case $S_i = U[2, 4]$). On the contrary, for the case $S_i = 1$, the tendency is the opposite (e.g. see Fig. 6.16). We can also see that, for small percentages of deviation from the optimal solution, the behaviour of this last layout is similar to the behaviour of the former ones, but it changes from deviations of around 10-15% or higher.

We found some variations in the additional analysis for the number of stages, i.e. for a small number of jobs the behaviour of any layout is quite similar, which could be foreseeable. Note that this tendency greatly depends on the processing times of jobs in each stage, so this behaviour may be different for different distributions of the processing times.

As conclusion of the analysis carried out in the section, we have shown that, although it highly depends on the specific layout of the shop, there are a number of scenarios where the HFSMO problem is different

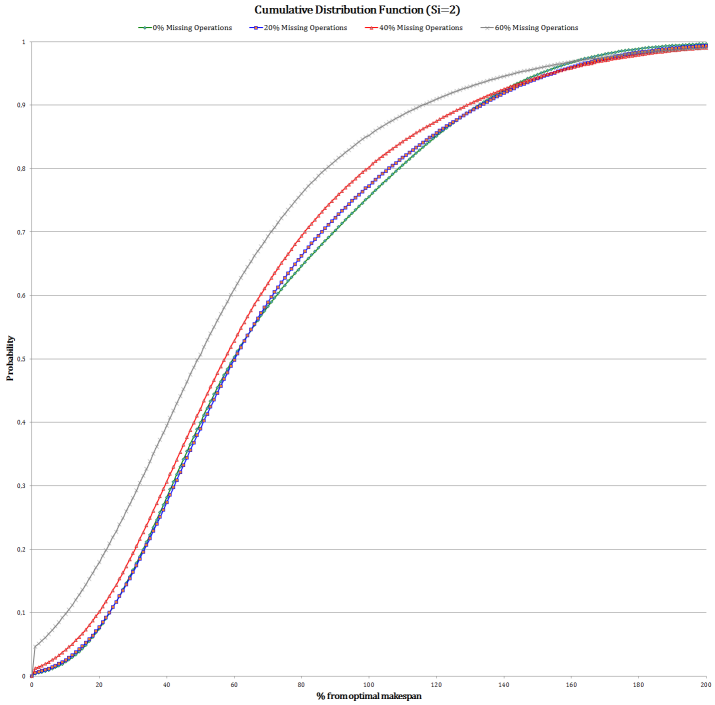

Figure 6.12: Hardness Analysis for $S_i = 2$ (missing operations)

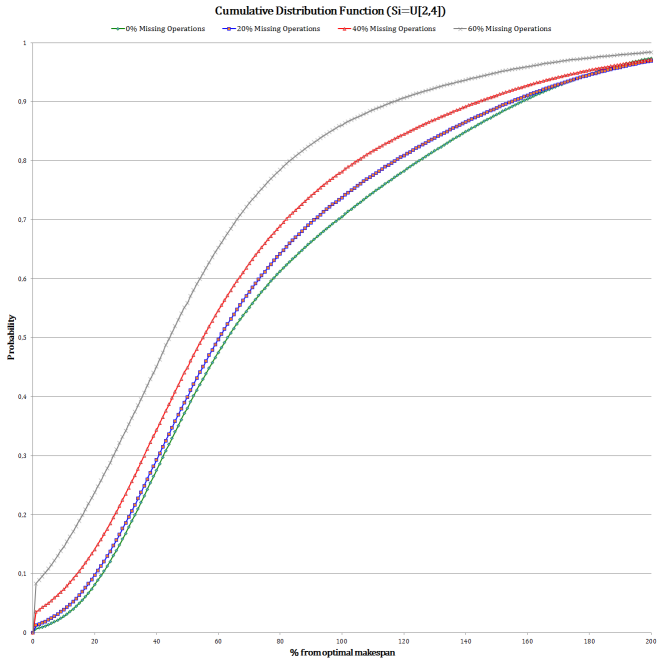

Figure 6.13: Hardness Analysis for $S_i = U\,[2, 4]$ (missing operations)

from the classical HFS problem, so it makes sense to use and develop heuristics specifically designed for this particular case. This is addressed in the next section.

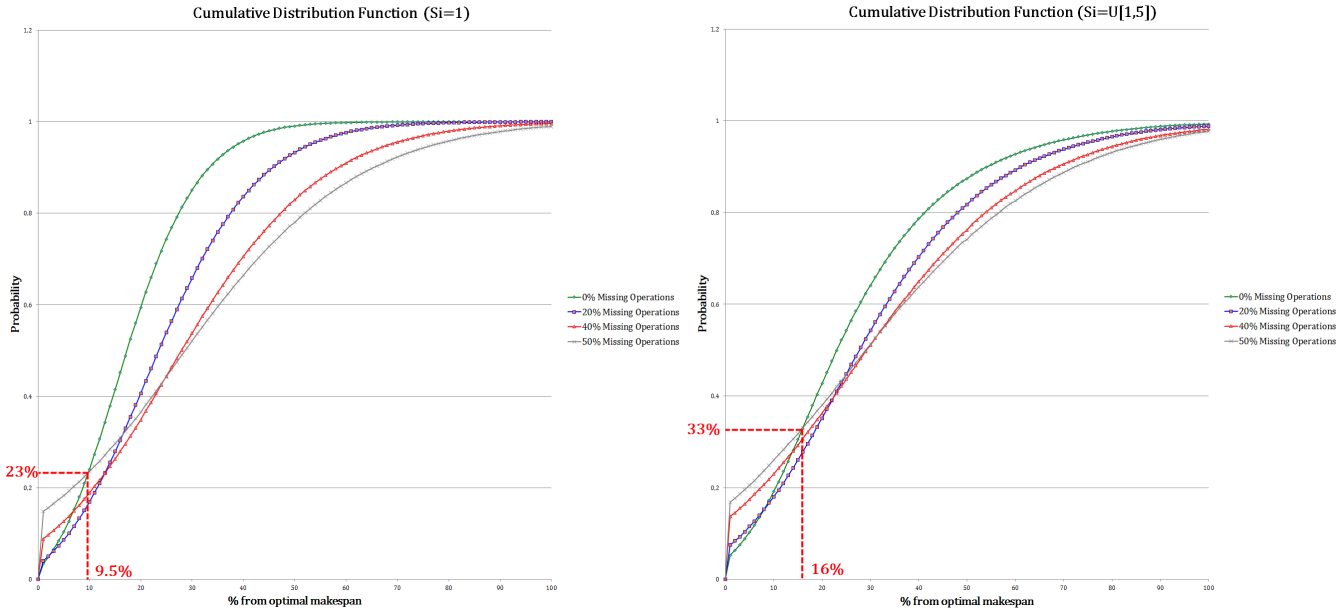

Figure 6.14: Hardness Analysis Comparison for $S_i = 1$ and $S_i = U[1, 5]$ (missing operations)

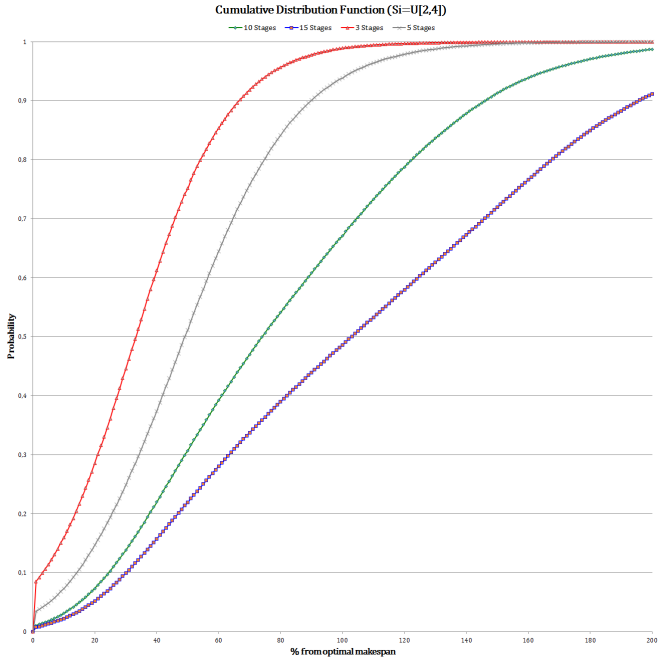

Figure 6.15: Hardness Analysis for $S_i = U[2, 4]$ (stages)

6.3.2 New Efficient Heuristics for the HFSMO Problem

6.3.2.1 Heuristics Proposal

In the previous section we analysed the convenience of developing new heuristics for the HSFMO problem. To do so we try to take advantage of the special features of the missing operations to develop a number

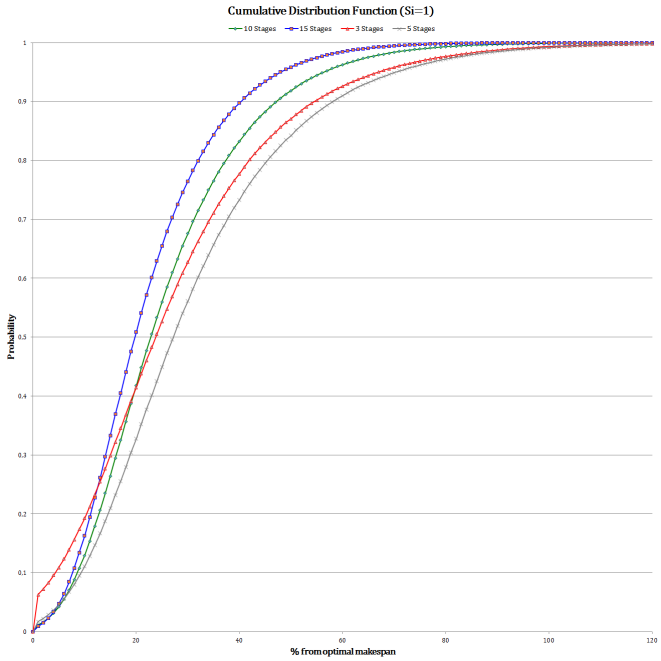

Figure 6.16: Hardness Analysis for $S_i = 1$ (stages)

of heuristics for this problem. First, we propose a number of modifications of two well-known dispatching rules, i.e. the Shortest Processing Time (SPT) rule, and the Longest Processing Time (LPT) rule. More specifically, the following dispatching rules are proposed:

- SPTswm (SPT - Stages Without Missing operations) and LPTswm (LPT - Stages Without Missing operations), in which we sort the jobs according to the sum of processing times divided by the number of stages without missing operations, the first in ascending order and the second in descending order. With this approach we try to remove the effect of processing times equal to zero.

- The next two dispatching rules are SPTw (SPT - Weighted) and LPTw (LPT - Weighted). These two heuristics first apply a weight to every stage according to the number of machines in each stage and then sort the processing times weighted by this factor in each stage, the first in ascending order and the second in descending order. The idea is to take into account the processing capacity of each stage, favouring those jobs with higher processing times in stages with more capacity and those with shorter processing times, respectively.

- We mixed the previous dispatching rules obtaining SPTwswm (SPT -Weighted Stages Without Missing operations) and LPTwswm (LPT - Weighted Stages Without Missing operations), where we apply the factors refering to the number of stages without missing operations and to the number of machines per stage.

```
%(DFF_N(a),DFF_R(a))
Π := NEH,Π := Rajendran
Δ^s := Jobs with missing operations in stage s
for s = 1 to S do
    while jobs in Δ^s do
        Remove job δ_i^s from Δ^s and Π
        Insert δ_i^s in a positions in Π to obtain sequences Π^n's
        Evaluate makespan of Π^n's
        Select Π^n* with minimum makespan
        Π = Π^n*
    end
end
Return Π
```

Figure 6.17: DFF Heuristics Pseudocode

- Finally, we apply the backward version of these heuristics as commented in Pan et al. (2014), i.e. we assign jobs to machines starting from the last stage instead of starting from the first and then they are assigned to the preceding stages by calculating their backward release times. To denote the backward version of the dispatching rules we add a b at the beginning of the name, e.g. bSPTswm is the backward version of the SPTswm rule.

In addition, we also propose a set of improvement heuristics, denoted as DFF_N(a) and DFF_R(a), that try to improve the solutions obtained by (Nawaz et al., 1983) and (Rajendran, 1993) by searching in the neighborhood of jobs with missing operations (see pseudocode in Fig. 6.17). More specifically, let Π_{NEH} and Π_{Raj} be the sequences obtained by the NEH heuristic (see pseudocode in Fig. 6.18) and Rajendran heuristic (see pseudocode in Fig. 6.19) respectively. Additionally, let q^s be the number of missing operations in stage s and let $\Delta^s := \delta_1^s, \ldots, \delta_{q^s}^s$ be the set of jobs containing missing operations in stage s. Then, the proposed heuristics first calculate Δ^s for each stage s and the initial sequences of jobs, denoted as Π. Note that DFF_N(a) heuristic uses Π_{NEH} as its initial solution whereas DFF_R(a) uses Π_{Raj}.

Once the initial solution has been obtained, we use the following procedure:

1. Starting from the first stage, i.e. $s = 1$, remove job δ_i^s from Δ^s.

2. Insert job δ_i^s in the first and last $a/2$ positions of sequence Π and denote the new sequence of jobs as Π^n.

3. Evaluate the objective function of Π^n. If the solution has been improved, Π is updated by Π^n.

4. The steps 1, 2 and 3 are repeated until there are no more jobs in Δ^s

%(NEH Heuristic)
Generate a job sequence $\Pi := [\pi_1, \pi_2, \cdots, \pi_n]$ using dispatching rule LPT
for $i = 1$ **to** n **do**
  Take job π_i from Π
  Move job π_i to all possible positions in Π and calculate makespan
  Select the sequence with minimum makespan
end
Return Π

Figure 6.18: NEH Heuristic Pseudocode

%(Rajendran Heuristic)
Generate a job sequence $\Pi := [\pi_1, \pi_2, \cdots, \pi_n]$ using dispatching rule LPT
Place first job in the partial sequence Π^*, i.e. $\Pi^* = [\pi_1]$
Number of jobs in π^* equal to 1 ($n^* = 1$)
for $i = 2$ **to** n **do**
  Calculate $lb = |(n^* + 1)/2|$ and $ub = (n^* + 1)$
  Remove next job from Π
  Insert the removed job in Π^* in the $lb < \rho < ub$ positions
  Evaluate makespan of partial sequences
  Select sequence with minimum makespan and update Π^*
  $n^* = n^* + 1$
end
Return Π

Figure 6.19: Rajendran Heuristic Pseudocode

Finally, two modifications of DFF_N(a) are proposed. DFF_N(100) with Random Insertions (DFF_N(100)_RI) works in the same manner as DFF_N(100) but, for each insertion of a job containing missing operations there is a small probability (10%) of selecting a job without missing operations to be inserted. With this proposal we want to know whether a better solution can be obtained by considering exclusively jobs with missing operations, or if we can obtain a better solution by including also jobs without missing operations. We also propose DFF Stage Dependent (DFF_SD), that works in the same manner as DFF_N(100), but instead of inserting jobs in all possible positions, the insertions are dependent on the stage where the missing operations appear, i.e. if a job contains missing operations in the first $\frac{s}{2}$ stages, the insertions are only made in the first $\frac{n}{2}$ positions of the sequence, and if the job contains the missing operations in the second $\frac{s}{2}$ stages, the insertions are made in the second $\frac{n}{2}$ positions of the sequence. The aim of this proposal is to know whether it is worth to insert jobs in all possible positions, or the improvement is dependent on the stage where missing operations appears.

Numerical Example for DFF_N(a)

To clarify how the proposed heuristics work, here we show a simple numerical example of the DFF_N(100%). Assume a hybrid flowshop composed of three stages with two machines in the first stage, one machine

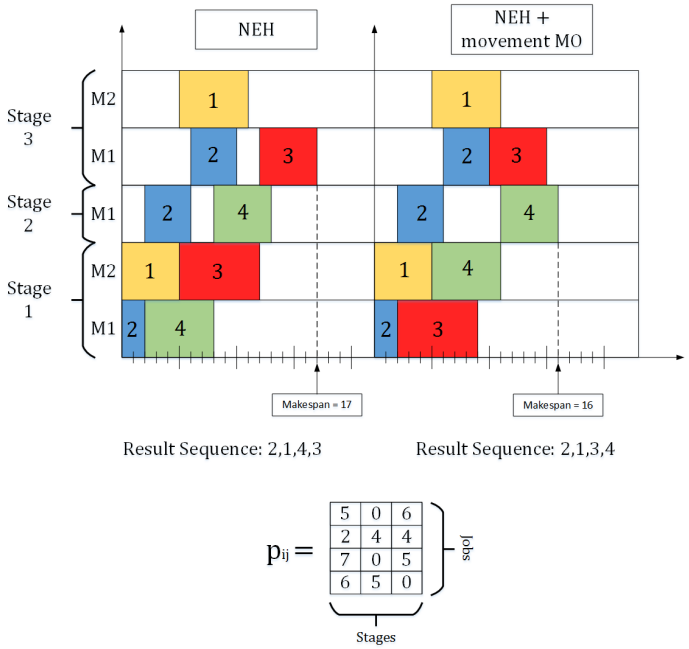

Figure 6.20: Example of DFF_N

in the second stage and two machines in the last stage. A set of four jobs whose processing times are depicted in Fig. 6.20 is to be scheduled. As we can see, missing operations appear both in second and third stages. In this example we can see how a reduction in the makespan can be achieved by applying the proposed heuristic. In the left part of Fig. 6.20 we can see the result of applying the NEH heuristic to the problem, obtaining a makespan value of 17 for sequence $\Pi = [2, 1, 4, 3]$. In the right part, we see that by applying one loop of the new heuristic an improvement of the makespan can be obtained. In this example we move the job with a missing operation in the last stage (job 3), and try it on the four possible positions, obtaining makespans values of 19,18,17 and 16. Therefore, we select the last possible sequence $\Pi = [2, 1, 3, 4]$ with an improved makespan.

6.3.2.2 Computational results

In this section we analyse the performance of our proposals by comparing them with a set of heuristics from the literature. To do so, we test a set of 56 different heuristics classified into 2 groups: dispatching rules (24) and improvement heuristics (32). Among the 24 dispatching rules we use the new proposals discussed in Section 6.3.2.1. In addition, the followings dispatching rules are tested:

- Shortest Processing Time (SPT). The jobs are ordered according to the sum of their duration in each stage in non descending order.
- Longest Processing Time (LPT). The jobs are ordered according to the sum of their duration in

each stage in descending order.

- Shortest Processing Time at the First Stage (SPTF). The jobs are ordered according to their duration in the first stage in non descending order.
- Longest Processing Time at the First Stage (LPTF). The jobs are ordered according to their duration in the first stage in descending order.
- Shortest Processing Time in the Bottleneck stage (SPTB). The jobs are ordered according to their duration in the bottleneck stage, i.e. the stage with the highest sum of processing times for all jobs, in non descending order.
- Longest Processing Tme in the Bottleneck stage (LPTB). The jobs are ordered according to their duration in the bottleneck stage in descending order.

We also added the backward version associated to all dispatching rules as commented in Section 6.3.2.1.

We also compare our heuristics against more complex procedures. We use the four heuristics for the HFS problem discussed in Section 6.1.1, and the best performing metaheuristics for the HFSMO problem by Saravanan et al. (2014) and Marichelvam and Prabaharan (2014). More specifically:

- We adapt the original NEH heuristic by applying the above dispatching rules as initial solutions. We denote them in the following as NEH_y, where y is the dispatching rule used as initial solution.
- As in NEH, we extend the analysis by using all previously described dispatching rules. We denote these heuristics as Raj_y, where y is the dispatching rule used as initial solution.
- For the heuristic by Kizilay et al. (2014), the parameter x is set to be equal to the number of jobs (n), as this case provides the best results according to makespan. Let us denote in the following WT1_NEH(n) and WT2_NEH(n) as WT1_NEH and WT2_NEH respectively.
- The Simulated Annealing algorithm by Saravanan et al. (2014) is included in the comparisons (henceforth SA_Saravanan).
- The IHGSS by Marichelvam and Prabaharan (2014) is included (henceforth IHGSS_Marichelvam).

To compare the different procedures, we designed a benchmark composed of 30 instances for each combination of the following factors. We considered 8 levels for the number of jobs, i.e. $n = \{5, 10, 20, 30, 40, 50, 100, 200\}$. The processing times were assumed to be drawn from a uniform distribution between 1 and 99. For the percentage of missing operations, we assumed 0%, 20%, 40% and 60% as discussed in Section 6.3.1. Based on

Table 6.2: Test Bed for Performance Analysis

Parameter	**Values**
n	5,10,20,30,40,50,100,200
$p_{i,j}$	$U[1,99]$
% Missing Operations	0%,20%,40%,60%
s	3,5,10,15,20
S_i	$U[1,5]$
Number of Instances	30

the benchmark by Carlier and Neron (2000) where authors use 5 and 10 stages, we use $S = \{3, 5, 10, 15\}$. Finally, with respect to the number of machines per stage, we based our benchmark in Naderi et al. (2010), where authors use a constant ($S_i = 2$) and an uniform distribution ($S_i = U[1, 4]$). We extend the analysis to $S_i = \{1, 2, U[1, 5], U[2, 4]\}$. A summary of the parameters used to test our heuristics is shown in Table 6.2. The results are evaluated by means of the average Relative Percentage Deviation (RPD) and the Average Computational Time, defining

$$RPD_i = \frac{C_{\max_i} - C^{best}_{\max}}{C^{best}_{\max}} \cdot 100$$

as the Relative Percentage Deviation of heuristic i for an specific instance, $C_{\max_i}$ as the makespan obtained by heuristic i for that instance and $C^{best}_{\max}$ as the best makespan obtained by any heuristic in that instance.

All algorithms have been coded in the same programming language (C#) and the set of instances was executed under the same conditions, i.e. using the same computer (Intel Core i7-3770 with 3.4 GHz and 16 GB RAM) and the same libraries. In the next subsections we present the results of the computational experience.

Dispatching rules

In Table 6.3, we can see the results of the comparison among the 24 dispatching rules for each level of missing operations, as well as the aggregated results. The table shows the ranking of the dispatching rules according to ARPD (we focus only on ARPD as the computational times are almost negligible). Note that, as we are presenting ARPD for instances with no missing operations, the shaded cells provide the same results as the cells just above.

For those instances with missing operations, there are 3 dispatching rules dominating the other 19: bSPTB,bSPT and LPT. In Table 6.4 we present the numerical results according to the size of the instances, classifying them into big (100 and 200 jobs), medium (30, 40 and 50 jobs), and small instances (5, 10 and 20 jobs). Although the best performing dispatching rules for all cases is bSPTB, depending on the size

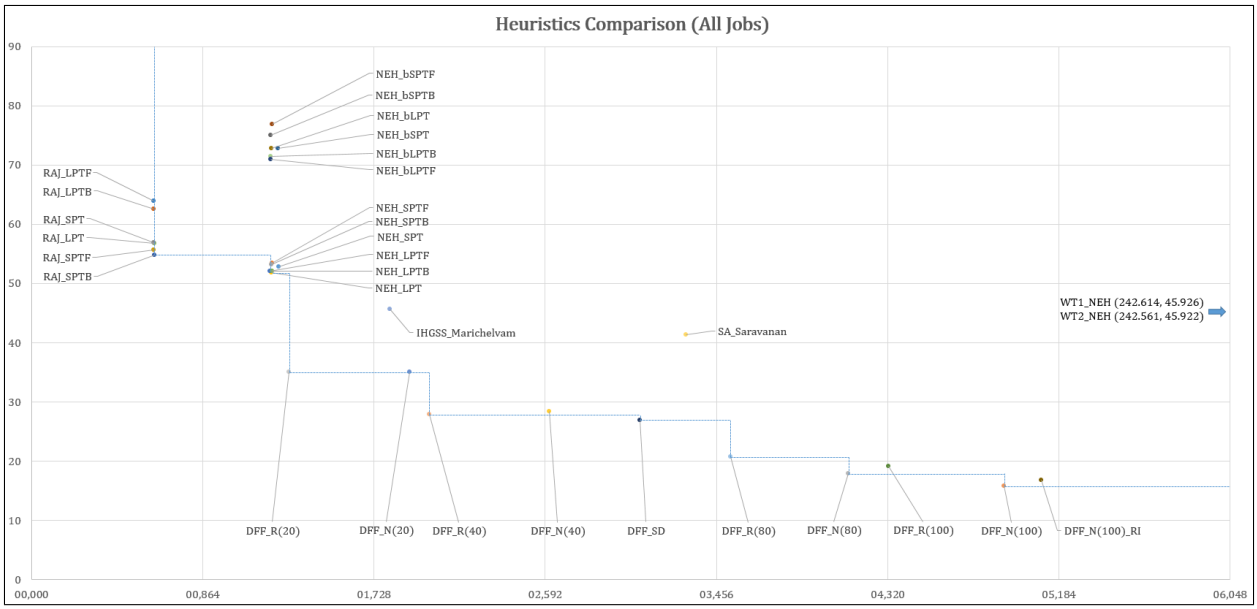

Figure 6.21: Heuristics Results for All Instances

of the instances and on if only missing operations or all jobs are considered together, in some cases LPT performs better than bSPT.

As it can be seen, the proposed dispatching rules are not the best performing ones, so we can conclude that more complex rules are needed to exploit better these features, as will be seen in the next subsection.

Improvement Heuristics

The results of performance of the 32 improvement heuristics are shown in Figs. 6.21 and 6.22. Results for the case of considering all instances and the case with only those instances with missing operations are presented, respectively. The behavior of the heuristics in both figures is similar, but the improvement obtained by the new heuristics in terms of ARPD is reduced when all instances are considered. This is foreseeable as our heuristics have the same performance as other heuristics when there are no missing operations. Taking this into account, we focus on the results of the instances containing missing operations (see Fig. 6.22). We define a number of groups according to Computational Time to ease the comparison between the heuristics.

A number of conclusions can be drawn from our comparison, which can be statistically justified by means of Holm's procedure (Holm (1979)). In this procedure, each hypothesis is analysed using a non-parametric Mann-Whitney test (see e.g. Fernandez-Viagas and Framinan (2015)). We first sort the hypotheses in non descending order of the p-values found in the test. And next, we check all of them and reject hypothesis i if its p-value is lower than $\frac{\alpha}{(k-i+1)}$, where k is the total number of hypothesis. The

Table 6.3: Dispatching Rules Performance

	0% Missing Operations		20% Missing Operations		40% Missing Operations		60% Missing Operations		All instances	
Ranking	Disp. Rule	ARPD	Disp. Rule	ARPD	Disp. Rule	ARPD	Disp. Rule	ARPD	Disp. Rule	ARPD
1	bSPTB	8,46	bSPTB	12,10	bSPTB	12,01	bSPTB	12,33	bSPTB	11,22
2	bSPTF	10,69	LPT	12,21	bSPT	13,21	LPT	12,95	LPT	12,58
3	LPT	10,92	bSPT	12,84	LPT	14,25	bSPT	13,67	bSPT	12,76
4	LPTswm	10,92	bSPTw	13,55	bSPTw	15,07	LPTswm	14,79	bSPTw	13,94
5	bSPT	11,33	LPTw	13,63	bSPTswm	15,09	bSPTswm	15,31	bSPTswm	14,03
6	bSPTswm	11,33	bSPTwswm	14,31	LPTswm	15,58	bSPTw	15,35	LPTswm	14,05
7	bSPTwswm	11,74	bSPTswm	14,39	LPTw	15,86	LPTw	15,80	LPTw	14,42
8	bSPTw	11,79	LPTswm	14,91	bSPTwswm	15,99	bSPTwswm	16,99	bSPTwswm	14,76
9	LPTwswm	12,05	LPTwswm	15,45	LPTwswm	17,08	bSPTF	17,00	bSPTF	15,11
10	LPTw	12,38	bSPTF	15,49	bSPTF	17,28	LPTwswm	17,25	LPTwswm	15,46
11	SPTF	12,78	SPTF	17,84	SPTF	19,40	SPTF	17,26	SPTF	16,82
12	SPTB	12,89	SPTB	19,30	SPTB	20,63	SPTB	19,06	SPTB	17,97
13	bLPT	17,98	LPTB	20,93	bLPTswm	21,63	bLPTswm	20,16	bLPTswm	20,77
14	bLPTswm	17,98	SPTswm	22,31	bLPTwswm	22,02	bLPTwswm	20,71	bLPTwswm	20,95
15	bLPTw	18,01	bLPTwswm	22,75	SPTswm	22,60	SPTswm	21,75	SPTswm	21,4
16	bLPTwswm	18,31	bLPTswm	23,31	SPTwswm	22,88	SPTwswm	22,15	SPTwswm	21,86
17	SPTw	18,83	SPTwswm	23,47	LPTB	24,40	LPTB	24,95	LPTB	22,37
18	SPTwswm	18,93	bLPTF	24,82	bLPTF	25,57	bLPT	25,08	bLPT	23,95
19	SPT	18,97	SPT	25,59	bLPT	26,65	bLPTF	25,54	bLPTF	23,98
20	SPTswm	18,97	LPTF	25,67	SPT	27,07	bLPTw	26,00	bLPTw	24,46
21	LPTB	19,18	bLPTw	25,78	bLPTB	27,15	SPT	26,79	SPT	24,6
22	bLPTF	19,99	bLPT	26,10	SPTw	27,85	bLPTB	27,18	SPTw	25,07
23	LPTF	20,13	SPTw	26,21	bLPTw	28,06	SPTw	27,41	bLPTB	25,39
24	bLPTB	20,61	bLPTB	26,62	LPTF	30,58	LPTF	33,32	LPTF	27,43

Table 6.4: Numerical Results (ARPD) for best performing dispatching rules

Size	Type	LPT	bSPTB	bSPT
BIG (100, 200 jobs)	**All Instances**	2.987	**1.923**	3.002
	Only Missing Operations	2.812	**2.044**	2.801
MEDIUM (30, 40, 50 jobs)	**All Instances**	8.541	**6.278**	8.056
	Only Missing Operations	8.704	**6.912**	8.192
SMALL (5, 10, 20 jobs)	**All Instances**	23.011	**22.372**	23.977
	Only Missing Operations	24.440	**24.117**	25.249

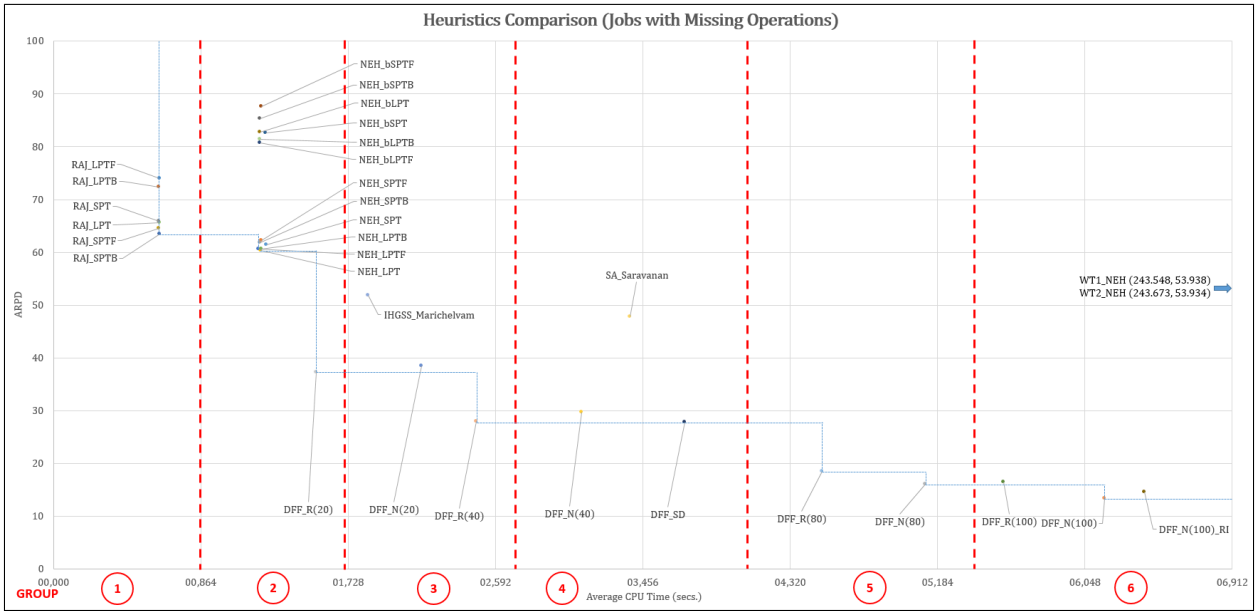

Figure 6.22: Heuristics Results for Instances with Missing Operations

results of the procedure for the conclusions shown below are presented in Table 6.5. We also present the mean and least significance difference (LSD) 95% intervals in Figure 6.23.

The following conclusions for the groups (and the hypotheses for the subsequent statistical analysis) have been analysed:

- Group 1. Here we consider all Raj heuristics. We see how these heuristics achieve the shortest computation times. However, the value of their ARPD is not as good as it is for the other heuristics. To analyse these heuristics we check if the following two hypotheses hold, i.e. in H_1 we compare the best performing Raj heuristic (Raj_SPTB) with the second one (Raj_SPTF) and in H_2 with the worst performing one (Raj_LPTF). According to Table 6.5, the first hypothesis cannot be rejected, meaning that we cannot assure that there is a significant difference between the two heuristics, while the second can be rejected, i.e. those two heuristics are significatively different.

  H_1: Raj_SPTB = Raj_SPTF

Table 6.5: Holm's Procedure

H_i	p-value	Mann-Whitney	$\frac{\alpha}{k-i+1}$	Holm's Procedure
H_2	0.000	R	0.0031	R
H_4	0.000	R	0.0033	R
H_5	0.000	R	0.0035	R
H_6	0.000	R	0.0038	R
H_7	0.000	R	0.0042	R
H_9	0.000	R	0.0045	R
H_{10}	0.000	R	0.0050	R
H_{11}	0.000	R	0.0056	R
H_{12}	0.000	R	0.0062	R
H_{13}	0.000	R	0.0071	R
H_{14}	0.000	R	0.0083	R
H_{15}	0.000	R	0.0100	R
H_{16}	0.000	R	0.0125	R
H_1	0.194		0.0167	
H_8	0.260		0.0250	
H_3	0.769		0.0500	

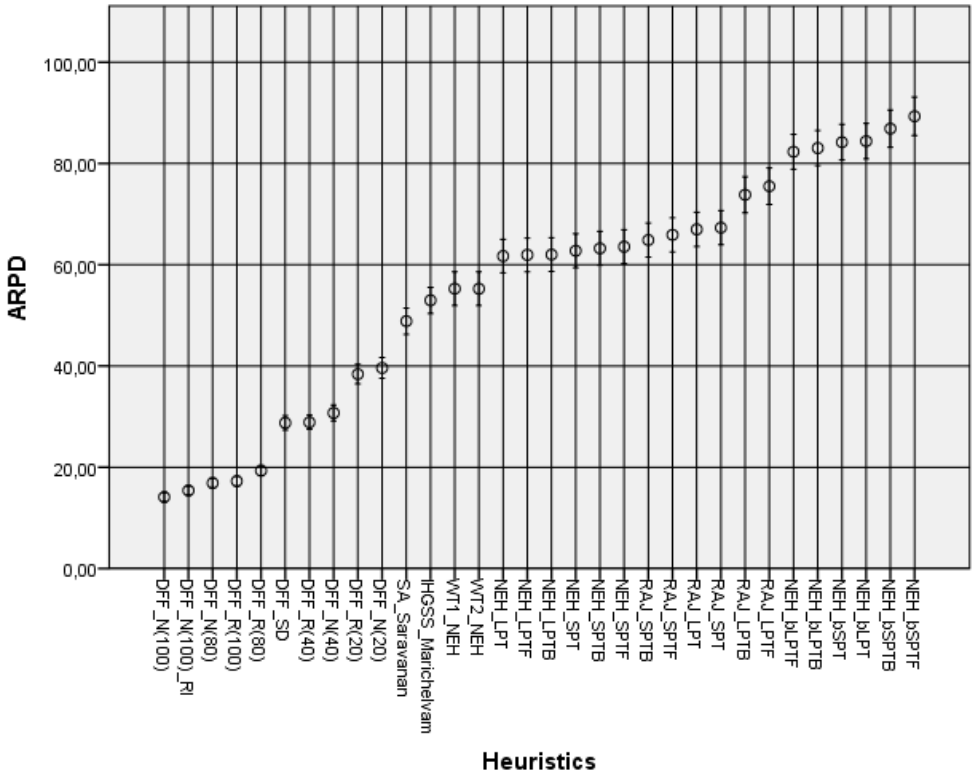

Figure 6.23: Mean and LSD intervals 95% for RPD

H_2: Raj_SPTB = Raj_LPTF

- Group 2. In this group we find NEH heuristics and DFF_R(20).The first conclusion is that NEH heuristics initialised with backward dispatching rules do not perform well for the problem under consideration, obtaining the worst ARPD and a computational time not very competitive. Among

NEH heuristics, the best performing one is NEH_LPT, although NEH_LPTB and NEH_LPTF have a similar performance. Finally we see that DFF_R(20) performs better with respect to ARPD , although it has a larger computation time. To check these conclusions we test the following three hypotheses, i.e. in H_3 we compare the best performing NEH heuristic (NEH_LPT) with the second one (NEH_LPTF), in H_4 we compare the best performing heuristic (NEH_LPT) with the worst performing one (NEH_bSPTF) and finally in H_5 we compare the best NEH heuristic (NEH_LPT) with one of our proposals (DFF_R(20)). According to Table 6.5, the first hypothesis cannot be rejected, i.e. NEH_LPT is not significatively different from NEH_LPTF,i.e. we cannot assure that NEH_LPT performs better than NEH_LPTF, but H_4 and H_5 can be rejected, i.e. there are significant differences between the best and worst NEH heuristics and between the best NEH and our proposal DFF_R(20), being our proposal the best performing heuristic of this group.

H_3: NEH_LPT = NEH_LPTF

H_4: NEH_LPT = NEH_bSPTF

H_5: NEH_LPT = DFF_R(20)

- Group 3. We have two of our proposals, i.e. DFF_N(20) and DFF_R(40), and the metaheuristic IHGSS_Marichelvam. We can conclude from Fig. 6.22 that the best performing one is DFF_R(40) although it takes some more CPU time than DFF_(20). IHGSS_Marichelvam takes less time than the other two heuristics but the achieved ARPD is also higher. To assert these conclusions we analyse the following two hypotheses, i.e. in H_6 we compare DFF_R(40) and DFF_N(20) and obtain, according to Table 6.5, that there are significative differences between the two heuristics. We also check H_7 where our proposal DFF_R(40) is compared with IHGSS_Marichelvam. It can be seen that there are significative differences between them. In the light of the results we can affirm that DFF_R(40) is the best performing heuristic of this group.

  H_6: DFF_R(40) = DFF_N(20)

  H_7: DFF_R(40)= IHGSS_Marichelvam

- Group 4. This case is similar to the previous one. We consider here our two propsals DFF_SD and DFF_N(40), and SA_Saravanan. The conclusions that can be drawn for this group are also similar as, from Figure 6.22 it can be seen that DFF_SD is the best performing heuristic of the group in terms of quality of solution, i.e. ARPD, although DFF_N(40) has a lower average CPU time. It can be also seen that SA_Saravanan is not efficient. To check these conclusions we include two new hypotheses: In H_8 we compare DFF_SD and DFF_N(40) and we obtain that there are not

significative differences between them, i.e. we cannot say that one is better than the other. In H_9 we compare DFF_SD and SA_Saravanan, and we find that their ARPD values are significatively different.

H_8: DFF_SD = DFF_N(40)

H_9: DFF_SD = SA_Saravanan

- Group 5. In this group we consider DFF_N(80) and DFF_R(80). We conclude that DFF heuristic initialised with NEH heuristic performs better than when initialised with Raj heuristic (see Fig. 6.22), although it needs more CPU time. From Table 6.5 we see that hypothesis H_{10} can be rejected, i.e. the compared heuristics are significatively different, being DFF_N(80) the best performing heuristic of the group.

  H_{10}: DFF_N(80) = DFF_R(80)

- Group 6. In this last group we include the most time-consuming heuristics, i.e. DFF_R(100), DFF_N(100), DFF_N(100)_RI, WT1_NEH and WT2_NEH. We see that DFF_N(100) performs better in ARPD than the other ones, and performs better in CPU time than all of them except from DFF_R(100). We compare DFF_N(100) against the other 4 heuristics, as can be seen in hypotheses H_{11}, H_{12}, H_{13}, and H_{14}. Regarding Table 6.5, we see that all the hypotheses can be rejected, meaning that there are significant differences between these heuristics, and therefore we conclude that our proposal DFF_N(100) is the best performing one.

  H_{11}: DFF_N(100) = DFF_R(100)

  H_{12}: DFF_N(100) = DFF_N(100)_RI

  H_{13}: DFF_N(100) = WT1_NEH

  H_{14}: DFF_N(100) = WT2_NEH

- Additional Analyses. We also compare the already existing heuristics for the problem under study, i.e. IHGSS_Marichelvam and SA_Saravanan, with the new proposals with better ARPDs in similar CPU times, i.e. DFF_R(20) and DFF_N(40) respectively. We see that the new proposals performs better for both cases. To confirm this conclusion we include hypotheses H_{15} and H_{16} to the study and according to Table 6.5, we see how there are significative differences.

  H_{15}:DFF_R(20) = IHGSS_Marichelvam

  H_{16}:DFF_R(40) = SA_Saravanan

6.4 Implementation Results

As we already commented, the development of the system was partly supported by a real project with a private company. However, the framework and a prototype of the system was already prepared before starting. Therefore, the biggest effort when developing the final system was to accurately gather all the requirements of the company and to incorporate them into the system. Among these requirements, apart from the use cases already described above, we can highlight:

- Possibility of configuring and managing the layout of the factory in an easy way. In the last years, due to an increase in the demand of products, the company has been incorporating new resources, so it was important to be able to add them to the system in an interactive way.
- Route management. For some products, it is possible to follow different pathways for its manufacturing, so this requirement was also implemented in the system. The configuration of the routes is made when defining the products in the system.
- Reporting. A reporting module for different profiles within the departments in the company was also a requirement for the system, e.g. information for the planner about the ratio of occupancy of the resources during the planning horizon, information for the human resources department about the working hours of each worker, etc.
- Quality issues. According to the guidelines of the quality department of the company, it was necessary to fill a questionnarie and to associate some documents to the finisihing of some specific tasks. So, after defining in the system which are these tasks and what should be introduced with each of them, the system prompts the user to fullfil these requirements every time he/she completes one of these tasks.
- Different technological aspects. The company was immerse in the updating of all its systems, so it was also necessary to allow for different types of connections to databases from the system. According to this, the possibility of configuring the data sources for the system was also included.

After some iterations with the department responsible of IT in the company and some adjustments in the system, we obtained the final version that was further deployed. From the feedback received we know that the system was in use and that some improvements on the functioning of planning were reached, such as an important improve in the time spent in doing the planning, a better communication between the planner and the other departments involved in the manufacturing process,etc. However we do not have at our disposal deeper information about the current state of the system in the company.

To finish this section, we just want to note that a number of different companies, from different sectors, have been interested in continuing with the development of our system, and we are currently trying to stablish new projects that allow us to improve the system in different aspects that will be discussed in the last chapter of this thesis.

6.5 Conclusions

In order to validate the framework proposed in Chapter 3 in a real environment, we present a manufacturing DSSTS that was developed for a plastic manufacturing company from Sevilla. We described the manufacturing process and the associated scheduling problem, known in literature as hybrid flowshop scheduling problem with missing operations. Then we made a detailed description of the main use cases of the DSSTS: the Alternatives Analysis – that offers an idea on the capacity of the plant to perform certain tasks–, the Long Term Scheduling – that allows the planner to know if it is possible to perform customer orders within the due dates–, Short Term Scheduling –that offers a detailed schedule of tasks–, Rescheduling – that permits the scheduler to change an ongoing schedule in the case a disturbance arises during its execution– and Control –that allows a real time monitoring of the execution of the schedule–. Next, we proposed a set of solution procedures to tackle this problem. To do so, first an analysis of the effect of missing operations on the HFS problem with makespan minimisation criterion was carried out. We studied the empirical hardness of this problem as compared to the classical HFS. The analysis showed that, depending on the prevalence of missing operations in the jobs, the problem is different than the classical HFS without missing operations. We also analysed a number of different factors (number of jobs, number of stages, etc.) influencing the structure of solutions of the problem. To take advantage of the special behaviour of the missing operations, we developed a number of dispatching rules and improvement heuristics that were compared with already existing ones both for the special case with missing operations, and for the general HFS case. From this comparison, we found that our proposals perform more efficiently than existing heuristics, therefore representing a new state-of-the art for the problem. These algorithms were embedded into the DSSTS. Finally, we provided some comments about the deployment of the system in the target plant.

Chapter 7

The Healthcare Case

To continue with the validation of the framework presented in Part II, this chapter applies it to a completely different sector, the healthcare. As in previous chapter, we describe the implementation of an operating room DSSTS, that was designed following the framework proposed in Chapter 3. We start by detailing the context where the DSSTS was deployed and describing the problem addressed, i.e. the operating room scheduling problem. Then, its main functionalities are described. Next, a set of efficient heuristics for this particular case is proposed, and finally, the main results of the implementation of the DSSTS are discussed.

7.1 ASSYST: An Operating Room DSSTS

The problem we address in this section is based on a real problem from a large University Hospital located in the south of Spain (University Hospital "Virgen del Rocío"). In this hospital there are 16 different SUs, each one with assigned resources in terms of ORs, supporting staff (such as anesthetists and nurses) and material. Our objective is to help SU Directors to solve the Surgery Planning within each SU. We focus on elective patients since emergency patients in the Hospital are intervened in specific ORs, i.e. there are a number of ORs intended for emergency surgeries.

In addition to the common constraints found in the traditional operating room scheduling problem that will be disscussed in Section 7.1.1, when taking the implemented DSSTS into practice we found a number of new features that need to be considered:

(i) It must accomplish with DPA (Data Protection Act) –i.e. the system must be secured by checking user identity and that the host is licensed before executing the DSSTS.

(ii) Since the SU director usually decides the schedule using his/her own laptop –sometimes out of working hours–, the required tool is conceived to be a standalone system. As a consequence, the DSSTS is not integrated with the Hospital Information System (HIS), but imports from it the relevant data of patients in the waiting list and the corresponding intervention data, such as expected duration, surgeon (or group of surgeons) in charge, OR (or group of ORs) where the patient can be intervened, dead line, etc.

(iii) The optimization engine should provide a plan that can be manually modified by the SU director, so he/she can incorporate *soft* constraints that cannot be easily integrated in the model, such as the preference of using the first hours of a shift for certain types of interventions (not only depending on the type of intervention, but on the specific patient), or some shifts in the beginning/end of the week due to the specific needs of post-surgery recovery. Therefore, easy manual fine-tuning of the solution is required.

(iv) The DSSTS should provide detailed analysis tools and drill-down capabilities so the SU director can analyze the so-called *scenario* –i.e. a surgery plan arisen from a waiting list and staffed ORs for an specific planning horizon- with great detail. Consequently, the system should be capable of handling different possible scenarios, that is: several solutions of the decision problem with the same/different data and using same/different parameter settings must be maintained so that the SU director may explore their feasibility, introduce manual changes, etc. and ultimately choose one as an *executable* schedule.

(v) The DSSTS is required to be flexible and extensible, so that it satisfies the currently identified business rules while makes it easy to add new ones. Consequently, the tool should be modular to allow incorporating new decision problems (models/ heuristics) to the system.

(vi) Since, in most SUs, surgeons are organized in groups –i.e. patients may be assigned to a group of surgeons instead of to a single surgeon-, the DSSTS should allow for setting groups of surgeons and defining surgeons' availabilities within each group.

7.1.1 Operating Room Scheduling Problem

The efficient management of Operating Rooms (ORs) in hospitals is key to deliver surgical services at a reasonable cost while accomplishing patients' satisfaction. Particularly, OR scheduling must take into account the availability of different/specialized ORs, the clinical staff (most notably surgeons), patients' availability, among other constraints. Furthermore, given the inherent variability of surgical interventions,

monitoring and establishing corrective actions (such us last-minute adjustments due to patient no-shows or cancellations) is also required. Despite their importance and complexity, decisions related to OR management are usually made according to managers' experience without considering the underlying optimization problems (Brunner et al., 2009).

In all SUs in the Hospital, it has been decided to adopt the so-called open scheduling planning strategy, i.e. interventions can be assigned to any OR and at any staffed hour of a working shift (Dexter et al., 2003; Guerriero and Guido, 2010), in contrast to the so-called block scheduling policy where OR capacity is distributed between surgeons following a pattern (Gupta, 2007). In order to guarantee continuity of care, each patient is to be intervened by the same surgeon who treats him/her from the beginning of the surgical process. Therefore, the decision problem is to assign the tuple patient-surgeon in the waiting list to a given OR and for a given shift. For this problem, the literature usually considers the following constraints (see Table 7.1):

- Resources Capacities, since both surgical facilities (such as ORs, Intensive Care Units –ICUs–), and surgical personnel (surgeons, anesthetists, nurses, porters, etc.) may not be fully available during the planning horizon. In our case, only surgeons' availability is considered, assuming that the rest of the required staff and material is available. Surgeons' availability are a critical factor for some SUs in the hospital, as their working duties include interventions and consultation.

- Time windows. Depending on the intervention type and urgency, it is necessary to carry out the intervention within a specific time window (defined by a release and a due date). In our case, both release and due dates are explicitly considered in the constraints of the model.

- Forbidden ORs. Some types of surgeries have to be executed only in specific ORs because they require special devices or material. In our case, there are certain surgery procedures (such as microsurgery interventions) which need some special equipment, so they can be performed only in certain ORs.

- Maximum number of ORs where a surgeon can intervene in a specific shift. Looking for the comfort of surgeons and to avoid problems related to their moves between ORs, in most SUs, the number of ORs where a surgeon can intervene in a shift is limited. This is also the case in our problem.

As it can be seen in Table 7.1, there are references dealing with some of these constraints, but not all of them at a time. In addition, specific constraints have to be taken into account in our problem: For practical reasons, both the number of different surgical teams and the number of patients scheduled within an OR time are limited. Finally, specific patients' availability has to be taken into account.

Policy	Solution Approach	Decision Model	Reference	Advance Scheduling	Planning Horizon	Constraints				
						Resource Capacity		Due Dates	Forbidden ORs	Max. number of ORs
						Surgical Personnel	Surgical Facilities			
Block	Exact	Deterministic	Ozkarahan (2000)	X	1-2 Weeks	Surgeon	OR,ICU	-	X	-
			Testi and Tanfani (2009)	X	Weekly	Surgeon	OR,ICU	-	-	-
			Zhiming (2011)	X	Daily	Surgeon	OR	-	-	-
	Approximate	Deterministic	Aringhieri et al. (2015)	X	Weekly	Surgeon	OR,Ward	X	-	-
			Chaabane et al. (2008)	X	Weekly	Surgeon	OR	X	X	-
			Fei et al. (2006)	X	Weekly	-	OR	X	-	-
			Fei et al. (2008)	X	Weekly	-	OR	X	-	-
			Fei et al. (2009)	X	Weekly	Surgeon	OR	X	X	-
			Jeang and Chiang (2010)	X	2 Weeks	Surgeon	OR	-	-	-
			Meskens et al. (2013)	-	Daily	Surgeon, Nurses, Anesth.	OR,ICU	X	X	-
			Riise and Burke (2011)	X*	Daily	Surgeon	OR	X	X	-
			Roland et al. (2010)	X*	Daily/Weekly	Surgeon, Nurses, Anesth.	OR	X	X	-
			Tanfani and Testi (2010)	X	Weekly	Surgeon	OR,ICU	-	-	-
			Wang et al. (2010)	X	Weekly	Surgeon	-	X	-	-
		Stochastic	Hans et al. (2008)	X	Daily/Weekly	-	OR	-	-	-
			Lamiri et al. (2007)	X	Weekly	-	OR	-	-	-
			Lamiri et al. (2008b)	X	Weekly	-	OR	-	-	-
			Lamiri et al. (2008c)	X	Weekly	-	OR	-	-	-
			Lamiri et al. (2009)	X	Weekly	-	OR	-	-	-
			Min and Yih (2010)	X	Weekly	-	OR,ICU	-	X	-
Open	Exact	Deterministic	Jebali et al. (2006)	X	Daily	Surgeon	OR,ICU	X	X	-
			Ogulata and Erol (2003)	X	Weekly	Surgeon	OR	-	-	-
			Pham and Klinkert (2008)	X*	2 Days	Surgeon	OR,ICU	X	X	-
	Approximate	Deterministic	Augusto et al. (2010)	-	1-2 Days	Porters	OR,ICU	-	-	-
			Chaabane et al. (2008)	X	Weekly	Surgeon	OR	X	X	-
			Fei et al. (2010)	X	Weekly	Surgeon	OR	X	-	-
			Guinet and Chaabane (2003)	X	Weekly	Surgeon	OR	X	-	-
			Lamiri et al. (2008a)	-	2 Days	Surgeons,Porters	OR,ICU	-	-	-
			Liu et al. (2011)	X	Weekly	Surgeon	OR	X	-	-
			Marques et al. (2012)	X*	Weekly	Surgeon	OR	X	-	-
			Marichelvam and Prabaharan (2014)	X*	Weekly	Surgeon	OR	X	-	-
			This paper	X	Up to 3 months	Surgeon	OR	X	X	X
		Stochastic	Batun et al. (2011)	X*	Daily	Surgeon	OR	-	-	-
			Lamiri and Xie (2006)	X	Weekly	-	OR	-	-	-
			Wang and Xu (2008)	X	2 Days	Surgeon,Nurses	OR	-	-	-
	Both		Marcon et al. (2003)	X	Daily	Surgeon	OR	-	-	X

Table 7.1: Contributions to operational level in literature. * Integrated Approach (Advance and Allocation Scheduling)

Regarding the objective function, usual goals considered in the literature include:

- Minimizing the delay in the surgery, i.e. minimizing the period of time between first consultation and intervention for each patient (Guinet and Chaabane, 2003; Ogulata and Erol, 2003).
- Minimizing overtime (Hans et al., 2008).
- Minimizing fixed patients costs (Fei et al., 2008; Jebali et al., 2006), i.e. minimizing those costs that are not related to the number of interventions that have to be carried out, like the labor cost of the OR team.
- Maximizing OR utilization (Hans et al., 2008; Ogulata and Erol, 2003), i.e. maximizing the percentage of available OR time used for interventions.
- Minimizing the risk of no realization (RNR) (Marcon et al., 2003), i.e. minimizing the probability that an intervention cannot be executed in its planned date.
- Maximizing patients satisfaction (Min and Yih, 2010; Pariente and Framinan, 2009; Ozkarahan, 2000; Sier et al., 1997), i.e. generating schedules with feasible cumulative patients' priority

In our case, the main goal –set by the director of the Hospital– is the maximization of an indicator of the quality of service combining two aspects: patient's medical priority and the need to fulfill the standards imposed by the Regional (Andalusian) Healthcare Administration. These standards establish that, for each type of illness and its corresponding surgery procedure, the time from diagnosis to intervention should not exceed a maximum number of days. If this number of days is exceeded, the Hospital may be liable for the expenses and complications associated to this delay. Such number is labeled *clinical guarantee* and is denoted by *cg* in the following.

The first aspect of the indicator —medical priority— is addressed by ranking each patient in the waiting list according to a medical priority valuation carried out by the SU director in view of the reports from the surgeons in charge of the intervention. This rank does not only take into account the type of illness of each patient, but also his/her associated medical risks. As a result, an integer number *mp* is assigned to each patient. Each SU uses specific ranks, i.e. while in some SUs patients are ranked either 1 (*normal*) or 2 (*preferential*), others use a number between 1 and 5, being 5 the highest priority. In order to make the rankings homogeneous among SUs, a normalized rank $mp^* = \frac{mp}{hmpv}$ is obtained, where *hmpv* is the highest medical priority value used within the SU. Medical priority–based rankings appear frequently in the literature. For instance, Sier et al. (1997) consider three classes of patients and correspondingly

penalize the delay of surgery cases, while Ozkarahan (1995, 2000) use the medical priority as one of the conflicting objectives within a goal programming approach.

The second aspect of the indicator -–waiting times-– is dealt with in the following manner: given *dwl* the number of days that a patient has been in the waiting list, and the clinical guarantee *cg* for the associated surgical procedure, a normalized indicator $dwl^* = \frac{dwl}{cg}$ is obtained.

The two aspects mentioned above are linearly combined so a so-called clinical weight parameter w_p is obtained for each patient, i.e. $w_p = a \cdot mp^* + (1-a) \cdot dwl^*$, where a is a parameter set by the SU Director in view of the internal objectives of each SU at any given planning period: for instance, if the waiting list is growing, the SU Director may wish to reinforce the fulfillment of the clinical guarantee, while in other situations, strictly following a medical priority is more desirable. Other work addressing elective surgery planning considering both medical priority and surgery deadline is Ogulata and Erol (2003).

7.2 Main Use Cases

Taking into account the above requirements and the proposed framework, the main use cases of the DSSTS (shown in Figure 7.1) are:

1. **Medium Term Estimation**, with the objective of generating a tentative surgical plan for a period of up to six months by assuming a weekly pattern (i.e. same ORs and surgeons availability in all weeks). The purpose is twofold: Check whether the available surgical resources pattern (ORs, Surgeons, and working shifts) is sufficient to accomplish the interventions in the waiting list in a proper manner, and to notify the patients with an estimated week for their interventions. To develop this plan, the heuristics methods described in Section 7.3.2 are employed. This case study corresponds to component *Decision Problems Handling* in the proposed framework.

2. **Short Term Planning**. The objective of this use case is to obtain a detailed surgical plan for a short planning period (typically the next two weeks) over a rolling-horizon basis. More specifically, at the end of each week, the SU Director imports the waiting list from the Hospital Information System, refines the availability pattern of resources along the next two weeks by incorporating specific events (closure of certain OR, punctual non-availability of a surgeon, etc.) and generates a detailed surgical plan for the next two weeks using either the MILP model of Section 7.3.1 or the heuristics presented in Section 7.3.2. The choice of exact/approximate methods is left to the SU director in view of the size of the problem. It is also possible to specify the maximum running time allowed to generate the planning so the DSSTS may choose the best method. This case study

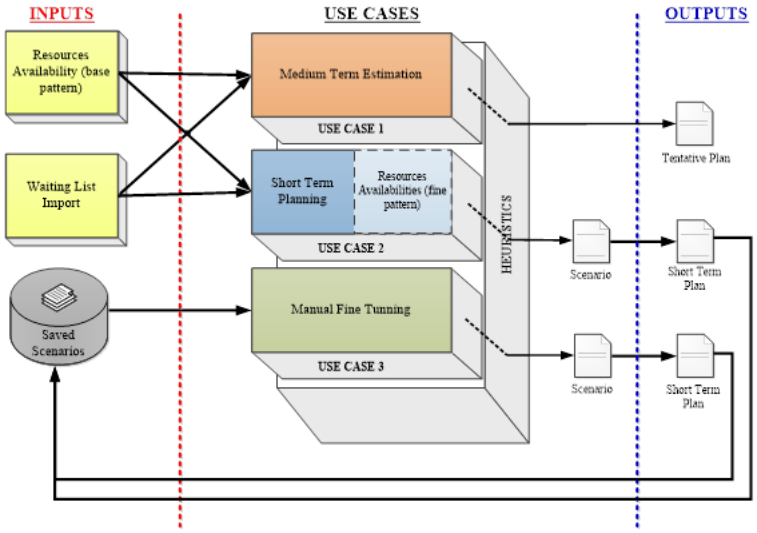

Figure 7.1: Main use cases

corresponds to components *Decision Problems Handling* and *Solution Approaches Library* in the proposed framework.

3. **Manual fine-tuning**. As stated before, a requirement for the DSSTS was that the SU director will be able to move any of the scheduled interventions within the short term surgical plan, whether to postpone it (e.g. a patient has flu or some health complication impeding the intervention), or to put them into an specific OR time. Moreover, not scheduled patients could also be manually allocated into a specific OR time. These manually allocated patients are considered frozen when invoking again the optimization engine. This case study also corresponds to component *Input/Output Interface* in the proposed framework.

7.2.1 Friendly Elective Surgical Planning

As mentioned before, the DSSTS allows for setting groups of surgeons and defining surgeons' availabilities inside each group. Similarly, ORs sharing certain properties –e.g. equipped for certain specific procedures- can be also grouped. In Figure 7.2 we show a screen of the DSSTS to configure such groups. Starting from this initial assignment, there is an easy procedure for refining availabilities within the planning horizon, to obtain the so-called *refined availability*. The DSSTS guides the SU director through a road map to specify the day-to-day availability of staffed ORs, which comprises both facilities' and surgeons' availabilities (see the sequence in the upper part of Figure 7.3).

As mentioned in the requirements, data from patients and their interventions are imported from the HIS. The last step in the sequence shown in Figure 7.3 allows specifying patients' unavailability in a very

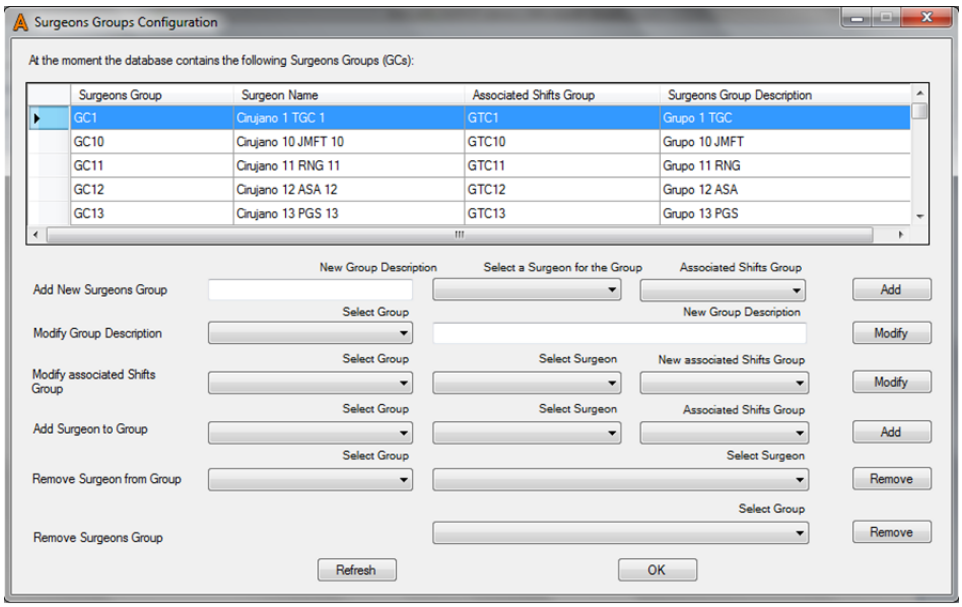

Figure 7.2: Generation of surgeons' groups and availabilities assignment

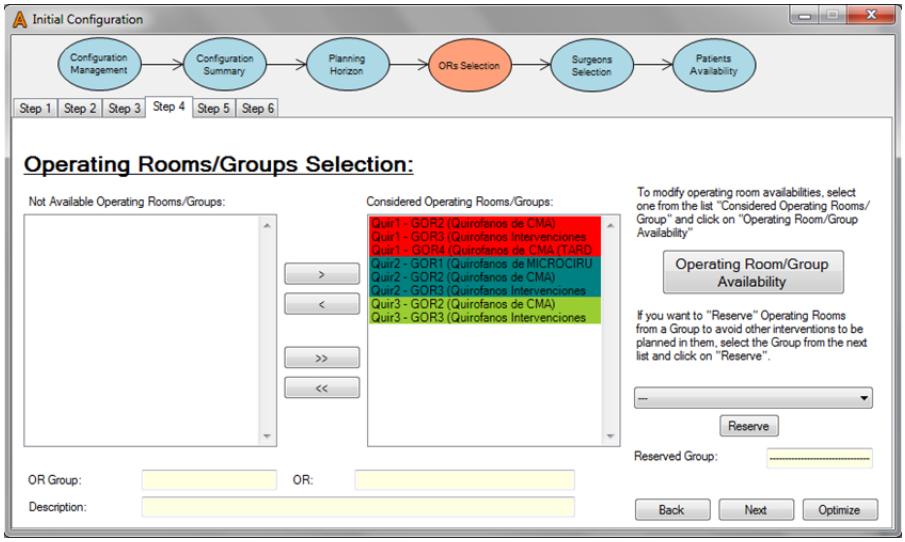

Figure 7.3: Availabilities refinement within the planning horizon: the ORs example

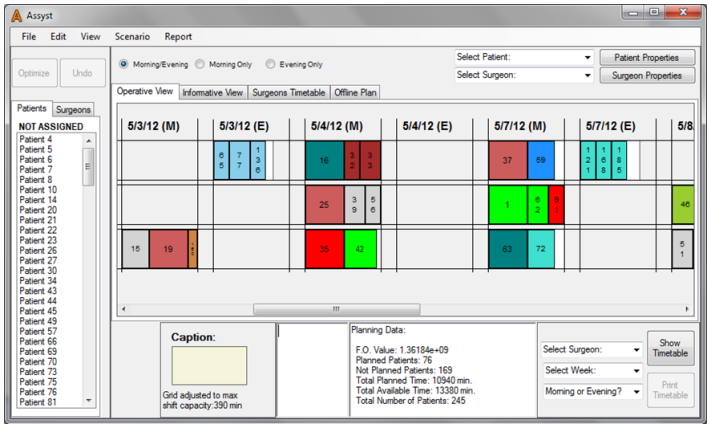

Figure 7.4: The user-friendly graphical interface: Example of short term planning within a three ORs surgical suite

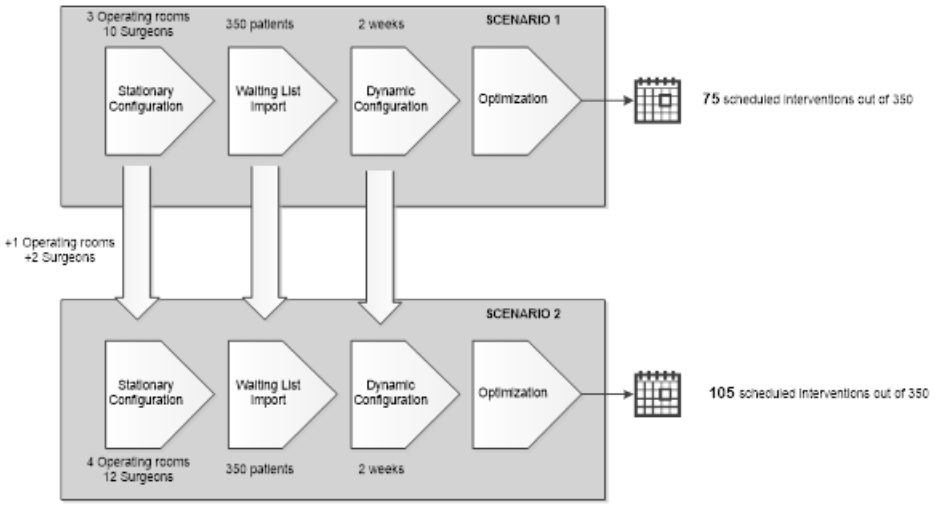

Figure 7.5: *What-if analysis*

intuitive manner (see parameter δ_{prt} in the MILP model in Section 7.3.1).

Detailed tools for analysis and drill-down capabilities have been also built in the DSSTS so SU directors can study their scenarios in greater detail. All use cases invoke the MILP model/heuristics for either scheduling or rescheduling. For manual fine-tuning, the SU director can *freeze* a number of formerly staffed and scheduled ORs working shifts so the interventions of patients who have already been notified remain unmodified. Once the optimization engine produces a solution (either exact or approximate), the resulting OR schedule is displayed in a user-friendly graphical interface so the SU director can visualize the available information of every intervention, the surgical timetable for every surgeon, and the graphic representation of the surgical schedules (sketched as a time-space matrix drawing, see Figure 7.4).

The above mentioned functionalities help the SU director to conduct *what-if analyses*. Figure 7.5 shows an example in which the SU director may use the DSSTS to assess the impact of using additional

ORs and surgeons in order to discuss with the Hospital Managers future budget/OR-time allocation for his/her SU.

7.3 Solution Procedures for the Case Study

In this section two solution approaches are presented for the operating room scheduling problem. First, a MILP model to optimally solve the problem is presented. And second, a set of heuristics to speed up the computational time required to solve the problem is presented.

7.3.1 MILP Model for the Operating Room Scheduling Problem

Taking into account the different requirements and constraints of the decision problem, we develop a mixed integer linear programming model which is presented next.

Mathematical Formulation

- *Sets of Indices*

  P Set of patienrs (interventions) in the waiting list, with elements $p \in P$ and cardinality $|P|$

  T Set of working shifts within the planning horizon, with elements $t \in T$ and cardinality $|T|$

  R Set of ORs, with elements $r \in R$ and cardinality $|R|$

  S Set of ORs, with elements $s \in S$ and cardinality $|S|$

- *Parameters*

  l_{rt} Regular capacity or OR r in working shift t

  c_{st} Time surgeon s is available to carry out interventions in working shift t

  u Maximum number of ORs where surgeons can perform surgeries in the same working shift

  m Maximum number of surgeons performing surgeries in the same OR and working shift

  n Maximum number of patients allocated to the same OR in a working shift

  d_p Expected duration for the surgery of patient p

  δ_{prt} Binary parameter yielding 1 if surgery of patient p can be performed in OR r in day t; 0 otherwise

  τ_{prt} Binary parameter yielding 1 if surgery of patient p is assigned to surgeon s; 0 otherwise

- *Variables*

  Z_{prt} 1 if patient p is to be intervened in OR r in working shift t; 0 otherwise

  X_{srt} 1 if surgeon s is allocated to OR r in working shift t; 0 otherwise

  L_{srt} OR time allocated to surgeon s in OR r in working shift t

The model is then:

$$\text{Max } (\sum_{p\in P} w_p \cdot \sum_{t\in T}\sum_{r\in R} \frac{Z_{prt}}{t}) \tag{7.1}$$

Subject to

$$L_{srt} = \sum_{p\in P} Z_{prt} \cdot d_p \quad , (\forall s \in S, \forall r \in R, \forall t \in T) \tag{7.2}$$

$$L_{srt} \leq l_{rt} \cdot X_{srt} \quad , (\forall s \in S, \forall r \in R, \forall t \in T) \tag{7.3}$$

$$L_{srt} \geq X_{srt} \quad , (\forall s \in S, \forall r \in R, \forall t \in T) \tag{7.4}$$

$$\sum_{r\in R} X_{srt} \leq u \quad , (\forall s \in S, \forall t \in T) \tag{7.5}$$

$$\sum_{s\in S} X_{srt} \leq m \quad , (\forall t \in T, \forall r \in R) \tag{7.6}$$

$$\sum_{p\in P} Z_{prt} \leq n \quad , (\forall t \in T, \forall r \in R) \tag{7.7}$$

$$\sum_{s\in S} L_{srt} \leq l_{rt} \quad , (\forall t \in T, \forall r \in R) \tag{7.8}$$

$$\sum_{r\in R} L_{srt} \leq c_{st} \quad , (\forall s \in S, \forall t \in T) \tag{7.9}$$

$$\sum_{t\in T}\sum_{r\in R} Z_{prt} = 1 \quad , (\forall p \in P | dl_p \leq |T|) \tag{7.10}$$

$$\sum_{t\in T}\sum_{r\in R} Z_{prt} \leq 1 \quad , (\forall p \in P | dl_p > |T|) \tag{7.11}$$

$$\sum_{t\in T}\sum_{r\in R} Z_{prt} \cdot t \leq dl_p \quad , (\forall p \in P | dl_p \leq |T|) \tag{7.12}$$

$$Z_{prt} = 0 \quad , (\forall p \in P, \forall r \in R, \forall t \in T | \delta_{prt} = 0) \tag{7.13}$$

$$Z_{prt} = 0 \quad , (\forall p \in P, \forall r \in R, \forall t \in T | \tau_{ps} = 0, s \in S) \tag{7.14}$$

$$Z_{prt}, X_{srt} \in 0, 1, L_{srt} \geq 0, (\forall p \in P, \forall r \in R, \forall t \in T) \tag{7.15}$$

Constraints (7.2) calculate the amount of OR time allocated to a surgeon to perform interventions within a working shift. Constraints (7.3) and (7.4) determine if surgeons are allocated to ORs and working shifts according to total time allocated to those surgeons in those ORs and those working shifts. Constraints (7.5) limit the number of different ORs allocated to a surgeon within the same working shift, whereas constraints (7.6) limit the number of different surgeons allocated to an OR time. Constraints (7.7) impose a bound for the number of patients scheduled in an OR time. Constraints (7.8) prohibit that

|P| = 3; |T| = 2; |R| = 1; |S| = 1				
Patient (p)	w_p	Solution		Objective Function
		Z_{p11}	Z_{p12}	
p_1	360	0	1	$360 \cdot (\frac{0}{1} + \frac{1}{2}) + 180 \cdot (\frac{0}{1} + \frac{0}{2}) + 480 \cdot (\frac{1}{1} + \frac{0}{2}) = 660$
p_2	180	0	0	
p_3	480	1	0	

Table 7.2: Objective function example

the total amount of time assigned to all surgeons in a time block is higher than the capacity of the time block. Analogously, constraints (7.9) prohibit that the total amount of time allocated to a surgeon in a working shift is higher than his/her availability for that working shift. The set of constraints (7.10) and (7.11) enforce that each intervention is performed at most once. If the due date of the patient is within the planning horizon, his/her intervention must be planned (7.10). However, if it is not, his/her intervention may or may not be planned (7.11). Constraints (7.12) ensure that the intervention of a patient with a due date within the planning horizon must be executed before his/her due date. In constraints (7.13) and (7.14) the allocation of a patient to an OR and a working shift is limited by the possibility of performing his/her intervention in that OR and working shift and the possibility of being intervened by his/her associated surgeon. Finally, equations (7.15) contain the variables definition.

Note that the objective function includes not only the above mentioned clinical weight but the specific date of interventions. Hence, equation (7.1) shows the objective in terms of a sum over planned patients of the product of their clinical weight parameter w_p and the inverse of their planned surgery date $\sum_{t \in T} \sum_{r \in R} \frac{Z_{prt}}{t}$ (see Table 7.2 for an explanatory example). The objective of our Mixed Integer Linear Programming (MILP) model can be seen as the maximization of the performed service level while attempting to bring higher w_p to sooner working shifts, thereby attempting both patient's satisfaction and quality of service.

7.3.2 Heuristics for the Operating Room Scheduling Problem

As it can be seen from Table 7.1, most authors propose approximate approaches as their priority is achieving good schedules in short time intervals rather than pursuing optimality. In our case, the use of heuristics is motivated by two issues:

- The complexity of the decision problem is such that, for most real-life cases with a relatively high number patients, ORs, and surgeons, exact approaches are able to find optimal solutions only after hours of computation time. These computational requirements may be acceptable if the decision problem is to be solved just once before being implemented (i.e. a two-weeks schedule is obtained from the MILP model and applied in an straightforward manner). However, in our case, the decision problem will be the output of an iterative procedure in which the SU director tries some *scenario*

consisting of a given resource availability, medical priorities, etc., and then the DSSTS provides a solution for this scenario, which is used by the SU director to build a new scenario (e.g. maybe the availability of some OR may be increased, or the number of interventions to be scheduled may increase in view of the spare capacity, etc.). A solution for this new scenario is then obtained by the DSSTS, and so forth until the SU director is satisfied with one of scenarios considered. Obviously, the SU director cannot wait for hours for one scenario to be solved by a MILP model. Therefore, we need to develop some heuristic able to produce fast and good solutions to the problem so they can be used during the *exploration* of scenarios, although the MILP model may be used to provide a the final schedule. The concept of scenario will be discussed with greater detail in the next section.

- When considering a medium-long term horizon, the high number of unforeseen events (such as emergencies Roland et al., 2010 or last minute cancellations Weinbroum et al., 2003) would possibly lead to the reschedule of planned interventions, so the advantages of using an exact approach vanish. This is another reason to use approximate approaches, particularly when the planning horizon is long enough so unforeseen events are more likely.

With this considerations in mind, we design three types of heuristics using a novel definition of the neighborhood structure. In these heuristics, a solution is represented by a specific arrangement of the waiting list. The proposed heuristics are the following:

1. Two-stage Sorting Bin-Packing (TSBP) heuristics. These procedures take into account both the fulfillment of time windows constraints and the objective function by prioritizing those patients whose latest surgery date (feasible with clinical guarantee) falls within the planning horizon. These heuristics consist of two stages:

   - Stage I. Patients in the waiting list are divided into two groups, those whose latest surgery date is within the planning horizon and the rest of patients. An initial waiting list is obtained by merging the first subset sorted in ascending order of the patients' latest surgery date and the second subset sorted according to a certain tuple $(Indicator(I), Criterion(C))$. In Figure 2 we present an example where the indicator I is the surgery duration (represented by a rectangle width) and the criterion C is the descending order.
   - Stage II. A surgical plan is constructed using one of the following variants of a Bin Packing (BP) algorithm:
     - *Next Fit (NF)*: intervention is planned in the last time block occupied, if possible; otherwise within the next available time block.

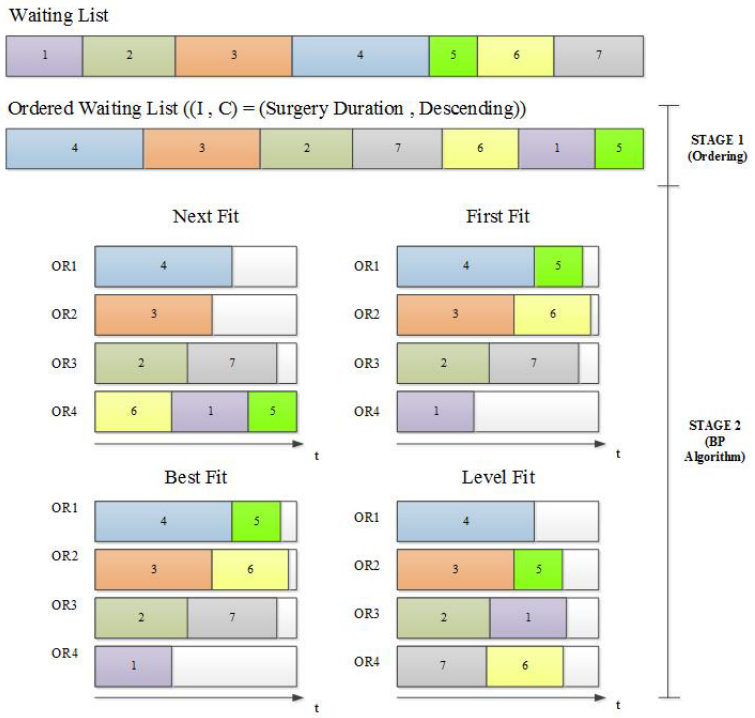

Figure 7.6: TSBP example

- *First Fit (FF)*: intervention is planned in the first time block it fits.
- *Best Fit (BF)*: intervention is planned in the time block that has the least amount of available time and it fits.
- *Level Fit (LF)*: intervention is planned in the time block that has the most amount of available time and it fits.

An example of this type of heuristics is shown in Figure 7.6.

2. Mixed Two-stage Sorting Bin-Packing (MIX-TSBP). In this type of heuristics the problem instance is solved according to several sorting tuples (i.e. different indicators and criteria) and a certain BP algorithm. This type of heuristics requires the choice of the aforementioned BP algorithm. In addition, we also consider a so-called MIX-TSBP heuristic, in which all algorithms and all sorting tuples are applied to the instance, selecting the combination of BP algorithm and (I, C) that yields the best results for the instance.

3. The Random Extraction-Insertion algorithm (REI) is an iterated greedy local search specifically designed for the problem, and based on the algorithm proposed by Ruiz and Stützle (2007) for the permutation flowshop scheduling problem. The initial waiting list is constructed from the best surgical schedule obtained by a constructive heuristic. The sorting procedure for obtaining a new

waiting list from an incumbent waiting list is composed by the following two stages:

- *Destruction stage.* It consists on randomly removing n patients from a given waiting list, obtaining a waiting list composed by $|P| - n$ patients.
- *Construction stage.* It consist on re-inserting the n patients (one by one) in the best position of the waiting list constructed in previous stage. Each insertion is evaluated using a bin-packing algorithm.

The resulting waiting list is considered as the incumbent waiting list if the objective function value improves the best value obtained so far. A simulated annealing-like acceptance criterion with a constant temperature is implemented to avoid the stagnation in the search procedure. The constant temperature is set such that moves which deteriorate the solution more than a percentage θ of the maximal deterioration with a probability smaller than φ (Lamiri et al., 2009). The termination criterion of REI is set based on the size of the problem (i.e. the length of the planning horizon, the number of ORs, and the number of surgeries on the waiting list).

7.4 Implementation results

The DSSTS was being deployed in several SUs in the hospital, although the most exhaustive results of their implementation were gathered in the Plastic Surgery Unit, where the pilot project took place. This Unit is composed by 16 surgeons who can perform any kind of surgery type within the specialty. Their OR resources consists of 4 multifunctional ORs for elective patients (three in the morning shift and one in the afternoon shift). The director of the Plastic Surgery Unit has been intensively using the DSSTS for more than one year with satisfactory results, and reported a number of direct benefits:

1. The usage of ORs per week has increased in the range of 10-20%, not only in the number of effective OR hours used but also in the number of planned patients and in the service level. This speaks for the quality of the schedules produced by the DSSTS, a fact that SU directors quoted as remarkable given the fact that they are well-seasoned professionals with years of experience in planning their Units.

2. In general, the plans produced by the DSSTS do not need to be manually modified by the SU director unless unforeseen events occur. According to the SU director, around 98% of the obtained two-week schedules were directly applicable without any modification.

3. The time required by the SU director to determine a surgical plan has been greatly reduced by approximately 30 hours per month, so in fact he has saved about 1 day per week that was previously devoted to establish next week's schedule.

4. SU director feels that the plans produced by the system are *fairer* that those produced manually, since the decision procedure was formally (mathematically) established and lacked subjective factors, and perceived this as an important advantage in order to justify their decisions in front of their staff and of the patients.

5. Long-term planning capabilities of the DSSTS have been extensively used to improve communication issues with patients, as the week or fortnight where each patient would be intervened can be easily estimated. This has resulted in a lower number of cancellations, due to the better adjustment between planning dates and real execution dates of interventions, and to the capability of re-scheduling interventions. Similarly, the tentative long-term plan has been used to obtain a long-term, reliable, timetable for ordering required medical tests (such as anesthesia tests) and a tentative agenda for their surgeons.

7.5 Conclusions

In order to apply the framework proposed in Chapter 3 to a real environment, an operating room DSSTS which is currently in use in one of the largest hospitals in Spain – the University Hospital "Virgen del Rocio" – is presented. A detailed analysis of the underlying decision problem, i.e. the operating room scheduling problem, is carried out. Next, the main use cases of the DSSTS are described. The DSSTS helps the responsible of each Surgical Unit in several related decisions: First, in a medium-term horizon to obtain tentative plans to determine a subset of patients to be put in standby, and to organize the material and human resources needed for their intervention. Secondly, a short-term horizon is used for detailed schedules, so optimal or quasi-optimal surgery dates are obtained in accordance with the constraints imposed by the surgical resources (ORs, surgeons), and patients' availability. In addition, the DSSTS allows users to fine-tune the schedule (e.g. to adapt to last-minute changes) by including a graphically-interactive user interface. In this chapter we also present a set of solution procedures, namely a MILP model with the objective of maximising the quality of service and a set of heuristics in order to reduce the computational time of the MILP model. These heuristics are embedded into the DSSTS. Finally, the main results of the implementation of the DSSTS in the target surgical units are discussed.

Part V

CONCLUSIONS

Chapter 8

Conclusions, results and future research lines

8.1 Conclusions

This Thesis focuses on Decision Support Systems for Task Scheduling (DSSTS). These systems are quite widespread both, in nowadays organizations and in literature contributions. The objective of this Thesis is to propose a common framework for the development of these DSSTS. In order to ensure the validity and range of application of this framework, its feasibility is analysed within two specific fields of applications, and two implementation case studies are conducted within these fields. In order to fulfill the general goals of the Thesis, a number of research objectives were established in Chapter 1. Next we present a review of these objectives and how they have been addressed in this document:

- ***O1***. *To propose a framework for the design and development of DSSTS.*

  A first step towards this objective is addressed in Chapter 2, where an analysis of the literature is carried out in order to classify the main issues regarding the implementations of DSSTS in practice. From this analysis, a number of guidelines for the development of a framework were obtained. Following these guidelines, a framework for the design and implementation of DSSTS was proposed in Chapter 3, using different perspectives, in order to cover all aspects required in the description of frameworks for information systems. A number of conclusions can be drawn from this framework:

  – The proposed framework fulfill all the guidelines obtained from the review of failures in the implementation.

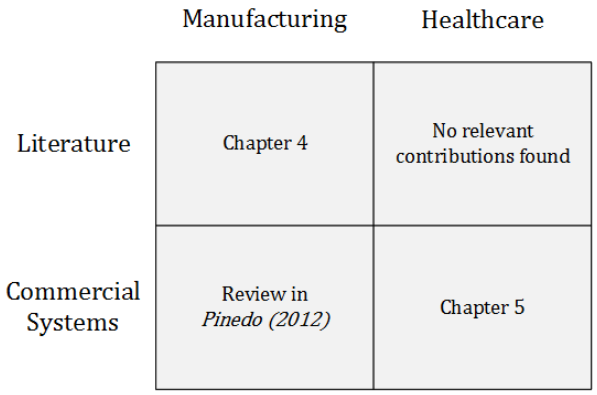

Figure 8.1: Review on existing DSSTS.

- The use of the framework proposed can ease and expedite the implementation of DSSTS in different contexts as developers does not need to design the system from scratch. Moreover, their efforts can be properly channeled as the most sensitive issues are known in advance.
- In these cases where interoperability among DSSTS is required, the use of the framework ease the communicaton between systems, i.e. it could serve as the basis of a standard for communication.

- ***O2***. *To analyse existing implementations of DSSTS in order to check the alignment of the framework proposed with the task scheduling systems implemented in the two sectors chosen for the evaluation of the framework.*

  This objective is addressed by carrying out two reviews of existing DSSTS in manufacturing and healthcare as shown in Figure 8.1. For the manufacturing scheduling case, in Chapter 4 a review of the literature is performed as no other review on this topic can be found. Regarding commercial tools, we refer to the review by Pinedo (2012). For the healthcare case, no relevant contributions are found when we try to carry out a literature review similar to that in the manufacturing case. Nevertheless, as no review on commercial DSSTS is found, in Chapter 5 we analyse and classify the most interesting operating room DSSTS found. A number of conclusions can be drawn:

  - Much more interest has been found in manufacturing DSSTS than in operating room DSSTS, in view of the number of contributions that can be found for each case.
  - DSSTS from both sectors fail in considering the most common issues described in Chapter 2 when implementing this type of systems.
  - The framework proposed is perfectly aligned with them, considering all the capabilities that were found in the reviews and including some additional features that can help in avoiding

implementation problems already discussed.

- ***O3***. *To conduct the design and implementation of two DSSTS according to the proposed framework in order to demonstrate its applicability.*

  This objective was addressed by developing a DSSTS for a real environment in each case. In Chapter 6 a DSSTS for the case of a plastic manufacturer is detailed. To do so, a detailed analysis of the decision problem to be tackled, i.e. the hybrid flowshop scheduling problem with missing operations, is performed, and a number of state-of-the-art heuristics are developed. In Chapter 7, a DSSTS for scheduling the intervenetions in three different surgical units in a hospital is presented. The decision problem is also discussed and a number of solution approaches to improve the efficiency of the surgical units are proposed.

  The conclusions of this objective can be divided for the two cases:

  - The Manufacturing Case.
    - The hardness of the hybrid flowshop scheduling problem with missing operations differs from the hardness of the traditional hybrid flowshop scheduling problem according to the studied parameters, namely number of stages, number of machines per stage, number of jobs and percentage of missing operations, and offers the possibility to achieve improvements by focusing on the specific characteristics of missing operations.
    - The heuristics developed for this problem outperforms the already existing algorithms, both in computational time and in quality of the solutions.
    - The DSSTS where these heuristics were embedded was successfully implemented in the target company (a plastic manufacturer).
    - The DSSTS can be assumed as validated according to the feedback received from the target company.
  - The Healthcare Case.
    - The heuristics developed for the target surgical units were validated by the surgical directors and their use improved the efficiency of the operating rooms.
    - The DSSTS was successfully implemented in the target surgical units and their use was properly accepted by the schedulers of the operating rooms.
    - The DSSTS can be assumed to be validated according to the feedback received from the target surgical units.

As a result of these implementations, the proposed framework can be considered to be validated, thus proving its applicability in different practical settings.

8.2 Contributions

This section summarizes the research contributions that have been generated during the development of this Thesis. In Section 8.2.1 we detail the contributions whose results are included in this Thesis. First the contributions that have been published (or that are currently in process) on international journals are shown. Next, the different conference proceedings that have been presented both in national and international conferences are shown. And finally, the research projects under which this Thesis has been produced are enumerated. In addition there are several works that, although are not considered within this Thesis, were published during its development. These contributions are listed in Section 8.2.2

8.2.1 Contributions from the Thesis

SCI indexed journals

- Dios, M., Framinan, J.M., "A review and classification of computer-based manufacturing scheduling", Computers and Industrial Engineering, 99, 229-249, 2016 (Impact Factor (2015): 2.086)
- Dios, M., Molina-Pariente, J.M., Fernandez-Viagas, V., Andrade-Pineda, J.L., Framinan, J.M., "A decision support system for operating room scheduling", Computers and Industrial Engineering,88,430-443,2015 (Impact Factor (2015): 2.086)
- Dios, M., Fernandez-Viagas, V., Framinan, J.M, "Efficient Heuristics for the Hybrid Flow Shop Scheduling Problem with Missing Operations" (Under review in *Computers and Industrial Engineering*)

Papers in conference proceedings

- Dios, M.; Framinan, J.M., "Constructive Heuristics Comparison in Hybrid Flow Shop Scheduling Environments with Missing Operations", International Conference on Industrial Engineering and Systems Management (IESM 2015)
- Dios, M., Fernández-Viagas, V., Perez-Gonzalez, Paz, Framinan, J.M., "Manufacturing Scheduling Systems: What are they made of?", Multidisciplinary International Scheduling Conference (MISTA 2015)

- Dios, M., Framinan, J.M., "A Review on Decision Support Systems for Manufacturing Scheduling", 14th International Conference on Project Management and Scheduling (PMS 2014)
- Molina, J.M., Dios, M., Andrade, J.L., Fernández, V., Framinan, J.M., Gómez-Cía, T., "Métodos avanzados de resolución para la planificación y programación de quirófanos", XV Congreso Nacional de Informática de la Salud (INFORSALUD 2012)
- Dios, M., Fernández, V., Molina, J.M., Andrade, J.L., Framinan, J.M., Gómez-Cía, T., "Assyst: Herramienta para el soporte a la toma de decisiones en planificación quirúrgica",XV Congreso Nacional de Informática de la Salud (INFORSALUD 2012)
- Dios, M., Fernandez, V., Molina, J.M., Andrade, J.L., Framinan, J.M., "Arquitectura de un sistema de soporte a la toma de decisiones para planificación de quirófanos" (poster), XV Congreso Nacional de Informática de la Salud (INFORSALUD 2012)

Research projects

- Scope – Sistemas Coordinados de Planificación y Ejecución de Pedidos. Proyecto de Excelencia de la Junta de Andalucía. Ref: P08-TEP-03630
- Scheduling & Control for Customer-Responsive Production. Plan Nacional 2010. Ref: DPI2010-15573
- Diseño Avanzado de Sistemas de Programación de la Producción Dinámicos, Robustos y Extendidos. Plan Estatal 2013-2016 Excelencia. Ref: DPI2013-44461-P
- Support: Surgical Processes – Planning, Optimization, Redesign and Testing. Proyecto de Excelencia de la Junta de Andalucía. Ref: P10-TEP-06067

RD Contracts

- Diseño y desarrollo de una herramienta de planificación y seguimiento de las operaciones en PUVENSA. Ref: ES-1192/2013
- e-Fábrica: Desarrollo de una metodología de gestión empresarial por procesos y gestión documental en una empresa manufacturera contrapedido con alto nivel de adaptación de diseño. Ref: PI-1366/2014
- Implantación de ASSYST en los Hospitales Universitarios Virgen del Rocío. Ref: PI-0661/2010
- Desarrollo de una Metodología Avanzada de Fabricación de Estructuras en Serie. Ref: PI-1044/2012

8.2.2 Contributions outside the Thesis

SCI indexed journals

- Dios, M., Gonzalez-R, P.L., Dios, D., Maffezzoli, A., "A mathematical modeling approach to optimize composite parts placement in autoclave", International Transactions in Operational Research, 24, 1-2, 115-141, 2017 (Impact Factor (2015): 1.255)

- Fernandez-Viagas, V., Dios, M., Framinan, J.M., "Efficient constructive and composite heuristics for the Permutation Flowshop to minimise total earliness and tardiness", Computers and Operations Research, 75, 38-48, 2016 (Impact Factor (2015): 1.988)

Papers in conference proceedings

- Dios, M., Framinan, J.M., "A MILP model for operating room scheduling considering PACU beds: A Decision Support Tool Prototype", 42th Annual Meeting EURO Working Group on Operational Research Applied to Health Services (ORAHS 2016)

- Fernández-Viagas, V., Dios, M., Framinan, J.M., "A constructive heuristic for the permutation flowshop to minimise total earliness and tardiness", 15th International Conference on Project Management and Scheduling (PMS 2016)

- Fernández-Viagas, V., Dios, M., Perez-Gonzalez, Paz, Framinan, J.M., "A framework of constructive heuristics for permutation-type scheduling problems", Multidisciplinary International Scheduling Conference (MISTA 2015)

- Perez-Gonzalez, Paz, Dios, M., Fernández-Viagas, V., Framinan, J.M., "Heuristic Methods for Single Machine Scheduling with Periodic Maintenance", Multidisciplinary International Scheduling Conference (MISTA 2015)

- Dios, M., Molina, J.M., Framinan, J.M., Hans, E., "A Decision Support System for Solving the Stochastic Operating Theater Tactical Problem", 40th Annual Meeting EURO Working Group on Operational Research Applied to Health Services (ORAHS 2014)

- Perez, P., Framinan, J.M., Dios, M., "Two-agent scheduling problema with flowtime objective: Analysis of the problema and exact method", 14th International Conference on Project Management and Scheduling (PMS 2014)

- Andrade, J.L., Molina, J.M., Dios, M., Fernández, V., Framinán, J.M., "GOL: Herramienta para el soporte a la toma de decisiones logísticas en una red de laboratorios clínicos", XV Congreso Nacional de Informática de la Salud (INFORSALUD 2012)

8.3 Future research lines

In this section we present some research issues that could be further addressed to continue the research line inititated with this Thesis.

1. Although the proposed framework has been properly validated, a normalized review of existing DSSTS in both fields would help in order to have a more accurate view of the existing DSSTS, which would allow to study the feasibility of a common design for DSSTS along the lines in the framework proposed.

   (a) Another interesting issue regarding the review of systems would be its extension to the most common solution approaches used to address scheduling. This would give us insights on how these solution approaches are used and would allow us to go into more details when describing the *Model Management Module* of the framework.

   (b) An interesting future research line would be the development of a standard for the data model of the DSSTS, that offer the possibility of going more into detail of the *Database Management Module*. This standard would open the opportunity of detailing a common communication protocol for these systems, what would help Information System developers to provide well structured interfaces to DSSTS.

   (c) In order to have the possibility of further detail the *User Dialogue Management Module*, it would be desirable to work in the development of standards for the representation of data and solutions in scheduling. Although, Gantt Charts are assumed as a good representation of solutions in scheduling, we already saw that for some cases this representation is not precise enough, so more research on this topic needs to be done.

2. In view of the data-intensive nature of the DSSTS, and given the fact that there are some standard data models (such as e.g. ISA/95) which, in principle, may support the data required for a DSSTS, an interesting research line would arise in order to verify whether these standards could support the framework described in the Thesis, and if so, to develop a roadmap for the implementation of the framework in accordance with these data models.

3. Finally, it would be also interesting to validate the DSSTS implemented in this Thesis in other organizations, i.e. in other manufacturing companies and healthcare institutions, in order to test the framework in other contexts. This would also allow to develop some of the functionalities described in the framework which have not been implemented in the two case studies in this Thesis.

Bibliography

Claudio F. Abreu, Jerrold H. May, William E. Spangler, and Luis G. Vargas. Conflict identification and reconciliation in a collaborative manufacturing scheduling task. *International Journal of Information Technology & Decision Making*, 07(01):147–174, 2008.

Heimo H. Adelsberger and John J. Kanet. The leitstand - a new tool for computer-integrated manufacturing. *Production and Inventory Management Journal*, 32(1):43, 1991.

A.T.M. Aerts, A. Jansen, L. Klieb, C. Noorlander, and G. Wolf. 'PLATE': A decision support system for resource constrained scheduling problems. *European Journal of Operational Research*, 79(2):158–166, 1994.

A. Almeida and G. Marreiros. A collaborative framework to support scheduling decisions. In *Industrial Informatics, 2006 IEEE International Conference on*, pages 979–984, 2006.

E. Angelidis, F.S. Pappert, and O. Rose. A prototype simulation tool for a framework for simulation-based optimization of assembly lines. In *Proceedings - Winter Simulation Conference*, pages 2378–2389, 2011.

P. Appelqvist and J.-M. Lehtonen. Combining optimisation and simulation for steel production scheduling. *Journal of Manufacturing Technology Management*, 16(2):197–210, 2005.

Roberto Aringhieri, Paolo Landa, Patrick Soriano, Elena Tanfani, and Angela Testi. A two level metaheuristic for the operating room scheduling and assignment problem. *Computers & Operations Research*, 54(0):21 – 34, 2015. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2014.08.014. URL `http://www.sciencedirect.com/science/article/pii/S030505481400224X`.

V. A. Armentano and D. P. Ronconi. Tabu search for total tardiness minimization in flowshop scheduling problems. *Computers and Operations Research*, 26(3):219–235, 1999.

A. Artiba and E.H. Aghezzaf. An architecture of a multi-model system for planning and scheduling. *International Journal of Computer Integrated Manufacturing*, 10(5):380–393, 1997.

V. Augusto, X. Xie, and V. Perdomo. Operating theatre scheduling with patient recovery in both operating rooms and recovery beds. *Computers & Industrial Engineering*, 58(2):231–238, 2010.

P.a Avgeriou and U.b Zdun. Architectural patterns revisited : A pattern language. In *Proceedings of the 10th European Conference on Pattern Languages of Programs*, 2005.

D. T. Bahlman and Johnson F. C. Using technology to improve and support communication and workflow processes. *Association of Operating Room Nurses Journal*, 82:65–73, 2005.

A.Y.a Barlatt, A.b Cohn, O.c Gusikhin, Y.c Fradkin, R.d Davidson, and J.e Batey. Ford motor company implements integrated planning and scheduling in a complex automotive manufacturing environment. *Interfaces*, 42(5):478–491, 2012.

S. Batun, B. T. Denton, T. R. Huschka, and A. J. Schaefer. Operating room pooling and parallel surgery processing under uncertainty. *INFORMS Journal on Computing*, 23(2):220–237, 2011.

J. Behnamian and S.M.T. Fatemi Ghomi. Hybrid flowshop scheduling with machine and resource-dependent processing times. *Applied Mathematical Modelling*, 35(3):1107–1123, 2011.

J. Belien, E. Demeulemeester, and B. Cardoen. A decision support system for cyclic master surgery scheduling with multiple objectives. *Journal of Scheduling*, 12:147–161, 2009.

M. Berglund and J. Karltun. Human, technological and organizational aspects influencing the production scheduling process. *International Journal of Production Economics*, 110:160 – 174, 2007.

Jacek Blazewicz, KlausH. Ecker, Erwin Pesch, Gunter Schmidt, and Jan Weglarz. Computer integrated production scheduling. In *Scheduling Computer and Manufacturing Processes*, pages 421–468. Springer Berlin Heidelberg, 2001.

Jacek Blazewicz, KlausH. Ecker, Erwin Pesch, Gunter Schmidt, and Jan Weglarz. Computer integrated production scheduling. In *Handbook on Scheduling*, International Handbook on Information Systems, pages 583–630. Springer Berlin Heidelberg, 2007.

A. Boccalatte, R. Minciardi, M. Paolucci, and R. Pesenti. Sked: An integrated decision support system for scheduling problems. In *Computer Integrated Manufacturing, 1992., Proceedings of the Third International Conference on*, pages 128–135, 1992.

A. Boccalatte, M. Paolucci, R. Pesenti, A. Piombo, and R. Prefumo. Hybrid system for short-term scheduling in manufacturing: a case study. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1189–1193, 1994.

Basilio Bona, Paolo Brandimarte, Cosimo Greco, and Giuseppe Menga. Hybrid hierarchical scheduling and control systems in manufacturing. *IEEE Transactions on Robotics and Automation*, 6(6):673–686, 1990.

A. Bonfill, A. Espuna, and L. Puigjaner. Decision support framework for coordinated production and transport scheduling in scm. *Computers & Chemical Engineering*, 32(6):1206 – 1224, 2008.

S.A.a Brah and L.L.b Loo. Heuristics for scheduling in a flow shop with multiple processors. *European Journal of Operational Research*, 113(1):113–122, 1999.

David Bredstrom, Jan T. Lundgren, Mikael Rannqvist, Dick Carlsson, and Andrew Mason. Supply chain optimization in the pulp mill industry - ip models, column generation and novel constraint branches. *European Journal of Operational Research*, 156(1):2 – 22, 2004.

J. O. Brunner, J. F. Bard, and R. Kolisch. Flexible shift scheduling of physicians. *Health Care Management Science*, 12(3):285–305, 2009.

P. Burke and P. Prosser. A distributed asynchronous system for predictive and reactive scheduling. *Artificial Intelligence in Engineering*, 6(3):106–124, 1991.

Jacques Carlier and Emmanuel Neron. An exact method for solving the multi-processor flow-shop. *RAIRO - Operations Research*, 34:1–25, 1 2000.

S. Chaabane, N. Meskens, A. Guinet, and M. Laurent. Comparison of two methods of operating theatre planning: Application in belgian hospital. *Journal of Systems Science and Systems Engineering*, 17(2): 171–186, 2008.

Felix T.S. Chan, K.C. Au, and P.L.Y. Chan. A decision support system for production scheduling in an ion plating cell. *Expert Systems with Applications*, 30(4):727 – 738, 2006.

W.-T. Chan and Z. Zeng. Coordinated production scheduling of prefabricated building components. In *Proceedings of the Congress*, pages 909–916, 2003.

M. J. Cheeseman, P. Swann, G. B. Hesketh, and S. Barnes. Adaptive manufacturing scheduling: a flexible and configurable agent-based prototype. *Production Planning & Control*, 16(5):479–487, 2005.

A. Collinot, C. Le Pape, and G. Pinoteau. Sonia: A knowledge-based scheduling system. *Artificial Intelligence in Engineering*, 3(2):86–94, 1988.

K.H. Concannon, K.I. Hunter, and J.M. Tremble. Simul8-planner simulation-based planning and scheduling. In *Winter Simulation Conference Proceedings*, volume 2, pages 1488–1493, 2003.

A.K.A. de Castro, P.R. Pinheiro, and G.G.C. de Souza. A scheduling process applied to newspaper production. In *Service Systems and Service Management, 2006 International Conference on*, volume 2, pages 1245–1250, 2006.

Benoît Saenz de Ugarte, Adnène Hajji, Robert Pellerin, and Abdelhakim Artiba. Development and integration of a reactive real-time decision support system in the aluminum industry. *Engineering Applications of Artificial Intelligence*, 22(6):897 – 905, 2009.

F. Dexter, R. D. Traub, and A. Macario. How to release allocated operating room time to increase efficiency: Predicting which surgical service will have the most underutilized operating room time. *Anesthesia and analgesia*, 96:507–512, 2003.

P. Esquirol, P. Lopez, L. Haudot, and M. Sicard. Constraint-oriented cooperative scheduling for aircraft manufacturing. *IEEE Expert*, 12(1):32–39, Jan 1997.

J.E. Everett. Iron ore production scheduling to improve product quality. *European Journal of Operational Research*, 129(2):355 – 361, 2001.

M.P.a Fanti, G.a Rotunno, G.b Stecco, W.b Ukovich, and S.b Mininel. An integrated system for production scheduling in steelmaking and casting plants. *IEEE Transactions on Automation Science and Engineering*, 13(2):1112–1128, 2016.

Hugh E. Fargher, Michael A. Kilgore, Paul J. Kline, and Richard A. Smith. Planner and scheduler for semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 7(2):117–126, 1994.

H. Fei, N. Meskens, and C. Chu. An operating theatre planning and scheduling problem in the case of a "block scheduling" strategy. In *International Conference on Service Systems and Service Management*, volume 1, pages 422–428, 2006.

H. Fei, C. Chu, N. Meskens, and A. Artiba. Solving surgical cases assignment problem by a branch-and-price approach. *International Journal of Production Economics*, 112(1):96–108, 2008.

H. Fei, C. Chu, and N. Meskens. Solving a tactical operating room planning problem by a column-generation-based heuristic procedure with four criteria. *Annals of Operations Research*, 166(1):91–108, 2009.

H. Fei, N. Meskens, and C. Chu. A planning and scheduling problem for an operating theatre using an open scheduling strategy. *Computers & Industrial Engineering*, 58:221–230, 2010.

Shan Feng, Ling Li, Ling Cen, and Jingping Huang. Using mlp networks to design a production scheduling system. *Computers & Operations Research*, 30(6):821 – 832, 2003.

Victor Fernandez-Viagas and Jose M. Framinan. A new set of high-performing heuristics to minimise flowtime in permutation flowshops. *Computers & Operations Research*, 53(0):68 – 80, 2015. ISSN 0305-0548.

G. Figueira, P. Amorim, L. GuimarÃ£es, M. Amorim-Lopes, F. Neves-Moreira, and B. Almada-Lobo. A decision support system for the operational production planning and scheduling of an integrated pulp and paper mill. *Computers and Chemical Engineering*, 77:85–104, 2015.

Mark A. Flower and Michael C. Cheselka. Amp-akzo company's simulation-based finite capacity scheduling system. In *Winter Simulation Conference Proceedings*, pages 1013–1019, 1994.

Mark S Fox. Isis: a retrospective. *Intelligent scheduling*, 1:3–28, 1994.

J. M. Framinan, R. Ruiz Usano, and R. Leisten. Sequencing conwip flow-shops: analysis and heuristics. *International Journal of Production Research*, 39(12):2735–2749, 2001.

J.M. Framinan, R. Leisten, and R. Ruiz. *Manufacturing Scheduling Systems: An Integrated View of Models, Methods, and Tools*. Springer, 2014.

J.M.a Framinan and R.b Ruiz. Architecture of manufacturing scheduling systems: Literature review and an integrated proposal. *European Journal of Operational Research*, 205(2):237–246, 2010.

J. C. Fransoo, T. Wäfler, and Wilson. *Behavioral Operations in Planning and Scheduling*. Springer, 2010.

Cong Gao and Lixin Tang. A decision support system for color-coating line in steel industry. In *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, pages 1463–1468, 2008.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

A. X. Garg, N. K. J. Adhikari, H. McDonald, M. P. Rosas-Arellano, P. J. Devereaux, J. Beyene, J. Sam, and R. B. Haynes. Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: A systematic review. *Journal of the American Medical Association*, 293(10): 1223–1238, 2005.

P.a Gazmuri and S.b Maturana. Developing and implementing a production planning dss for cti using structured modeling. *Interfaces*, 31(4):22–36, 2001.

Jozsef. Geiger. Decision support for multi-objective flow shop scheduling by the pareto iterated local search methodology. *Computers and Industrial Engineering*, 61(3):805–812, 2011.

Martin Josef Geiger. MOOPPS: An optimization system for multi objective scheduling. In *Proceedings of the Metaheuristics International Conference*, pages 403–408, 2005.

R Goldman, M Boddy, and M Ringer. Constraint-based scheduling for batch manufacturing. In *Proceedings of the Artifical Intelligence and Manufacturing Research Planning Workshop*, 1996.

R.P. Goldman and M.S. Boddy. A constraint-based scheduler for batch manufacturing. *IEEE Expert*, 12 (1):49–56, Jan 1997.

C.a b c Guerin, J.-M.b c Hoc, and N.b c Mebarki. The nature of expertise in industrial scheduling: Strategic and tactical processes, constraint and object management. *International Journal of Industrial Ergonomics*, 42(5):457–468, 2012.

F. Guerriero and R. Guido. Operational research in the management of the operating theatre: A survey. *Health Care Management Science*, 14(1):89–114, 2010.

A. Guinet and S. Chaabane. Operating theatre planning. *International Journal of Production Econocmics*, 85:69–81, 2003.

Z.X.a Guo, E.W.T.a Ngai, C.b Yang, and X.a Liang. An RFID-based intelligent decision support system architecture for production monitoring and scheduling in a distributed manufacturing environment. *International Journal of Production Economics*, 159:16–28, 2015.

A.K. Gupta, A.I. Sivakumar, and S. Sarawgi. Shop floor scheduling with simulation based proactive decision support. In *Simulation Conference, 2002. Proceedings of the Winter*, volume 2, pages 1897–1902 vol.2, Dec 2002.

D. Gupta. Surgical suites' operations management. *Production and Operations Management*, 16(6): 689–700, 2007.

Jatinder N.D. Gupta. Two-stage, hybrid flowshop scheduling problem. *Journal of the Operational Research Society*, 39(4):359–364, 1988.

K. Hadavi, M.S. Shahraray, and K. Voigt. Reds-a dynamic planning, scheduling, and control system for manufacturing. *Journal of Manufacturing Systems*, 9(4):332–344, 1990.

K. Hadavi, W.-L. Hsu, T. Chen, and C.-N. Lee. An architecture for real-time distributed scheduling. *AI Magazine*, 13(3):46–56, 1992.

E. W. Hans, G. Wullink, M. van Houdenhoven, and G. Kazeimer. Robust surgery loading. *European Journal of Operational Research*, 185:1038–1050, 2008.

E. W. Hans, M. V. Houdenhoven, and P. J. H. Hulshof. A framework for health care planning and control. Memorandum 1938, University of Twente, 2011.

G.P. Henning and Cerda J. Knowledge-based predictive and reactive scheduling in industrial environments. *Computers and Chemical Engineering*, 24(9-10):2315–2338, 2000.

Jeffrey W. Herrmann. *Handbook of production scheduling*. Springer US, 2006.

K. Hindle and M. Duffin. Simul8-planner for composites manufacturing. In *Proceedings - Winter Simulation Conference*, pages 1779–1784, 2006.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

W.-L.a Hsu, M.J.a Prietula, G.L.a Thompson, and P.b Si Ow. A mixed-initiative scheduling workbench integrating ai, or and hci. *Decision Support Systems*, 9(3):245–257, 1993.

D. J. Iaconetti, S. A. Lussos, J. Koch, and C. Falcone. Anesthesia information management systems and operating room management: The innova fairfax hospital experience. *Seminars in Anesthesia, Perioperative Medicine and Pain*, 23:104–114, 2004.

S.a c Jackson, J.R.a b c d Wilson, and B.L.a e f g MacCarthy. A new model of scheduling in manufacturing: Tasks, roles, and monitoring. *Human Factors*, 46(3):533–550, 2004.

S. Jacobi, E. Leon-Soto, C. Madrigal-Mora, and K. Fischer. MasDISPO: A multiagent decision support system for steel production and control. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence and the 19th Innovative Applications of Artificial Intelligence Conference*, volume 2, pages 1707–1714, 2007.

A. Jeang and A. J. Chiang. Economic and quality scheduling for effective utilization of operating rooms. *Journal of Medical Systems*, 36(3):1–18, 2010.

A. Jebali, A. B. H. Alouane, and P. Ladet. Operating rooms scheduling. *International Journal of Production Economics*, 99:52–62, 2006.

Gunnar Johannsen. Design of intelligent human-machine interfaces. In *Proceedings of the IEEE International Workshop on Robot and Human Communication*, pages 18–25, 1994.

J. Józefowska and A. Zimniak. Optimization tool for short-term production planning and scheduling. *International Journal of Production Economics*, 112(1):109 – 120, 2008. Special Section on Recent Developments in the Design, Control, Planning and Scheduling of Productive Systems.

N.I.a Karacapilidis and C.P.b Pappis. Production planning and control in textile industry: A case study. *Computers in Industry*, 30(2):127–144, 1996.

R. Kaushal, K. G. Shojania, and D. W. Bates. Effects of computerized physician order entry and clinical decision support systems on medication safety: A systematic review. *Archives of Internal Medicine*, 163(12):1409–1416, 2003.

K. G. Kempf. *Intelligent Scheduling*, chapter Intelligently Scheduling Semiconductor Wafer Fabrication, pages 517–545. Morgan Kaufmann Publishers Inc., 1994.

R.M.a Kerr and R.V.b Ebsary. Implementation of an expert system for production scheduling. *European Journal of Operational Research*, 33(1):17–29, 1988.

Pinar Keskinocak, Frederick Wu, Richard Goodwin, Sesh Murthy, Rama Akkiraju, Santhosh Kumaran, and Annap Derebail. Scheduling solutions for the paper industry. *Operations Research*, 50(2):249–259, 2002.

S. Kharraja, P. Albert, and S. Chaabane. Block scheduling: Toward a master surgical schedule. In *International Conference on Service Systems and Service Management*, pages 429–435, October 2006 2006.

D. Kizilay, M.F. Tasgetiren, Quan ke Pan, and Ling Wang. An iterated greedy algorithm for the hybrid flowshop problem with makespan criterion. In *Computational Intelligence in Production and Logistics Systems (CIPLS), 2014 IEEE Symposium on*, pages 16–23, Dec 2014.

Chien-Ho Ko and Shu-Fan Wang. Ga-based decision support systems for precast production planning. *Automation in Construction*, 19(7):907 – 916, 2010.

A.W.J. Kolen and A. Woerlee. Vips: a decision support system for visual interactive production scheduling. Designing Decision Support Systems Notes 8805, Eindhoven University of Technology, 1988.

P.a Korosec, U.b Bole, and G.a b Papa. A multi-objective approach to the application of real-world production scheduling. *Expert Systems with Applications*, 40(15):5839–5853, 2013.

D.B. Kotak, M. Fleetwood, H. Tamoto, and W.A. Gruver. Operational scheduling for rough mills using a virtual manufacturing environment. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 1, pages 140–145 vol.1, 2001.

G. E. Krasner and S. T. Pope. A cookbook for using the model-view-controller user interface paradigm in smalltalk-80. *Journal of Object Oriented Programming*, 1(3):26–49, 1988.

Manish Kumar and Sunil Rajotia. Integration of process planning and scheduling in a job shop environment. *The International Journal of Advanced Manufacturing Technology*, 28(1-2):109–116, 2006.

W.-H. Kuo and S.-L. Hwang. A prototype of a real-time support system in the scheduling of production systems. *International Journal of Industrial Ergonomics*, 21(2):133–143, 1998.

M.E.a Kurz and R.G.b Askin. Comparing scheduling rules for flexible flow lines. *International Journal of Production Economics*, 85(3):371–388, 2003.

A.c Lamatsch, M.c Morlock, K.a Neumann, and T.b Rubach. Schedule-an expert-like system for machine scheduling. *Annals of Operations Research*, 16(1):425–438, 1988.

M. Lamiri and X. Xie. Operating rooms planning using lagrangian relaxation technique. In *IEEE International Conference on Automation Science and Engineering*, pages 176–181, 8-10 Oct. 2006.

M. Lamiri, J. Dreo, and X. Xie. Operating room planning with random surgery times. In *IEEE International Conference on Automation Science and Engineering*, pages 521–526, 2007.

M. Lamiri, V. Augusto, and X. Xie. Patients scheduling in a hospital operating theatre. In *4th IEEE Conference on Automation Science and Engineering*, pages 627–632, August 23-26, 2008 2008a.

M. Lamiri, X. Xie, A. Dolgui, and F. Grimaud. A stochastic model for operating room planning with elective and emergency demand for surgery. *European Journal of Operational Research*, 185(3):1026–1037, 2008b.

M. Lamiri, X. Xie, and S. Zhang. Column generation approach to operating theater planning with elective and emergency patients. *IIE Transactions*, 40:838–852, 2008c.

M. Lamiri, F. Grimaud, and X. Xie. Optimization methods for a stochastic surgery planning problem. *International Journal of Production Econocmics*, 120:400–410, 2009.

Eugene L. Lawler. Scheduling a single machine to minimize the number of late jobs. Technical report, EECS Department, University of California, Berkeley, 1983.

C. Le Pape. *Intelligent Scheduling*, chapter Scheduling as Intelligent Control of Decision-Making and Constraint Propagation, pages 67–99. Morgan Kaufmann Publishers Inc., 1994.

R. Leisten and M. Kolbe. A note on scheduling jobs with missing operations in permutation flow shops. *International Journal of Production Research*, 36(9):2627–2630, 1998.

V.J. Leon and B. Ramamoorthy. An adaptable problem-space-based search method for flexible flow line scheduling. *IIE Transactions*, 29(2):115–125, 1997.

Y.K. Leung, K.L. Choy, and C.K. Kwong. A real-time hybrid information-sharing and decision support system for the mould industry. *The Journal of High Technology Management Research*, 21(1):64 – 77, 2010. Exploring Technological Innovation.

H.a Li, Z.a Li, L.X.b Li, and B.a Hu. A production rescheduling expert simulation system. *European Journal of Operational Research*, 124(2):283–293, 2000.

R. Linn and W. Zhang. Hybrid flow shop scheduling: a survey. *Computers and Industrial Engineering*, 37(1):57–61, 1999.

Y. Liu, C. Chu, and K. Wang. A new heuristic algorithm for the operating room scheduling problem. *Computers & Industrial Engineering*, 61:865–871, 2011.

Omar Lopez and Israel Villar. A multi-agent system to construct production orders by employing an expert system and a neural network. *Expert Systems with Applications*, 36(2, Part 2):2937 – 2946, 2009.

B.L.a b MacCarthy, J.R.a Wilson, and S.a b Crawford. Human performance in industrial scheduling: A framework for understanding. *Human Factors and Ergonomics In Manufacturing*, 11(4):299–320, 2001.

A. Madureira, S. Gomes, B. Cunha, J.P. Pereira, J.M. Santos, and I. Pereira. Prototype of an adaptive decision support system for interactive scheduling with metacognition and user modeling experience. In *2014 6th World Congress on Nature and Biologically Inspired Computing, NaBIC 2014*, pages 145–150, 2014a.

A.a Madureira, I.a Pereira, P.a Pereira, and A.b Abraham. Negotiation mechanism for self-organized scheduling system with collective intelligence. *Neurocomputing*, 132:97–110, 2014b.

E. Marcon, S. Kharraja, and G. Simonnet. The operating theatre planning by the follow-up of the risk of no realization. *International Journal of Accounting Information Systems*, 85:83–90, 2003.

M.K.a Marichelvam and T.b Prabaharan. Performance evaluation of an improved hybrid genetic scatter search (ihgss) algorithm for multistage hybrid flow shop scheduling problems with missing operations. *International Journal of Industrial and Systems Engineering*, 16:120–141, 2014.

Jose Marinho, Alexandre Braganca, and Carlos Ramos. Decision support system for dynamic production scheduling. In *Proceedings of the IEEE International Symposium on Assembly and Task Planning*, pages 424–429, 1999.

Ines Marques, M.Eugenia Captivo, and Margarida Vaz Pato. An integer programming approach to elective surgery scheduling. *OR Spectrum*, 34(2):407–427, 2012. ISSN 0171-6468. doi: 10.1007/s00291-011-0279-7. URL `http://dx.doi.org/10.1007/s00291-011-0279-7`.

Bob Marriott. Production scheduling systems using provisa. In *Winter Simulation Conference Proceedings*, pages 522–526, 1994.

Sergio Maturana, Enzo Pizani, and Jorge Vera. Scheduling production for a sawmill: A comparison of a mathematical model versus a heuristic. *Computers & Industrial Engineering*, 59(4):667 – 674, 2010.

J.H. May and L.G. Vargas. Simpson: An intelligent assistant for short-term manufacturing scheduling. *European Journal of Operational Research*, 88(2):269–286, 1996.

B. L. McCarthy and Jiyin Liu. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*, 31(1):59–79, 1993.

S.Thomas McCormick, Michael L. Pinedo, Scott Shenker, and Barry Wolf. Sequencing in an assembly line with blocking to minimize cycle time. *Operations Research*, 37(6):925–935, 1989.

Kenneth N McKay and John A Buzacott. The application of computerized production control systems in job shop environments. *Computers in Industry*, 42:79 – 97, 2000.

K.N. McKay and V. C. S. Wiers. Unifying the theory and practice of production scheduling. *Journal of Manufacturing Systems*, 18(4):241–255, 1999.

K.N.a McKay and V.C.S.b Wiers. Integrated decision support for planning, scheduling, and dispatching tasks in a focused factory. *Computers in Industry*, 50(1):5–14, 2003.

K.N.a McKay, F.R.b Safayeni, and J.A.c Buzacott. "common sense" realities of planning and scheduling in printed circuit board production. *International Journal of Production Research*, 33(6):1587–1603, 1995.

R.F.a McPherson and K.P.b White Jr. A framework for developing intelligent real-time scheduling systems. *Human Factors and Ergonomics In Manufacturing*, 16(4):385–408, 2006.

J. Meneses, M. Boixadós, L. Valiente, and M. Armayones. Construcción de estrategias sistemáticas para la búsqueda exhaustiva de información en internet: un marco de toma de decisiones aplicado a la información sobre psicología de la salud. *Information Research*, 10:231, 2005.

Nadine Meskens, David Duvivier, and Arnauld Hanset. Multi-objective operating room scheduling considering desiderata of the surgical team. *Decision Support Systems*, 55(2):650 – 659, 2013. ISSN 0167-9236. doi: http://dx.doi.org/10.1016/j.dss.2012.10.019. URL `http://www.sciencedirect.com/science/article/pii/S0167923612002746`. 1. Analytics and Modeling for Better HealthCare 2. Decision Making in Healthcare.

K.S. Metaxiotis, J.E. Psarras, and D.T. Askounis. Genesys: an expert system for production scheduling. *Industrial Management & Data Systems*, 102(6):309–317, 2002.

D. Min and Y. Yih. Scheduling elective surgery under uncertainty and downstream capacity constraints. *European Journal of Operational Research*, 206:642–652, 2010.

H.a Missbauer, W.b Hauber, and W.c Stadler. A scheduling system for the steelmaking-continuous casting process. a case study from the steel-making industry. *International Journal of Production Research*, 47 (15):4147–4172, 2009.

D. Mourtzis, M. Doukas, and E. Vlachou. A mobile application for knowledge-enriched short-term scheduling of complex products. *Logistics Research*, 9(1):1–17, 2016.

S.A. Munawar, Mangesh D. Kapadi, S.C. Patwardhan, K.P. Madhavan, S. Pragathieswaran, P. Lingathurai, and Ravindra D. Gudi. Integration of planning and scheduling in multisite plants: Application to paper manufacturing. In Luis Puigjaner and Antonio Espuna, editors, *European Symposium on Computer Aided Process Engineering 15, 38th European Symposium of the Working Party on Computer Aided Process Engineering*, volume 20 of *Computer Aided Chemical Engineering*, pages 1621 – 1626. Elsevier, 2005.

S. Murthy, R. Akkiraju, R. Goodwin, P. Keskinocak, J. Rachlin, W. Frederick, J. Yeh, R. Fuhrer, S. Kumaran, A. Aggarwal, M. Sturzenbecker, R. Jayaraman, and R. Daigle. Cooperative multiobjective decision support for the paper industry. *Interfaces*, 29(5):5–30, 1999.

K.J Musselman. Complex scheduling of a printing process. *Computers & Industrial Engineering*, 39:273 – 291, 2001.

B.a Naderi, R.b Ruiz, and M.c Zandieh. Algorithms for a realistic variant of flowshop scheduling. *Computers and Operations Research*, 37(2):236–246, 2010.

M.a Nawaz, E.E.b Enscore Jr., and I.b Ham. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95, 1983.

Inneke Van Nieuwenhuyse, Liesje De Boeck, Marc Lambrecht, and Nico J. Vandaele. Advanced resource planning as a decision support module for erp. *Computers in Industry*, 62(1):1 – 8, 2011.

B.C. Niew, B.S. Lim, and N.C. Ho. Knowledge based master production scheduler. In *IEE Conference Publication*, pages 88–93, 1990.

M. Numao. *Intelligent Scheduling*, chapter Development of a Cooperative Scheduling System for the Steel-Making Process, pages 607–629. Morgan Kaufmann Publishers Inc., 1994.

M. Numao and S. Morishita. A scheduling environment for steel-making processes. In *Proceedings of the Fifth Conference on Artificial Intelligence Applications*, pages 279–286, 1989.

M. Numao and S. Morishita. Cooperative scheduling and its application to steelmaking processes. *Industrial Electronics, IEEE Transactions on*, 38(2):150–155, Apr 1991.

Masayuki Numao and Shin-ichi Morishita. Scheplan - a scheduling expert for steel-making process. In *Proceedings of the International Workshop on Artificial Intelligence for Industrial Applications*, pages 467–472, 1988.

D.a O'Donoghue, E.b Healy, and H.b Sorensen. Development of a rule-based finite capacity scheduling system. *International Journal of Production Economics*, 36(2):221–227, 1994.

S. N. Ogulata and R. Erol. A hierarchical multiple criteria mathematical programming approach for scheduling general surgery operations in large hospitals. *Journal of Medical Systems*, 27(3):259–270, 2003.

Mustafa Ozbayrak and Robert Bell. A knowledge-based decision support system for the management of parts and tools in fms. *Decision Support Systems*, 35(4):487 – 515, 2003.

I. Ozkarahan. Allocation of surgical procedures to operating rooms. *Journal of Medical Systems*, 19(4): 333–352, 1995.

Irem Ozkarahan. Allocation of surgeries to operating rooms by goal programing. *Journal of Medical Systems*, 24(6):339–378, 2000.

Quan-Ke Pan, Ling Wang, Jun-Qing Li, and Jun-Hua Duan. A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation. *Omega*, 45(0):42 – 56, 2014.

J. M. Molina Pariente and J. M. Framinan. Testing planning policies for solving the elective case scheduling phase: A real application. In *International Conference on Operational Research Applied to Health Services - ORAHS*, 2009.

Bela Patkai. Cooperative scheduling system for paper machinery manufacturing. In *IECON Proceedings (Industrial Electronics Conference)*, volume 4, pages 2518–2521, 1998.

P. Perez-Gonzalez and J.M. Framinan. Single machine interfering jobs problem with flowtime objective. *Journal of Intelligent Manufacturing*, 2015. doi: 10.1007/s10845-015-1141-6.

D. N Pham and A. Klinkert. Surgical case scheduling as a generalized job shop scheduling problem. *European Journal of Operational Research*, 185:1011–1025, 2008.

J. Piairo, A. Madureira, J.P. Pereira, and I. Pereira. A user-centered interface for scheduling problem definition. *Advances in Intelligent Systems and Computing*, 206 AISC:1063–1073, 2013.

M.a Pinedo and B.P.-C.a b Yen. On the design and development of object-oriented scheduling systems. *Annals of Operations Research*, 70:359–378, 1997.

Michael L. Pinedo. *Scheduling. Theory, Algotihms and Systems*. Springer New York, 4th edition, 2012.

M. Prietula, W.-L. Hsu, P.S. Ow, and G.L. Thompson. *Intelligent Scheduling*, chapter MacMerl: Mixed-Initiative Scheduling with Coincident Problem Spaces, pages 655–683. Morgan Kaufmann Publishers Inc., 1994.

C. Rajendran. Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. *International Journal of Production Economics*, 29(1):65–73, 1993.

Chandrasekharan Rajendran and Hans Ziegler. A performance analysis of dispatching rules and a heuristic in static flowshops with missing operations of jobs. *European Journal of Operational Research*, 131(3): 622 – 634, 2001.

J. Rasmusen, A. M. Pejtersen, and L. P. Goodstein. *Cognitive Systems Engineering*. Wiley, 1994.

James Reason. *Human Error*. University of Cambridge, 2003.

Imma Ribas, Rainer Leisten, and Jose M. Framinan. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, 37(8):1439 – 1454, 2010.

A. Riise and E. K. Burke. Local search for the surgery admission planning. *Journal of Heuristics*, 17: 389–414, 2011.

B. Roland, C. Di Martinelly, F. Riane, and Y. Pochet. Scheduling an operating theatre under human resource constraints. *Computers & Industrial Engineering*, 58:212–220, 2010.

R.a Romero-Silva, J.b Santos, and M.a Hurtado. A framework for studying practical production scheduling. *Production Planning and Control*, 26(6):438–450, 2015.

R Ruiz and T Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049, 2007.

R.a Ruiz, F.S.b Serifoglu, and T.c Urlings. Modeling realistic hybrid flexible flowshop scheduling problems. *Computers and Operations Research*, 35(4):1151–1175, 2008.

Ruben Ruiz and Jose Antonio Vazquez-Rodriguez. The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205(1):1 – 18, 2010.

N. Sadeh. Micro-boss: A micro-opportunistic factory scheduler. *Expert Systems With Applications*, 6(3): 377–392, 1993.

N. Sadeh. *Intelligent Scheduling*, chapter Micro-Opportunistic Scheduling: The Micro-Boss Factory Scheduler, pages 99–137. Morgan Kaufmann Publishers Inc., 1994.

S. Sadi-Nezhad and S.B. Darian. Production scheduling for products on different machines with setup costs and times. *International Journal of Engineering and Technology*, 2(6):410–418, 2010.

P. Santibanez, M. Begen, and D. Atkins. Surgical block scheduling in a system of hospitals: An application to resource and wait list management in a british columbia health authority. *Healthcare Management Science*, 10:269–282, 2007.

M.a Saravanan, S.b Sridhar, and N.b Harikannan. Application of meta-heuristics for minimization of makespan in multi-stage hybrid flow shop scheduling problems with missing operations. *International Review of Mechanical Engineering*, 8(5):916–923, 2014.

J.a b Sauer and R.a c Bruns. Knowledge-based scheduling systems in industry and medicine. *IEEE Expert-Intelligent Systems and their Applications*, 12(1):24–31, 1997.

Jurgen Sauer. *Scheduling of Production Processes*, chapter Meta-scheduling using dynamic scheduling knowledge, pages 151–162. Ellis Horwood, 1993.

Jurgen Sauer, Gerd Suelmann, and Hans-Jurgen Appelrath. Multi-site scheduling with fuzzy concepts. *International Journal of Approximate Reasoning*, 19(1-2):145 – 160, 1998.

S. Sauer and Engels G. Mvc-based modeling support for embedded real-time systems: Position statement. In *Workshop on Object-Oriented Modeling of Embedded Realtime Systems*, 1999.

Daryl V. Savell, Rafael A. Perez, and Song W. Koh. Scheduling semiconductor wafer production: An expert system implementation. *IEEE Expert*, 4(3):9–15, 1989.

Cem Saydan and W. Douglas Cooper. A decision support system for scheduling jobs on multi-port dyeing machines. *International Journal of Operations & Production Management*, 22:1054–1065, 2002.

H. Shams and K. Zamanifar. Mvcc: An architectural pattern for developing context-aware frameworks. In *Proceedings of the 11th International Conference on Mobile Systems and Pervasive Computing*, volume 34, pages 344–351, 2014.

D.E.a Shobrys and O.C.b White. Planning, scheduling and control systems: Why can they not work together. *Computers and Chemical Engineering*, 24(2-7):163–173, 2000.

D. Sier, P. Tobin, and C. McGurk. Scheduling surgical procedures. *Journal of the Operational Research Society*, 48:884–891, 1997.

Cristovao Silva. Combining ad hoc decision-making behaviour with formal planning and scheduling rules: a case study in the synthetic fibre production industry. *Production Planning & Control*, 20(7):636–648, 2009.

Cees De Snoo, Wout Van Wezel, and René J. Jorna. An empirical investigation of scheduling performance criteria. *Journal of Operations Management*, 29(3):181 – 193, 2011.

G. Sotiris, S. Athanasios, and T. Ilias. A decision support system for detailed production scheduling in a greek metal forming industry. *MIBES Transactions*, 2(1):41–59, 2008.

J. F. Sowa and J.A. Zachman. Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, 31:590 – 616, 1992.

M.G.a Speranza and A.P.b Woerlee. A decision support system for operational production scheduling. *European Journal of Operational Research*, 55(3):329–343, 1991.

R. H. Sprague. A framework for the development of decision support systems. *MIS Quarterly*, 4(4):1–26, 1980.

J. Sridhar and C. Rajendran. Scheduling in a cellular manufacturing system: a simulated annealing approach. *International Journal of Production Research*, 31(12):2927–2945, 1993.

M. Stevenson, Y. Huang, and L.C. Hendry. The development and application of an interactive end-user training tool: Part of an implementation strategy for workload control. *Production Planning and Control*, 20(7):622–635, 2009.

Paul P.M. Stoop and Vincent C.S. Wiers. The complexity of scheduling in practice. *International Journal of Operations & Production Management*, 16(10):37–53, 1996.

Kari Stuart and Erhan Kozan. Reactive scheduling model for the operating theatre. *Flexible Services and Manufacturing Journal*, 24(4):400–421, 2012.

E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2): 278–285, 1993.

E. D. Taillard. Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 47(1):65–74, 1990.

E. Tanfani and A. Testi. A preassignment heuristic algorithm for the master surgical schedule problem (mssp). *Annals of Operations Research*, 178(1):105–119, 2010.

Lixin Tang, Xiaoxia Zhang, and Qingxin Guo. Two hybrid metaheuristic algorithms for hot rolling scheduling. *ISIJ International*, 49(4):529–538, 2009.

J.C. Taunton and C.M. Ready. Intelligent dynamic production scheduling. *Food Research International*, 27(2):111–116, 1994.

A. Testi and E. Tanfani. Tactical and operational decisions for operating room planning: Efficiency and welfare implications. *Health Care Management Science*, 12:363–373, 2009.

A. Testi, E. Tanfani, and G. Torre. A three-phase approach for operating theatre schedules. *Healthcare Management Science*, 10:163–172, 2007.

V. T'kindt, J.-C. Billaut, J.-L. Bouquard, C. Lenté, P. Martineau, E. Néron, C. Proust, and C. Tacquard. The e-ocea project: towards an internet decision system for scheduling problems. *Decision Support Systems*, 40(2):329 – 337, 2005.

Mariem Trojet, Fehmi HMida, and Pierre Lopez. Project scheduling under resource constraints: Application of the cumulative global constraint in a decision support framework. *Computers & Industrial Engineering*, 61(2):357 – 363, 2011.

C.-T.a Tseng, C.-J.b Liao, and T.-X.b Liao. A note on two-stage hybrid flowshop scheduling with missing operations. *Computers and Industrial Engineering*, 54(3):695–704, 2008.

C.a Upton and F.b Quilligan. GREYBOX scheduling: Designing a joint cognitive system for sustainable manufacturing. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 897–900, 2014.

Bharath S. Vaidyanathan, David M. Miller, and Young H. Park. Application of discrete event simulation in production scheduling. In *Winter Simulation Conference Proceedings*, volume 2, pages 965–971, 1998.

M. L. R. Varela, R. A. Ribeiro, and S. Carmo-Silva. A manufacturing scheduling web-based decision support system. In *Proceedings of the 14º Congresso da APDIO*, pages 157–164, 7-9 September 2009.

Guilherme E. Vieira, Jeffrey W. Herrmann, and Edward Lin. Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling*, 6(1):39–62, 2003.

A. Vignier, J.-C. Billaut, and C. Proust. Les problèmes d'ordonnancement de type flow-shop hybride : état de l'art. *RAIRO - Operations Research*, 33(2):117–183, 3 1999.

T.a b Vogel, B.b Almada-Lobo, and C.a Almeder. Integrated versus hierarchical approach to aggregate production planning and master production scheduling. *OR Spectrum*, pages 1–37, 2016.

D. Wang and J. Xu. A fuzzy multi-objective optimizing scheduling for operation room in hospital. In *IEEE International Conference on Industrial Engineering and Engineering Management*, pages 614–618, 2008.

Kesheng Wang and Wenfen Lin. The application of integrated intelligent systems (iis) to production scheduling. In Frank Plonka and Gustav Olling, editors, *Computer Applications in Production and Engineering*, IFIP : The International Federation for Information Processing, pages 578–591. Springer US, 1997.

Liupu Wang, Juexin Wang, Michael Wang, Yong Li, Yanchun Liang, and Dong Xu. Using internet search engines to obtain medical information: A comparative study. *J Med Internet Res*, 14(3):e74, 2012.

Yu Wang, Jiafu Tang, and Gang Qu. A genetic algorithm for solving patient- priority- based elective surgery scheduling problem. In Kang Li, Minrui Fei, Li Jia, and GeorgeW. Irwin, editors, *Life System Modeling and Intelligent Computing*, volume 6329 of *Lecture Notes in Computer Science*, pages 297–304. Springer Berlin Heidelberg, 2010.

K.Heinz Weigl. Combining simulation and scheduling: an application in the office furniture industry using arena and preactor. In *Winter Simulation Conference Proceedings*, pages 977–981, 1995.

A. A. Weinbroum, P. Ekstein, and T. Ezri. Efficiency of the operating room suite. *The American Journal of Surgery*, 185:244–250, 2003.

Alexander J Weintraub, Andrew Zozom Jr, Thorn J Hodgson, and Denis Cormier. A simulation-based finite capacity scheduling system. In *Proceedings of the 29th conference on Winter simulation*, pages 838–844, 1997.

V.C.S. Wiers. A review of the applicability of or and ai scheduling techniques in practice. *Omega*, 25(2): 145–153, 1997.

P. Xiao, P. Hu, J. Moss, C. F. De Winter, D. Venekamp, C.F. Mackenzie, J. Seagull, and S. Perkins. Opportunities and challenges in improving surgical work flow. *Cognition, Technology & Work*, 10: 313–321, 2008.

M.-F. Yang and Y. Lin. The coordinated scheduling support system of production and delivery. *American Journal of Applied Sciences*, 6(4):601–607, 2009.

Benjamin Ping-Chang Yen. Interactive scheduling agent on the internet. In *Proceedings of the Hawaii International Conference on System Sciences*, volume 4, pages 220–229, 1997.

J.A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 26:276 – 292, 1987.

Xiaoxia Zhang, Liwen Dong, and Qiuying Bai. A decision support system with ct_aco algorithm for the hot rolling scheduling. In *Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference on*, volume 1, pages 65–68, 2010.

J.H. Zhao and J. Lin. Expert system aided multi-agent intelligent ant colony optimization system. *Applied Mechanics and Materials*, 121-126:2021–2025, 2011.

J. Zheng, Y. Takahashi, Y. Kobayashi, and T. Sato. Towards developing a self-explanatory scheduling system based on a hybrid approach. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 10(6):840–843, 2016.

Z. Zhiming. A two-stage scheduling approach of operation room considering uncertain operation time. In *International Conference on Information Science and Technology*, pages 1225–1228, China, March 26-28 2011.

M. Zhongyi, M. Younus, and L. Yongjin. Automated planning and scheduling system for the composite component manufacturing workshop. *Engineering Letters*, 19(1):1, 2011.

Y. Zong, T. Yang, and J.P. Ignizio. An expert system using an exchange heuristic for the resource-constrained scheduling problem. *Expert Systems With Applications*, 6(3):327–348, 1993.

Günther Zäpfel, Roland Braune, and Michael Bögl. *Metaheuristic Search Concepts. A Tutorial with Applications to Production and Logistics*. Springer, 2009.