Universidade do Minho
Escola de Engenharia

Cláudia Sofia Rodrigues Duarte

Single line for assembly just-in-sequence multiple models

Junho de 2013

Cláudia Sofia Rodrigues Duarte **Single line for assembly just-in-sequence multiple models**

UMinho | 2013

Universidade do Minho
Escola de Engenharia

Cláudia Sofia Rodrigues Duarte

Single line for assembly just-in-sequence multiple models

Programa Doutoral em Líderes para as Indústrias Tecnológicas

Trabalho efetuado sob a orientação do
Professor José Manuel Vasconcelos Valério de Carvalho
Universidade do Minho
e co-orientação da
Professora Ana Paula Ferreira Dias Barbosa Póvoa
Instituto Superior Técnico

Junho de 2013

“Always do your best. What you plant now, you will harvest later.”

Og Mandino

ACKNOWLEDGMENTS

First of all I would like to thank my grandparents. They left to a better place in the middle of this journey and if I am finishing this work is because they continue to support me. Thank you for everything, you will always be part of who I am.

I would also like to thank my parents that supported me and encouraged me during this journey. Thank you also to my sisters and to my fiancé. They had a lot of patience with my mood fluctuations. It was not easy but with their help it was easier.

My inclusion in this project was only possible thanks to Professor António Cunha that helped me choosing the project at the beginning of my PhD.

Regarding my advisors, it was inspiring to see Professor Valério de Carvalho working and his commitment to this project. Professor Ana Paula Póvoa was an inspiration, her practical overview of the issues was very helpful while trying to achieve the goals of this project.

I would also like to thank Professor Stanley Gershwin for his advices at the beginning of this project and for the support during my stay at MIT.

The author, Cláudia Sofia Rodrigues Duarte, was supported by the Portuguese Foundation for Science and Technology (SFRH / BD / 43010 / 2008) and the MIT Portugal Program.

SINGLE LINE FOR ASSEMBLY JUST-IN-SEQUENCE MULTIPLE MODELS – ABSTRACT

The automotive industry is under a deep competitive reorganization process that manifests itself both on the demand and on the supply side. The competitiveness of this reorganization is highly dependent on a flexible production system, able to produce on demand, different vehicles (models) on a single assembly line. Due to demand requirements, production has to adjust faster to new models, each one with a large number of individual feature variants, and complexity grows in production. In addition, lean manufacturing principles introduced Just-in-Sequence as a further key issue of modern automotive production. In order to explore the single line concept related with the Just-in-Sequence principle, Car Sequencing policies avoiding blockage and starvation caused by product variety are needed.

The main goal of this project was the development of a mathematical model and a computational tool to define the car sequence in the final assembly line in a daily production run. The car sequence depends on the daily demand, called the production mix, and should avoid line stoppages and minimize the number of workers needed to complete the sequence in the minimum time.

This goal was achieved and we have created a new exact approach for Car Sequencing that considers limited capacity, special markets priorities and clustering of colors. An Integer Programming model was developed, but when considering clustering colors it became more complex and hard to solve. To overcome this difficulty a heuristic procedure was presented. Results show that the use of this new heuristic integrated with the exact integer model is a good approach for the Car Sequencing problems. The models were run in the software IBM ILOG 12.2 framework in a Intel® Core™2 Duo CPU T9600 Toshiba laptop @ 2.80GHz with 6 GB of RAM and we obtained good results for the heuristic in less than half an hour, for three hundred cars and seventeen options.

As a result of our work we show that the clustering of colors improves the performance of the global manufacturing system and that our tool can be used daily for Car Sequencing in automotive companies.

SINGLE LINE FOR ASSEMBLY JUST-IN-SEQUENCE MULTIPLE MODELS – RESUMO

Atualmente, a indústria automóvel encontra-se sobre profunda reorganização como resultado das alterações na procura dos automóveis. Estas reestruturações serão tanto mais competitivas quanto mais flexível for o sistema de produção, tendo capacidade de se adaptar a diferentes procuras, de diferentes variantes de carros com graus de complexidade diferentes e sendo capaz de produzir estes carros numa linha única. Os princípios Lean introduziram o conceito just-in-sequence como a chave para a modernização e para a capacidade de adaptação a esta nova realidade da indústria automóvel. O conceito de linha única associada ao conceito just-in-sequence levou à procura de políticas de sequenciação de carros que evitassem o bloqueio e a paragem das linhas.

O principal objetivo desta tese de doutoramento foi desenvolver um modelo matemático e uma ferramenta computacional para definir a sequência dos carros na montagem final, num dia de produção. A sequência de carros depende da procura diária e deve evitar paragens de linha e minimizar o número de operários necessários para completar a sequência no mínimo tempo possível.

Este objetivo foi alcançado e foi desenvolvida uma nova abordagem exata para sequenciar carros considerando capacidade limitada, prioridades para mercados especiais e o agrupamento de carros da mesma cor. Foi desenvolvido um modelo de programação inteira, mas quando se considerou o agrupamento de carros da mesma cor, o modelo tornou-se mais complexo e difícil de resolver. Por este motivo, foi criado um modelo heurístico integrado com o modelo inteiro exato, que resulta numa boa abordagem para os problemas de sequenciação. Os modelos foram testados no software IBM ILOG 12.2 num computador Intel® Core™2 Duo CPU T9600 Toshiba laptop @ 2.80GHz with 6 GB of RAM e obtiveram-se bons resultados em menos de meia hora, para trezentos carros e dezassete opções.

Como resultado deste trabalho demonstramos que agrupar os carros por cores na montagem final melhora a performance global do sistema de produção e que a nossa ferramenta pode ser usada diariamente para sequenciar os carros a produzir na montagem final da indústria automóvel.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The production in the same line of different cars just-in-sequence causes problems that raise important research challenges. Such challenges are related, amongst other aspects, to the balancing, sequencing, space allocation for feeding and work management in the single line. Also, the operation of a multiproduct single line is characterized by a high variability.

The above issues can be tackled with an appropriate design of the plant using the right approaches for balancing and sequencing complex and high variability work flows, together with well placed buffers to help reaching the adequate throughput with a minor increase of the Work in Progress while minimizing costs.

The studies that focus on the sequencing problems applied to the automotive industry do not exist in a large number in the literature. Most of them were developed in the ROADEF'2005, where Renault proposed a challenge to researchers in this field. The majority of these studies were developed considering heuristic approaches as Local Search and Ant Colony Optimization. Only few works as Drexl and Kimms (2001) and Prandstetter (2005) applied Exact techniques to the Sequencing problem. For this reason this is an area where improvements can be done and this is an opportunity to develop a relevant work in industrial environments trying to apply exact mathematical models to real problems. We will explore such opportunity for the above reasons and also as a result of the current manufacturing processes in the automotive industry and the necessity to react to diversified market needs. In the automotive sector, particularly, where several changes have been made at the production and logistic levels this is a relevant issue. Therefore, the design and planning of mixed model assembly lines appears nowadays as a core competence to be addressed where sequencing issues dealing with blockage and starvation caused by product variety need to be accounted for.

To study these issues we worked in strict relation with an European car manufacturer that holds a plant in Portugal. For confidentiality reasons we can not reveal the name of the company but all the developments achieved were shared and are being improved with this partner.

1.1 Problem Definition

Analyzing the economic situation in 2008 it is possible to identify that the European automotive industry faces new challenges related to the growing of competitiveness. As an important playmaker in the European automotive industry, our automotive partner had a huge impact in this project. In 2008 they needed to address volume, models flexibility and cost reduction to meet the new market challenges. Because of this, they decided to implement the concept of a single line, allowing to assemble just-in-sequence multiple models in a single assembly line. This final assembly line is producing, since 2010, four different models. However, two of these models are similar and are usually considered as one unique model.

In 2008, three possible areas of work were identified considering that in 2010 our automotive partner should begin the production of cars in a single assembly line. These areas are represented in Figure 1 and involve the production mix, the line balancing and the stochastic lines analysis. These three main areas are currently important research areas in automotive industries worldwide.

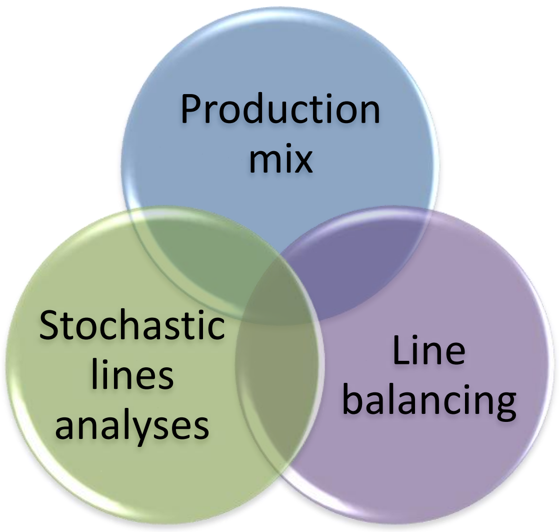


Figure 1: Possible areas of research.

These areas lead us to the identification of many problems that could be addressed including:

- definition of sequencing rules to improve the line performance;
- definition of a sequence, dependent on the daily demand;
- optimization of the line balancing to improve the line stoppage times;
- improving the line performance taking into account the production mix and line balancing - iterative process involving the three areas;
- and many others...

However, we and our automotive partner in several meetings have decided that the present project should focus on sequencing cars for the final assembly line so as to ensure the delivery of the vehicles just in sequence, without degrading the efficiency of the system. This problem is also an actual recent research problem in the scientific community with a high margin for improvements. Thence, in this project we intend to formulate:

- the decision problem that consists in deciding whether it is possible to find a sequence that satisfies all capacity constraints;
- the optimization problem that involves finding a minimum cost sequence, where the cost function evaluates constraint violations.

It is important that this formulation could give results in less than half an hour, allowing the planning and re-planning of the sequence for each shift, quickly enough not to disturb the current sequence that already entered in production. The re-planning is necessary when, for example, an urgent order is requested or when a supplier fails a delivery.

Summarizing, the research questions can be written as following:

- Is it possible to build a Car Sequencing model that given a daily demand determines the best sequence considering, by order of importance, the number of times that a capacity constraint is violated, that special cars should come first in the sequence and finally the spread of cars with the same color?
- Is it possible that this model gives solutions in less than half an hour, allowing the use of this tool daily for Car Sequencing in automotive companies?

1.2 Main Contributions

In the last years the mathematical programming software achieved great improvements. In the past it was only possible to solve small problems, but nowadays with computer and software performance improvements exact methods could be feasible. In order to contribute to the development of an exact method that could provide efficient solutions to real problems we explore the Car Sequencing problem trying to solve it using exact methods.

We defined in a multi-objective perspective that a good sequence is the one that considers by order of importance:

- Minimization of the number of times that a capacity constraint is violated using the ROADEF'2005 method (described in detail on subsection 4.3.2);
- Trying to place all the special cars first in the sequence (described in detail on subsection 4.3.3);
- Minimization of the spread of cars with the same color (described in detail on subsection 4.3.4).

The concept of an assembly sequence that minimizes the sum of the spread values of cars with the same color is different from the traditional concept of minimizing the number of color changes in the assembly sequence, presented for example in Prandstetter and Raidl (2008). Instead of minimizing just the number of color changes, our approach, tries to place the cars of a given color as close as possible in the assembly line. The detailed explanation of this concept can be found on subsection 4.3.4. As a result of our work we show that this new concept improves the performance of the global manufacturing system. However, the mathematical model that considers the spread of cars with the same color is more complex and harder to solve. For this reason, a new heuristic, also based on exact methods, was developed. The new heuristic provides good results in the 30 minutes constraint defined in the research questions, when tested for three different production mixes that might occur in different periods of the year.

Therefore as a result of this research, we prove that the developed tool can be used daily for Car Sequencing in automotive companies.

1.3 THESIS OVERVIEW

This PhD thesis is divided in four main parts:

- The characterization of automotive manufacturing systems considering our automotive partner as a case study that supports this characterization;
- The literature review of sequencing models;
- The mathematical model development;
- The conclusions and future work.

The first part presents the characterization of automotive production systems. This characterization is made in section 2. Based on the necessities of our automotive partner we decided to focus on the Car Sequencing Models as mentioned in section 1. Thus, we started the second part of this PhD thesis with a literature review that includes the characterization of assembly lines, in subsection 3.1. The literature review also includes the existing sequence models, in subsection 3.2, and the detailed literature review of the Car Sequencing Models in subsection 3.3. The third part of this work begins with an introduction to mathematical models made in subsections 4.1 and 4.2. The developed mathematical programming model is described in subsection 4.3. The model that considers the spread values, characterized in subsection 4.3.4, is harder to solve and for this reason a consolidation of the model was tried as shown in subsection 5.1. This consolidation improved the computational times in 45% but we believed that better results could be achieved with a new heuristic, based on exact methods. The heuristic is described in subsection 5.2. To test the quality of the solutions and the robustness of the approach, we developed a random generator that produces instances that resemble a daily production mix, and ran computational tests. This random generator and the process to create the instances are described in subsections 6.1 and 6.2. The new heuristic provides good results in acceptable computational times for three different classes of instances of 300 cars (maximum available capacity by shift of our automotive partner), each class resembling the production mix that occurs in a different period of the year. These results are presented in subsection 6.3. The last part of this PhD thesis begins in section 7 with the impacts of this new model and the conclusions. As a conclusion of this PhD work it seems to be possible to develop a tool that can be used daily for Car Sequencing in a real case. Finally, some directions for future work are presented in subsection 7.1.

2 STUDY AND CHARACTERIZATION OF THE CASE-STUDY MANUFACTURING SYSTEM

Our automotive partner was our guide in the development of our mathematical model. They helped us deciding the main aspects that should be taken into account when developing a generic model to sequence cars in a car manufacturer final assembly. Their experience in the area was motivating and the real examples given to test the model, of about 200 cars, were also of extreme relevance proving that the developed model can be applied to real automotive productions systems.

Data collection in our automotive partner was essential to understand which parameters and characteristics are important to develop a generic model to apply in Car Sequencing. This data includes information on production system inputs and outputs (see Figure 2).

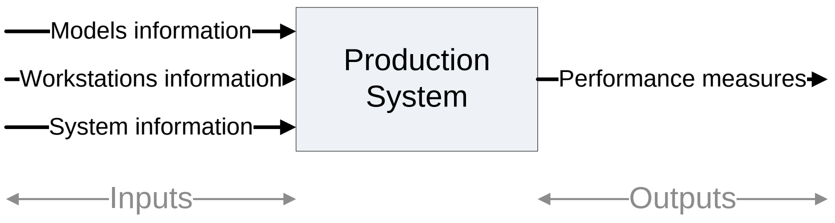


Figure 2: Generic representation of the data collected.

The data, represented in the Figure 2, includes information on:

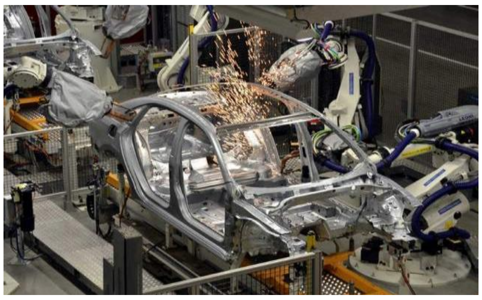
- *Car Models:* which cars types are produced, the operations needed to produce each unit, operation times and assembly constraints;
- *Workstations:* how many, dimension of each station and time needed to complete the operation in each station (cycle time);
- *System:* current balancing and sequencing approaches including cycle time, sequencing constraints and software used to sequence the cars;
- *Performance Measures:* number of capacity constraint violation in each sequence, number of special cars that should come first in the sequence, spread of each car color in the sequence and time to obtain a car sequence in the software used to develop the model.

This data was collected during a period of around three months. To maximize the efficiency during the data collection we have used maps and information of our automotive partner, daily, and we have designed and supervised the data collection system. This took a lot of time but was necessary to guarantee that the retrieved information was correct and was valid to be used for this thesis project.

A summary of important information obtained from the data collection, is described in the following subsections.

2.1 Car Manufacturing Processes

The car manufacturing process, as shown in Figure 3, is composed by three main areas: body, paint and final assembly.



Body Area (Bomey 2012)

Painting area (Pavarin 2009)

Final Assembly area (Dal Poggetto 2008)

Figure 3: Main car manufacturing processes.

In the Body Area the structure of the car is built by welding processes that join metal stampings and pressings. In this area, setup times on the presses are high, therefore the best sequence is the one that minimizes the setup times.

In the Painting area the body is treated, prepared and painted. An important objective in this area is to minimize the amount of solvent used to clean the painting nozzles for the following main reasons:

- When a color is changed the nozzles need to be cleaned with solvent;
- For maintenance reasons after an amount of utilizations the nozzles need to be cleaned with solvent;
- For quality reasons, when a defined maximum number of body cars with the same color are painted in a row, the nozzles need to be cleaned with solvent.

The time needed to clean the nozzles also represents a setup time that should be minimized.

In the Final Assembly area the mechanical, electrical and trim components are added to the car. Some of these components may require more work content operations. The cars that need these components should be dispersed throughout the sequence to smooth workload at the affected workstations. The objective is to ensure load balancing and component supply to minimize the times needed to build the sequence in the final assembly.

On average, 70% of the car value is added in the final assembly line. Consequently, Car Sequencing problems in the literature are more focused on finding a good sequence using the final assembly constraints that ensures load balancing and component supply (Gravel, Gagné, and Price 2005). Based on this, in the following section, a detailed explanation of the final assembly structure is described.

2.2 STRUCTURE OF FINAL ASSEMBLY

The final assembly of our automotive partner has the structure represented in Figure 4.

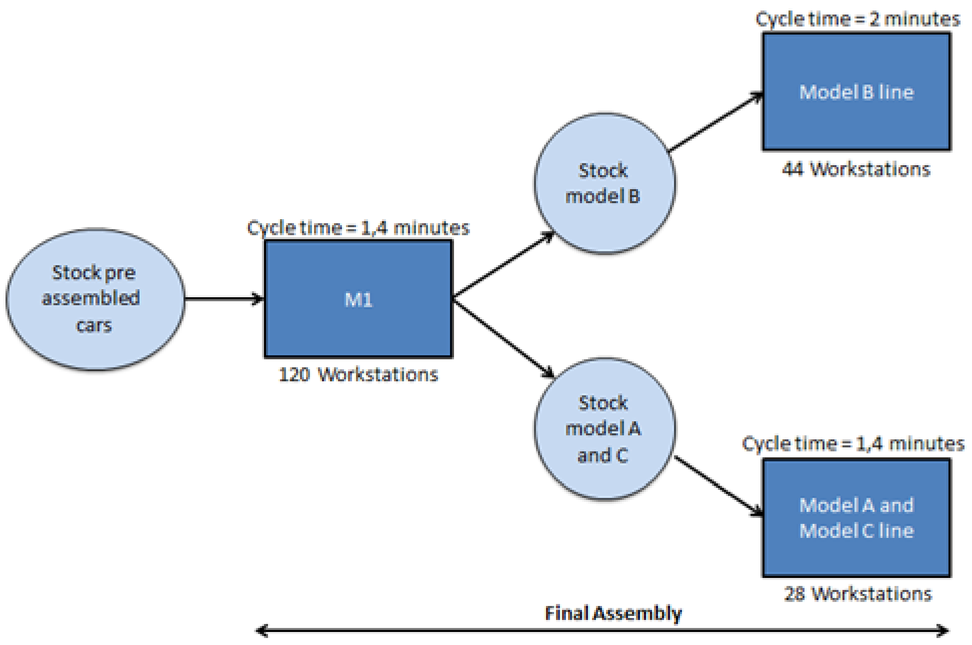


Figure 4: Structure of our automotive partner Final Assembly.

The division in two lines occurs just before the roof assembly of the cabriolet Model B. This operation takes more than the cycle time and this is the reason for the split. The split involves the duplication of some resources, for example, in the two lines exists a workstation to assemble the seats. These operations need specific tools that must exist in the two lines of the split.

Other implication of the split is the buffer needed between M1 and the Model B line. Suppose that there are two B Models in a row. The first leaves M1, with a cycle time of 1,4 min/unit, and enters in the Model B line, with a cycle time of 2,0 min/unit, which is slower. This will imply the existence of a buffer to prevent the stoppage of M1 when there are 2 B Models in a row.

The assembly lines in automotive industries are composed by a group of stations disposed in a line with a transportation system that assures the cars movement between the first and the last station. At each station workers execute different operations depending on the car model, while cars pass through the station. The stations have a length that corresponds to the production cycle time. After finishing the work in the car that is crossing the station, the worker returns to the beginning of his station (border of the station) or he stops near the next car. A representation of the workstations can be seen in Figure 5.

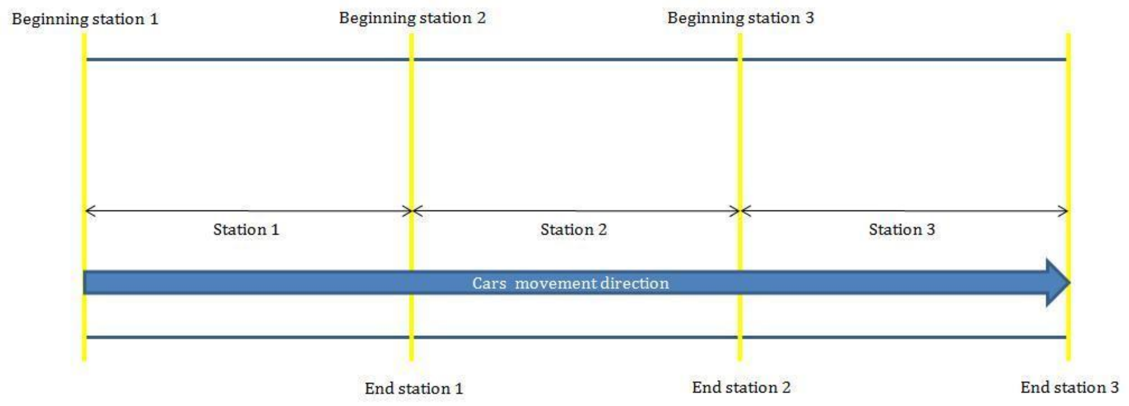


Figure 5: Assembly line and workstations representation.

Whenever the work defined for a station is not completed, there are two possible strategies to deal with it (Tsai 1995). In United States strategy, workers are not allowed to cross stations and utility workers are employed on an ad hoc basis to finish the work. In Japan strategy, the worker is able to stop the line, pushing a button, when he is not able to finish the work. Our automotive partner is trying to implement the United States strategy and for this reason it is important that the developed model in this PhD thesis considers the number of extra workers needed to assemble a sequence that do not obey to all the capacity constraints (check the details in subsection 4.3.2).

In order to simplify the problem, some assumptions were made in this thesis work:

- There are no buffers between the stations;
- The model-mix cannot be changed in the line (static problem);
- Multiple models are produced in the line. These models have different components, different operations and different processing times;
- There are a group of rules for sequencing the cars (see Table 1).

One of the identified problems is related to the time that cars spend in the workstations (cycle time) at the final assembly. This time is constant since the workstations have the same dimension. The cycle time is related to the velocity of the conveyor. If, for some reason, the time to complete the operations in each station exceeds the cycle time, the next model has to compensate this time. The following constraints, based on the constraints of our automotive partner, have this aspect into account and should be respected in the sequence. We will consider them to test our model.

Table 1: Cars constraints.

Option	Option Code	Models	Constraints (m:n – m out of n)
Model C	K8P	Model C	1:2
Model B	K8K	Model B	1:3
Right hand drive	LOR	Model A, B and C	1:3
Electric sliding door	GZ6	Model C	1:4
Tow bar	1D2/1M6	Model B and C	1:4
R 20/R Line	6EJ	Model A	1:26
Electric Tailgate	4E7	Model C	1:6
Japan cars	B29	Model A and C	1:6
Active Suspension	2H1	Model A, B and C	1:3
Alarm	7AS	Model C	1:4
4x4	1X1	Model C	1:16
Amplifier	9VE	Model A, B and C	1:3
Bad Floor Package	1SB	Model A	1:6
Head line with big console	7N3	Model C	1:2
3rd row seats	5KT	Model C	1:2
Slides	6L6	Model C	1:2

In the following subsection we will describe the demand characteristics along the year taking into consideration the demand of our automotive partner.

2.3 Demand characteristics along the year

The highest demand predicted is 698 cars by day. The production rate for the highest demand is:

- Model A: 44000 units/year (192 units/day);
- Model B: 35700 units/year (165 units/day);
- Model C that includes two similar MPV models: 52600 units/year (229 units/day) + 25732 units/year (112 units/day);
- Total: 158032 units/year (698 units/day).

Thus we have the units to be sequenced described in Table 2.

Table 2: Demand by day.

Product	Quantity by day
Model A	192
Model B	165
Model C	341

Based on this there are, approximately, $4.95*10^{1683}$ (698!) different possible sequences.

The production mix (A:B:C) shows the demand of cars by model, that have to be produced. Thus, and based on the above information, the production mix by day is on average 192:165:341. This production mix is used to establish the sequence in which cars are assembled in the line. The daily mix is calculated as a function of the demand for each model and follows the process represented in the Figure 6.

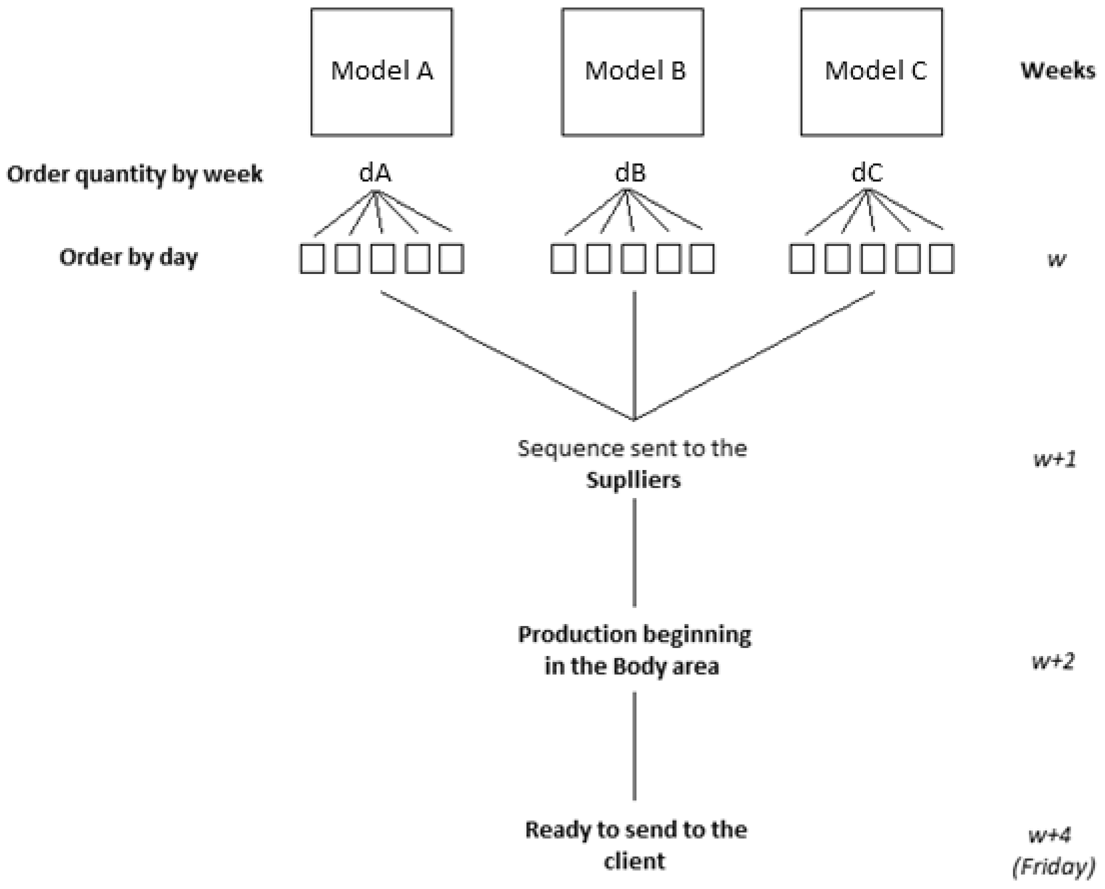

Model A
Model B
Model C
Weeks
Order quantity by week
dA
dB
dC
Order by day
w
Sequence sent to the Suplliers
w+1
Production beginning in the Body area
w+2
Ready to send to the client
w+4 (Friday)

Figure 6: Our partner process since the order of cars until the shipment to clients.

As shown, in Figure 6, the orders are sent in a week basis. These orders are first divided by each day of the week. One week later the sequence is sent to the suppliers. The production begins in the Body Area one week after the sequence has been sent to the suppliers. After two weeks, on Friday, the cars ordered four weeks before are ready to be sent to the clients. All quantities planned by day have to be ready to send to the client in the end of the last shift, according to the number of shifts needed to assemble the cars. If, for some reason, there is a priority, this one has to be mentioned to be considered in the mathematical model that will determine the sequence.

The demand for each car model varies along the year. Model B is the one with larger demand variability since it is a cabriolet model and presents a seasonal demand. The most critical period is before the summer, when the demand for Model B is higher.

The most important aspects of automotive manufacturing systems and some important details, of our automotive partner, to build our model, were explained in this subsection. In the following sections we will present a revision of the literature in the areas of assembly lines and Car Sequencing Models.

3 SEQUENCING MODELS FOR ASSEMBLY LINES

In this section, we will present the literature review in three main areas. The first one is about assembly lines. Since we are focusing in Car Sequencing Models in single assembly lines, it makes sense to present the main assembly line concepts to understand how assembly lines work. The second one is about sequencing models. In the literature the Car Sequencing is not the only type of sequencing model. Thus it makes sense to introduce the other types of sequencing models. Finally, we will present the literature review for Car Sequencing problems which is the main issue of this thesis.

3.1 ASSEMBLY LINES

Assembly lines are a special kind of flow-line production systems. In an assembly line there is a sequence of tasks each one with a process time and a set of precedence relations. These tasks are performed by operators and the work pieces have to pass through the workers one after another, in sequence.

When Henry Ford implemented the concept of assembly line to produce the Ford T he began a revolution that resulted in the modern automated assembly line concept. This type of lines is ideal to apply the concept of mass production that is based on the production of a small number of standardized products in large quantities. This concept allows (Scholl 1999):

- High capacity utilization;
- Small throughput times;
- Small in-process-inventories;
- Regular flow of materials that simplifies the materials control;
- Less space is needed for storage and material movement because work pieces are transferred in the lines;
- Small number of movements by the operator;
- Strict division of work that results in high specialization of labor and associated learning effects. Less skilled workers are needed and they can be trained more quickly.

However, there are also disadvantages in the concept of line assembly (Scholl 1999):

- The initial investments are high, especially if automatic equipment is needed;
- Changes in the processes are expensive because all the line is designed for the current products;
- The strict division of work creates simple and repetitive work. This influences negatively the workers satisfaction and origins high absenteeism and high turnover rates of employees;
- If one machine fails all the system may stop. For this reason, the quality control is very important.

Nowadays, the reasons that made this model prosper are disappearing. The abundance of resources and closed markets are not a reality anymore. Today the products life cycle is short, the markets are opened and there is a highly competitive context. Therefore, there is the need for an adaption to this new reality. Furthermore, in our days, the assembly lines have different technology levels according to different needs in terms of flexibility and volume requirements. Heilala and Voho (2001) classified the assembly line production strategies in:

- *Sequential manual assembly line* – the process is decomposed in small process steps. The tasks are very simple and because of this the system has great potential for automation. Manual tasks allow flexibility if the operators have the right knowledge;
- *Parallel manual assembly line* – one operator or a group of operators do all the assembly steps and are responsible for assembling one type of product. This factor increases job satisfaction and, consequently, the quality of the final product. There is potential for flexibility if the operators have the right knowledge;
- *Semi-automatic assembly lines* – part of the production system is automated. Here the most critical activities in terms of time, quality, and others, are automated. The key factors are the connections between operators and the automatic system, and the kind of pallet conveyor system used;

- *Flexible automatic assembly lines* – the assembly process is automated. This kind of system is indicated for high volume products in relatively big lots;
- *Dedicated automatic assembly lines* - the assembly process is automated. This kind of system is indicated for mass production (high production volume and low variety of products).

A new concept of assembly lines has emerged from the necessity of the companies to adapt to the new markets requirements. Meet these requirements imply to produce efficiently with high diversification of customer demands at a low cost. This new concept results in the Mixed-Model assembly lines (Duplaga and Bragg 1998). Such lines describe a change in the paradigm of production. The companies used to build cars in large lots, in rigid assembly lines, and are now changing to Mixed-Model assembly lines. Our automotive partner is one of the examples and is changing to Mixed-Model assembly lines because these lines are more flexible. With them it is possible to produce a high variety of models in the same assembly line. The effective utilization of these lines means that two key problems have to be solved:

- *the Line Balancing problem* - to allocate operations to workstations taking into account not only the workload but the logistics activities;
- *the Mixed-Model Sequencing problem* - to sequence the models.

In the literature, a common approach, to solve balance and sequencing problems assumes that the actual arrival sequence is randomly distributed according to the demand proportions of various models. Using this assumption, Bukchin, Dar-El and Rubinovitz (2002) solved the design of mixed model/just-in-sequence assembly lines taking into account the layout and workers flexibility issues in balancing problems. Their objective was to maximize the system efficiency and minimize the cycle time by reducing blockage and starvation between stations and designing the line to a make-to-order environment. In a situation where the model sequence can be arranged without compromising lead time, both line balancing and sequencing should be considered simultaneously to achieve a better performance of the assembly line. The work made by Kim, Kim and Kim (2000) tried to do that. Some other researchers, as Duplaga and Bragg (1998) and Ponnambalam, Aravindan and Naidu (2000), argued that line balancing and sequencing should be solved in different time frames. Becker and Scholl (2006) considered that the balancing decisions have a time horizon of several months and the sequencing problem arises per shift, day or week according to the demand. In this work the balancing is firstly determined based on an average model-mix and the sequencing problem is then solved considering a frozen line balancing. Drexl and Kimms (2001) choose to focus in the sequencing problem. They developed an approach that considers Car Sequencing and Level Scheduling applying an exact technique.

3.2 INTRODUCTION TO SEQUENCING MODELS

One of the goals of Mixed-Model assembly lines is to find an intermixed sequence of different types of products that satisfies the demand of all models and fulfills the company objectives in terms of costs and time (Scholl 1999).

As scheduling problems, sequencing problems are combinatorial and may have more than one objective function that minimizes, for example, the number of violated capacity constraints (Gravel, Gagné, and Price 2005).

Boysen, Fliedner and Scholl (2009) divided the sequencing models in three different groups assuming that the operations are assigned to workstations and that the system is only dependent on the models sequence. Such groups are:

- *Mixed-Model Sequencing* – that aims to avoid/minimize sequence-dependent work overload based on operational characteristics as, for example, operation times and station border;
- *Car Sequencing* – which objective is to minimize sequence-dependent work overload taking into account the sequence and the work overload;
- *Level Scheduling* – which goal is to minimize the differences between actual and ideal rates and keep a constant rate of usage of the parts used in the line (JIT - just-in-time - concepts).

Nowadays, companies need solutions to apply in real situations where the objective is to minimize the work overload and the material requirements. So, Car Sequencing and Level Scheduling appear as important approaches for practical application and in particularly for the automotive industries. To meet this requirement, an extra type of models, called Hybrid Mixed-Models appeared whose goal is to achieve simultaneously the minimization of work overload and the leveling part usage (Boysen, Fliedner, and Scholl 2009).

In the following subsections the sequencing models approaches are going to be presented.

3.2.1 Mixed-Model Sequencing

Mixed-Models aim to minimize sequence-dependent work overload. Sequencing problems appear when the operations in a workstation take more time than the cycle time. In this case the next models in the sequence have to compensate this overload, otherwise, the line will stop or additional workers will be needed because the workers will not be able to finish their sequence tasks without passing the station borders (Scholl 1999, Boysen, Fliedner, and Scholl 2009). Nevertheless, the ideal situation is to find a solution that mixes the models, respecting the clients demand and compensating the differences of production times in workstations. The Mixed-Model Sequencing problem tries to find this flexible solution taking into account the processing times, worker movements, station borders and other operational characteristics (Scholl 1999).

3.2.2 Car Sequencing Models

Car Sequencing Models arise from practical applications required by the automotive industry, nevertheless, these models can be applied to other types of industries. Within the automotive industry, these models are usually applied in automotive assembly lines, but can also be applied in the other two consecutive areas of automotive industries: body and Painting area.

In the assembly line, the goal of Car Sequencing Models is to control the succession of work intensive model options to avoid work overload, using sequencing rules of the type $H_o{:}N_o$ (Boysen, Fliedner, and Scholl 2009). These rules mean that at most H_o out of N_o successively sequenced cars may require option $o \in O$, where O is the set of available options. The use of these rules allows expressing the problem formulation as a constraint satisfaction problem that enforces the sequencing rules observance.

3.2.3 Level Scheduling

Level Scheduling is part of the Toyota Production System (Monden 1993). The main goal of these models is to minimize the safety stocks in order to obtain a just-in-time material supply to the production system (Boysen, Fliedner, and Scholl 2009).

Nowadays, companies need solutions to apply in real situations where the objective is to minimize the work overload and the material requirements. To meet this need, an extra type of models, called Hybrid Mixed-Models, arose.

3.2.4 Hybrid Mixed-Models

The goal of these models is to achieve simultaneously the objectives of the 3 models presented previously. So, the minimization of work overload and leveling part usage are the targets to be achieved by this approach. Consequently, other operational characteristics need to be added (Boysen, Fliedner, and Scholl 2009):

- Setup operations;
- Due dates;
- Assembly line balancing.

These models consider capacity and material aspects which are important for practical application but at the cost of adding more complexity to the existing models. For example, Drexl and Kimms (2001) developed an approach that considers Car Sequencing and Level Scheduling applying an exact technique called Column Generation, used in Branch and Price approaches.

Following we will present the literature review of the Car Sequencing problem since this is the technique that we will apply to solve this problem in this PhD thesis.

3.3 CAR SEQUENCING PROBLEM IN DETAIL

The sequencing problem analyzed throughout this PhD work is the Car Sequencing problem. Despite the little introduction that was made in subsection 3.2.2, a more detailed description of the main problem characteristics will be addressed now.

The Car Sequencing problem is usually defined as a linear programming problem as the one following described (Drexl and Kimms 2001).

$$\min\left[\sum_{o}^{O}\sum_{t=1}^{T}\text{pen}[o,s] * w_o\right]$$

Subject to

$$\sum_{v}^{V}\sum_{j=s}^{s+N_o-1} x_{v,j} * o_{o,v} - \text{pen}[o,s] \leq H_o,\ o \in O\ ,\ s \in \{1\ldots T\text{-}N_o+1\}$$

(...)

This formulation includes a set of options $o \in O$ and a set of variants, $v \in V$, requested by the clients. The options and the variants are the inputs for the model. Each car is placed in a position $t \in T$, being $|T|$ the total number of cars. Each variant v is composed by a set of options o. The variable $x_{v,j}$ becomes 1 if variant v is scheduled in period t, otherwise is 0. This problem intends to minimize the penalties, $pen_{o,s}$, associated with the violation of capacity constraints in one range of car slots. These capacity constraints are usually defined using sequencing rules of the type H_o:N_o, meaning that at most H_o out of N_o successively sequenced cars may require option $o \in O$, where O is the set of available options, as explained before. The use of these rules allows expressing the problem formulation as a constraint satisfaction problem that enforces the sequencing rules observance as shown in the equations above. For each option, if a capacity constraint is violated, a weight, w_o, is associated by increasing the value of the objective function.

The constraints and the objective function are related to operational characteristics. In terms of objective function there are two kinds of approaches (Boysen, Fliedner, and Scholl 2009):

- *Without an objective function* - this is not an optimization problem but a feasibility problem. It is usually seen in the literature as a Constraint Satisfaction Programming problem, where the constraints represent a subset of the problem, declaring allowed or forbidden value combinations and each one providing a local view of the whole problem. The solution of these kind of problems satisfies all the constraints.
- *With an objective function* - the goal is to minimize the violations of sequencing rules and there are several approaches (Boysen, Fliedner, and Scholl 2009) as:
    - minimization of the number of positions where a violation occurs, known as the Sliding Window approach (Gottlieb, Puchta, and Solnon 2003);
    - minimization of the number of all excessive options in violated N_o, etc.

The relevant operational characteristics can be classified in terms of:

- *Number of options* - special characteristics of a car;
- *Hard and soft sequencing rules* - hard rules are related to critical options and cannot be violated, soft rules can be violated. These two rules can be used in the same model or not depending on the problem;
- *Kind of sequencing rules* – there are other rules in addition to conventional H_o:N_o rules as, for example, the restriction of the maximum number of direct successions (Boysen, Fliedner, and Scholl 2009);
- *Assignment restrictions* - are related to the production cycles available for assign model copies. For example, for a group of models there are specified cycles where these models can be assigned to (Boysen, Fliedner, and Scholl 2009).

Gravel, Gagné and Price (2005) and Prandstetter and Raidl (2008) proposed an integer linear programming approach for the Car Sequencing problem that solves benchmark instances of combinatorial problems in acceptable time and proves the solutions optimality. Nevertheless, two major exact solution techniques are used to solve combinatorial problems. These are Branch and Bound and the Constraint Programming. To address the complexity of this problem, Boysen, Fliedner and Scholl (2009) suggested a mix between traditional combinatorial optimization and Constraint Programming , as well as the inclusion of more efficient heuristics and exact solution procedures.

In 2005, Renault proposed a challenge, the ROADEF'2005, to researchers in this field. This challenge is an extension of the classical Car Sequencing problem. The goal was to schedule cars along an assembly line considering two types of capacity constraints, imposed by the assembly line and according to their priority, and considering paint batching constraints (Solnon et al. 2008).

To stimulate the competition and provide benchmarks for researchers working in this field the researchers Ian Gent and Toby Walsh created the CSPLib library. They published it in the Internet (Gent and Walsh 2005). This library offers test problems for constraint solvers to help focusing the research into more structured problems keeping the researchers away from purely random problems.

The efficiency of the approaches used in Car Sequencing problems decreases as the problem size and difficulty increases. Another relevant aspect is that when there is not a feasible solution, long computational times are needed to reach this conclusion (Gravel, Gagné, and Price 2005). To solve these models, exact and heuristics approaches have been used. A generic description of some of these techniques can be seen in the following subsections.

3.3.1 Exact techniques

Exact techniques provide an optimal solution if sufficiently time and memory space is given to the model while running in a computer. This means that the solution found is the better value for the objective function in a given problem. In the Car Sequencing problem of the ROADEF'2005 Challenge this means, for example, that the optimal solution minimizes the number of capacity constraints violated and the number of color changes. If there is more than one optimal solution only one is obtained using this kind of techniques (Prandstetter and Raidl 2008).

These techniques work reasonably fast for small problems. However, in larger problems, usually real problems, these techniques are not useful due to the associated computational times. One approach to improve computational times is to use Exact techniques to prove new bounds and integrate them with heuristic and Meta-Heuristic techniques (Solnon et al. 2008).

For the reasons explained above, the Exact techniques area is complex and has few results for Car Sequencing problem. So, this area is not fully explored and only few authors studied it deeply. The most relevant authors that tried to solve the Car Sequencing problem using Exact techniques are A. Drexl, A. Kimms, M. Fliedner, M. Prandstetter and N. Boysen.

Examples of Exact techniques applied in Car Sequencing problems are:

- Simplex (Prandstetter 2005);
- Branch and Bound (Fliedner and Boysen 2008);
- Branch and Cut (Fliedner and Boysen 2008);
- Branch and Price (Drexl and Kimms 2001, Barnhart et al. 1998);
- Constraint Programming (Solnon et al. 2008).

These techniques will be explained following.

3.3.1.1 Simplex Method

The Simplex Method was thought by Jean Baptiste Joseph Fourier, a French mathematician and physicist known for initiating the investigation of Fourier series. This idea was mechanized algebraically by George Bernard Dantzig an American mathematical scientist who made important contributions to operations research, computer sciences, economics, and statistics (Schrijver 1998). Basically, the idea of the Simplex Method is to perform successive trips on the polyhedron represented by a linear program, from vertex to vertex along edges, until an optimal vertex is reached. The choice of vertex at each step is largely determined by the requirement that this vertex does improve the solution.

Regarding the Car Sequencing problem this technique is used in several softwares to solve mathematical problems. One example is the CPLEX solver from IBM ILOG OPL program used to solve the model from Prandstetter (2005). We will also use the IBM ILOG OPL solver to test our instances with our models.

3.3.1.2 Branch and Bound

The Branch and Bound technique is probably the most used exact solution technique for mixed integer linear problems (Fliedner and Boysen 2008). This algorithm enumerates all candidate solutions to an optimal integer solution, doing successive partitions of the solution space and cutting the search tree by considering limits calculated along the enumeration. These cuts are calculated using upper and lower estimated bounds of the quantity being optimized. The efficiency of the Branch and Bound algorithm depends on the capacity of detect and fathom subtrees that do not lead to an optimal solution. If a formulation has a symmetric tree, for example, it is important to reformulate the problem to avoid enumeration of similar solutions at different nodes of the tree. This fact will allow to reduce or eliminate this symmetry and improve the algorithm performance (Barnhart et al. 1998). When problems are larger or complex, strong cutting planes embedded into the Branch and Bound tree, resulting in Branch and Cut algorithms, typically improve the algorithm efficiency (Bradley, Hax, and Magnanti 1977).

Fliedner and Boysen (2008) tried to solve the Car Sequencing problem developing a special Branch and Bound algorithm which exploits the problem structure in order to reduce combinatorial complexity. Drexl, Kimms and Matthieβen (2006) proposed a dedicated Branch and Bound algorithm to solve the Car Sequencing and Level Scheduling problem from their previous work Drexl and Kimms (2001).

3.3.1.3 Branch and Cut

The Branch and Cut technique is a mix between the Branch and Bound algorithm and the Cutting Plane method. It is also defined as a generalization of the Branch and Bound technique with linear programming relaxations that allows the separation and cut throughout the Branch and Bound tree (Barnhart et al. 1998). In this technique classes of valid inequalities are not used in the linear programming relaxation because of the huge number of constraints to handle efficiently that will not lead to an optimal solution. During the linear programming relaxation if an optimal solution to the linear programming relaxation is infeasible, a separation problem (subproblem) is solved to try to find violated inequalities and add them to the linear programming in order to cut off the infeasible solution. This is repeated until no violated inequalities are found. After this, the tree branching is performed (Barnhart et al. 1998).

The CPLEX from IBM ILOG OPL solver uses a Branch and Cut algorithm which splits the major integer problem in a series of smaller linear programming sub problems.

3.3.1.4 Branch and Price

The Branch and Price algorithm uses a similar strategy to the Branch and Cut algorithm. The difference is that in Branch and Price the procedure focuses on column generation whether in the Branch and Cut the focuses is in row generation. Column generation is used to solve the linear programming relaxation within the Branch and Bound technique (Alvelos 2005). Nevertheless, column generation and row generation are complementary procedures very useful to improve linear programming relaxation. There are some algorithms that use these two approaches (Barnhart et al. 1998). In Branch and Price sets of columns are not used in the linear programming relaxation because of the huge number of columns to handle efficiently that will have their associated variable equal to zero in an optimal solution (Barnhart et al. 1998). To verify the optimality of the linear programming solution a pricing problem (sub problem), that is a separation problem for the dual linear programming, is solved to identify columns to enter the basis. If the columns are found, the linear programming is reoptimized. After this, the tree branching is executed.

Drexl and Kimms (2001), applied Branch and Price algorithms in their work to solve the linear programming relaxation from their Car Sequencing and Leveling model.

3.3.2 HEURISTIC TECHNIQUES

Heuristic techniques are approaches designed to solve a given problem faster than the Exact techniques. Although, heuristic techniques do not guarantee the optimal solution to that problem, it is intended to gain in computational performance and/or conceptual simplicity. So, heuristics are used to solve real difficult problems reasonably well in a reasonable amount of time.

Heuristic techniques as Meta-Heuristics can also be used to provide good initial solutions to use in exact approaches reducing the search space more efficiently (Solnon et al. 2008).

3.3.2.1 Greedy Heuristics

As the name suggests with this procedure the sequence is built in a greedy way. This means that at each stage a locally optimum is chosen which may or may not lead to a globally optimum solution of a given problem. Once a car is moved into a position, it cannot be removed.

Gottlieb, Puchta and Solnon (2003) made an evaluation of some Greedy Heuristics used to solve the Car Sequencing problem, such as:

- *Random Choice* – choose randomly a car from sequence;
- *Static Highest Utilization Rates* – choose the car with highest utilization rate;
- *Dynamic Highest Utilization Rates* – the difference between the static and dynamic highest utilization rates is that in the dynamic the utilization rates are updated each time a car is added;
- *Static Sum of Utilization Rates* – here the sum of utilization rates of the required options is considered;
- *Dynamic Sum of Utilization Rates*- the difference between the static and dynamic sum of utilization rates is that in the dynamic the sum of utilization rates is updated each time a car is added;
- *Dynamic Even Distribution* – in this heuristic the first car is selected randomly among the cars requiring the maximum number of options. The other ones are selected considering a distribution of the options and selecting the cars that on average require an option that is in minority in the sequence under construction.

Local Search, Ant Colony Optimization and many other techniques, use Greedy Heuristics, similar to the ones presented before, to compute initial solutions (Gottlieb, Puchta, and Solnon 2003, Bautista, Pereira, and Adenso-Díaz 2008).

3.3.2.2 Local Search Algorithms

In Local Search Algorithms the search space is explored moving from one solution to another solution in the space of candidate solutions until a given stoppage criterion is reached. Examples of stoppage criteria are:

- an optimal sequence is found;
- a time bound is achieved;
- a maximum number of moves is performed.

The algorithm starts from a candidate solution and then iteratively moves to a neighbor solution. This is only possible if a neighborhood relation is defined on the search space. The choice of which one to move to is taken using only information about the solutions in the neighborhood of the current one, hence the name Local Search (Puchta and Gottlieb 2002).

The approach depends on (Solnon et al. 2008):

- *the way of construction of the initial sequence* - usually random permutation of the vehicles to produce;
- *the neighborhood considered at each move* - different moves can be considered (Solnon et al. 2008, Puchta and Gottlieb 2002):
    - *Insert or Forward/backward Insert* - removes a group of cars from its current position and inserts it after or before the current position;
    - *Swap* - the position of two cars is exchanged;
    - *Transposition or SwapT* - is a special case of Swap. Means that two consecutive cars are exchanged;
    - *SwapS* - is also a special case of Swap. Means that two cars which option requirements are different in one or two options, exchange their positions;
    - *Lin2Opt* - reverses the positions of a subsequence of cars;
    - *Random or Shuffle* - randomly re-arranges a subsequence of cars.

- *the search strategy* - given a neighborhood, different heuristics or Meta-Heuristics can be chosen to decide which is the next move in the following iteration (Solnon et al. 2008).

The different kind of movements considered affect only part of the sequence. For this reason, it is faster to evaluate the change in the objective function due to the change in positions instead of evaluating the whole sequence after each move. Transposition and SwapS, in particular, are evaluated very quickly because Transposition only changes two neighboring positions and SwapS changes one or two options, meaning that only constraints for these options will be evaluated (Puchta and Gottlieb 2002).

Estellon, Gardi and Nouioua (2008) won the ROADEF'2005 Challenge developing a fast Local Search method. The main contribution of this paper, beyond the good results, is the explanation about how to make the exploration efficiently by maintaining special data structures.

3.3.3 Meta-Heuristic techniques

Meta-Heuristics are a result of combining heuristics to optimize a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. These techniques have been most generally applied to problems classified as NP-Hard or NP-Complete by the theory of computational complexity, as Car Sequencing problems. Some examples of Meta-Heuristics applied in Car Sequencing problems are now presented:

- Ant Colony Optimization Algorithms (Gottlieb, Puchta, and Solnon 2003);
- Genetic Algorithms (Warwick and Tsang 1995);
- Simulated Annealing (Briant, Naddef, and Mounie 2008);
- Tabu Search (Cordeau, Laporte, and Pasin 2008, Reis 2007, Warwick and Tsang 1995).

These techniques are going to be explained in the following subsections.

3.3.3.1 Ant Colony Optimization

The Ant Colony Optimization algorithm is based on the behavior of real ant colonies. The idea is to solve the problem as a search for a minimum cost path and use artificial ants to search for good paths (Gottlieb, Puchta, and Solnon 2003). The behavior of these artificial ants is similar to the behavior of real ant colonies because in artificial ant colonies they lay pheromone trails on components of the minimum cost path and they choose the best path taking into account the probabilities that depend on pheromone trails that have been previously laid by the colony (Solnon et al. 2008). Different authors applied this method in the Car Sequencing problem. In the algorithm of Gottlieb, Puchta and Solnon (2003) a greedy heuristic is used to achieve pheromone trails. In Gottlieb, Puchta and Solnon (2003) it is also shown that the Local Search performance of Puchta and Gottlieb (2002) is worse than the Ant Colony Optimization algorithm developed by them, for small computational time limits. For larger limits both approaches have similar results. Gravel, Gagné and Price (2005) presented another Ant Colony Optimization algorithm that integrates a Local Search procedure to improve the solutions constructed by the ants.

3.3.3.2 Genetic Algorithms

The Genetic Algorithms are inspired in the nature biological processes and evolution and have also been applied to the Car Sequencing problem (Joly and Frein 2008). This search technique is stochastic and explores combinatorial search spaces simulating the evolution and recombining candidate solutions, called population, which are associated with fitness values related to a specific domain of the objective function (Warwick and Tsang 1995). The goal of the Genetic Algorithms is to combine solutions to obtain new ones considering the existence of mutations (Joly and Frein 2008). Each candidate solution is called individual, thus the idea is to combine two individuals to obtain a new one, like a crossover between two parents.

In Warwick and Tsang (1995) the search space is explored through selection, cross-over and mutation, the main genetic operators. At each generation, sequences are combined by cross-over operations. The new sequences may not satisfy the constraints, so they are greedily repaired by mutation operators and each offspring hill-climbed by a swap function similar to the one used in Local Search approaches. This approach has been shown to work well in problems characterized by low utilization percentages. Nevertheless, in larger problems the number of successful runs decreases (Warwick and Tsang 1995). The main steps of this technique are (Joly and Frein 2008):

- *Generate an initial solution* - usually random permutation of the vehicles to produce but other techniques can be used;
- *Parent selection* - considering, for example, that the better the objective function, the greater is the probability of being selected;
- *Determine the crossover operator* - some well known methods include:
    - *Order based crossover* - builds an offspring choosing a sub-sequence from one parent and preserving the relative order of vehicles from the other parent;
    - *Partially mapped crossover operator* - builds an offspring choosing some vehicles from one parent and preserving the order and position of as many vehicles as possible from the other parent;
    - *Cycle crossover* - builds an offspring considering that each vehicle and its position comes from one of the parents;
    - *Uniform order crossover* - builds an offspring choosing a sub-sequence from one parent. Vehicles in this subsequence are permuted with vehicles of the same position in the other parent sequence.

- *Determine the mutation operator* - using the following operators often used in literature (Zinflou, Gagné, and Gravel 2008):
    - *Reflection* - consists in randomly select two positions and reverse the subsequence between these two positions;
    - *Random_swap* - consists in randomly exchange the positions of two cars that have different characteristics;
    - *Group_exchange*- consists in randomly exchange the position of two subsequences of consecutive cars with the same characteristic;
    - *Block_reflection* - consists in selecting a subsequence of consecutive cars with the same characteristic and inverting the position of the cars included in this subsequence.
- *Population evaluation* - this evaluation is made using the value of the objective function for each individual;
- *Selection of the surviving population* - this selection is made considering that the better the individuals, the greater is the probability for them to survive in the next generation.

As Zinflou, Gagné and Gravel (2008) mentioned, Genetic Algorithms are efficient approaches if the different mechanisms of the algorithm, as crossover and/or mutation operators, were designed to deal with the specificities of the problem.

3.3.3.3 Simulated Annealing

The Simulated Annealing technique takes inspiration from the annealing process in metallurgy. By analogy with this process, a set of solutions of a certain problem is associated with the energy state. The objective function corresponds to the physical energy of the solid and the ground state corresponds to a global optimal solution (Reis 2007). This neighborhood search heuristic allows an escape from local optima by accepting solutions that may not be better than the last solution found. The probability of preserving a worse solution depends on the temperature parameter that initially is defined by the user as a high value that decreases during the run of the algorithm. Consequently, the probability of accepting worsening moves decreases during the run of the algorithm (Joly and Frein 2008). Thus, this probability depends on the difference between the value of the objective function of the last step solution and the current value of the objective function, and on the temperature parameter (Reis 2007). The result is a good approximation to the global optimum of a given function in a large search space.

This technique needs an initial solution that is usually obtained by random permutation of the vehicles to be produced. Then at each step a solution is created and if it improves the objective function value it is accepted, otherwise it may be accepted depending on the temperature parameter value, as explained before.

Simulated Annealing guarantees a convergence after running a sufficiently large number of iterations. Though this is not very helpful, since the annealing time required to ensure a significant probability of success will usually exceed the time required for a complete search of the decision space (Reis 2007). If an acceptable good solution found in a fixed amount of time is preferable than the best possible solution that takes a long time, Simulated Annealing may be more effective than exhaustive enumeration (Brailsford, Potts, and Smith 1999). Comparing with other heuristics, Simulated Annealing is usually expensive in a computational sense. Nevertheless, the other heuristics tend to generate solutions far from the optimum.

Briant, Naddef and Mounie (2008) developed an algorithm that uses two different methods in two different phases to solve the Car Sequencing problem from ROADEF'2005 Challenge. In the first one they used a Greedy algorithm to minimize the number of color changes. In the second one, a dynamic Simulated Annealing procedure was used to optimize all the criteria considered in the model: capacity constraints and color changes.

3.3.3.4 Tabu Search

The Tabu Search technique uses a Local Search procedure to move from a solution to another in the neighborhood until a defined criteria has been satisfied. This Local Search procedure uses a tabu list to restrict the moves between solutions (Warwick and Tsang 1995). The tabu list is a short-term memory that contains solutions that have been visited recently and forbidding moves. The way that it is managed plays a crucial part in the effectiveness and efficiency of the technique. For instance, if the number of forbidden moves is too high the algorithm may miss good solutions. To avoid this situation the algorithm, usually, has a criteria that override the tabu status of certain movements (Reis 2007). The efficiency can also be improved using methods that exploits the long-term memory of the search process either by recovering the best solutions obtained so far or recovering the attributes of that solutions (Reis 2007).

Cordeau, Laporte and Pasin (2008) developed and iterated Tabu Search algorithm to solve the Car Sequencing problem from ROADEF'2005 Challenge. The algorithm can start from any car permutation and the paper results show that the heuristic was flexible, easy to implement, and fast since it gives results in less than a second on test instances with more than 1000 cars. Although, it has obtained slightly worse solutions than the best algorithm found in ROADEF'2005 Challenge from Estellon, Gardi, and Nouioua (2008).

3.3.4 Remarks

This PhD thesis work focuses on the Car Sequencing problem for the reasons explained in section 1. Some authors as Prandstetter and Raidl (2008), referred above, used Constraint Programming and Integer Programming models to help solving the Car Sequencing problem. These models solved using the Exact techniques, mentioned on subsection 3.3.1, reach their limit when one hundred or so vehicles with few options are considered (Estellon, Gardi, and Nouioua 2008).

The Car Sequencing problem is a NP-hard (non-deterministic polynomial-time hard) problem as Kis (2004) proved and as years before Gent (1998) proved as being NP-complete. For this reason, it was necessary to develop new strategies to solve Car Sequencing problems effectively. Several heuristics have been proposed as Greedy algorithms (Gottlieb, Puchta, and Solnon 2003), Local Search (Estellon, Gardi, and Nouioua 2008, Gottlieb, Puchta, and Solnon 2003), Ant Colony Optimization (Gottlieb, Puchta, and Solnon 2003), Genetic Algorithms (Solnon et al. 2008), Simulated Annealing (Briant, Naddef, and Mounie 2008) and Tabu Search (Cordeau, Laporte, and Pasin 2008).

With the improvements made in mathematical programming software in the last years we decided to develop an exact method that could provide efficient solutions to real problems. This is clearly an area where improvements need to be reached to solve exact models effectively. Therefore, in the following section, we will present an introduction to mathematical models concepts as well as a new Integer Programming model for Car Sequencing.

4 MATHEMATICAL MODELS

Mathematical models are used to describe the reality, to test ideas and to make predictions about the reality. These models are usually used to model industrial processes, traffic patterns, message transmission, linguistic characteristics, atmospheric circulation patterns, stress distribution in engineering structures, the growth and development of landforms, and other processes in science and engineering. Modeling these processes, we are able to do experiments on mathematical representations of the real world without interfere in it (Vries 2001).

A mathematical model usually describes a system or a reality, using a set of variables, that represent some properties of the system, and a set of equations that establish relationships between the variables. But represent the reality in a model is often very complex resulting in a trade-off between simplicity and accuracy. Regarding this matter Howard Wilson Emmons, professor from the Mechanical Engineer Department of the University of Harvard, said that we should:

"... not to produce the most comprehensive descriptive model but to produce the simplest possible model that incorporates the major features of the phenomenon of interest".

Howard Wilson Emmons

One of the challenges of the modeling process is to describe the reality precisely without compromise the simplicity of the model.

Mathematical programming approaches that define mathematical models can be linear or nonlinear according to linearity or not of the constraints and/or the objective function. These approaches can be classified as (Wolsey 1998):

- *Mixed Integer Programming* - if some but not all the variables are integer;

$$\max[\mathrm{cx} + \mathrm{hy}]$$

Subject to

$Ax + Gy \leq b$

$x \geq 0$

$y \geq 0 \; and \; integer$

where A is an m by n matrix, G is an m by p matrix, c an n-dimensional row vector, h is a p row-vector, b an m-dimensional column vector, x an n-dimensional column vector of variables unknown and y is a p column-vector of integer variables.

- *Integer Programming* - if all variables are integer;

$$\max[\mathrm{cx}]$$

Subject to

$Ax \leq b$

$x \geq 0 \; and \; integer$

- *0-1 or Binary Integer Programming* - if all variables have 0-1 values;

$$\max[\mathrm{cx}]$$

Subject to

$Ax \leq b$

$x \in \{0,1\}^n$

- *Combinatorial Optimization* - if the problem is to find a minimum weight feasible subset.

$$\min_{S \subseteq N} \left[\sum_{j \in S} c_j : S \in F \right]$$

where *N* is a finite set *N* = {1, ... ,*n*}, c_j are weights for each *j*∈*N* and *S* is a set of feasible subsets of *N*.

Being the variables of a Car Sequencing problem integers, Integer Programming is going to be used to solve our model. For this reason this approach will be explained in detail in the following subsection.

4.1 Integer Programming

Integer Programming deals with mathematical optimization problems in which all variables are discrete or integer (Wolsey 1998, Schrijver 1998). The application in problems of real life is extensive and includes Car Sequencing, Vehicle Routing, Scheduling problems, Production Planning problems, Telecommunications and Cutting problems, among others.

Car Sequencing problems, for example, are combinatorial because the optimal solution is a subset of a finite set, and for this reason, can be solved by enumeration. The maximum capacity of our automotive partner is 300 cars per shift. Thus our goal is to find the better sequence for each shift among all the possible combinations. To enumerate all the possible solutions for sequence these 300 cars per shift, is necessary to calculate the number of possible solutions. This corresponds to 300! that is approximately, $3.06*10^{614}$. As we can conclude, enumeration can only be helpful for small problems and that is why improvements in the models and algorithms are needed to achieve solutions faster.

As mentioned in subsection 3.3.1, many of the real life problems are hard to solve. For this reason, industries that use Integer Programming models, usually, stop the model when the first solution that satisfies the constraints is found. This can result in losses of mega-dollars and for this reason better models, better algorithms and better software are needed (Wolsey 2003).

In the last years, a lot of improvements were made due to a combination of improved modeling, superior linear programming software, faster computers, new Cutting Plane theory and algorithms, new heuristic methods and Branch and Cut and Integer Programming decomposition algorithms (Wolsey 1998). The use of inequalities to improve formulations and obtain tighter bounds is one of the areas with the most progress in the last years as shown in Wolsey (1998), Wolsey (2003) and Junger et al. (2010).

Nevertheless, it is necessary to distinguish equivalent formulations because they can result in faster or slower times to achieve results. We can see an example of equivalent formulations in the Figure 7.

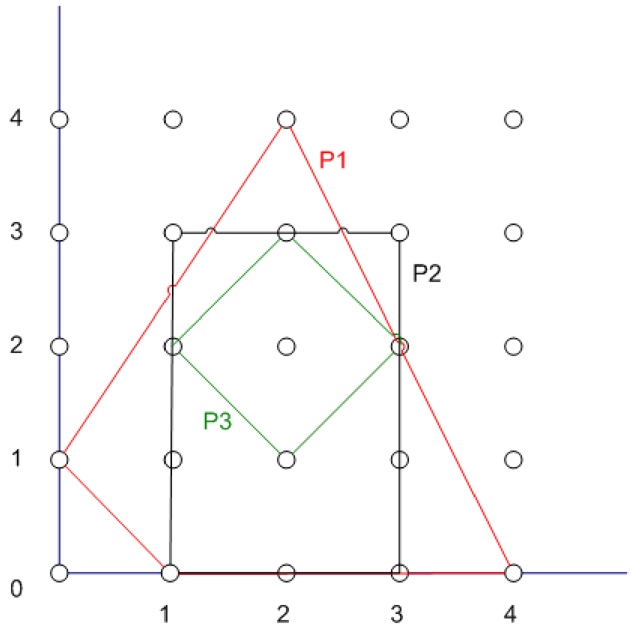


Figure 7: Graphic representation of 3 formulations, P1, P2 and P3, of the same problem.

Geometry can help us find which the best formulation is. Looking at Figure 7 we can distinguish three different formulations, P1, P2 and P3. Formulation P3 is better because if we solve a linear program over P3 the optimal solution is an extreme point (Wolsey 1998).

In most cases the ideal formulation has an enormous (exponential) number of inequalities that need to be described, turning the characterization of *conv(X)* a very difficult task. Therefore, it is important to distinguish which formulation is better.

Recently, Constraint Programming is also being used to model Car Sequencing problems. For this reason, and despite not being applied in our model, it makes sense to do an introduction about this technique in the following subsection.

4.2 Constraint Programming

This is a generic technique that aims to optimize a function, or to find a feasible solution, subject to constraints over discrete and/or continuous variables. In other words, is a technique used to solve constraint satisfaction problems. Through Constraint Programming language, it is possible to solve a constraint satisfaction problem specifying only the variables and constraints that the algorithm is going to solve, using generic algorithms called constraint solvers. These solvers usually employ a systematic exploration of the search space that enumerates assignments of values to variables (Solnon et al. 2008). Lookahead algorithms are effective reducing the size of the search space. After this, a constraint propagation is applied, to restrict the domains of other variables whose values are not fixed yet, until a solution is found or until it is proven that the problem has no solution (Brailsford, Potts, and Smith 1999).

Several works that use Constraint Programming to solve the Car Sequencing problem consider that a solution is valid when all the capacity constraints are satisfied in the final solution. Nevertheless, Bergen, van Beek and Carchrae (2001) proposed a Constraint Programming model for a Car Sequencing problem that includes hard and soft constraints that can be violated at a cost. To solve this model they applied three different approximation algorithms: Local Search algorithm, backtracking algorithm and Branch and Bound algorithm. The Branch and Bound algorithm was the one with better results when tested with six real world instances.

In several works, Constraint Programming shown to be an effective technique to solve easy or small Car Sequencing instances as the others Exact techniques, but it has not yet been useful for harder or larger instances (Solnon et al. 2008). Nevertheless, it has the advantage of its declarative nature, allowing constraints to be expressed more easily.

Were identified in this subsection the basic concepts of Constraint Programming. These concepts are applied in mathematical programming software that help solving the mathematical models. Mathematical programming software has improved in the last years. We observed in the literature that it is practical now to solve problems with a dozen of options and about 300 cars (Prandstetter and Raidl 2008). Therefore, we took a step further, and explored the concept of an assembly sequence that minimizes not only the capacity constraints violations but also the sum of the spread values of cars with the same color. The model developed for the Car Sequencing will now be presented and the special features will be explained in detail.

4.3 NEW INTEGER PROGRAMMING MODEL FOR CAR SEQUENCING

This subsection explains in detail the developed mathematical model as well as the basic concepts of our mathematical formulation.

4.3.1 MATHEMATICAL FORMULATION

As mentioned before, a mathematical model was developed for the Car Sequencing problem. This one used as a starting point the OPL Model example for Car Sequencing included in the IBM ILOG 12.2 CPLEX Optimization Studio, and the model presented by Drexl and Kimms (2001). Our model considers that:

- Cars in production are placed on an assembly line;
- Cars move through various stations that install options on the cars, such as air conditioning and radios;
- The assembly line can thus be viewed as composed by slots and each car must be allocated to a single slot;
- Cars cannot be allocated arbitrarily, since it exists limited capacity and therefore car options must be considered. To each option that limits the production capacity a capacity constraint is associated;
- Cars with special colors to special markets should be placed first in the sequence;
- Cars with the same color should be placed together in the sequence. In this work we are measuring the spread of cars with the same color trying to minimize it.

The objective function considers three special features:

- The number of times that a capacity constraint is violated using the ROADEF'2005 method (Prandstetter 2005). Each capacity constraint has different levels of priority according to the extra time needed to assemble that option;
- The special cars that should come first in the sequence;
- The spread of cars with the same color.

The OPL Model for Car Sequencing is a Constraint Programming model. This simplifies the algorithm but increases the computation time. So, we have decided to change the model to use an algorithm based in the CPLEX algorithm from IBM ILOG 12.2 CPLEX Optimization Studio. This allows us to reduce the computational times from more than one day to four seconds for a test set of 698 cars, considering only the number of times that a capacity constraint is violated, as we will show later on.

It is important to underline that the CP Optimizer engine uses two techniques for solving optimization problems: search strategies and constraint propagation while the CPLEX implements optimizers based on the simplex algorithms (both primal and dual simplex) as well as primal-dual logarithmic barrier algorithms and a sifting algorithm. For problems with integer variables the CPLEX uses essentially the tree search as well as Branch and Cut algorithms.

The set of parameters and variables used in the developed model are shown in Table 3. The variables are used to store the results of the model or to model some special characteristics.

Table 3: Parameters and Variables of our new Integer Programming approach for Car Sequencing.

Parameters	Variables
O - set of options, index *o*	*cost* - objective function value
V - set of variants, index *v*	$pen_{o,t}$ - penalty associated with constraint violations
C - set of colors, index *c*	$x_{v,t}\begin{cases}1, & \text{if variant } v \text{ is scheduled in period } t\\ 0, & \text{otherwise}\end{cases}$
dc - number of colors that should come in the first positions of the sequence (these colors should come in first place in the matrix that represents $c_{c,v}$)	$start_{c,t}\begin{cases}1, & \text{between the first and last position of the car with color}\\ 0, & \text{otherwise}\end{cases}$
dt - number of the first positions of the sequence that should have the cars with the *dc* colors ($\sum_v^V D_v * c_{c,v}, \forall\, c \in dc$)	$end_{c,t}\begin{cases}1, & \text{after the last position of the car with color } c\\ 0, & \text{otherwise}\end{cases}$
T - positions in the sequence = $\sum_v^V D_v$, index *t*	
D_v- demand of variant *v*	
w_o - extra time that option *o* takes into a critical workstation (1+extra time)	
H_o:N_o - at most H_o out of N_o successively sequenced may require option $o \in O$	
$o_{o,v}\begin{cases}1, & \text{if option } o \text{ is part of variant } v\\ 0, & \text{otherwise}\end{cases}$	
$c_{c,v}\begin{cases}1, & \text{if variant } v \text{ has the color } c\\ 0, & \text{otherwise}\end{cases}$	
α - weight associated with the part of the objective function that evaluates capacity constraint violations	
β - weight associated with the part of the objective function that evaluates if special colors are placed in the first *dt* positions of the sequence	

As shown in Table 3 the new Integer Programming formulation includes a set of options $o \in O$, a set of colors $c \in C$ and a set of variants $v \in V$, requested by the clients, that are the inputs of the model. Each variant v is composed by a set of options o and exactly one color c. The 0-1 matrix $o_{o,v}$ includes the options of each variant. If option o is part of the variant v, the value is 1, otherwise is 0. The elements of the 0-1 matrix $c_{c,v}$, define the color of each variant. If color c is part of the variant v, the value is 1, otherwise is 0. Each car is placed in a position $t \in T$, being $|T|$ the total number of cars. Furthermore, there is an input representing the number of colors of the special cars $dc \in C$. The first colors in the matrix $c_{c,v}$ are the colors of the special cars dc. The number of special cars dt is equal to $\sum_{v}^{V} D_v * c_{c,v}, \forall c \in dc$. The total demand is equal to $\sum_{v}^{V} D_v$, being D_v the demand of each variant, and corresponds to the number of positions in the sequence T. Each option has a capacity constraint associated, $H_o{:}N_o$, meaning that at most H_o out of N_o successively sequenced, may require option $o \in O$. Each capacity constraint has different levels of priority according to the extra time needed to assembly that option $w_o \in IR^+$. The w_o will also allow the calculation of the extra workers needed to complete the job when a capacity constraint is violated and where they are needed considering the position of the car in the sequence and the capacity constraint violated.

The problem has a hierarchical multi-objective function, composed by three terms (Table 4 and Figure 8). The first priority is to minimize the capacity constraints violations. The second is to place the special cars first in the sequence. The third priority represents the spread problem. These objectives may be conflicting. For this reason it was decided to have a single objective function with a proper choice of weights for each term. The α weight is attributed to the part of the objective function that evaluates capacity constraint violations. The β weight is attributed to the part of the objective function that evaluates if special colors are placed in the first dt positions of the sequence. A weight equal to one is attributed to the spread problem part of the objective function.

The Integer Programming formulation is defined in Table 4.

Table 4: New Integer Programming approach for Car Sequencing.

	Our model	
Objective function	$\min x + y + z$	(1)
	$x = \alpha \sum_{o \in O} \sum_{t \in T} \left(pen_{o,t} * w_o \right)$	(2)
	$y = \beta \sum_{v \in V} \sum_{c \in d_c} \sum_{t=dt+1}^{T} \left(c_{c,t} * x_{v,t} * (t - dt) \right)$	(3)
	$z = \sum_{c \in C} \sum_{t \in T} \left(start_{c,t} - end_{c,t} \right)$	(4)
Constraints	$\sum_{v \in V} x_{v,t} = 1 \quad , \forall t \in T$	(5)
	$\sum_{t \in T} x_{v,t} = D_v \quad , \forall v \in V$	(6)
	$\sum_{v \in V} \sum_{j=k}^{j+N_o-1} \left(x_{v,j} * o_{o,v} \right) - pen_{o,k} \leq H_o \quad , \forall o \in O, \forall k \in \{1 \dots T - N_o + 1\}$	(7)
	$\sum_{v \in V} \left(c_{c,v} * x_{v,t} \right) \leq start_{c,t} - end_{c,t} \quad , \forall c \in C, \forall t \in T$	(8)
	$start_{c,t} \leq start_{c,t+1} \quad , \forall c \in C, \forall t \in \{1 \dots T - 1\}$	(9)
	$end_{c,t} \leq end_{c,t+1} \quad , \forall c \in C, \forall t \in \{1 \dots T - 1\}$	(10)
	$pen_{o,t} \geq 0 \quad , \forall o \in O, \forall t \in T$	(11)
	$x_{v,t} \in \{0,1\} \quad , \forall v \in V, \forall t \in T$	(12)
	$start_{c,t} \in \{0,1\} \quad , \forall c \in C, \forall t \in T$	(13)
	$end_{c,t} \in \{0,1\} \quad , \forall c \in C, \forall t \in T$	(14)

Comparing the OPL model with our formulation, the OPL model does not consider an objective function being a Constraint Satisfaction Programming problem. In opposition our model is an optimization problem, which goal is to minimize the objective defined in the objective function (1). The OPL model solution satisfies all the constraints, and ours as explained before minimizes the violations of capacity constraints (2), penalizes the special cars that are not placed in the first positions of the sequence (3) and evaluates the cars color spread (4). Equations (3) and (4) intend to put together the cars with the same color. The difference between equations (3) and (4) is that the equation (3) intends to put the cars with special color together and in the first places of the sequence, while equation (4) only intends to put the cars with special color together.

The objective function (1) considers the number of violations of a capacity constraint, using the ROADEF'2005 method (Prandstetter 2005) in equation (2), the displacement of cars for special markets in equation (3), and the sum of the spread of cars with the same color, in equation (4). Considering the constraints, equation (5) guarantees that exactly one car is produced in each period *t*. To ensure that the number of cars produced by variant corresponds to the customer demand, equation (6) was implemented. Constraints (7) and (11) guarantee that each capacity constraint violation is correctly counted, using the ROADEF'2005 method (Prandstetter 2005) and the variable $pen_{o,t}$. Constraints (8-10) guarantee that the function $(start_{c,t} - end_{c,t})$ takes the value 1 in the periods *t* in which cars with color *c* are produced, and 0 otherwise. Constraints (12-14) enforce variables to be binary.

Accordingly, and as referred before, we have a hierarchical multi-objective function that includes three measures as Figure 8 suggests.

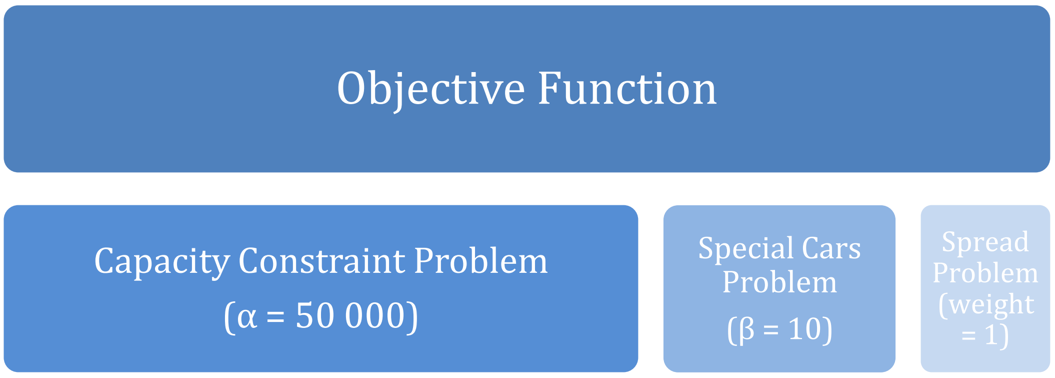

Objective Function
Capacity Constraint Problem (α = 50 000)
Special Cars Problem (β = 10)
Spread Problem (weight = 1)

Figure 8: Hierarchical multi-objective function.

The first priority is to minimize the capacity constraints violations and has associated an α weight equal to 50000. The second priority is to place the special cars first in the sequence and has associated a β weight equal to 10. These weights are higher because, according to our industrial partner, obeying capacity constraints and the displacement of special cars is more important for final assembly goals than the color spread. Thus, the third priority is the spread problem that has a weight equal to 1. As mentioned before, a proper choice of these weights will enforce, for instance, that solutions with an extra capacity constraint violation will never occur if it is possible to have a solution without that extra capacity constraint violation. That is why, testing these parameters in the IBM ILOG software, we achieve the values α equal to 50000 and β equal 10.

In the following subsections we will explain carefully how we count the capacity constraint violations and the concepts of "special cars come first" and "color spread".

4.3.2 Counting the capacity constraint violations

In the literature are mentioned, at least, three methods that count the capacity constraint violation: the Sliding Window mentioned before in the subsection 3.3 from Gravel, Gagné and Price (2005), the ROADEF'2005 approach explained in Prandstetter (2005) and the FB method from Fliedner and Boysen (2008).

In the Sliding Window approach only the subsequences of length N_o, where a violation occurs, are counted. The constraint that represents this approach is the following (Gravel, Gagné, and Price 2005).

$$\sum_{v=1}^{V} \sum_{j=k}^{k+N_o-1} \left(x_{v,j} * o_{o,v}\right) - (N_o - H_o)pen_{o,k} \leq H_o, \forall \mathrm{o} \in \mathrm{O}\ , \forall \mathrm{k} \in \{1 \ldots \mathrm{T}\text{-}\mathrm{N_o}+1\}$$

In Golle, Rothlauf and Boysen (2011) the difference between N_o and H_o is replaced by a biggest integer (B). But the use of the difference between N_o and H_o is enough to guarantee that only the subsequences of length N_o, where a violation occurs, are counted.

The ROADEF'2005 approach introduced in the ROADEF Challenge 2005 counts the number of violations following the equation (7) from Table 4.

The FB method counts all option occurrences leading to a rule violation. The following constraint represents this approach (Fliedner and Boysen 2008).

$$\sum_{v=1}^{V} \sum_{j=k}^{\min\{k+N_o-1,T\}} \left(x_{v,j} * o_{o,v}\right) - \left(1 - \sum_{v=1}^{v} o_{o,v} * x_{v,k}\right) * B - B * pen_{o,k} \leq H_o, \forall \mathrm{o} \in \mathrm{O}\ , \forall \mathrm{k} \in \{1 \ldots \mathrm{T}\}$$

To analyze the differences between the approaches, we will compare the results of the constraint violation evaluation for each method, in Table 5. All the calculations are in the Appendix I. Remember that the objective function is to minimize the sum of penalties for all the methods.

Table 5: Comparison between different methods to calculate capacity constraint violations.

Sequences	ROADEF'2005 method	Sliding Window method	FB method (considering $B = N_o - H_o$)
a**bbb**aaa**bbb**aa (option b - 1:3)	8	6	4
bbbaaaaaa**bbb** (option b - 1:3)	6	4	4
a**bb**a**b**aa**bb**a**b**a (option b - 1:3)	6	6	4
a**bb**aa**bb**aa**bb**a (option b - 1:3)	6	6	3

Analyzing the results summarized on Table 5, the ROADEF'2005 approach is the one that penalizes more violations of capacity constraints. This method counts the number of violations in each subsequence of N_o. The Sliding Window method, as the ROADEF'2005, tends to double count some rule violations and weights them differently depending on their position in the sequence. That is why the first and second sequences have different values despite the sequence of cars b be the same in the two cases and the only difference is the group position of cars b. This should not happen and that is why Fliedner and Boysen (2008) created the FB method. However the FB method, for example, does not distinguish the sequence 1 from the sequence 3. As can be seen, sequence 3 is not so bad as sequence 1, because it allow workers to rest, not obliging them to do 3 $b's$ in a row. The ROADEF'2005 method is the only approach that distinguishes this case.

Based on this performance study and taking into account the goal of the problem under study we have decided to adopt the ROADEF'2005 approach in the model developed. The main reason discussed with our automotive partner is the fact that this approach is sensitive to the number of cars that exist in a row.

4.3.3 SPECIAL CARS COME FIRST

Our industrial partner defined that it is important to assemble first in the sequence cars with special colors, destined to special markets. These cars have to be delivered in the beginning of the day and that is why it is important to produce them first in the sequence. Being the mathematical programming a powerful tool to model special features we decided to introduce a penalty in the objective function of the mathematical programming model, as shown in the Figure 9.

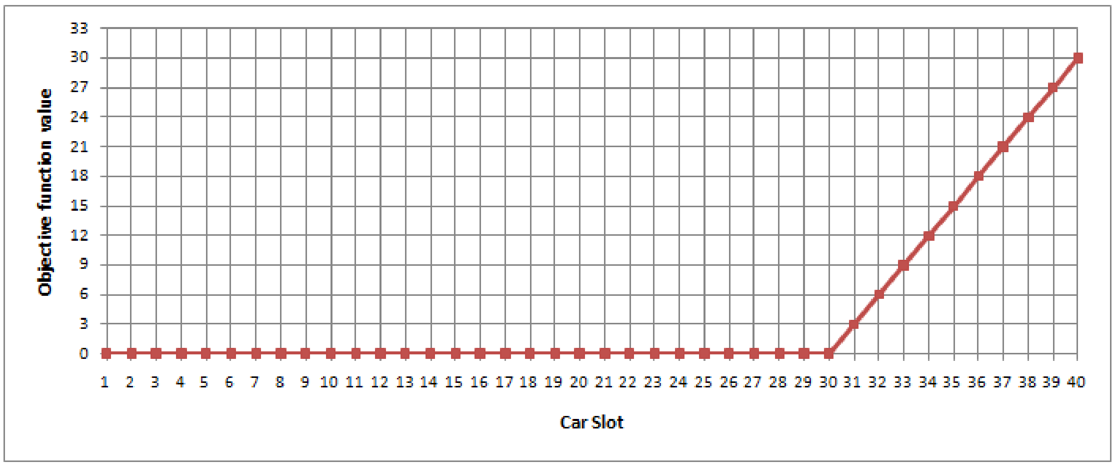


Figure 9: Representation of the objective function for 30 special cars and 40 car slots, considering that special cars should come first in the sequence.

As an example, suppose that 30 special cars are to be produced on a given day. If they do not come first in the sequence, a penalty in the optimization function is triggered as in Figure 9. Special cars until the 30th position, inclusively, will not have penalties. On the other hand, if a special car is sequenced after slot 30, the value of the penalty corresponds to the difference between the slot (car position) and the number of special cars, multiplied by a constant, chosen to be three in this example.

Notice that it may not be possible to place the first 30 cars first in the sequence without violating capacity constraints. Therefore, as these objectives may be conflicting, were given different weights in the objective function as explained before in subsection 4.3.1.

4.3.4 MINIMIZATION OF THE COLOR SPREAD SUM

The model that considers capacity constraint violations and that special cars should come first, works well, but during the model conceptualization we have considered important to include colors spread in the model. This could improve the synchronization between painting and final assembly, allowing to reduce the Work in Progress, to diminish the consumption of solvents in the Painting area because of the decrease in color changes, as well as reducing costs with suppliers of the final assembly. If it would be possible to assemble by car color, our suppliers that deliver painted pieces, could deliver also by color, saving money on the sequence rearranging to deliver at our automotive partner final assembly.

Previous models in the literature, as the model of Prandstetter and Raidl (2008), considered the number of color changes. They assumed that there is a color change in a car sequence if two adjacent cars have different colors. Notice that the number of color changes is the same when two cars are just two slots apart (with another color in between) or when they are three hundred slots apart. For this reason, we introduced the concept of an assembly sequence that minimizes the sum of the spread values of cars with the same color, trying to place the cars of a given color as close as possible in the assembly sequence. Solutions with reduced color spread allow to batch cars with the same color in the Painting area and we claim that this concept improves the performance of the global system.

A solution provided by our model is shown in Figure 10. The first row represents the cars colors and the second one represents the car option considering only the car model. Yellow cars, in the second row, have a capacity constraint of 1:2 and orange cars have a capacity constraint of 2:3. This example has 187 cars of 3 types, 32 variants, 10 options, 11 colors, 74 special cars, α equal to 50000 and β equal to 10.

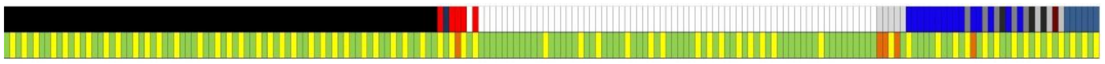


Figure 10: Solution provided by the new Integer Programming formulation.

In the example shown in Figure 10 the special cars were the ones with black color. All the special cars are placed first in the sequence and are all together. For this reason the corresponding color spread is equal to zero. On the other hand, for example, to avoid the violation of capacity constraints, there are two cars placed among the red cars. Therefore, the red color spread is equal to seven (two plus five red cars). Globally this solution, non-optimal, shows a total color spread of 210 (23 plus 187 cars) and was obtained in, approximately, one hour and a half.

The strategy is then to use the batch sequence of the final assembly to determine the order in which cars are painted. In this example the batch sequence would be black, red, white, etc. After the cars being painted, they would enter the assembly line using the sequence determined before by the model. Before entering to final assembly, changes in positions are needed, violating the batch painting sequence. Nevertheless, these violations are allowed, because the company wants to keep always a number of cars in the buffer between the painting and the final assembly to avoid stock-outs. The changes in positions are also necessary because, as mentioned before, the priority in the final assembly is to avoid the violation of capacity constraints to prevent line stoppages.

This strategy may also provide better logistics with the suppliers, for instance, in the case of the bumpers, which have an attribute color and are supplied in a sequence that follows the final assembly sequence plan. Delivering bumpers when colors are clustered may enable the supplier to batch production and to reduce the operations needed to organize the sequence, reducing their costs and even reducing sequence supply errors.

This new model that counts the spread of each color has an increased complexity, and presents larger computational times to obtain a solution. For this reason, improvements in the model have been studied, as it will be described in the following section. These model improvements were achieved trying to consolidate the model including the search of valid inequalities and bounds that, when found, improve computational times. A heuristic approach was also studied to try to achieve better results.

5 Model Improvements

Model improvements can be achieved looking for a model consolidation by improving the bounds with valid inequalities and adding strong inequalities. Therefore, lower and upper bounds and additional inequalities were studied to strengthen the model and to improve computational times.

Another way of improving the model is to develop a heuristic to integrate with it. But first of all we will describe our achievements when trying to consolidate the model.

5.1 Model Consolidation

We started this approach using three different cutting planes based on the capacity constraint part of the objective function represented by the equation (2), on the special cars term of the objective function represented by the equation (3), and on the color spread part of the objective function represented by the equation (4). Our first approach was to solve each problem independently and introduce the results as new constraints as shown in the following constraints.

$$\alpha \sum_{o \in O} \sum_{t \in T} \left(pen_{o,t} * w_o \right) \leq optimal\ solution\ capacity \text{ constraint problem} \tag{15}$$

$$\beta \sum_{v \in V} \sum_{c \in d_c} \sum_{t=d_t+1}^{T} \left(c_{c,t} * x_{v,t} * (t - dt) \right) \leq optimal\ solution \text{ special cars problem} \tag{16}$$

For each color we have introduced a new constraint with the optimal solution of the capacity constraint and the color spread model. So, considering the spread inferior limit, we will have as many constraints as the number of colors.

$$\sum_{t \in T} (start_{c,t} - end_{c,t}) \geq optimal\ solution\ color\ spread\ problem, \forall c \in C \tag{17}$$

Following we will present the improvements of each cut in the objective function value for four different instances with the characteristics described in the Table 6.

Table 6: Instances to evaluate cuts improvements (also used in section 6).

Instance number	Number Cars	Mix (A:B:C)	Number Options	Number Variants	Number Colors	Number Special Cars	Average utilization rate	Options with utilization rate > 1
1	300	37:121:142	14	115	16	79	0,24	-
17	300	61:114:125	14	97	16	53	0,24	-
23	300	50:100:150	15	104	18	57	0.2	-
24	300	58:77:165	14	116	19	62	0.23	O1 (1.10)

The instances characteristics include a number of cars equal to 300 in the four instances. This value corresponds to the maximum capacity by shift in our automotive partner. The mix value corresponds to the demand of each model (A:B:C) and the sum is always equal to 300. The number of options, variants, colors and special cars vary according to the distributions of the demand. The demand characteristics are explained in detail in subsection 6.2. There are 17 possible options with associated capacity constraints as expressed in the following table.

Table 7: Instances characteristics - options and capacity constraints.

Option	Capacity constraint
Option 1	1:2
Option 2	1:2
Option 3	1:3
Option 4	1:3
Option 5	1:4
Option6	1:4
Option 7	1:26
Option 8	1:6
Option 9	1:6
Option 10	1:3
Option 11	1:4
Option 12	1:16
Option 13	1:3
Option 14	1:6
Option 15	1:2
Option 16	1:2
Option 17	1:2

The results of applying the cuts to the model are in Table 8. The data that originates it, is on Appendix II. The Table 8 includes, for each instance, the best solutions after 5, 20 and 30 minutes. Three different use cases were tested separately for each of the four instances:

- First case – constraint (17);
- Second case - constraints (15) and (16); and
- Third case – constraints (15), (16) and (17).

The percentage of improvement was calculated according to the next function.

$$\frac{Result\ model\ without\ cuts - result\ model\ with\ cut}{result\ model\ without\ cuts}$$

Table 8: Results of strengthening the model.

		% Improvement Constraint (17)	% Improvement Constraints (15)+(16)	% Improvement Constraints (15)+(16)+(17)
I 1	Best solution after 5 minutes	99.27%	0.00%	100.00%
	Best solution after 20 minutes	4.55%	0.00%	-68.18%
	Best solution after 30 minutes	9.09%	0.00%	0.00%
I 17	Best solution after 5 minutes	24.53%	0.00%	100.00%
	Best solution after 20 minutes	100.00%	100.00%	100.00%
	Best solution after 30 minutes	100.00%	100.00%	100.00%
I 23	Best solution after 5 minutes	99.99%	0.00%	99.99%
	Best solution after 20 minutes	99.99%	0.00%	99.99%
	Best solution after 30 minutes	0.27%	0.00%	0.27%
I 24	Best solution after 5 minutes	0.01%	0.00%	0.00%
	Best solution after 20 minutes	0.03%	0.00%	0.00%
	Best solution after 30 minutes	0.03%	0.00%	0.00%
	Average improvements	44.81%	16.67%	44.34%

The average improvements are considered calculating the average of the results in each test for each instance. We have concluded that the constraint represented by the constraint (17) have improved the results in about 45% on average. Constraints (15) and (16) have improved the model in about 17% and constraints (15), (16) and (17) have improved the model in 44%. Despite the improvements of constraint (17) and constraints (15), (16) and (17) be similar, constraint (17) is slightly better since it gives better solutions in all instances while constraints (15), (16) and (17) do not provide solutions within 30 minutes for instance 24.

The results for the capacity constraint and special cars problems, that lead to the inclusion of constraints (15) and (16), are easily found, solving the capacity constraint and the special cars problems independently. These results are obtained in a few seconds in the IBM ILOG 12.2.

The results of the spread problem for each color that lead to the inclusion of constraint (17) are however difficult to achieve. Easily these take more than 5 minutes to reach an optimal solution. Based on this fact and in order to overcome this drawback we have developed the following strategy.

```
If "Result of the capacity constraint model" = 0 then
    Solve the problem using the procedure below
else "Solve the problem using the model in IBM ILOG 12.2"
    If "the time to solve the problem is > 5 minutes" then
        "Use the inferior limit achieved"
        If "Inferior limit is integer" then
            "Use the integer inferior limit achieved"
        Else "Use the next integer value"
        End if
    Else "Use the optimal solution"
    End if
End if
```

As described in the procedure above if the result of the capacity constraint is equal to zero it is better to find the result of the spread problem for each color using the procedure that will be described following. This new procedure achieves the correct value in less than one minute for each color if we follow the tips described next. One is to calculate the minimum number of slots that a considered number of cars will occupy, without violating capacity constraints, following the next function.

$$\mathrm{N_o}(\mathrm{n}-1)+1$$

This function has to be used carefully since it is necessary to take into account the capacity constraints of each car. Other cars that obey to other capacity constraints can be allocated to the slots between the slots occupied by the cars sequenced before. To calculate the number of free slots between the first and last car that is being analyzed we have to use the following function.

$$(\mathrm{N_o}-1)(\mathrm{n}-1)$$

As an example consider 12 cars (a) with the capacity constraint 1:2, 4 cars (b) with capacity constraint 1:3 and 3 cars (c) without capacity constraints. The procedure is the following:

1. Choose the option with a higher number of cars.
   In this case are the a's
2. Calculate the number of slots needed to sequence these cars
   $N_o(n-1)+1$ = 2(12-1) + 1 = 23
3. Calculate the number of free slots between the first and last car
   $(N_o-1)(n-1)$ = (2-1)(12-1) = 11
   Illustrating this sequence we have:
   a__a__a__a__a__a__a__a__a__a__a__a (11 free slots + 12 *a* cars)
4. Choose the next option with the higher number of cars
   In this case are the b's
5. Compare the number of cars with the number of free slots calculated in 3.
   In this case 4 < 11

6. Calculate the number of slots needed to sequence these cars

   $N_o(n-1)+1 = 3(4\text{-}1) + 1 = 10$

6.1 If the number of cars and slots calculated in 5 and 6 is inferior to the number of free slots we will not need more slots to sequence these cars.

6.1.1 "Based on the illustration" include the cars in the free slots.

aba__aba__aba__aba__a__a__a__a

6.1.2 Repeat the process from 4-7 until you do not have cars to sequence

6.2 If the number of cars and slots calculated in 5 and 6 is higher you will need more slots to sequence these cars after the last car sequenced.

6.2.1 Imagine that you have 7 *b's*. In this case the result of the point 5 would be 7<11 and the result of the point 6 would be 3(7-1) + 1 = 19>11. "Based on the illustration" include the cars in the free slots and create the new slots to include the cars that do not fit in the empty slots.

aba__aba__aba__aba__aba__aba__b

6.2.2 Repeat the process from 4-6 until you do not have cars to sequence.

Notice that, if the result of the capacity constraint is different than zero it is advisable not to use this procedure, because the cars where the capacity constraints are violated can vary depending on the color that we are optimizing. For this reason, in these cases, we have done it using the IBM ILOG 12.2 framework to do all this calculations, as described in the procedure presented before.

Despite the good results when using constraint (17) in the model, we think that the processing times can be improved without big penalizations for our solution, bringing better times and acceptable solutions for our industrial partner. For this reason, a heuristic, described in following subsection, was created to be integrated with the exact "capacity constraint, special cars and spread model" explained in subsection 4.3.

5.2 New Heuristic Approach for Car Sequencing

Preliminary computational results have shown that the model that considers the capacity constraints and the special cars issue is solved with CPLEX 12.2, provided by IBM-ILOG in a few seconds for 300 cars on average. These preliminary results can be consulted in Appendix III. However when the color spread is considered the model becomes much harder to solve as can be seen in the results described on the column “Without heuristic” from Appendix IV. For this reason a heuristic approach based on the integer model presented in subsection 4.3 was pursued. Our strategy enabled us to keep the original framework, which is a powerful modeling tool, allowing at the same time to take into consideration new features.

The basic idea behind the heuristic is an approximate measure of the spread value. Instead of measuring the spread by considering the total number of slots occupied by the cars of a given color, the heuristic measures the spread by considering a total number of intervals, with 5, 10 or 25 slots for example, occupied by the cars of a given color. The heuristic simplifies the problem by dividing the whole sequence into intervals, thus reducing the number of variables and consequently the problem complexity.

To reduce the number of variables of the final matrix of colors and slots, the concept of number of intervals was created. Each interval represents a group of cars from a group of slots according with the following equation.

$$\text{number of intervals} = \frac{\text{number slots}}{\text{interval width}}$$

In each interval the cars color of each slot are clustered, as the following example explains (Figure 11).

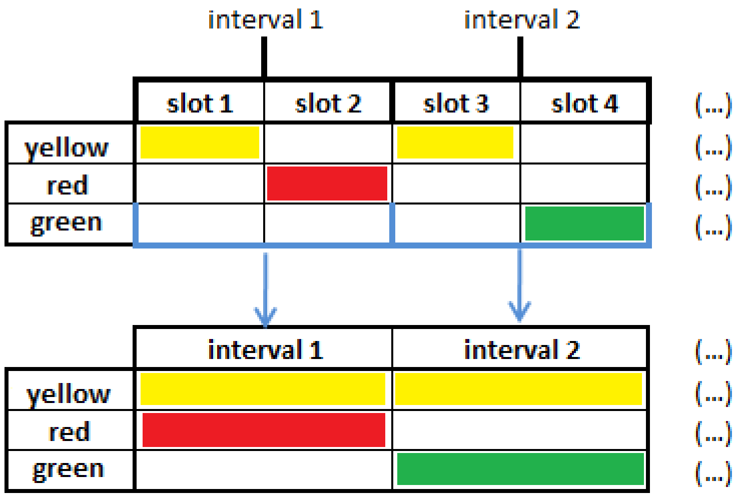


Figure 11: Clustering of cars color of each slot, considering 3 colors and a number of intervals equal to 2 in a range of 4 cars.

The correct number of intervals is evaluated considering the computational results. The number of intervals should be as closest as possible to the number of slots, to achieve results as close as possible to the optimal solution of the real problem. It is estimated that this approximation should follow the next constraint, that will be explored in subsection 5.2.2.

$$\text{spread real problem} \leq \text{spread heurisitic} * \text{interval width}$$

This means that the spread of the real problem is at most the width of the interval multiplied by the value of the heuristic spread.

In summary, the heuristic simplifies the problem by dividing the whole sequence in small intervals, thus reducing the number of variables and consequently the problem complexity. On the other hand, instead of an optimal global solution the output will be an optimal local solution.

5.2.1 HEURISTIC INTEGER PROGRAMMING FORMULATION

The new formulation for the problem includes a similar objective function represented by equations (1-3) from subsection 4.3.1, and the constraints (5-7) and (11-12) of the integer linear program also described in subsection 4.3.1. However, the variables *start* and *end* will have a different definition due to the concept of number of intervals. So the parameters *nint* and *nlarge* were created: *nint* represents the number of intervals; *nlarge* represents the width of the interval and is obtained taking into consideration the next equation.

$$nlarge = \begin{cases} \dfrac{T}{nint} + 1 \,, t\%nint! = 0 \\ \dfrac{T}{nint} \,, otherwise \end{cases}$$

A new decision variable was created to represent the colors of each interval and the variables $start_{c,t}$ and $end_{c,t}$ were modified as following:

- $start_{c,s}$ - 0-1 decision variable. Becomes 1 when the first interval of cars, *s*, with $s \in \{1 \ldots nint\}$, with the color *c* is scheduled and remains 1 until the last interval;
- $end_{c,s}$ - 0-1 decision variable. Becomes 1 in the next interval of cars, *s*, with $s \in \{1 \ldots nint\}$, after the last interval of cars, *s*, with the color *c* is scheduled and remains 1 until the last interval of cars *s*;
- $bint_{c,s}$ - 0-1 decision variable. Becomes 1 if color *c* is scheduled in the period *s*, with $s \in \{1 \ldots nint\}$.

Thus, our new Integer Programming formulation that includes the heuristic is defined in Table 9.

Table 9: New Heuristic Integer Programming approach for Car Sequencing.

	Our model	
Objective function	$\min x + y + z$	(1)
	$x = \alpha \sum_{o \in O} \sum_{t \in T} (pen_{o,t} * w_o)$	(2)
	$y = \beta \sum_{v \in V} \sum_{c \in d_c} \sum_{t=d_t+1}^{T} \left(c_{c,t} * x_{v,t} * (t - d_t)\right)$	(3)
	$z = \sum_{c \in C} \sum_{t \in nint} (start_{c,t} - end_{c,t})$	(4a)
Constraints	$\sum_{v \in V} x_{v,t} = 1 \quad, \forall t \in T$	(5)
	$\sum_{t \in T} x_{v,t} = D_v \quad, \forall v \in V$	(6)
	$\sum_{v \in V} \sum_{j=k}^{j+N_o-1} (x_{v,j} * o_{o,v}) - pen_{o,k} \leq H_o \quad, \forall o \in O, \forall k \in \{1 \dots T - N_o + 1\}$	(7)
	$\sum_{v \in V} (c_{c,v} * x_{v,t}) \leq bint_{c,((T-1)\backslash nlarge)+1} \quad, \forall c \in C, \forall t \in T$	(15)
	$bint_{c,s} \leq start_{c,s} - end_{c,s} \quad, \forall c \in C, \forall s \in \{1 \dots nint\}$	(8a)
	$start_{c,s} \leq start_{c,s+1} \quad, \forall c \in C, \forall s \in \{1 \dots nint - 1\}$	(9a)
	$end_{c,s} \leq end_{c,s+1} \quad, \forall c \in C, \forall s \in \{1 \dots nint - 1\}$	(10a)
	$pen_{o,t} \geq 0 \quad, \forall o \in O, \forall t \in T$	(11)
	$x_{v,t} \in \{0,1\} \quad, \forall v \in V, \forall t \in T$	(12)
	$start_{c,s} \in \{0,1\} \quad, \forall c \in C, \forall s \in \{1 \dots nint\}$	(13a)
	$end_{c,s} \in \{0,1\} \quad, \forall c \in C, \forall s \in \{1 \dots nint\}$	(14a)
	$bint_{c,s} \in \{0,1\} \quad, \forall c \in C, \forall s \in \{1 \dots nint\}$	(18)

In the following subsection we will present in detail how the approximation to estimate the spread of the heuristic was achieved.

5.2.2 SPREAD ESTIMATION

The original problem minimizes the value of the color spread, i.e., the sum of the lengths of all the color intervals in terms of positions. On the other hand, the heuristic is a modified problem that minimizes the value of the interval color spread, i.e., the sum of the lengths of all the color blocks in terms of intervals. It can be solved for any choice of *nint* intervals each with *nlarge* positions. In the following analysis, we consider that *nint * nlarge = T,* being *nint, nlarge* and *T* integer values. The heuristic is still an Integer Programming problem, but it was denoted as such, because it provides an approximate solution for the original problem.

We will show that it is possible to derive an approximation guarantee for the value of the color spread of the solution obtained by the heuristic. Of course, when we solve the modified problem, the optimal solution found may be the optimal solution for the original problem. However, when this does not happen, the color spread of the heuristic solution (measured in terms of the color spread in the original problem), does never exceed by more than a pre-defined amount the value of color spread of the optimal solution of the original problem, as shown below.

The result has a general scope, and is valid for any problem where a strategy of minimizing spread is applied. When addressing this sequencing problem, this analysis is valid when we just consider, in the objective function, the value of the spread, and ignore the other terms.

Let x^i represents the solution that minimizes the color spread in the original problem, and $s(x^i)$ the corresponding optimal value of the color spread. When we solve the heuristic, for a given choice of *nint,* and minimize the interval color spread, attaining the optimal integer solution for the modified problem, we get a solution that will be denoted as x_{nint}^h. Let $b_{nint}(x_{nint}^h)$ be the (optimal) number of color intervals in the modified problem. This solution will have a corresponding value of color spread (measured in the original problem) equal to $s(x_{nint}^h)$.

Let $b_{nint}(x^i)$ be the number of intervals occupied by the optimal solution of the original problem, x^i, that has fewer intervals occupied. In fact we may have alternative optimal solutions in the original problem in terms of the value of spread, but one (or some) occupying fewer intervals. Note that the number of color intervals of the optimal solution of the modified problem, $b_{nint}(x_{nint}^h)$, can never have a number of blocks greater than $b_{nint}(x^i)$, because the solution x^i is a valid solution at hand when we solve the modified problem. Therefore $b_{nint}(x_{nint}^h) <= b_{nint}(x^i)$.

Let us analyze first the case when $b_{nint}(x_{nint}^h) = b_{nint}(x^i) = b$. The relation $s(x_{nint}^h) >= s(x^i)$ holds, because we may have alternative optimal solutions to the modified problem with different values of spread in terms of the original problem. We want to show that $s(x_{nint}^h) <= K^{max}\ s(x^i)$, where K^{max} is the worst case approximation guarantee. Clearly, the optimal solution of the modified problem has a color spread that obeys the following inequality:

$$s(x_{nint}^h) <= b * nlarge, \tag{19}$$

otherwise it would occupy more than $b=b_{nint}(x^i)$ blocks, and it would not be the optimal solution of the modified problem.

Clearly we would like to have an a priori value for K^{max} that does not depend on the value of b, which is unknown a priori. We can derive a rough value with a simple analysis. Consider the worst case situation in which all the b blocks have just one position occupied in the solution of the original problem. Then, $s(x_{nint}^h) <= nlarge\ s(x^i)$.

However, when the number of intervals is larger than the number of colors, $nint >= C$, which is a more common situation, we can derive a tighter result. In this case, due to the structure of the solutions, both in the original problem and in the heuristic, in which the color interval remains active from the first position until the last position, we can have, at most, $2C$ intervals in the solution of the original problem with just one position filled, at both ends of each color block, while the intervals in the middle are part of the color interval, and contribute with *nlarge* positions. Therefore, for an optimum solution of the original problem that occupies b intervals, the value of the spread:

$$s(x^i) >= 2C + (b - 2C)*nlarge. \tag{20}$$

Furthermore, it cannot be smaller than T:

$$s(x^i) >= T, \tag{21}$$

which combines into:

$$s(x^i\ ) \geq \max\{T, b * nlarge + 2C(1 - nlarge)\} \tag{22}$$

Combining this relation with (17), we obtain:

$$s(x^h_{nint}) \leq \frac{b*nlarge}{\max\{T,\ b*nlarge+2C(1-nlarge)\}}\ s(x^i\ ) \tag{23}$$

The approximation guarantee *K(b)*, in $s(x_{nint}{}^h) <= K(b)\ s(x^i)$, is a piecewise function that depends on the value of *b*. There is a breakpoint, *bp*, when *b*nlarge+2C(1-nlarge) = T*, meaning:

$$bp = \frac{T+\ 2C(nlarge-1)}{nlarge} \tag{24}$$

Notice that *nint <= b <= C * nint*. This range can be divided in two parts: in the first part, $b \in [nint, bp]$, where (21) holds, and, in the second part, $b \in [bp, C * nint]$, where (20) holds. It is easy to check that, for the approximation guarantee *K(b)*, the value of *K(b)* increases as *b* increases in the first part, while, in the second part, the value of *K(b)* decreases as *b* increases. Therefore, the maximum value of $K(b)= K^{max}$, which corresponds to the worst case approximation guarantee, occurs for one of the integers neighboring the breakpoint *bp*, and so it can be calculated beforehand, making it possible to choose a suitable value of *nlarge* in order not to incur an approximation larger than a pre-defined value.

Another case happens when $b_{nint}(x_{nint}^h) < b_{nint}(x^i)$, meaning that there is a solution $x^{i'}$ to the original problem that is not optimal (having a spread s($x^{i'}$) larger than the optimum, s(x^i), but occupying fewer blocks. Clearly the solution $x^{i'}$ is available when we solve the heuristic. The ratio to calculate, *K(b),* has, in the numerator, the value of the largest spread (in terms of positions) of the heuristic solution and, in the denominator, the value of the spread of the original solution. If there is a solution to the modified problem that occupies fewer blocks, the heuristic will find it, and the worst value in the numerator will be smaller. On the other hand, the value in the denominator will be larger. Therefore, the value of *K(b)* indicated in (23) is larger, and will provide a valid approximation guarantee *Q.E.D.*.

Considering the example shown in the following Figure 12, with *T* equal to 30, *nlarge* equal to 5, *nint* equal to 6 and *C* equal to 3. Assuming that the optimal solution is the one given in the second diagram with a spread $s(x^i)$ equal to 30 and occupying *b* equal to 6 intervals. The heuristic will also find a solution occupying 8 intervals (the intervals shown in Figure 12), but the heuristic solution might be the one in the third diagram, with a spread (in terms of positions) equal to 38. This heuristic solution has a spread that exceeds the optimal spread by 26.6%.

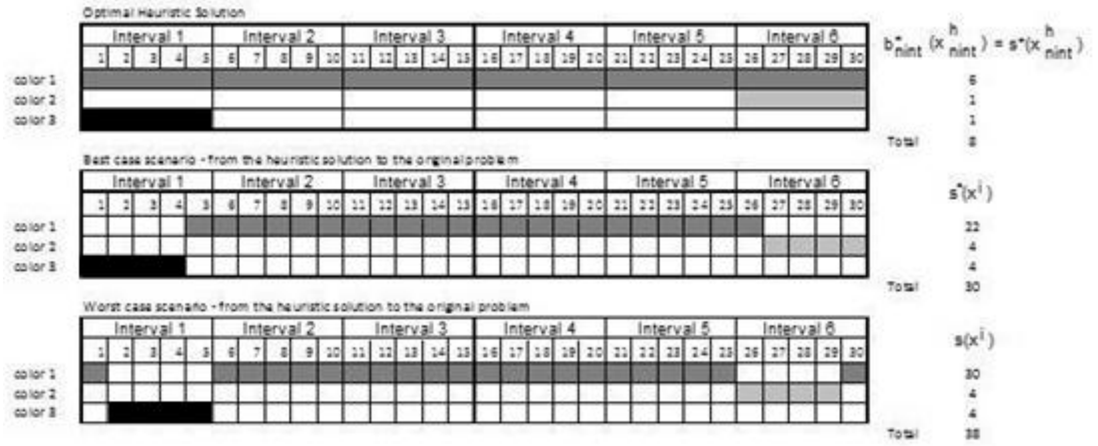

Optimal Heuristic Solution
Interval 1
Interval 2
Interval 3
Interval 4
Interval 5
Interval 6
color 1
color 2
color 3
$b^*_{nint}(x^h_{nint}) = s^*(x^h_{nint})$
6
1
1
Total 8
Best case scenario - from the heuristic solution to the original problem
$s^*(x^i)$
22
4
4
Total 30
Worst case scenario - from the heuristic solution to the original problem
$s(x^i)$
30
4
4
Total 38

Figure 12: Example with the spread of the heuristic and best and worst scenario for the spread of the real problem.

For this example, the values of *K(b)* are in Table 10.

Table 10: Values of *K(b)* for the example represented by Figure 12.

b	6	7	8	9	10	11	12	13	14	15	16	17	18
K(b)	1.00	1.17	1.33	1.50	1.67	1.77	1.67	1.59	1.52	1.47	1.43	1.39	1.36

For an optimal solution with b equal to 8, the value of *K(8)* is equal to 1.33. In the example, the heuristic solution shown is 26.6% within the optimal. Furthermore, the choice of *nlarge* equal to 5 guarantees an approximation calculated as follows. The breakpoint occurs for 10.8. In (21), *K(10)* is equal to 1.67 and *K(11)* equal to 1.77. Therefore, the value of the spread (in terms of positions) of the solution obtained from the heuristic will never be worse than the optimal solution by more than 77%.

Consider another example used in the computational tests, with T equal to 300, *nlarge* equal to 5, *nint* equal to 60 and C equal to 11. The breakpoint occurs for 77.6. Calculations done as above lead to $s\left(x_{nint}^{h}\right) \leq 1.291\, s(x_i)$. This means that the value of the spread (in terms of positions) of the solution obtained from the heuristic will never be worse than the optimal solution by more than 29.1% .

6 COMPUTATIONAL RESULTS

In order to be sure that, as a result of this research, it is possible to develop a robust tool that can be used daily for Car Sequencing at automotive assembly companies, we have decided to test the models with data that simulates our industrial partner's environment. This data was created using a Random Generator developed for this project. This Random Generator allows the creation of "random instances" based on our industrial partner's demand of 42 weeks from 2010 and 2011 and considering different scenarios according to our goals.

6.1 THE RANDOM GENERATOR

As mentioned before, to test the model a Random Generator was created using Microsoft Visual Basic 6.5. This Random Generator creates scenarios to simulate our industrial partner's environment and gives the following data that will be used as parameters into the models described in subsection 4.3 and in section 5.

- Demand of each model (Models A, B and C);
- Demand of each variant;
- Matrix with options of each variant;
- Matrix with color of each variant.

The demand for each model (Models A, B and C) is created independently, considering all the possible combinations of options and taking into account the demand percentage of the cars with different options per day. These percentages vary according to the scenarios in consideration.

Firstly, to calculate the demand of each model, we took into account the percentage of demand of each model to calculate the probability of being a Model A, a Model B and a Model C. As demands are independent and the model cannot be, for example, a Model A and a Model B at the same time, the probabilities correspond to the demand percentage of each model, for the considered scenario.

Secondly and finally, 300 random values between 0 and 1 are generated, since the maximum available capacity of our automotive partner is 300 per shift. Based on the random value obtained, a decision will be made:

- if the number is in the interval [0 , Model B probability], the car is a Model B;
- if the number is in the interval [Model B probability, Model B probability + Model C probability], the car is a Model C;
- otherwise, it is a Model A.

The demand of each variant is calculated using the same strategy. However, it is necessary to calculate the probabilities of occurrence of each variant taking into consideration the demand of each option. These probabilities are calculated considering the examples described in Table 11.

Table 11: Calculation of variant probabilities - example.

	Variant 1	Variant 2	Variant 3
Option 1 (O1)	1	0	1
Option 2 (O2)	0	0	1
Option 3 (O3)	0	0	1
Option 4 (O4)	1	0	1
Probability of occurrence	% demand O1 * (1-% demand O2) * (1-% demand O3) * % demand O4	(1-% demand) * (1-% demand O2) * (1-% demand O3) * (1-% demand O4)	% demand O1 * % demandO2 * % demand O3 * % demand O4

Finally, to calculate the demand of each variant, *x* random values according to all possible variants of each model are created. Based on the random value a decision will be made:

- if the number is in the interval [0 , Probability Var1], the car is a Variant 1;
- if the number is in the interval [Probability Var1 , Probability Var1+ Probability Var2], the car is a Variant 2;
- if the number is in the interval [Probability Var1+ Probability Var2 , Probability Var1 + Probability Var2 + Probability Var3], the car is a Variant 3;
- and so on, until the second last variant;
- otherwise, it belongs to the last variant.

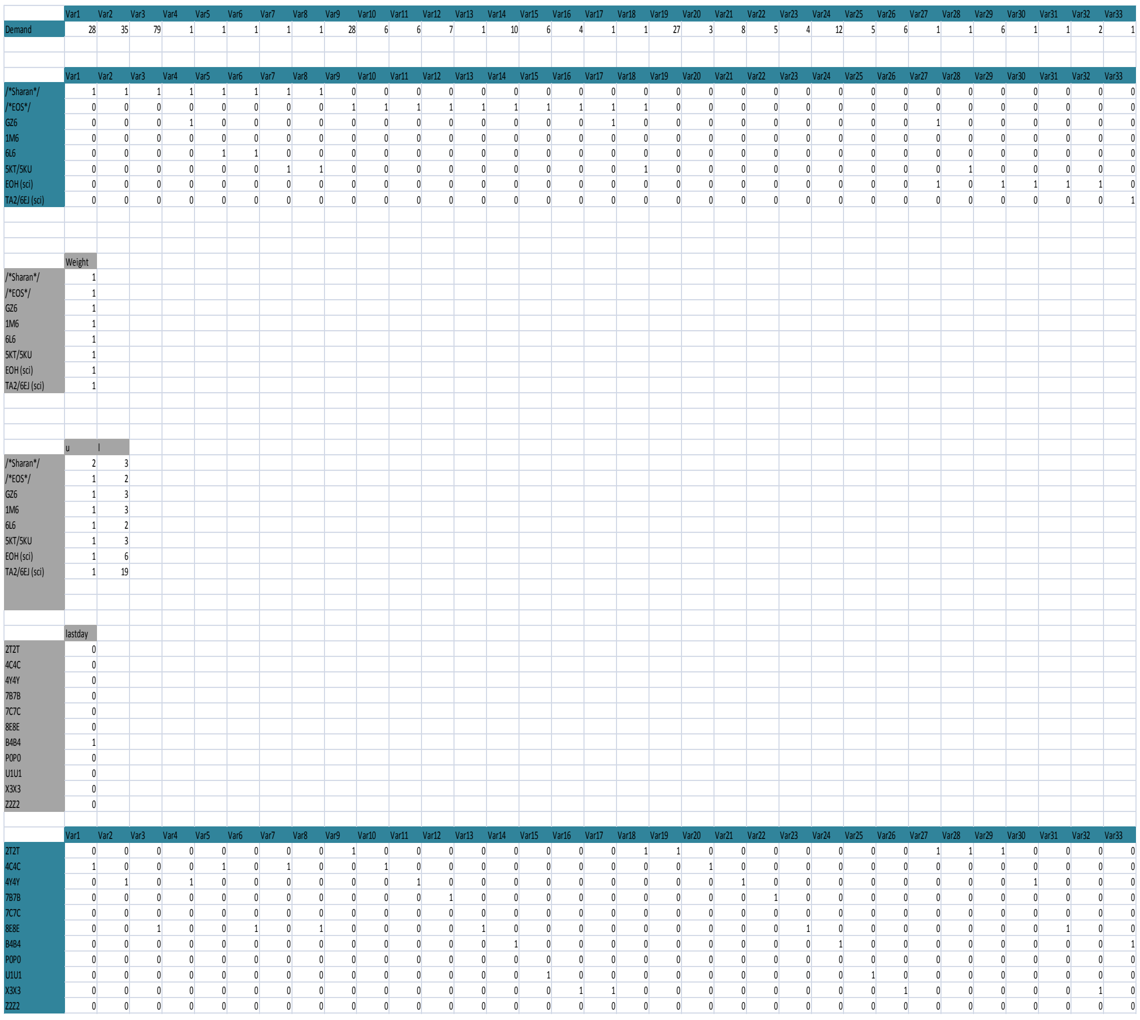
The demand of each color is calculated using the same strategy. However, it is necessary to calculate the probabilities of occurrence of each color taking into consideration the demand of each color. As in the case of the models probabilities, the demand of each color is independent and a variant has only one color. Thus, the probabilities correspond to the demand percentage of each color, for the considered scenario.

To calculate de demand of each color, *x* random values according to all possible variants of each model are created. Based on the random value a decision will be made:

- if the number is in the interval [0 , Probability Color1], the car has the color 1;
- if the number is in the interval [Probability Color1 , Probability Color1 + Probability Color2], the car has the color 2;
- if the number is in the interval [Probability Color1 + Probability Color2, Probability Color1 + Probability Color2 + Probability Color3], the car has the color 3;
- and so on until the second last variant;
- otherwise, it belongs to the last color.

Based on the calculated demands of car models, variants and colors, the matrixes with the options of each variant and with the color of each variant are created.

The final result, represented in Figure 13, will be used as an input to the models presented in subsection 4.3 and in section 5.



	Var1	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9	Var10	Var11	Var12	Var13	Var14	Var15	Var16	Var17	Var18	Var19	Var20	Var21	Var22	Var23	Var24	Var25	Var26	Var27	Var28	Var29	Var30	Var31	Var32	Var33
Demand	28	35	79	1	1	1	1	1	28	6	6	7	1	10	6	4	1	1	27	3	8	5	4	12	5	6	1	1	6	1	1	2	1

	Var1	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9	Var10	Var11	Var12	Var13	Var14	Var15	Var16	Var17	Var18	Var19	Var20	Var21	Var22	Var23	Var24	Var25	Var26	Var27	Var28	Var29	Var30	Var31	Var32	Var33
/*Sharan*/	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
/*EOS*/	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GZ6	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1M6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6L6	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5KT/5KU	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
EOH (sci)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0
TA2/6EJ (sci)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	Weight
/*Sharan*/	1
/*EOS*/	1
GZ6	1
1M6	1
6L6	1
5KT/5KU	1
EOH (sci)	1
TA2/6EJ (sci)	1

	u	l
/*Sharan*/	2	3
/*EOS*/	1	2
GZ6	1	3
1M6	1	3
6L6	1	2
5KT/5KU	1	3
EOH (sci)	1	6
TA2/6EJ (sci)	1	19

	lastday
2T2T	0
4C4C	0
4Y4Y	0
7B7B	0
7C7C	0
8E8E	0
B4B4	1
P0P0	0
U1U1	0
X3X3	0
Z2Z2	0

	Var1	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9	Var10	Var11	Var12	Var13	Var14	Var15	Var16	Var17	Var18	Var19	Var20	Var21	Var22	Var23	Var24	Var25	Var26	Var27	Var28	Var29	Var30	Var31	Var32	Var33
2T2T	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0
4C4C	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4Y4Y	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
7B7B	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
7C7C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8E8E	0	0	1	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
B4B4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
P0P0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
U1U1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
X3X3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
Z2Z2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13: Results of the Random Generator - Layout.

6.2 THE INSTANCES

Three models of cars were considered, models A, B and C. Three different possible scenarios were taken into consideration by assuming peaks for each model. The instances generated and tested are in the Table 12.

Table 12: Instances characteristics.

Instance number	Number Cars	Mix (A:B:C)	Number Options	Number Variants	Number Colors	Number Special Cars	Average utilization rate	Options with utilization rate > 1
1	300	37:121:142	14	115	16	79	0,24	
2	300	74:51:175	14	103	19	52	0,24	O1 (1,17)
3	300	48:99:153	14	113	17	69	0,23	O1 (1,02)
4	300	54:83:163	15	121	19	65	0,23	O1 (1,09)
5	300	52:105:143	14	102	18	53	0,2	
6	300	48:103:149	15	109	18	80	0,21	
7	300	56:109:135	15	107	18	59	0,23	
8	300	82:77:141	14	107	21	62	0,26	
9	300	56:112:132	14	108	20	57	0,22	
10	300	72:37:191	14	109	17	49	0,26	O1 (1,27)
11	300	42:99:159	14	110	17	76	0,23	O1 (1,06)
12	300	66:41:193	14	113	18	50	0,25	O1 (1,29)
13	300	61:95:144	14	95	18	50	0,21	
14	300	40:101:159	14	106	15	77	0,24	O1 (1,06)
15	300	61:102:137	14	103	19	59	0,25	
16	300	83:77:140	15	102	21	54	0,21	
17	300	61:114:125	14	97	16	53	0,24	
18	300	57:124:119	15	98	19	43	0,18	
19	300	61:88:151	15	107	18	51	0.21	O1 (1,01)
20	300	52:97:151	15	116	19	68	0.25	O1 (1,01)
21	300	39:116:146	14	111	16	82	0.23	
22	300	41:146:113	14	110	18	86	0.25	
23	300	50:100:150	15	104	18	57	0.2	O1
24	300	58:77:165	14	116	19	62	0.23	O1 (1,10)
25	300	47:106:147	14	115	20	64	0.24	
26	300	60:96:144	16	109	19	70	0.24	
27	300	76:82:142	15	110	21	68	0.23	

As explained before in subsection 4.1, the instances characteristics include a number of cars equal to 300 in all instances. This value corresponds to the maximum capacity by shift in our automotive partner. The mix value corresponds to the demand of each model (A:B:C) and the sum is always equal to 300. The number of options, variants, colors and special cars vary according to the distributions of the demand. There are 17 possible options with associated capacity constraints as expressed in the Table 7. The average utilization rate is calculated taking into consideration the following function.

$$\frac{demand\ option}{total\ demand * \ H_o/N_o}$$

6.3 Results and Analysis of Computational Tests

The instances created by the random generator, as mentioned before, were tested using the IBM ILOG 12.2 framework in a Intel® Core™2 Duo CPU T9600 Toshiba laptop @ 2.80GHz with 6 GB of RAM.

Firstly, we solved the model just with the capacity constraints as part of the objective function (equation 2 – subsection 4.3.1). Then we have addressed the model variant that considers capacity constraint and special cars, taking into account the equations (2) and (3) of subsection 4.3.1 as part of the objective function. Finally, we have solved the global problem that includes the capacity constraint, special cars and color spread model considering the equations (2), (3) and (4a) of the subsection 5.2.1 as part of the objective function. For the global problem we have searched the best solutions obtained after 5, 20 and 30 minutes, acceptable times for achieving a solution for planning the sequence just before each shift. Still, the model can be ran for longer times aiming at better solution values.

To analyze the results we have decided to use box plots since they appear as an easy way to see data organized considering statistical values. Also, they are useful for seeing a big picture of the data without being overly detailed (Siegel 2012).

This statistical tool is a standardized way of displaying the distribution of data based on five sample statistics:

- *Minimum* – the smallest data value, excluding outliers and extremes (0th percentile);
- *First quartile* – the 25th percentile, ¼ of the way in from the minimum;
- *Median* - the 50th percentile, in the middle;
- *Third quartile*- the 75th percentile, ¾ of the way in from the minimum;
- *Maximum* - the highest data value, excluding outliers and extremes (100th percentile).

Values that are far from the middle data set are considered outliers. As we used IBM SPSS version 20 and this software considers two types of outliers we will designated them as outliers and extremes. Outliers are values which are between one and a half and three box lengths from either end of the box and considering a normal distribution of the data. Extreme values are more than three box lengths from either end of the box.

In the following subsections we will present the results and analyzes of the capacity constraint model, capacity constraint + special cars model and global model.

6.3.1 *Results and Analyzes of The Capacity Constraint Model and of the Capacity Constraint + Special Cars Model*

In the Appendix III the results for the capacity constraint model and of the capacity constraint + special cars model can be found. These results are presented in a table similar to the following example (Table 13).

Table 13: Structure of the table from Appendix III.

			Instances
Capacity constraint problem	Optimal solution	**Time (s)**	
		Capacity constraint violations	
		Spread	
		Node	
		Solutions found	
Capacity constraint + special cars problem	Optimal Solution	**Time (s)**	
		Capacity constraint violations	
		Position last special cars	
		Spread	
		Node	
		Solutions found	

Each row of the Table 13 means the following:

- *Time (s)* – time, in seconds, to solve the problem optimally;
- *Capacity constraint violations* - number of capacity constraints violated. This number is minimum when the model finds the optimal solution;
- *Position last special cars* – without brackets is the position of the last special car. Between brackets is the number of special cars. When an optimal solution is found, we can see if it is possible to have all the special cars together;
- *Spread* – sum of the distances between the first and last car of each color. This number is not minimum because we are not taking this value into consideration in the objective function;
- *Node* – node where the optimal solution is found. When an optimal solution is not found in 30 minutes represents the node where the search stopped;
- *Solutions* found – number of solutions found by the model until the optimal solution is found or within 30 minutes.

The results presented in detail in Appendix III show that 2 instances out of 27 did not obtain optimal solutions for the capacity constraint model and for the capacity constraint + special cars model. This represents 7% of the results. We can also conclude that usually, when an optimal solution is not found for the capacity constraint model, an optimal solution is also not found for the capacity constraint + special cars model.

Considering the instances that achieved an optimal solution in 30 minutes we have analyzed the results using box plots. These box plots are represented in the Figure 14.

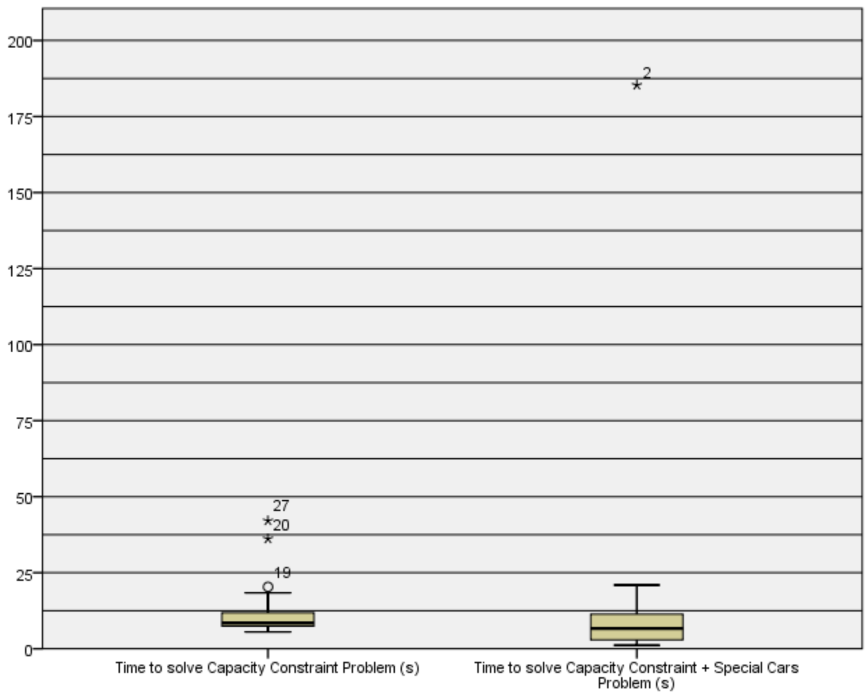


Figure 14: Box plots with the time to solve the Capacity Constraint Problem and the Capacity Constraint + Special Cars Problem.

From the analysis of the Figure 14 we can conclude that the median of the time to solve the Capacity Constraint and the Special Cars Problem is lower than the time to solve the Capacity Constraint Problem. So we can assume that the time values to achieve optimal solutions tend to be smaller in the case of the Capacity Constraint and of the Special Cars Problem. However, the size of the whiskers tells us that the dispersion of the data is higher in that case. The highest value of the sample, without considering outliers, is highest also in the case of the Capacity Constraint and the Special Cars Problem. The solutions of these two problems have outliers and extremes. In this case there is only one outlier which is instance 19 for the Capacity Constraint Problem and there are 3 extremes: instances 20 and 27 for the Capacity Constraint Problem; and instance 2 for the Capacity Constraint and Special Cars Problem.

Following, in Figure 15, we will present the spread distributions for the Capacity Constraint Problem and for the Capacity Constraint and Special Cars Problem.

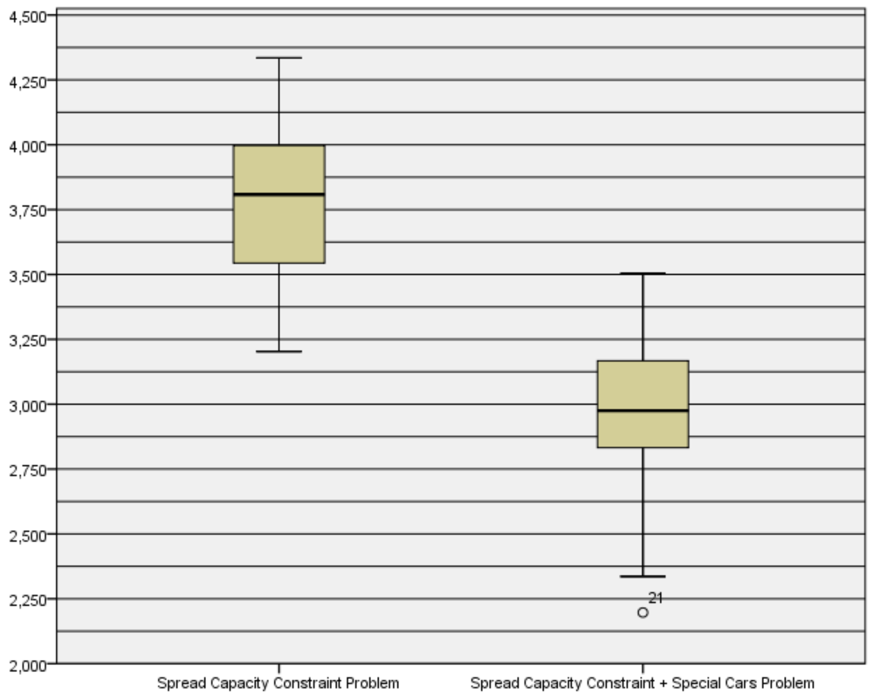


Figure 15: Box plots with the spread of the Capacity Constraint Problem and the Capacity Constraint + Special Cars Problem.

Analyzing Figure 15 we can conclude that the median spread of the Capacity Constraint and the Special Cars Problem is lower than the spread of the Capacity Constraint Problem. So we can assume that the spread values of optimal solutions tend to be smaller in the case of the Capacity Constraint and the Special Cars Problem. However, the size of the whiskers indicates that the dispersion of the data is similar in these cases. The highest value of the sample, without considering outliers or extremes, is highest also in the case of the Capacity Constraint Problem. The solutions of the Capacity Constraint and the Special Cars Model have one outlier for instance 21 meaning that the spread value is between one and a half and three box lengths from the beginning of the box, considering a normal distribution of the data.

Considering all the data from Appendix III, we built a summary table presented in Table 14.

Table 14: Summary table with the results of the capacity constraint model and of the capacity constraint + special cars model.

Instance number	Final solution with capacity constraints violations	Final solution without all the special cars in the first positions	Running times of the capacity constraint model (s)	Running times of the capacity constraint + special cars model (s)
1			9.25	20.93
2	x	x	9.56	185.33
3	x	x	18.35	2.98
4	x		11.81	15.43
5			6.58	8.42
6		x	8.31	3.68
7		x	8.10	2.42
8	Optimal solution not found in 1800 seconds			
9		x	6.72	1.76
10	x		8.30	11.39
11	x		9.52	10.87
12	x		7.24	12.65
13		x	6.96	2.01
14	x	x	8.35	2.92
15		x	7.38	6.66
16	Optimal solution not found in 1800 seconds			
17			5.54	1.28
18	x	x	5.68	1.14
19	x	x	20.34	9.31
20	x	x	36.16	2.96
21			8.55	2.37
22		x	8.46	8.47
23		x	14.27	7.05
24	x		10.70	11.39
25		x	8.33	3.10
26		x	12.61	4.43
27		x	42.09	16.74

Analyzing the results shown in Table 14 it seems that the existence of capacity constraints violations in the optimal solution of the capacity constraint problem does not make the problem more complex, since the running times to achieve optimal solutions are similar to the running times of instances without capacity constraint violations. Another possible conclusion is that the addition of special cars to the Capacity Constraint Problem does not add complexity to the problem because the median of the time values to achieve optimal solutions is inferior in this case.

Analyzing the nodes where the optimal solutions were found, for the Capacity Constraint Problem, only in instance 27 the solution was not found in the node 0. This represents 4% of the instances. This instance is the one that took more time to achieve a solution. This fact suggests that, maybe, the time to solve these kind of instances is higher. For the Capacity Constraint and the Special Cars Problem only the solution of instance 2 was not found in node 0. Again it seems that maybe the time to solve the instances that do not find the optimal in node 0, is higher than the usual, because this instance represents an extreme (see Figure 14).

6.3.2 Results and Analyses of Global Model

We will present now the analyses of the results for the global problem. These results are presented in Appendix IV in a table similar to the following example (Table 15).

Table 15: Structure of the table from Appendix IV.

			Instance 1			
			Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals
Capacity constraint + special cars + spread problem	Best solution after 5 minutes	Spread				
		GAP				
		Node				
		Solutions found				
	Best solution after 20 minutes	Spread				
		GAP				
		Node				
		Solutions found				
	Best solution after 30 minutes	Spread				
		GAP				
		Node				
		Solutions found				
	Optimal Solution	Time (s)				

These results have the following information when optimal solutions are not found within 5, 20 and 30 minutes:

- *Spread after 5, 20 and 30 minutes* – sum of the distances between the first and last car of each color. This number is minimum when the model founds the optimal solution for the "Without heuristic" Problem. When the optimal solution is found for the problem that considers the heuristic, a minimum value of spread, taking into account the number of intervals as explained in subsection 5.2.1, is found;
- *GAP after 5, 20 and 30 minutes* – is calculated using the following equation:

$$GAP = \frac{Current\ value\ of\ the\ objective\ function\ - inferior\ limite\ of\ the\ objective\ function}{Current\ value\ of\ the\ objective\ function}$$

- *Node after 5, 20 and 30 minutes* – represents the node where the search was after 5, 20 and 30 minutes;
- *Solutions found* – number of solutions found by the model after 5, 20 and 30 minutes.
    - When a solution is not found within 5 minutes, in the line "Best solution after 5 minutes" will appear "Not found";
    - When a solution is not found within 20 minutes, in the lines "Best solution after 5 minutes" and "Best solution after 20 minutes" will appear "Not found";
    - When a solution is not found within 30 minutes, in the lines "Best solution after 5 minutes", "Best solution after 20 minutes" and "Best solution after 30 minutes" will appear "Not found".

When an optimal solution is achieved within 30 minutes appears the following information in the line “Optimal Solution”:

- *Time (s)* – time, in seconds, to solve the problem optimally;
- *Capacity constraint violations* - number of capacity constraints violated. This number is minimum when the model finds the optimal solution;
- *Position last special cars* – without brackets is the position of the last special car. Between brackets is the number of special cars. When an optimal solution is found, we can see if it is possible to have all the special cars together;
- *Spread* – sum of the distances between the first and last car (considering the first and last car) of each color. This number is minimum because we are taking this value into consideration in the objective function;
- *Node* – node where the optimal solution is found;
- *Solutions found* – number of solutions found by the model until the optimal solution is found.

    - When an optimal solution is achieved before 5 minutes of running time, in the lines “Best solution after 5 minutes”, “Best solution after 20 minutes” and “Best solution after 30 minutes” will appear “Optimal solution found before”;
    - When an optimal solution is achieved between 5 and 20 minutes of running time, in the lines “Best solution after 20 minutes” and “Best solution after 30 minutes” will appear “Optimal solution found before”;
    - When an optimal solution is achieved between 20 and 30 minutes of running time, in the line “Best solution after 30 minutes” will appear “Optimal solution found before”.

Following we will present the box plots with the results for the spread of the global problem in the first 5, 20 and 30 minutes of running time. The meaning of the x axis is the following:

- The first letter represents which kind of results we are analyzing: S – Spread; G – GAP; Nd – Node; Nr – Number of solutions (see Figure 16);

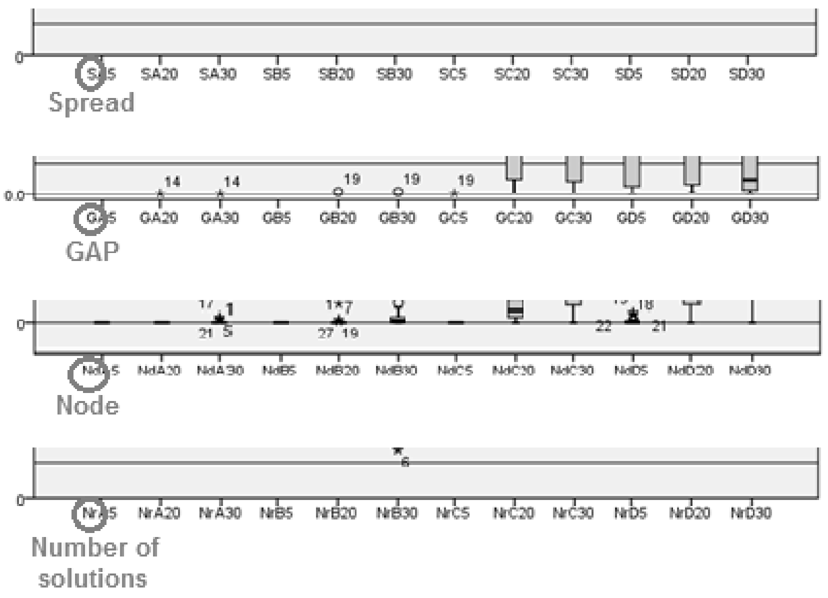


Figure 16: Example with the meaning of the first letter from the x axis in the box plot graphics.

- The second letter represents which problem we are analyzing: A-without heuristic; B – 100 intervals Heuristic; C – 10 intervals Heuristic; D – 5 intervals Heuristic (see Figure 17);

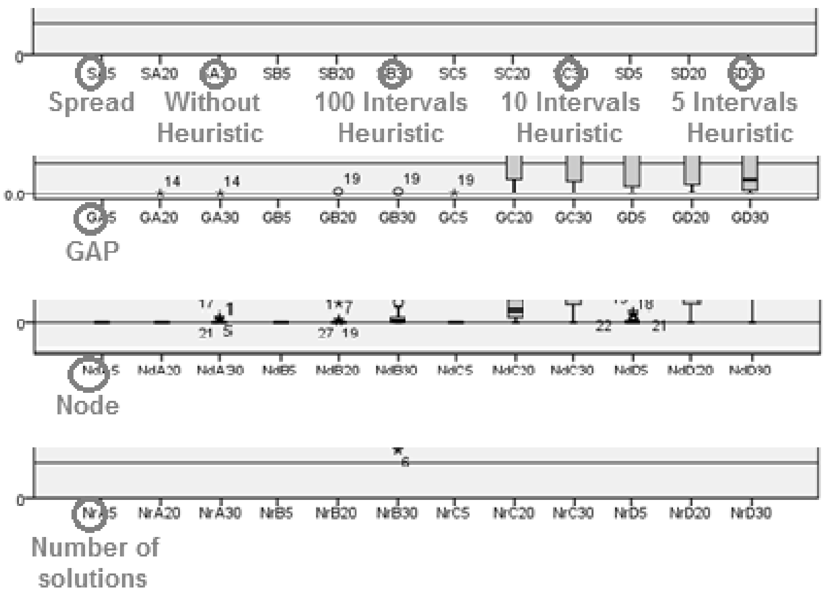


Figure 17: Example with the meaning of the second letter from the x axis in the box plot graphics.

- The third letter represents the running time to achieve those results (see Figure 18).

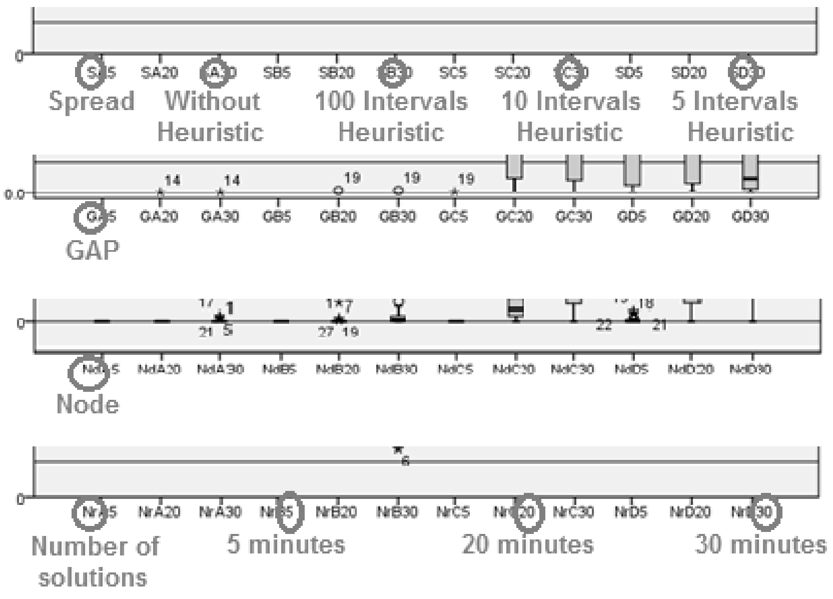


Figure 18: Example with the meaning of the numbers from the x axis in the box plot graphics.

In the next picture, Figure 19, we will present the box plots representing the spread results.

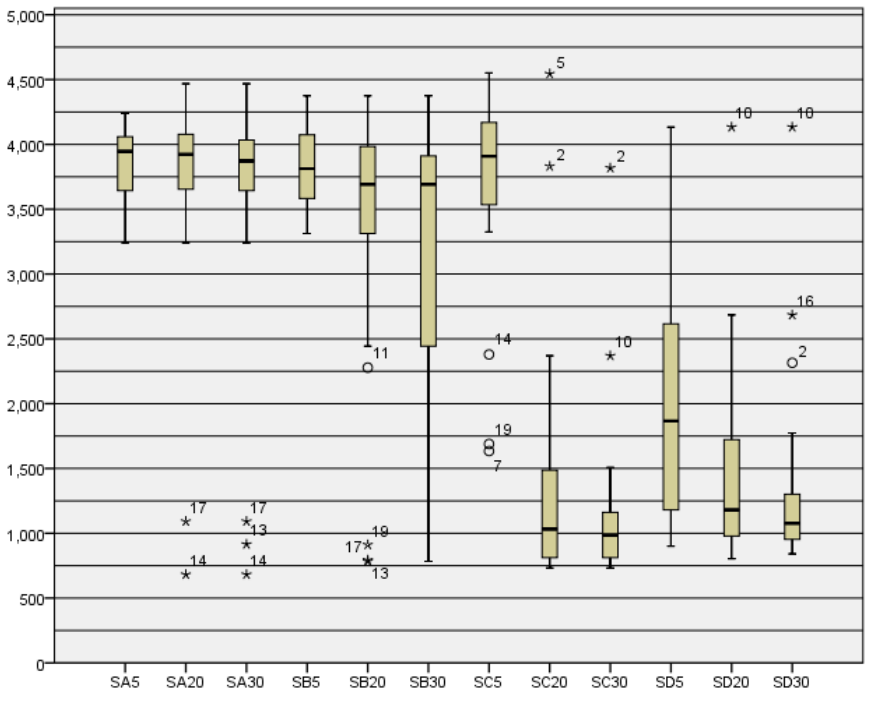


Figure 19: Box plots with the spread of the global problem in the first 5, 20 and 30 minutes of running time.

Analyzing the data from Figure 19 we can conclude that the median of the 10 and 5 Intervals Heuristics in the first 20 and 30 minutes of running time is lower than in the other cases. So we can assume that the spread values tend to be smaller in these cases. The size of the whiskers tells us that the dispersion of the data is similar in these cases because the size of the whiskers is similar when comparing 10 and 5 Intervals Heuristics in the first 20 minutes and 10 and 5 Intervals Heuristics in the first 30 minutes. Comparing the 10 and 5 Intervals Heuristics in the first 20 and 30 minutes of running time the highest value of the sample, without considering outliers or extremes, is for the case of the 5 Intervals Heuristic. For this reason, we can conclude that for the spread, the 10 Intervals Heuristic is slightly better.

Analyzing the spread values, there are outliers and extremes in the tests SA5, SB5, SB30 and SD5. The outliers and extremes with smaller values of spread, for example, in the case of instances 13, 14 and 17 for the SA30 problem, are not a problem because this means that the model in these cases achieved smaller values of spread than the usual. The outliers and extremes with higher values of spread, as in the case of instances 2 and 5 for the SC20 problem, mean that sometimes the model achieved higher values of spread than the usual within the expected time.

Following we will present the box plots with the GAP for the global problem in the first 5, 20 and 30 minutes of running time (Figure 20).

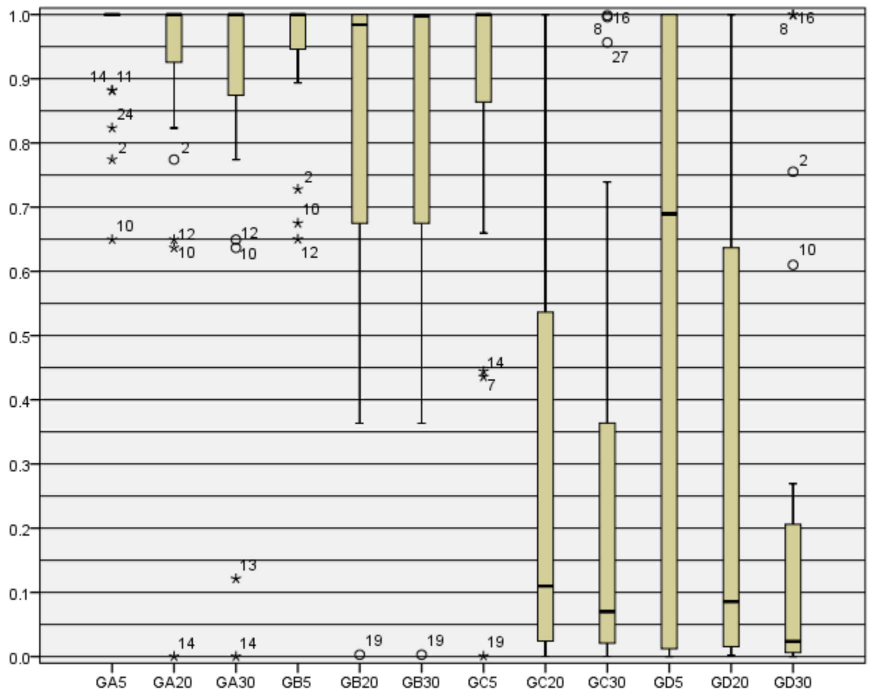


Figure 20: Box plots with the GAP of the global problem in the first 5, 20 and 30 minutes of running time.

Analyzing the data from Figure 20 we can conclude that the median of the 10 and 5 Intervals Heuristics in the first 20 and 30 minutes of running time is lower than in the other cases. So we can assume that the GAP values tend to be smaller in these cases. The size of the whiskers tells us that the dispersion of the data in the case GD30 is small because the size of the whiskers is smaller. Comparing GC20, GC30, GD20 and GD30 the median tend to be smaller in the cases of 5 intervals Heuristic (GD20 and GD30), so we can conclude that for the GAP, the 5 Intervals Heuristic is slightly better.

Problems GC20, GD5 and GD20 do not have outliers and extremes. The outliers and extremes with smaller values of GAP as in the case of instances 2, 10, 12 and 14 for the GA20 problem are not a problem because the model in these cases achieved a smaller GAP than the usual within the expected time. The outliers and extremes with higher values of GAP as in the case of instances 8, 16 and 27 for the GC30 problem mean that sometimes the model takes more than the usual to approximate the solutions to the inferior limit within the expected time.

In the following figure we will present the results in terms of node achieved within the considered time, in each kind of test (Figure 21).

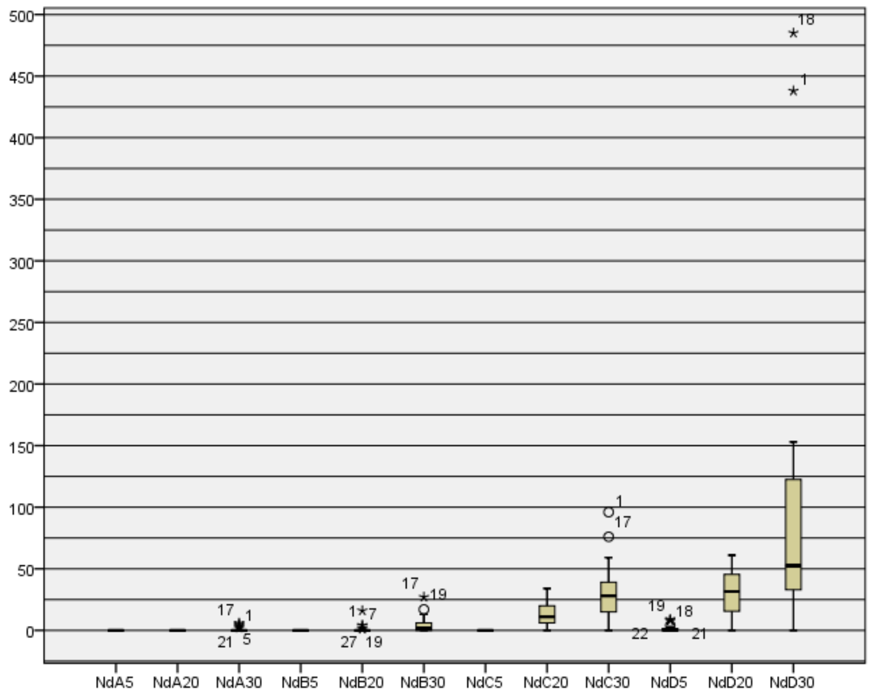


Figure 21: Box plots of the global problem with the node where the model was searching in the first 5, 20 and 30 minutes of running time.

Analyzing Figure 21 we can see that in the first 5 minutes the model is usually searching in the node 0. After 20 minutes only in the cases of 10 and 5 intervals heuristic the search leaves the node 0. We can conclude also that the median of the 10 and 5 Intervals Heuristics in the first 20 and 30 minutes of running time is higher than in the other cases.

Problems NdA30, NdB20, NdB30, NdC30 and NdD5 have outliers and extremes. These outliers and extremes are always for nodes higher than the usual, as in the case of instances 1, 5, 17 and 21 for the NdA30 problem, meaning that sometimes the model searches in more nodes than the usual within the expected time.

Following we will present the box plots with the number of solutions found in the first 5, 20 and 30 minutes of running time (Figure 22).

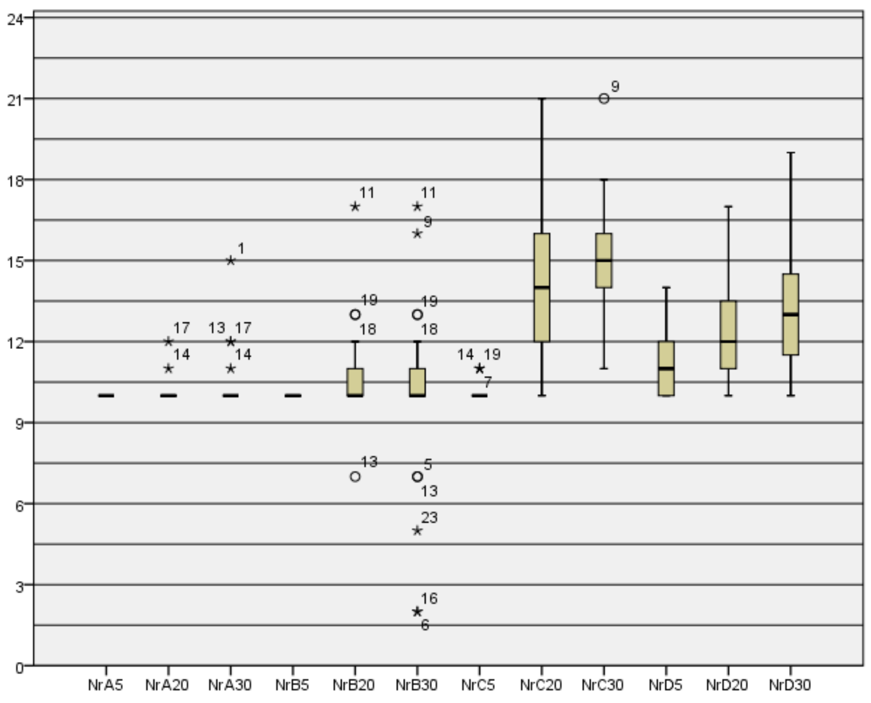


Figure 22: Number of solutions found in the first 5, 20 and 30 minutes of running time.

In the first 7 problems analyzed we can see that usually the number of solutions found is 10 because the median has that value. For the 10 Interval Heuristic in the first 20 and 30 minutes and for the 5 Interval Heuristic in the first 5, 20 and 30 minutes the values are higher meaning that more solutions were found within the considered running times.

Problems NrA20, NrA30, NrB20, NrB30, NrC5 and NrC30 have outliers and extremes. The outliers and extremes with a smaller number of solutions found as in the case of instances 5, 6, 13, 16 and 23 for the NrB30 problem, mean that in these cases the number of solutions found is lower than the usual. The outliers and extremes with a bigger number of solutions found as in the case of instances 9, 11, 18 and 19 for the NrB30 problem, mean that the number of solutions found is higher than the usual.

Analyzing optimal solutions in each problem for all instances, optimal solutions were achieved in the following cases:

- Instance 3 – 5 and 10 Intervals Heuristics;
- Instance 4 – 5 and 10 Intervals Heuristics;
- Instance 11 – 5 and 10 Intervals Heuristics;
- Instance 12 – 5 and 10 Intervals Heuristics;
- Instance 14 – 5 and 10 Intervals Heuristics;
- Instance 19 – 5 Interval Heuristic;
- Instance 24 – 5 and 10 Intervals Heuristics.

This means that optimal solutions are only found using the global model and with a number of intervals equal to 5 and 10. And only in one case (instance 19) an optimal solution was found only with a number of intervals equal to 5. This can help us concluding that the 10 Intervals Heuristic is enough to produce good results when it is possible to find optimal solutions within 30 minutes, with exception for the case of the instance 19.

In the next box plots we will analyze the time, the spread, the node and the number of solutions found before finding an optimal solution.

6.3.3 *Results and Analyses of Optimal Solutions*

The box plots presented in this subsection represent the results when an optimal solution is achieved. Here we are not analyzing each instance and each kind of problem as we have done in subsections 6.3.1 and 6.3.2, but we are just looking for the results that achieved optimal solutions.

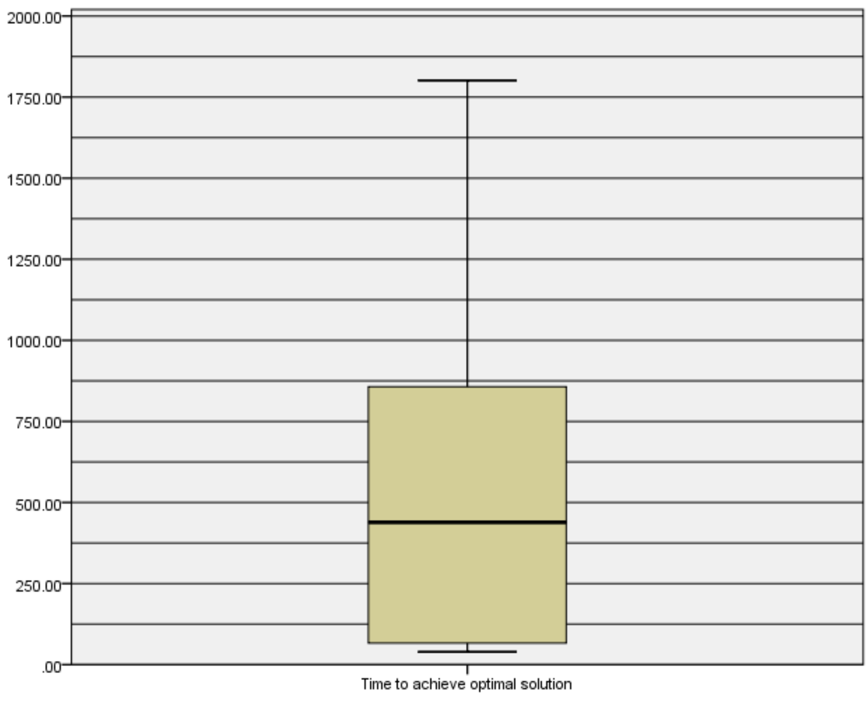


Figure 23: Box plot with the time, in seconds, to achieve optimal solutions.

The box plot represented in Figure 23 shows the minimum value of the sample is 39.52 seconds and the maximum value is 1801.02 seconds. The median is equal to 438.85 seconds, less than 10 minutes. The box plot is skewed down. The top whisker is also much longer than the bottom whisker. This means that the dispersion of values of the sample, smaller than the median, is inferior than the ones above the median.

Following we will present the box plot for the spread values of optimal solutions (Figure 24).

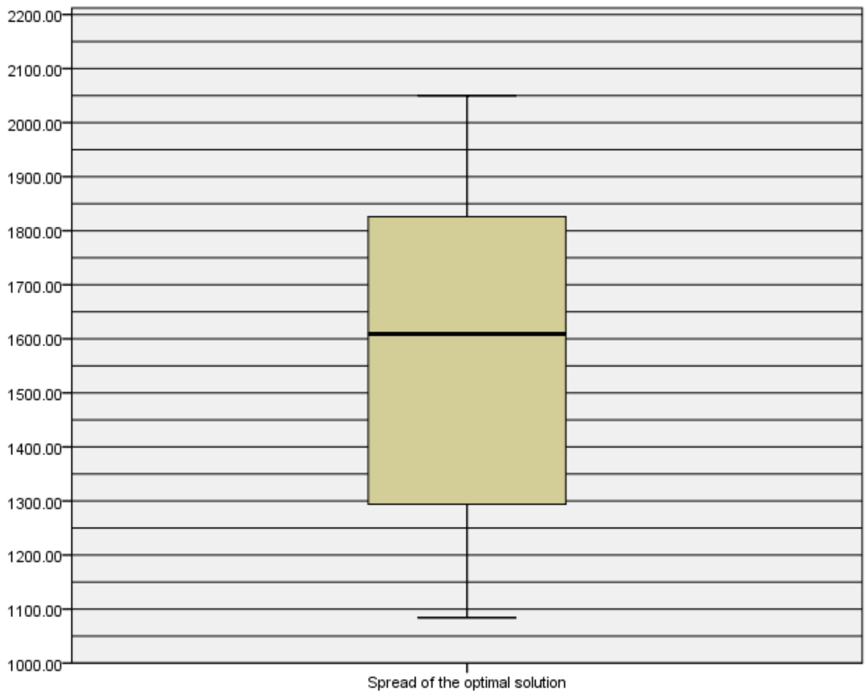


Figure 24: Box plot with the optimal spread of the optimal solutions.

Analyzing the Figure 24 we can conclude that the minimum value of the sample is 1084 and the maximum value is 2050. The median is equal to 1609 and is closer to the top of the box. This means that the dispersion of values of the sample, higher than the median, is inferior than the ones above the median.

The next picture represents the box plot with the values of the nodes where the optimal solutions were found (Figure 25).

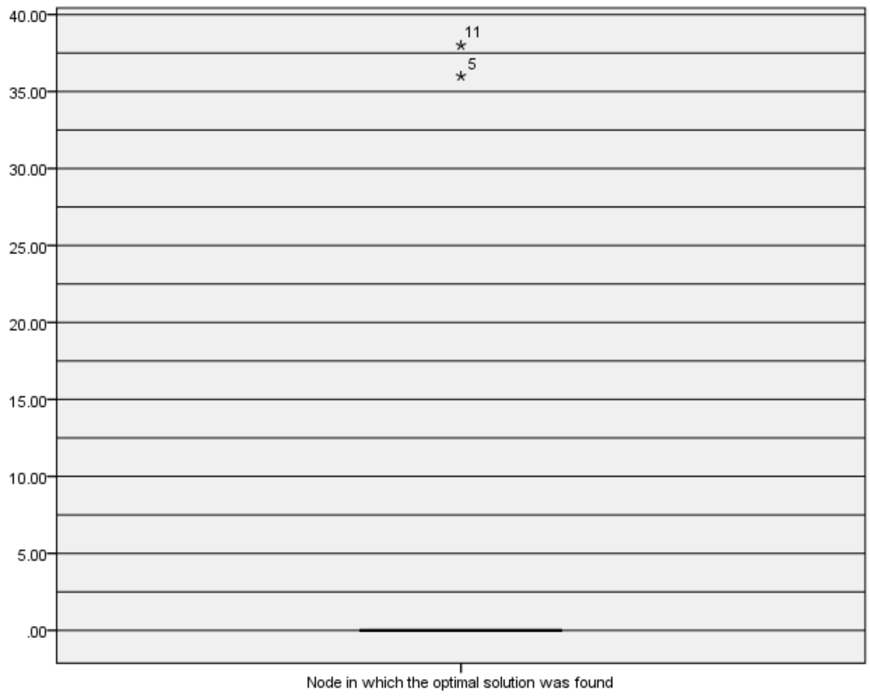


Figure 25: Box plot representing the nodes where the optimal solutions were found.

Examining Figure 25 we can conclude that only in two instances the optimal solution was not found in the node 0. These instances are the number 5 and the number 11. So we can affirm that in 92.59% of the instances the optimal solution was found in the node 0.

Following we will present the results in terms of number of solutions found until the achievement of the optimal solution (Figure 26).

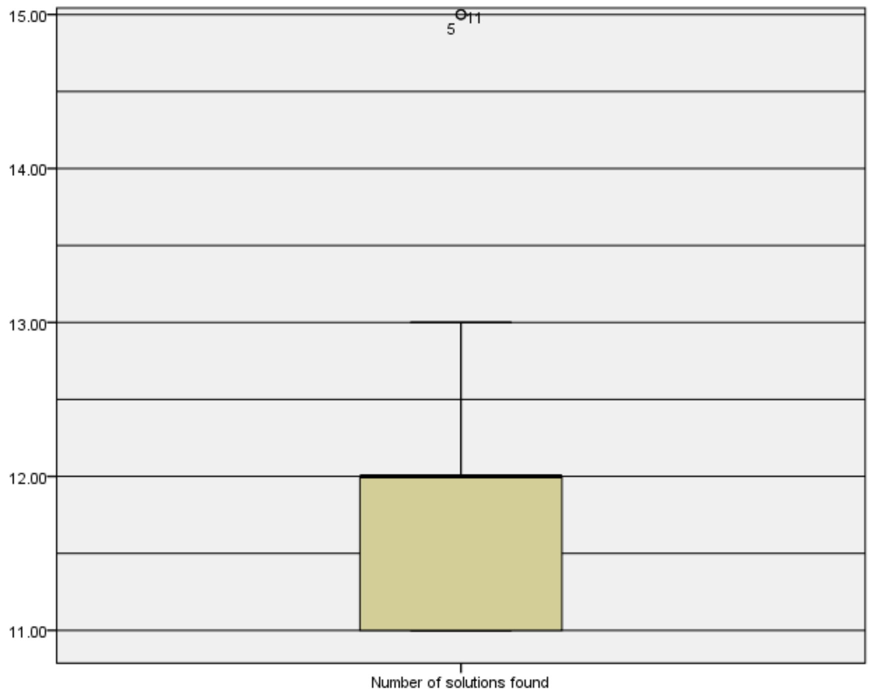


Figure 26: Box plot representing the number of solutions found to achieve the optimal solution.

Analyzing Figure 26 the minimum value of the sample is 11 and the maximum value is 15. The median is equal to 12. Only instances 5 and 11 are outliers meaning that these values deviate from the standard values of the sample.

6.3.4 FINAL ANALYSES

Considering the capacity constraint problem and the capacity constraint and special cars problems, the optimal solutions can be obtained in less than 1 minute, on average, for all instances. The complexity of the problem increases when the spread is considered because times to achieve optimal solutions are higher. The median is approximately 438,85 seconds for 26% of the instances. In 74% of the instances the time to find the optimal solution is more than 30 minutes.

The 10 Intervals Heuristic and the 5 Intervals Heuristic led to better results in terms of spread and GAP. The 10 Intervals Heuristic brings slightly better results in terms of spread but the 5 Intervals Heuristic gives us results closer to the optimum because the results have a slightly inferior GAP. Thus, analyzing the inclusion of the spread in the model, the heuristic approach gives us always better results, in 30 minutes, in terms of spread and GAP. Thus we can say that solutions that significantly reduce the color spread can be consistently obtained within 30 minutes, using the heuristic and a number of intervals equal to 10. If we consider the GAP, a number of intervals equal to 5 brings results slightly closer to the optimum when compared with a number of intervals equal to 10. Even though, further research could show better choices in terms of the number of intervals. Concluding, the heuristic allow us to find better solutions in less time when comparing it to the exact approach without intervals.

7 CONCLUSIONS

Nowadays markets are highly competitive and for this reason the design and planning of mixed model assembly lines appears as a core competence, where sequencing issues, dealing with blockage and starvation caused by product variety, need to be accounted for.

We intend with this PhD thesis to enrich the mathematical models that exist in the literature to solve Car Sequencing problems. These models applied to the automotive industry do not exist in a large number in the scientific community and most of them were initiated and/or developed in the ROADEF'2005.

Our new model for Car Sequencing includes a new hierarchical exact approach and a heuristic that:

- tries to obey to all the capacity constraints, finding a solution that minimizes the number of capacity constraint violations;
- places all special cars first in the sequence;
- when capacity constraint violations cannot be avoided, will allow to find the number of extra workers (relief men) as the workstation and correct time when the extra workers are needed;
- cluster cars with the same color, minimizing the color spread.

Considering that the color spread, in the model, is a new concept developed in this PhD thesis. We believe that, at least, the following issues may be envisaged with car sequence plans that take into account color spread:

- Better synchronization between painting and final assembly
    - If the car sequence plan for the final assembly has one blue car first in the sequence (meaning to assemble the car in the early morning) and one last blue car in the sequence (meaning to assemble the car late in the afternoon), both blue cars have to be painted in a batch in the previous day, and stored in the Random Access Storage (RAS), between the Painting and the Assembly Line;
    - On the other hand, if both cars appear close in the sequence, a better synchronization between Painting and Final Assembly may be obtained, possibly reducing Work in Progress. Therefore, solutions that minimize color spread and cluster cars in groups may render paint batch scheduling easier.
- Better logistics with suppliers
    - Considering, for instance, the case of bumpers, which have an attribute color, and are supplied in a sequence that follows the car sequence plan. Delivering bumpers when colors are clustered may reduce the supplier operations needed to reorganize the sequence, reducing their costs and even reducing sequence supply errors.

The model results show that solutions that satisfy just the capacity constraints and special cars can be obtained in less than a 1 minute, on average. Considering the global problem, the model was strengthened improving the times in 45%, and a new heuristic was created to improve the computational times allowing to achieve results in less than 30 minutes. Solutions that significantly reduce the color spread can be consistently obtained within 30 minutes, using the heuristic and a number of intervals equal to 10. If we evaluate solutions near the heuristic optimum, a number of intervals equal to 5 brings results slightly closer to the heuristic optimum when compared with a number of intervals equal to 10.

Concluding, the results show that our model is robust and can be used as a tool to create production plans for sequencing cars in final assembly automotive industries. When capacity constraint violations cannot be avoided, it allows to easily find the number of extra workers (relief men), as well as the workstations and the correct times when the extra workers are needed. Also, as shown in this PhD thesis, the color spread concept allows improving the global performance of the cars manufacturers production systems.

The research questions developed on subsection 1.1 were answered with success since:

- We built a Car Sequencing model that, given a daily demand, determines the best sequence considering, by order of importance, the number of times that a capacity constraint is violated, that special cars should come first in the sequence and the minimization of the spread of cars with the same color;
- Our model is able to determine solutions in less than half of an hour allowing the use of our model for Car Sequencing in automotive companies.

In the following subsection we will indicate some paths for further research.

7.1 FUTURE WORK

This model was developed to be applied to sequence cars in final assembly of cars manufacturers. Nevertheless, when we decided to consider colors as in the ROADEF'2005 challenge, we saw that this model will allow working in the sequence of the Painting area creating batches to paint the cars. This model, allows the reduction of solvent consumption, while at the same time will also help to improve the synchronization between painting and final assembly. Although, we did not consider the painting batch size, the rework and other effects that can change the sequence of the Painting area, we will consider it for future work.

In our literature review we considered four types of sequencing models: Mixed-Model Sequencing, Car Sequencing, Level Scheduling and Hybrid Mixed-Models. However, in conversations with our industry partner, we decided to focus on Car Sequencing because their main objective was to minimize line stoppages. During the project we decided to improve the Car Sequencing method and we thought in two possible alternatives: considering leveling of part supply or considering colors. We decided to consider colors as in ROADEF'2005 challenge because it will allow to work in the sequence of the Painting area and in the sequence of the final assembly. Also, better results can be obtained applying the developed model in the final assembly logistic. A good example that shows that better results can be obtained is the case of bumpers, where the leveling of part supply without considering colors may increase the costs of rearrange the sequence. In our model, supplier constraints can also be considered when a part cannot be delivered in a row for supplier reasons, including them as capacity constraints. Nevertheless, the Level Scheduling and Hybrid Mixed-Models can be explored in future improvements of the model presented in this PhD thesis. It would be also interesting to include Mixed-Model Sequencing for cases where the design or re-design of the assembly line is needed.

A deeper study of the weights of the multi-objective function should be done, in order to accomplish a better refinement in the achievement of better/faster optimal solutions.

The study of new cutting planes and strong valid inequalities for Car Sequencing is a challenging problem poorly studied until now. We tried to start an analysis on this work calculating the inferior limits for the Car Sequencing problem and we consider it a challenging problem for future research. Our calculations for the inferior limits can be modeled for example as a mathematical programming problem but further research is necessary.

Finally, further research could show better choices in terms of the number of intervals for the developed heuristic.

8 Bibliography

Alvelos, Filipe Pereira Pinto Cunha. 2005. *Branch-and-Price and Multicommodity Flows*, Production and Systems Department, University of Minho, Guimarães.

Barnhart, Cynthia, Ellis L Johnson, George L Nemhauser, Martin W P Savelsbergh, and Pamela H Vance. 1998. "Branch and Price: Column Generation for solving huge integer programs." *Operations Research* no. 46 (3):316-329.

Bautista, Joaquín, Jordi Pereira, and Belarmino Adenso-Díaz. 2008. "A GRASP approach for the extended car sequencing problem." *Journal of Scheduling* no. 11:3-16.

Becker, C, and A Scholl. 2006. "A survey on problems and methods in generalized assembly line balancing." *European Journal of Operational Research* no. 168:694–715.

Bergen, M E, P van Beek, and T Carchrae. 2001. Constraint-based vehicle assembly line sequencing. Paper read at 14th Canadian Conference on Artificial Intelligence.

Bomey, Nathan. 2010. *Tennessee Volkswagen plant puts focus on quality*. Detroit Free Press 2012 [cited 10/08/2012 2010]. Available from http://www.freep.com/article/20120801/BUSINESS01/308010029/Tennessee-Volkswagen-plant-puts-focus-on-quality.

Boysen, Nils, Malte Fliedner, and Armin Scholl. 2009. "Sequencing mixed-model assembly lines: Survey, classification and model critique." *European Journal of Operational Research* no. 192:349-373.

Bradley, Stephen P, Arnoldo C Hax, and Thomas L Magnanti. 1977. *Applied mathematical programming*: Addison-Wesley Publishing Co. Inc.

Brailsford, Sally C, Chris N Potts, and Barbara M Smith. 1999. "Constraint satisfaction problems: Algorithms and applications." *European Journal of Operational Research* no. 119:557-581.

Briant, Olivier, Denis Naddef, and Grégory Mounie. 2008. Greedy approach and multi-criteria simulated annealing for the car sequencing problem *European Journal of Operational Research* (191): 993-1003.

Bukchin, J, E M Dar-El, and J Rubinovitz. 2002. "Mixed model assembly line design in a make-to-order environment." *Computers & Industrial Engineering* no. 41:405-421.

Cordeau, Jean-François, Gilbert Laporte, and Federico Pasin. 2008. "Iterated tabu search for the car sequencing problem." *European Journal of Operational Research* no. 191:945-956.

Dal Poggetto, Priscila. 2012. *Produção de veículos cai 28.6% em relação a Novembro de 2007*. globo.com 2008 [cited 10/08/2012 2012]. Available from http://g1.globo.com/Noticias/Carros/0,,MUL910688-9658,00-PRODUCAO+DE+VEICULOS+CAI+EM+RELACAO+A+NOVEMBRO+DE.html.

Drexl, Andreas, and Alf Kimms. 2001. "Sequencing JIT Mixed-Model Assembly Lines Under Station-Load and Part-Usage Constraints." *Management Science* no. 47 (3):480–491.

Drexl, Andreas, Alf Kimms, and Lars MatthieBen. 2006. "Algorithms for the car sequencing and the level scheduling problem." *Journal of Scheduling* no. 9 (2):153-176.

Duplaga, E A, and D J Bragg. 1998. "Mixed-model assembly line sequencing heuristics for smoothing component parts usage: a comparative analysis." *International Journal Production Research* no. 36 (8):2209-2224.

Estellon, Bertrand, Frédéric Gardi, and Karim Nouioua. 2008. "Two local search approaches for solving real-life car sequencing problems." *European Journal of Operational Research* no. 191:928-944.

Fliedner, Malte, and Nils Boysen. 2008. "Solving the car sequencing problem via Branch & Bound." *European Journal of Operational Research* no. 191:1023-1042.

Gent, Ian P. 1998. Two Results on Car sequencing Problems. APES.

Gent, Ian P, and Toby Walsh. 2005. CSPlib: a problem library for constraints. In *CSPlib: a problem library for constraints*.

Golle, Uli, Franz Rothlauf, and Nils Boysen. 2011. Iterative Beam Search for Car Sequencing. In *Working Papers in Information Systems and Business Administration*. Johannes Gutenberg-University Mains.

Gottlieb, Jens, Markus Puchta, and Christine Solnon. 2003. "A study of greedy, local search and ant colony optimization approaches for car sequencing problems." In *Applications of Evolutionary Computing*. Springer Berlin / Heidelberg.

Gravel, M, C Gagné, and W L Price. 2005. "Review and comparision of three methods for the solution of the car sequencing problem." *Journal of the Operational Research Society* no. 56:1287-1295.

Heilala, Juhani, and Paavo Voho. 2001. "Modular reconfigurable flexible final assembly systems." *Assembly Automation* no. 21 (1):20-30.

Joly, Alexandre, and Yannick Frein. 2008. "Heuristics for an industrial car sequencing problem considering paint and assembly shop objectives." *Computers & Industrial Engineering* no. 55:295-310.

Junger, Michael, Thomas M Liebling, Denis Naddef, George L Nemhauser, William R Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A Wolsey. 2010. "50 Years of Integer Programming 1958-2008

Kim, Yeo Keun, Jae Yun Kim, and Yeongho Kim. 2000. "A Coevolutionary Algorithm for Balancing and Sequencing in Mixed Model Assembly Lines." *Applied Intelligence* no. 13 (3):247-258.

Kis, Tamás. 2004. "On the complexity of the car sequencing problem." *Operations Research Letters* no. 32:331-335.

Monden, Yasuhiro. 1993. *Toyota production system*. Norcross: Industrial Engineering Press, Institute of Industrial Engineers.

Pavarin, Guilherme. 2012. *Pintura Aqua-tech da Mazda promete lavar a alma do ambiente* 2009 [cited 10/08/2012 2012]. Available from http://info.abril.com.br/noticias/blogs/bitnocarro/carros-do-futuro/pintura-aqua-tech-da-mazda-promete-lavar-a-alma-do-ambiente/.

Ponnambalam, S, P Aravindan, and G Naidu. 2000. "A multi-objective genetic algorithm for solving assembly line balancing problem." *Advanced Manuf. Techn* no. 16:341-352.

Prandstetter, Matthias. 2005. *Exact and heuristic methods for solving the Car Sequencing Problem*, Institute of Computer Graphics and Algorithms, Vienna University, Vienna.

Prandstetter, Matthias, and Gunther R Raidl. 2008. "An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem." *European Journal of Operational Research* no. 191:1004–1022.

Puchta, Markus, and Jens Gottlieb. 2002. "Solving Car Sequencing Problems by Local Optimization." *Applications of Evolutionary Computing* no. 2279:181-188.

Reis, Ricardo José de Oliveira. 2007. *Solving the Car Sequencing Problem from a Multiobjective Perspective*, Instituto Superior Técnico, Lisbon.

Scholl, Armin. 1999. *Balancing and Sequencing of Assembly Lines*. New York: Physica-Verlag Heidelberg.

Schrijver, Alexander. 1998. *Theory of Linear and Integer Programming*. England: John Wiley & Sons Ltd.

Siegel, Andrew F. 2012. *Practical Business Statistics*. 6th ed: Elsevier Inc.

Solnon, Christine, Van Dat Cung, Alain Nguyen, and Christian Artigues. 2008. "The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem." *European Journal of Operational Research* no. 191:912-927.

Tsai, Li-Hui. 1995. "Mixed-model sequencing to minimize utility work and the risk of conveyor stoppage." *Management Science* no. 41 (3):485-495.

Vries, Gerda. 2001. What is mathematical modelling? Department of Mathematical Sciences: University of Alberta.

Warwick, Terry, and Edward P K Tsang. 1995. "Tackling Car Sequencing Problems Using a Generic Genetic Algorithm." *Evolutionary Computation* no. 3 (3):267-298.

Wolsey, Laurence A. 2003. "Strong formulations for mixed integer programs: valid inequalities and extended formulations." *Mathematical Programming* no. 97 (1-2):423-447.

Wolsey, Laurence A. 1998. *Integer Programming*. United States of America: Wiley-Interscience.

Zinflou, A, C Gagné, and M Gravel. 2008. "Design of an Efficient Genetic Algorithm to Solve the Industrial Car Sequencing Problem." In, edited by Witold Kosiński, 377-400. Vienna: I-Tech Education and Publishing.

Appendixes

Appendix I: Comparison of different methods to calculate capacity constraint violations

Sequences	Subsequences for calculations	ROADEF'2005 method	Sliding Window method	FB method (considering $\boldsymbol{B = N_o - H_o}$)
a**bbb**aaa**bbb**aa (option b - 1:3)	abb	1	$2 \leq 1 + 2pen \Leftrightarrow pen = 1$	$2 - (1 - 0) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	bbb	2	$3 \leq 1 + 2pen \Leftrightarrow pen = 1$	$3 - (1 - 1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 1$
	bba	1	$2 \leq 1 + 2pen \Leftrightarrow pen = 1$	$2 - (1 - 1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 1$
	baa	0	$1 \leq 1 + 2pen \Leftrightarrow pen = 0$	$1 - (1 - 1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	aaa	0	$0 \leq 1 + 2pen \Leftrightarrow pen = 0$	$0 - (1 - 0) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	aab	0	$1 \leq 1 + 2pen \Leftrightarrow pen = 0$	$1 - (1 - 0) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	abb	1	$2 \leq 1 + 2pen \Leftrightarrow pen = 1$	$2 - (1 - 0) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	bbb	2	$3 \leq 1 + 2pen \Leftrightarrow pen = 1$	$3 - (1 - 1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 1$
	bba	1	$2 \leq 1 + 2pen \Leftrightarrow pen = 1$	$2 - (1 - 1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 1$
	baa	0	$1 \leq 1 + 2pen \Leftrightarrow pen = 0$	$1 - (1 - 1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$

	aa	not calculated in this approach	not calculated in this approach	$0-(1-0)*2$ $\leq 1+2pen$ $\Leftrightarrow pen=0$
	a	not calculated in this approach	not calculated in this approach	$0-(1-0)*2$ $\leq 1+2pen$ $\Leftrightarrow pen=0$
	Total	8	6	4
bbbaaaaaa**bbb** (option b - 1:3)	bbb	2	$3 \leq 1+2pen$ $\Leftrightarrow pen=1$	$3-(1-1)*2$ $\leq 1+2pen$ $\Leftrightarrow pen=1$
	bba	1	$2 \leq 1+2pen$ $\Leftrightarrow pen=1$	$2-(1-1)*2$ $\leq 1+2pen$ $\Leftrightarrow pen=1$
	baa	0	$1 \leq 1+2pen$ $\Leftrightarrow pen=0$	$1-(1-1)*2$ $\leq 1+2pen$ $\Leftrightarrow pen=0$
	aaa	0	$0 \leq 1+2pen$ $\Leftrightarrow pen=0$	$0-(1-0)*2$ $\leq 1+2pen$ $\Leftrightarrow pen=0$
	aaa	0	$0 \leq 1+2pen$ $\Leftrightarrow pen=0$	$0-(1-0)*2$ $\leq 1+2pen$ $\Leftrightarrow pen=0$
	aaa	0	$0 \leq 1+2pen$ $\Leftrightarrow pen=0$	$0-(1-0)*2$ $\leq 1+2pen$ $\Leftrightarrow pen=0$
	aaa	0	$0 \leq 1+2pen$ $\Leftrightarrow pen=0$	$0-(1-0)*2$ $\leq 1+2pen$ $\Leftrightarrow pen=0$
	aab	0	$1 \leq 1+2pen$ $\Leftrightarrow pen=0$	$1-(1-0)*2$ $\leq 1+2pen$ $\Leftrightarrow pen=0$
	abb	1	$2 \leq 1+2pen$ $\Leftrightarrow pen=1$	$2-(1-0)*2$ $\leq 1+2pen$ $\Leftrightarrow pen=0$

	bbb	2	$3 \leq 1 + 2pen$ $\Leftrightarrow pen = 1$	$3 - (1 - 1) * 2$ $\leq 1 + 2pen$ $\Leftrightarrow pen = 1$
	bb	not calculated in this approach	not calculated in this approach	$2 - (1 - 1) * 2$ $\leq 1 + 2pen$ $\Leftrightarrow pen = 1$
	b	not calculated in this approach	not calculated in this approach	$1 - (1 - 1) * 2$ $\leq 1 + 2pen$ $\Leftrightarrow pen = 0$
	Total	6	4	4
	abb	1	$2 \leq 1 + 2pen$ $\Leftrightarrow pen = 1$	$2 - (1 - 0) * 2$ $\leq 1 + 2pen$ $\Leftrightarrow pen = 0$
	bba	1	$2 \leq 1 + 2pen$ $\Leftrightarrow pen = 1$	$2 - (1 - 1) * 2$ $\leq 1 + 2pen$ $\Leftrightarrow pen = 1$
	bab	1	$2 \leq 1 + 2pen$ $\Leftrightarrow pen = 1$	$2 - (1 - 1) * 2$ $\leq 1 + 2pen$ $\Leftrightarrow pen = 1$
	aba	0	$1 \leq 1 + 2pen$ $\Leftrightarrow pen = 0$	
a**bb**a**b**aa**bb**a**b**a (option b - 1:3)	baa	0	$1 \leq 1 + 2pen$ $\Leftrightarrow pen = 0$	$y1 - (1 - 1)$ $* 2 \leq 1 + 2pen$ $\Leftrightarrow pen = 0$
	aab	0	$1 \leq 1 + 2pen$ $\Leftrightarrow pen = 0$	$1 - (1 - 0) * 2$ $\leq 1 + 2pen$ $\Leftrightarrow pen = 0$
	abb	1	$2 \leq 1 + 2pen$ $\Leftrightarrow pen = 1$	$2 - (1 - 0) * 2$ $\leq 1 + 2pen$ $\Leftrightarrow pen = 0$
	bba	1	$2 \leq 1 + 2pen$ $\Leftrightarrow pen = 1$	$2 - (1 - 1) * 2$ $\leq 1 + 2pen$ $\Leftrightarrow pen = 1$

	bab	1	$2 \leq 1 + 2pen \Leftrightarrow pen = 1$	$2 - (1 - 1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 1$
	aba	0	$1 \leq 1 + 2pen \Leftrightarrow pen = 0$	$1 - (1 - 0) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	ba	not calculated in this approach	not calculated in this approach	$1 - (1 - 1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	a	not calculated in this approach	not calculated in this approach	$0 - (1 - 0) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	Total	6	6	4
a**bb**aa**bb**aa**bb**a (option b - 1:3)	abb	1	$2 \leq 1 + 2pen \Leftrightarrow pen = 1$	$2 - (1 - 0) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	bba	1	$2 \leq 1 + 2pen \Leftrightarrow pen = 1$	$2 - (1 - 1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 1$
	baa	0	$1 \leq 1 + 2pen \Leftrightarrow pen = 0$	$1 - (1 - 1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	aab	0	$1 \leq 1 + 2pen \Leftrightarrow pen = 0$	$1 - (1 - 0) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	abb	1	$2 \leq 1 + 2pen \Leftrightarrow pen = 1$	$2 - (1 - 0) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	bba	1	$2 \leq 1 + 2pen \Leftrightarrow pen = 1$	$2 - (1 - 1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 1$
	baa	0	$1 \leq 1 + 2pen \Leftrightarrow pen = 0$	$1 - (1 - 1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$

	aab	0	$1 \leq 1 + 2pen \Leftrightarrow pen = 0$	$1 - (1-0) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	abb	1	$2 \leq 1 + 2pen \Leftrightarrow pen = 1$	$2 - (1-0) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	bba	1	$2 \leq 1 + 2pen \Leftrightarrow pen = 1$	$2 - (1-1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 1$
	ba	not calculated in this approach	not calculated in this approach	$1 - (1-1) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	a	not calculated in this approach	not calculated in this approach	$0 - (1-0) * 2 \leq 1 + 2pen \Leftrightarrow pen = 0$
	Total	6	6	3

Appendix II: Cuts Improvements Evaluation

		Without cuts	Constraint 17	Constraints 15+16	Constraints 15+16+17	% Improvement Constraint 17	% Improvement Constraints 15+16	% Improvement Constraints 15+16+17
Instance 1	Best solution after 5 minutes	6857566	50308	6857566	37	99.27%	0.00%	100.00%
	Best solution after 20 minutes	22	21	22	37	4.55%	0.00%	-68.18%
	Best solution after 30 minutes	22	20	22	22	9.09%	0.00%	0.00%
	Optimal Solution	Solution not found	Solution not found	Solution not found	Solution not found	Solution not found	Solution not found	Solution not found
Instance 17	Best solution after 5 minutes	9552247	7209104	Solution not found	21	24.53%	0.00%	100.00%
	Best solution after 20 minutes	9552247	20	21	21	100.00%	100.00%	100.00%
	Best solution after 30 minutes	9552247	20	21	21	100.00%	100.00%	100.00%
	Optimal Solution	Solution not found	Solution not found	Solution not found	Solution not found	Solution not found	Solution not found	Solution not found
Instance 23	Best solution after 5 minutes	6553367	749	Solution not found	758	99.99%	0.00%	99.99%
	Best solution after 20 minutes	6553367	747	Solution not found	746	99.99%	0.00%	99.99%
	Best solution after 30 minutes	746	744	Solution not found	744	0.27%	0.00%	0.27%
	Optimal Solution	Solution not found	Solution not found	Solution not found	Solution not found			
Instance 24	Best solution after 5 minutes	1450486	1450270	Solution not found	Solution not found	0.01%	0.00%	0.00%
	Best solution after 20 minutes	1450486	1450035	Solution not found	Solution not found	0.03%	0.00%	0.00%
	Best solution after 30 minutes	1450486	1450035	Solution not found	Solution not found	0.03%	0.00%	0.00%
	Optimal Solution	Solution not found	Solution not found	Solution not found	Solution not found	Solution not found	Solution not found	Solution not found
					Average improvements	44.81%	16.67%	44.34%

Appendix III: Results of the capacity constraint model and the capacity constraint + special cars model

			Instances																	
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Capacity constraint problem	Optimal solution	Time (s)	9.25	9.56	18.35	11.81	6.58	8.31	8.10	Optimal Solution not obtained in 1800 seconds Best solution: 4 Inferior limit: 0 Spread: 4706 Node: 2596 Solutions found: 15	6.72	8.30	9.52	7.24	6.96	8.35	7.38	Optimal Solution not obtained in 1800 seconds Best solution:5 Inferior limit: 0 Spread: 4469 Node: 21338 Solutions found: 18	5.54	5.68
		Capacity constraint violations	0	49	5	25	0	0	0		0	81	17	85	0	17	0		0	0
		Spread	3203	4012	3485	3759	3997	3544	3656		3604	4048	3266	3914	3650	3411	3672		3874	3978
		Node	0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0
		Solutions found	11	11	11	11	11	11	11		11	11	11	11	11	11	11		11	11
Capacity constraint + special cars problem	Optimal Solution	Time (s)	20.93	185.33	2.98	15.43	8.42	3.68	2.42	Optimal Solution not obtained in 1800 seconds Best solution: 4 violations + 71 (62) Inferior limit: 0 violations + 73 (62) Spread: 3521 Node: 32713 Solutions found: 7	1.76	11.39	10.87	12.65	2.01	2.92	6.66	Optimal Solution not obtained in 1800 seconds Best solution: 5 violations + 71 (54) Inferior limit: 0 violations + 75 (54) Spread: 3575 Node: 27448 Solutions found: 8	1.28	1.14
		Capacity constraint violations	0	49	5	25	0	0	0		0	81	17	85	0	17	0		0	0
		Position last special cars	79 (79)	55 (52)	78 (69)	65 (65)	53 (53)	85 (80)	71 (59)		55 (57)	49 (49)	76 (76)	50 (50)	50 (51)	82 (77)	71 (59)		53 (53)	55 (43)
		Spread	2336	3317	2833	2832	3123	2863	3324		3007	3016	2591	2977	2966	2371	3167		2881	2953
		Node	0	300	0	0	0	0	0		0	0	0	0	0	0	0		0	0
		Solutions found	12	17	1	2	2	1	1		1	2	2	2	1	1	2		1	1

			Instances								
			19	20	21	22	23	24	25	26	27
Capacity constraint problem	**Optimal solution**	**Time (s)**	20.34	36.16	8.55	8.46	14.27	10.70	8.33	12.61	42.09
		Capacity constraint violations	1	1	0	0	0	29	0	0	0
		Spread	4047	3856	3242	3231	3808	3936	4146	4099	4335
		Node	0	0	0	0	0	0	0	0	38
		Solutions found	12	12	11	11	11	11	11	11	13
Capacity constraint + special cars problem	**Optimal Solution**	**Time (s)**	9.31	3.96	2.37	8.47	7.05	11.39	3.10	4.43	16.74
		Capacity constraint violations	1	1	0	0	0	29	0	0	0
		Position last special cars	78 (51)	92 (68)	82 (82)	109 (86)	73 (57)	62 (62)	77 (64)	95 (70)	97 (68)
		Spread	3324	2786	2197	2663	2975	3118	3206	3503	3294
		Node	0	0	0	0	0	0	0	0	0
		Solutions found	2	1	1	2	1	1	1	1	1

Appendix IV: Results of the model that considers capacity constraints, special cars and cars color

			Instance 1				Instance 2				Instance 3				Instance 4			
			Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals
Capacity constraint + special cars + spread problem	Best solution after 5 minutes	Spread	3242	3435	3324	1291	3665	3813	3831	2315	Not found	Not found	Optimal solution found before	Optimal solution found before	Not found	Not found	4007	2548
		GAP	99.99%	99.99%	99.99%	99.96%	77.41%	72.79%	75.86%	76.46%							86.38%	84.53%
		Node	0	0	0	0	0	0	0	0							0	0
		Solutions found	10	10	10	10	10	10	10	10							10	10
	Best solution after 20 minutes	Spread	3242	3435	778	804	3665	3813	3831	2315	3872	3857	Optimal solution found before	Optimal solution found before	4468	3984	Optimal solution found before	Optimal solution found before
		GAP	99.99%	99.99%	36.36%	14.29%	77.41%	72.79%	75.86%	75.52%	97.00%	96.83%			86.71%	84.50%		
		Node	0	4	34	49	0	0	0	16	0	0			0	0		
		Solutions found	10	10	18	15	10	10	10	11	10	10			10	10		
	Best solution after 30 minutes	Spread	3241	3435	778	842	3665	3813	3818	2315	3872	3857	Optimal solution found before	Optimal solution found before	4468	3984	Optimal solution found before	Optimal solution found before
		GAP	99.99%	99.99%	36.36%	10.00%	77.41%	72.79%	73.93%	75.52%	97.00%	96.83%			86.71%	84.50%		
		Node	3	10	96	438	0	0	3	24	0	0			0	0		
		Solutions found	15	10	18	16	10	10	16	11	10	10			10	10		
	Optimal Solution	Time (s)	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	58.59	57.64	Not found	Not found	731.57	1118.17
		Capacity constraint violations											5	5			25	25
		Position last special cars											78 (69)	78 (69)			65 (65)	65 (65)
		Spread											1278	1911			2032	1653
		Node											0	0			0	0
		Solutions Found											11	11			12	13

			Instance 5				Instance 6				Instance 7				Instance 8			
			Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals
Capacity constraint + special cars + spread problem	Best solution after 5 minutes	Spread	4032	Not found	4544	1220	3923	Not found	3864	2062	3643	3689	1634	1056	4235	4375	4552	2862
		GAP	99.99%		100.00%	26.92%	99.99%		100.00%	100.00%	99.99%	99.99%	44.42%	0.68%	100.00%	100.00%	100.00%	100%
		Node	0		0	2	0		0	0	0	0	0	1	0	0	0	0
		Solutions Found	10		10	12	10		10	10	10	10	11	11	10	10	10	10
	Best solution after 20 minutes	Spread	4032	Not found	4544	1220	3923	Not found	777	975	3643	3689	798	1056	4235	4375	1487	1686
		GAP	99.99%		100.00%	26.92%	99.99%		5.79%	2.65%	99.99%	99.99%	1.10%	0.68%	100.00%	100.00%	99.87%	99.93%
		Node	0		0	36	0		20	38	0	4	21	42	0	0	6	17
		Solutions Found	10		10	12	10		14	11	10	12	17	11	10	10	11	13
	Best solution after 30 minutes	Spread	4032	3840	1187	1220	3923	1066	777	975	3643	3689	798	1056	4235	4375	1487	1774
		GAP	99.99%	100.00%	48.89%	26.92%	99.99%	58.35%	5.79%	3%	99.99%	99.99%	1.10%	0.68%	100.00%	100.00%	99.87%	99.90%
		Node	4	4	2	53	0	1	39	102	0	13	21	141	0	0	21	36
		Solutions Found	10	7	16	12	10	2	14	11	10	12	17	11	10	10	11	17
	Optimal Solution	Time (s)	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found
		Capacity constraint violations																
		Position last special cars																
		Spread																
		Node																
		Solutions Found																

			Instance 9				Instance 10				Instance 11				Instance 12			
			Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals
Capacity constraint + special cars + spread problem	Best solution after 5 minutes	Spread	3818	3911	3929	2938	3644	3871	3934	4133	3763	3698	3534	1646	Not found	3696	Optimal solution found before	Optimal solution found before
		GAP	99.99%	100.00%	100.00%	100.00%	64.94%	67.48%	65.97%	61.43%	88.13%	89.84%	89.19%	0.01%		65.01%		
		Node	0	0	0	0	0	0	0	0	0	0	0	0		0		
		Solutions Found	10	10	10	10	10	10	10	11	10	10	10	11		10		
	Best solution after 20 minutes	Spread	3818	3911	987	1092	3644	3871	2369	4133	3763	2278	1170	Optimal solution found before	4171	3696	Optimal solution found before	Optimal solution found before
		GAP	99.99%	100.00%	7.02%	2.63%	64.94%	67.48%	8.18%	61%	88.13%	66.78%	22.76%		63.67%	65.01%		
		Node	0	0	12	50	0	0	7	15	0	0	18		0	0		
		Solutions Found	10	10	21	17	10	10	16	11	10	17	14		10	10		
	Best solution after 30 minutes	Spread	3818	3911	987	916	3644	3871	2369	4133	3763	2278	Optimal solution found before	Optimal solution found before	4171	3696	Optimal solution found before	Optimal solution found before
		GAP	99.99%	99.99%	7.02%	1%	64.94%	67.48%	8.18%	61%	88.13%	66.78%			63.67%	65.01%		
		Node	0	4	49	81	0	0	15	30	0	2			0	0		
		Solutions Found	10	16	21	19	10	10	16	11	10	17			10	10		
	Optimal Solution	Time (s)	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	1801.02	438.85	Not found	Not found	66.58	78.2
		Capacity constraint violations											17	17			85	85
		Position last special cars											76 (76)	76 (76)			50 (50)	50 (50)
		Spread											1294	1484			1408	2050
		Node											36	0			0	0
		Solutions Found											15	12			11	11

			Instance 13				Instance 14				Instance 15				Instance 16			
			Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals
Capacity constraint + special cars + spread problem	Best solution after 5 minutes	Spread	4123	Not found	3952	915	3411	3476	2380	Optimal solution found before	3970	4061	3908	2430	4224	Not found	4513	2684
		GAP	99.99%		100.00%	12.12%	88.21%	89.35%	43.57%		99.99%	99.99%	100.00%	100.00%	99.98%		99.98%	99.98%
		Node	0		0	0	0	0	0		0	0	0	0	0		0	0
		Solutions Found	10		10	12	10	10	11		10	10	10	10	10		10	10
	Best solution after 20 minutes	Spread	4123	793	864	915	682	3476	Optimal solution found before	Optimal solution found before	3970	4061	1139	1352	4224	Not found	1507	2684
		GAP	99.99%	46.79%	24.44%	12.12%	0.03%	89.35%			99.99%	99.99%	3.78%	1.74%	99.98%		99.59%	99.98%
		Node	0	0	10	42	0	0			0	0	19	28	0		6	0
		Solutions Found	10	7	15	12	11	10			10	10	16	16	10		14	10
	Best solution after 30 minutes	Spread	99.99%	793	864	932	682	3476	Optimal solution found before	Optimal solution found before	3970	4061	1139	1352	4224	4307	1507	2684
		GAP	0	46.79%	24.44%	9.38%	0.03%	89.35%			99.99%	99.99%	3.78%	1.74%	99.98%	99.98%	99.59%	99.98%
		Node	10	8	29	142	6	0			0	0	37	62	0	2	23	0
		Solutions Found	99.99%	7	15	13	11	10			10	10	16	16	10	2	14	10
	Optimal Solution	Time (s)	Not found	Not found	Not found	Not found	Not found	Not found	856.87	39.52	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found
		Capacity constraint violations							17	17								
		Position last special cars							82 (77)	82 (77)								
		Spread							1100	1655								
		Node							0	0								
		Solutions Found							12	11								

			Instance 17				Instance 18				Instance 19				Instance 20			
			Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals
Capacity constraint + special cars + spread problem	Best solution after 5 minutes	Spread	4059	3790	3772	3822	4009	3468	3887	1142	Not found	3888	1689	1306	Not found	4115	3961	2260
		GAP	99.99%	100.00%	100.00%	100.00%	99.98%	99.99%	99.99%	1.12%		99.37%	0.07%	0.02%		99.41%	99.37%	99.37%
		Node	0	0	0	0	0	0	0	4		0	0	9		0	0	0
		Solutions Found	10	10	10	10	10	10	10	13		10	11	13		10	10	10
	Best solution after 20 minutes	Spread	1092	783	733	910	4009	3446	836	1142	3772	911	1094	Optimal solution found before	3936	2444	1769	1758
		GAP	99.04%	36.33%	13.79%	5.00%	99.98%	99.99%	2.41%	1.12%	99.32%	0.29%	0.03%		99.44%	95.34%	53.65%	66.35%
		Node	0	16	30	8	0	0	27	61	0	2	20		0	0	0	0
		Solutions Found	12	11	16	13	10	13	14	13	10	13	14		10	11	11	11
	Best solution after 30 minutes	Spread	1092	783	733	910	4009	3446	836	1142	3772	911	1094	Optimal solution found before	3936	2444	847	1046
		GAP	99%	36.33%	13.79%	5.00%	99.98%	99.99%	2.41%	1.12%	99.32%	0.29%	0.03%		99.44%	95.34%	0.02%	0.01%
		Node	5	27	76	8	0	6	59	485	0	17	28		0	0	0	0
		Solutions Found	12	11	16	13	10	13	14	13	10	13	14		10	11	16	13
	Optimal Solution	Time (s)	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	747.67	Not found	Not found	Not found	Not found
		Capacity constraint violations												1				
		Position last special cars												78 (51)				
		Spread												1084				
		Node												38				
		Solutions Found												15				

			Instance 21				Instance 22				Instance 23				Instance 24			
			Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals
Capacity constraint + special cars + spread problem	Best solution after 5 minutes	Spread	3419	3343	3513	900	3303	3312	3528	1792	4003	Not found	3832	1293	4119	Not found	4169	1940
		GAP	100.00%	100.00%	100.00%	14.29%	99.98%	99.98%	99.98%	2.60%	99.98%		100.00%	1.34%	82.33%		81.07%	0.02%
		Node	0	0	0	8	0	0	0	8	0		0	0	0		0	0
		Solutions Found	10	10	10	13	10	10	10	11	10		10	11	10		10	11
	Best solution after 20 minutes	Spread	3419	3343	812	900	3303	3312	744	1792	4003	Not found	1161	1293	4119	Not found	Optimal solution found before	Optimal solution found before
		GAP	100.00%	100.00%	33.33%	14.29%	99.98%	99.98%	0.48%	2.60%	99.98%		3.00%	1.34%	82.33%			
		Node	0	0	0	57	0	0	25	19	0		9	34	0			
		Solutions Found	10	10	14	13	10	10	13	11	10		16	11	10			
	Best solution after 30 minutes	Spread	3419	3343	812	900	3303	3312	744	1092	4003	3862	1161	1215	4119	Not found	Optimal solution found before	Optimal solution found before
		GAP	100.00%	100.00%	33.33%	14.29%	99.98%	99.98%	0.48%	0.41%	99.98%	99.98%	3.00%	1.07%	82.33%			
		Node	2	0	8	153	0	6	47	36	0	2	34	52	0			
		Solutions Found	10	10	14	13	10	10	13	13	10	5	16	13	10			
	Optimal Solution	Time (s)	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	867.37	358.51
		Capacity constraint violations															29	29
		Position last special cars															62 (62)	62 (62)
		Spread															1826	1609
		Node															0	0
		Solutions Found															12	12

			Instance 25				Instance 26				Instance 27			
			Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals	Without heuristic	With heuristic 100 intervals	With heuristic 10 intervals	With heuristic 5 intervals
Capacity constraint + special cars + spread problem	Best solution after 5 minutes	Spread	**4036**	**4174**	**4236**	**983**	3852	4086	4199	1061	4239	4146	4297	2900
		GAP	**99.99%**	**99.99%**	**99.99%**	**99.49%**	**99.98%**	**99.98%**	**99.98%**	0.17%	99.97%	99.97%	99.97%	99.98%
		Node	**0**	**0**	**0**	**0**	**0**	**0**	**0**	3	**0**	**0**	**0**	**0**
		Solutions Found	**10**	**10**	**10**	**12**	**10**	**10**	**10**	14	**10**	**10**	**10**	**10**
	Best solution after 20 minutes	Spread	**4036**	**4174**	**943**	**983**	3852	4086	1030	1061	4239	4146	1034	1404
		GAP	**99.99%**	**99.99%**	**2.08%**	**99.49%**	**99.98%**	**99.98%**	0.75%	0.17%	99.97%	99.97%	95.65%	0.44%
		Node	**0**	**0**	**9**	**29**	**0**	**0**	20	51	**0**	**1**	1	12
		Solutions Found	**10**	**10**	**15**	**12**	**10**	**10**	11	14	**10**	**11**	12	14
	Best solution after 30 minutes	Spread	**4036**	**4174**	**943**	**999**	3852	1697	1030	1061	4239	4146	1034	1251
		GAP	**99.99%**	**99.99%**	**2.08%**	**0.58%**	**99.98%**	99.48%	0.75%	0.17%	99.97%	99.97%	95.65%	0.31%
		Node	**0**	**0**	**23**	**42**	**0**	1	36	104	**0**	**6**	14	47
		Solutions Found	**10**	**10**	**15**	**15**	**10**	11	11	14	**10**	**11**	12	14
	Optimal Solution	Time (s)	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found
		Capacity constraint violations												
		Position last special cars												
		Spread												
		Node												
		Solutions Found												