P.PORTO

SCHOOL OF
MANAGEMENT AND
TECHNOLOGY
POLYTECHNIC OF
PORTO

M

MASTER
METHODS FOR BUSINESS DECISION MAKING

Optimization of Aluminium Profiles Production Planning

Elisabete Sofia Queirós Almeida

11/2019

Elisabete Sofia Queirós Almeida. Optimization of Aluminium Profiles Production Planning

11/2019

P.PORTO

SCHOOL OF MANAGEMENT AND TECHNOLOGY POLYTECHNIC OF PORTO

P.PORTO

MASTER METHODS FOR BUSINESS DECISION MAKING

M

Optimization of Aluminium Profiles Production Planning
Elisabete Sofia Queirós Almeida

SCHOOL OF
MANAGEMENT AND
TECHNOLOGY
POLYTECHNIC OF
PORTO

P.PORTO

M

MASTER
METHODS FOR BUSINESS DECISION MAKING

Optimization of Aluminium Profiles Production Planning

Elisabete Sofia Queirós Almeida

Supervisors
PhD Eliana Oliveira da Costa e Silva
PhD Aldina Isabel de Azevedo Correia
PhD Maria de Fátima de Almeida Ferreira

Jury

President

PhD Alexandra Maria da Silva Braga

School of Management and Technology - Polytechnic of Porto

Supervisors

PhD Eliana Oliveira da Costa e Silva

School of Management and Technology - Polytechnic of Porto

PhD Aldina Isabel de Azevedo Correia

School of Management and Technology - Polytechnic of Porto

PhD Maria de Fatima de Almeida Ferreira

School of Management and Technology - Polytechnic of Porto

External Examiner

PhD Rui Jorge Ferreira Soares Borges Lopes

University of Aveiro

“O sucesso é ir de fracasso em fracasso sem perder entusiasmo”

(Winston Churchill)

Acknowledgements

Um trabalho de mestrado é uma longa viagem, que inclui uma trajetória permeada por inúmeros desafios, tristezas, incertezas, alegrias e muitos percalços pelo caminho. Trilhar este caminho só foi possível com o apoio e incentivo de várias pessoas, a quem dedico especialmente este projeto e às quais estarei eternamente grata.

Especialmente às minhas orientadoras, Professora Doutora Eliana Costa e Silva, Professora Doutora Aldina Correia e Professora Doutora Fátima Almeida, pela sua orientação, total apoio, disponibilidade, ensinamentos, incentivos e dedicação incondicional. Pela partilha do saber e valioso contributo na elaboração desta dissertação, muito obrigada!

Obrigada também à instituição, coordenação do curso, bem como aos restantes professores, por me proporcionarem ferramentas e material necessário para uma boa aprendizagem.

Aos meus pais, António e Fernanda, pelas palavras de incentivo e encorajamento valorizando o meu potencial, nos momentos mais difíceis. Sem vocês não teria esta oportunidade de lutar pelos meus sonhos e objetivos.

Ao meu irmão, Miguel, e restantes familiares, pela ajuda, apoio, incentivo, colaboração e encorajamento.

Ao meu namorado, Filipe, por todo o incentivo, carinho, companheirismo durante todo este longo percurso. Obrigada por sempre acreditares nas minhas capacidades e por todo o amor e dedicação.

A minha profunda gratidão aos meus amigos, sem vocês não teria tido a mesma força de vontade, obrigada pelas palavras de conforto e apoio.

Finalmente, ao Professor Doutor Rui Borges Lopes pelas criticas construtivas que enumerou na arguência do presente trabalho e que contribuíram para esta versão final.

Agradeço por todo o apoio, palavras de incentivo constantes, carinho e amizade, pela oportunidade de crescimento académico e também pessoal.

A Todos, Muito Obrigada!

Resumo

No atual ambiente empresarial, a crescente competitividade impulsiona as empresas a implementar estratégias de otimização para assegurar ou melhorar a sua posição no mercado.

Nesse sentido é crucial tomar as melhores decisões do ponto de vista do planeamento da produção. A produção de perfis de alumínio apresenta vários desafios ao responsável da produção. Este trabalho aborda um caso real de uma empresa do setor metalúrgico, cuja área de negócio é o desenvolvimento e produção de perfis de alumínio para aplicação em diversas áreas, como obras de engenharia, arquitetura e indústria em geral. O objetivo é minimizar o desperdício ocorrido na produção, designado de sucata, através da minimização dos tempos de preparação, que ocorrem quando a matriz é trocada, respeitando os prazos de entrega do produto e garantindo a qualidade. Este estudo de caso centra-se num problema de sequenciamento de flow shop que envolve tempos de preparação dependentes da sequência decorrentes da necessidade de alterar as ferramentas usadas no processo de extrusão de alumínio. Um modelo de programação inteira mista é desenvolvido e implementado para responder ao desafio da empresa. O problema será formulado na linguagem de modelagem AMPL e será usado o solver Gurobi para resolver instâncias reais extraídas dos dados providenciados pela empresa. Os resultados obtidos com o modelo desenvolvido são comparados com a regra de despacho FIFO, em termos da soma dos tempos de preparação (e consequentemente do número de trocas de matrizes), bem como o cumprimento do prazo de entrega. Além disso, embora o procedimento atual da empresa não seja conhecido, os resultados obtidos são comparados com a média de trocas de matriz obtidas a partir dos dados históricos de produção fornecidos pela empresa. Conclui-se que, utilizando as soluções obtidas com o modelo desenvolvido, a quantidade de sucata gerada durante o processo de produção é minimizada, uma vez que, o modelo garante a minimização da soma dos tempos de preparação e minimiza o número de vezes que ocorrem trocas de matriz.

Palavras Chave:: Escalonamento, Tempo de Processamento, Otimização, Programação Linear Inteira Mista, Extrusão de Alumínio, Sucata.

Abstract

In the current business world, the increasing competitiveness forces companies to adopt optimization strategies to ensure or improve their market position. It is crucial to take the best decisions from the production planning point of view.

The production of aluminium profiles poses several challenges to the production manager. This work addresses a real case of a company that operates in the aluminium market, whose core business is the development and production of aluminium profiles for application in several areas, such as, engineering, architecture and industry works in general. The aim is to minimize production waste, commonly known as scrap, through minimization of setup times, that occur when the die is changed, while respecting product delivery times and maintaining quality. This case study focuses on a scheduling flow shop problem involving sequence-dependent setup times arising from the need to change the tools used in the process of aluminium extrusion. A mixed integer programming model is developed and implemented for answering the company's challenge. The problem is formulated the AMPL modeling language and the Gurobi solver is used to solve real instances extracted from data provided by the company. The results obtained with the developed model are compared to the dispatching rule FIFO, in terms the sum of the setup times (and consequently the number of exchanges of dies), as well as the fulfillment of the deadline. Furthermore, although the company's current procedure is not known, the results obtained are compared with the average of die changes obtained from the historical production data provided by the company. It is observed that by using the solutions obtained with the developed model, the quantity of scrap that is generated during the production process is minimized, since the model guarantees the minimization of the sum of the setup times, therefore minimizes the number of times there are die changes.

Keywords: Scheduling, Setup Times, Optimization, Mixed Integer Linear Programming, Aluminium Extrusion, Scrap.

Contents

List of Figures

List of Tables

Acronyms

CR Critical Ratio

EDD Earliest Due Date

FIFO First In First Out

FSP Flowshop Scheduling Problem

JSP Job Shop Scheduling Problem

LP Linear Programming

MP Mathematical Programming

MILP Mixed Integer Linear Programming

MINLP Mixed Integer Nonlinear Programming

MIP Mixed Integer Programming

NLP Nonlinear Programming

OR Operational Research

SPT Shortest Processing Time

SQP Quadratic Programming

Chapter 1

Introduction

1.1 Study Contextualization

Nowadays, the increasing globalization of different types of markets leads companies to invest repeatedly in order to improve and optimize all processes that comprise their activities. Companies have to be permanently making decisions, either at a productive level or with human resources, aiming for constant cost savings, and for continuous improvement of their services. Industrial needs are becoming always more complex pushed by an ever more demanding market and an increasingly fierce competition. In fact, companies are required to meet tight deadlines that are set with the customers while maintaining high quality products. Failing to do so can result in significant financial and productivity losses, and ultimately in losing the customer [29]. The productive activities of companies are one of the areas that most studies and developments have generated, providing an increasingly efficient and effective production. It is precisely in this area that production planning and scheduling is encompassed. This aims to structure, as best as possible, the production process of various products, organizing and scheduling the different jobs that constitute it.

Thus, effective sequencing and scheduling have become a requirement for industries to maintain competitive positions into today fast-changing business environment. Production scheduling is the major component for the efficient management of production. This class of problems is among the most studied in Operational Research (OR) field and presents applications in several areas [29]. However, the reality of the production systems is more complex than the theoretical classical models and there is a noticeable gap between the theory and the application of the existing methods [29].

Scheduling refers to the allocation of resources to jobs over a set of given time periods, optimizing one or more criteria [69]. Finding the best schedule can be very easy or very difficult, depending on the shop environment, the process constraints and the performance indicators. The main goal is to find a schedule that satisfies all resource constraints while minimizing the objective.

According to Harjunkoski *et al.* [40] the major decisions scheduling problem are: **(i)** selection and sizing of tasks; **(ii)** assignment of tasks to processing units; and **(iii)** sequencing and/or timing of tasks on each unit (see Figure 1.1).

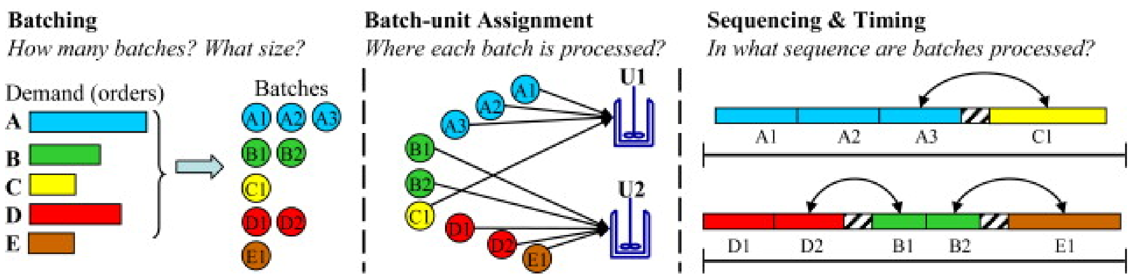


Figure 1.1: Decisions in batch continuous process scheduling. Source: [40].

Production scheduling problems have been the subject of many research works in the last decades and feature practical relevance to process industries [40]. Fuchigami and Rangel [29] in 2018, review articles published in the last 25 years and identify case studies in various productive sectors. The authors consider that most of the production scheduling literature is theoretical and does not model many of the complexities experienced in practice.

The first published work in production scheduling is the Johnson's research publication in 1954, addressing the Flow Shop Scheduling Problem (FSP) [46]. In the classical FSP, a set of jobs (indexed by j) flow through multiple stages in the same machine (indexed by i) order, where each stage consists of only one machine. The author presents a simple algorithm that optimally solves the FSP in polynomial time for the special case where $i = 2$. The Johnson's original paper assumed that setup times are included in the processing times. This is a common assumption for this type of problems.

According to Allahverdi [3], more than 90% of the literature on scheduling problems ignores setup times/costs. However the author emphasizes that setup times/costs need to be explicitly considered while scheduling decisions are made in order to increase productivity, eliminate waste, improve resource utilization, meet deadlines, increased profitability and satisfaction and increased customer satisfaction. Setup time represents the time loss incurred at each operation changeover and appear in a infinity of industrial and service applications. Allahverdi and Soroush [6] emphasize the importance of reducing setup times/setup costs across various industries, including the aluminium industry.

Scheduling problems have been receiving considerable attention since the mid–1960 and three comprehensive literature reviews have been published regarding the research on scheduling problems with setup times/costs. On the other hand, globalization and constant changes in the business environment are relevant factors that increase exportation. The national metallurgy and metal mechanics industry, is characterized by being the most exporting industry in the Portuguese economy, operating in more than 200 markets, such as in Spain, Germany, France, United Kingdom, Italy, Angola and United States of America [2].

Specifically, in the aluminium industry, profiles are mostly produced from aluminium alloys by the extrusion process. Saha [78] points that extrusion is a process of mechanical conformation by plastic deformation of an aluminium billet, in which the material is subjected to high pressures, applied by means of a punch, and forced to pass through the hole of a die, in order to reduce and/or change the shape of its cross-section.

1.2 The Case Study

This work addresses a real case of a company that operates in the aluminium market, whose core business is the development and production of aluminium profiles for application in engineering, architecture and industry works in general.

In this company, after the customers' orders have been received, the production manager has to develop a production plan. Workers takes into consideration the specific characteristics of the customers' order, the production capability of the factory, the existing stock, the deadline for the delivery of the final product, among other factors. The aim is to scheduling the production of orders in a way to minimize the waste, commonly known as scrap, respecting product delivery times and maintaining quality. Always that a die change is made, scrap is generate, and a setup time is verified. So, the scrap can be reduced through minimization of setup times, which occur when the die is change. This case study focuses on a FSP involving sequence-dependent setup times arising from the need to change the tools used in the process of aluminium extrusion. For this, a mathematical model will be developed for schedule the profiles production, to minimize the setup times.

1.3 Contributions of this Project

With the elaboration of this project, some contributions appear. Most of the works found of the literature on scheduling problems ignores setup times [3]. In practice, all scheduling related decisions are mainly derived by managers or operators. In order to systematically improve their decisions, computer aided tools can be useful to increase productivity, taking into consideration the involvement of parameters and the dynamic demand changes. Harjunkoski *et al.* [40] emphasize a need to further develop and apply optimization solutions in process industries. Since the existing commercial software does not answer to the company's specifications, the modelling of this industrial real challenge enhances the innovative contribution of the present work. In fact, a more dynamic framework, that supplements models of manufacturing with insights from the field of operational planning, is essential.

Another contribution is related to sustainability in the aluminum industry. Energy conservation and emissions reduction have been a fundamental concern for the sustainability of the aluminium industry, and an energy efficiency in the aluminium extrusion process can will play an important part in achieving this goal [39]. Reducing the generation of scrap in extrusion process is an energy efficiency measures in the aluminium industry [39]. Moreover, improvements in operations scheduling affect resource efficiency, and thus reduce the resource consumption and waste generation of the production process [51].

Furthermore, with this work, the aim is also contributing to the academia–industry knowledge transfer. These problems belong to the class of combinatorial optimization problems which are strongly NP–hard (Non–deterministic Polynomial time hardness), whose complexity is exponential [69].

1.4 Outline

The present document is structured in six chapters, as presented in Figure 1.1.

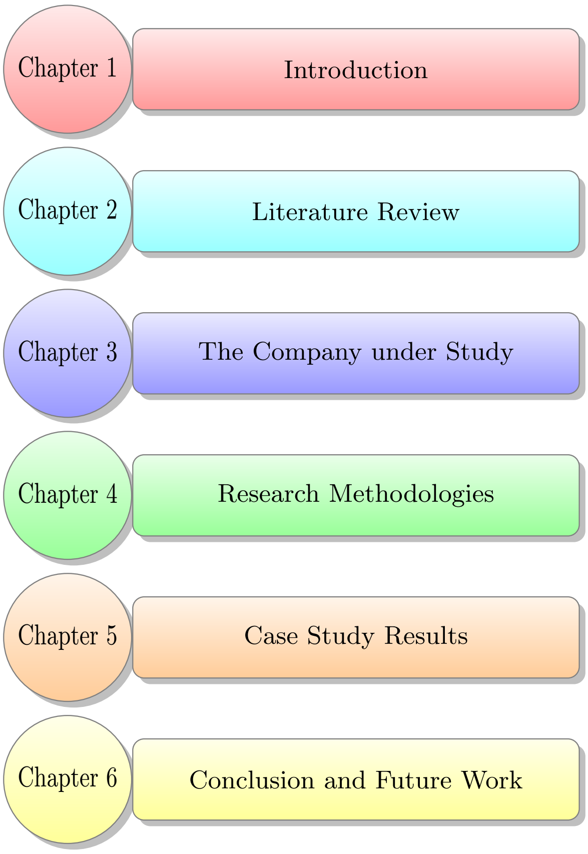


Figure 1.2: Document structure

This first chapter, includes the introduction, which provides an overview of the scheduling problem, the case study that will be addressed, the objectives that are intended to achieve, and the contributions of this project. Finally, the structure and organization of the project are exposed. In Chapter 2 a literature review of the aluminium extrusion process and production planning and scheduling is made. The company characterization, problem description and dataset are presented in Chapter 3. Chapter 4 presents the general datasets analysis and the developed mathematical model. In Chapter 5 several instances are solved to illustrate the efficiency and applicability of the suggested approach. Finally, in Chapter 6 the main conclusions and future research avenues are presented.

Chapter 2

Literature Review

In this section are exposed the concepts related with the aluminium extrusion process, production planning and scheduling. Companies are focused on improving their market competitiveness through increased operational performance therefore, production planning and scheduling is crucial and represents one of the main challenges that managers face currently.

2.1 The Aluminium Extrusion Process

In industry, extrusion is an attractive manufacturing process due to its high profitability. The first mention on the principles of extrusion is due to the famous hydraulic engineer Joseph Bramah who described a press in which the aluminium, that was kept molten in an iron pan, was forced by a pump to pass through a tube. This allowed to register the pioneer patent, with the principles of extrusion, in 1797 [81].

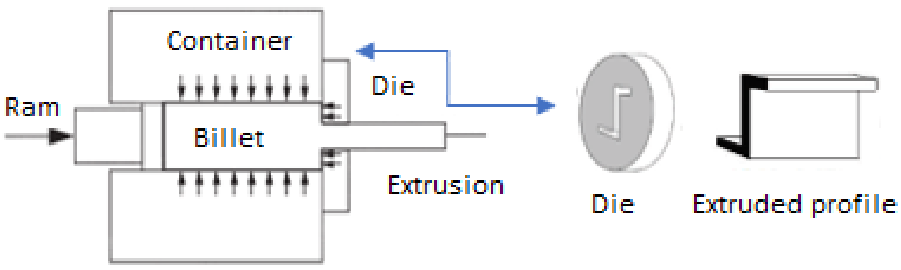


Figure 2.1: Direct extrusion process (Adapted from [78]).

Extrusion is a continuous process of production and can be defined as "a plastic deformation process in which a block of metal (billet) is forced to flow by compression through the die opening of a smaller cross–sectional area than that of the original billet" [78, p. 12] (see Figure 2.1).

This continuous process can be of two types: direct or indirect with the difference, in simple description, that the billet is the moving part in direct extrusion, whereas the die is the moving part in indirect extrusion [39], as depicted in Figure 2.2.

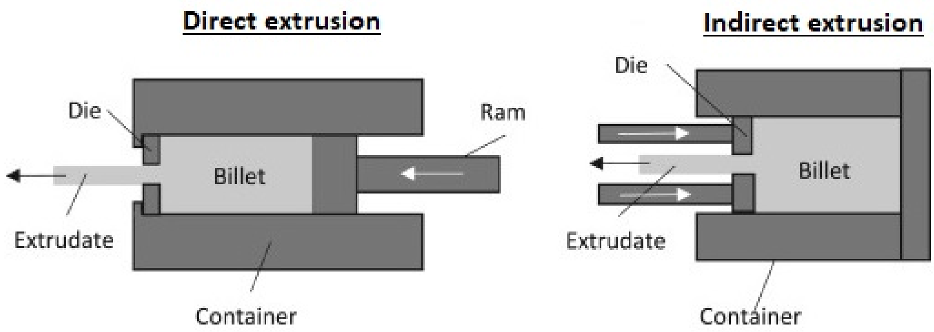


Figure 2.2: Schematic drawing of direct and indirect extrusion (Adapted from [39]).

In direct extrusion, as in the company under study, the die is located at the forward end of the container and the aluminium alloy to be extruded is pushed towards it, hence moving relative to the container. In the case of indirect extrusion, the die is placed on the end of the ram, which is bored out to allow passage of the extruded section, and moves through the container [78, 81].

Extrusion can be cold or hot [78], in a hot aluminium extrusion, aluminium billet and extrusion tools such as containers and die require preheating. During the extrusion process, the material is under high pressure and is normally driven, within the limits of the material, at high temperatures to reduce the material's resistance to deformation [8]. However, along the extrusion process temperature control is required. Temperature management plays an important role in the extrusion of aluminium, mainly the extrudate temperature at the die exit, because it largely determines the microstructure, mechanical properties and surface quality of the extruded product [101].

Extrusion is a complex process that is affected by several variables during extrusion but also in pre–extrusion and post–extrusion. All the variables have to be controlled, since they directly influence the quality of the extruded product and the scrap production. Saha [78] point that the quality of the extruded product is influenced by: the extruder type; the pressure capacity and size of container; the frictional effects at the die or both container and die; the type, layout, and design of die; the length of the billet and the type of alloy; the temperature of the billet and the container; the extrusion ratio; the die and tooling temperature, as well as speed of extrusion. Ferrás *et al.* [26] presents an empirical study concerning the extrusion process of the company in study, and analyzing the dataset provided concluded that variables concerning with extrusion temperature, time, speed, pressure and die geometry are crucial to improve and control the scrap production.

Extrusion dies, used for producing aluminium profiles, may be steel circular, rectangular or tubular, and may vary in shape from simple to quite complicated, whose inscribed configuration gives the profile the desired shape [8]. The complexity of an extruded profile, can be illustrated as a function of the ratio of the perimeter to the cross–sectional area of the part. The shapes generally is classified into three groups: solid, semi–hollow and hollow, and the complexity increases from solid to semi–hollow to hollow shapes [8].

In the extrusion process, the die is inserted into the container together, if necessary, with various supporting accessories, such as: backer, bolster and feed plate [78]. These accessories are also constructed of hardened steel and provide support for the die during the extrusion process, contributing to improved tolerance control and extrusion speed [78].

The performance of the extrusion is usually limited by typical failure mechanisms related to the material. Saha [78] enumerate the most common mechanisms of failure:

- Hot;
- Plastic deformation and cracking:
- The material must have high resistance to tempering;
- High hot hardness (high yield strength);
- Good wear resistance nitriding and hard and thin coatings;
- Good toughness.

Dies, for extrusion of aluminium alloys, are exposed to severe thermal, chemical and mechanical conditions. Consequently, in the direct extrusion process, die wear occurs, exposed by the loss or progressive removal of material from a die surface [8, 78]. This wear affects the process size, shape, and quality of material flowing through the die. Therefore, there is a maximum time of use of each die and tool, in order to the quality of the extruded product is always high. According to Arif *et al.* [8] various parameters have influence on the service lifetime of the extrusion die, namely the grade of the tool steel used and heat treatments; extruded profile shape complexity and die construction; die maintenance; chemical and structure composition of billet and temperature of the extruded material; technological parameters related to the hot extrusion and wear resistance characteristics of the die bearing surface layer.

In current extrusion production, continuous billet–to–billet extrusion is often used. In this process, the new billet is welded onto the back surface of the old billet in the welding chamber or feeder ring [53]. In this process transverse welds occur which introduce a discontinuity to the weld interface in the extruded product and in many structural applications can cause quality problems. At this point, it is crucial to minimize the transverse weld length of the extruded aluminium. The main purpose of this continuous extrusion production is to keep the length of the transverse welding interface to a minimum at the same time providing a high quality weld in the extruded that is strong enough to withstand subsequent processing [55]. Transverse weld length it depends strongly both on the type of alloy and on the geometries of the corners of the die or the speed with which the materials leave the corners of the die [53]. Furthermore, the relationship between ram speed and puller speed is central [55]. The puller is close loop linked with the press cycle and it stops automatically when the press is stopped, at any time during and at the end of the extrusion process. Saha [78] states that there are many advantages of using a puller system, namely:

- Reduction of labor and increase of production;
- Reduction of the amount of scrap;
- Controlling the flow of metal through the die;
- The possibility of cutting the billet in various lengths without stopping the press.

Finally, after extrusion process, the extruded profiles are subjected to heat and/or surface treatment, which should also be considered in the production planning. This final procedure confers to the profiles greater resistance to corrosion and oxidation, in order to achieving good mechanical strength. Usually it is done by mechanical, anodizing, electrolytic, and painting (both powder and liquid) processes [78].

2.2 Production Planning and Scheduling

The current business world is subject to constant change, and the ability to respond quickly to these changes has increasingly become a critical success factor. The increasing competitiveness forces companies to adopt more efficient processes to ensure their market position. The production planning system has as its primary objective to ensure that the desired products are produced at the right time, at a minimal cost (not necessarily economic), in the exact quantities, while maintaining the established quality levels [48].

Vercellis [91] and Arenales *et al.* [7] consider that the best way to proceed is to divide the production planning process into three distinct levels: strategic planning, tactical planning and operational planning. In this sense, it is crucial to take the best decisions from the production planning point of view. In the current more globally oriented market, strategic, tactical and operational issues regarding markets, products and production are fundamental to understand for any company to stay competitive [65].

Strategic planning is at the top of the pyramid and concerns long term decisions that provide the overall direction and direction of the company. The current situation of the company, the goals to be achieved and the strategies to follow to achieve those same goals should therefore be characterized. Examples of strategic level decisions include reducing production costs, choosing suppliers, locating facilities, and so on.

The tactical planning deals with the type of aggregate planning and the amount of production over a period of weeks up to six months. This type of planning acts in each functional area of the company, serving as a kind of connection between strategic and operational plans. It is in tactical planning that goals are created and the appropriate environment is built so that the actions determined in strategic planning can be realized. Examples of such plans include purchase and production decisions, inventory policy, or transportation policy.

Finally, operational planning serves as a regulator of day–to–day decisions and therefore has a short term planning horizon. Operational planning is responsible for decisions, such as scheduling, that is, the exact order in which operations are to be performed and the sizing of production lots. In fact, Fuchigami and Rangel [29] consider that for a company to remain competitive in demanding consumer markets, one of the most important activities, at the operational level, is production scheduling. The present work focus on decision making issues related to operational planning.

Scheduling is a critical issue in process operations and is crucial for improving production performance. Scheduling is a classic combinatorial problem much studied by researchers in OR [38]. Scheduling problems are very challenging from the mathematical point of view and also highly relevant and useful for industry, which explains the large research interest and the progress that has been achieved during the last years. A general overview concerning scheduling problems are review in papers, such as, for example Graves [37], Floudas and Lin [27], Méndez *et al.* [59], Maravelias [57] and Harjunkoski *et al.* [40], Fuchigami and Rangel [29].

Scheduling is not only the sequencing, but also determining the starting and completing time of each operation based on the sequence. Pinedo [69, p. 12] defines scheduling as a decision making process that deals with the allocation of resources to jobs over given time periods in order to optimize one or more objectives.

According to Pinedo [69] all schedule can be classified into one of the following three types of schedules: non–delay schedule, active schedule and semi–active schedule. Firstly it is important to know the concepts of sequence, schedule and scheduling policy. While the sequence usually corresponds to a permutation of the n jobs or the order in which jobs are to be processed on a given machine, the schedule usually corresponds to the allocation of jobs to a given set of machines at certain times. And lastly, the concept of a scheduling policy appears in stochastic settings, and aims to "prescribe" the most appropriate action for all states in which the system can be found. Deterministic models only consider the sequences and the schedules.

The first class of schedules, non–delay schedule, happens when there are free machines whenever an operation needs to be processed. This scheduling class prohibits unforced idleness. The second class, active schedule, is imposed when it is not possible to construct another schedule, through process order changes on the machines, leading to at least one operation ending earlier and none ending later than the starting schedule. The third class, semi–active schedule, identifies escalations where no operation can be completed earlier without changing the processing order on either machine.

Scheduling problems have different characteristics. As a way of classifying, them is useful to resort to a certain nomenclature, in order to make it easier to specify, represent and solve. Graham *et al.* [34] initially introduced the $\alpha|\beta|\gamma$ nomenclature for classification of scheduling problems.

α

Notation	Description	Notation	Description
1	Single Machine	FFm	m–stage flexible (hybrid) flow shop
P	Parallel machines (identical)	AFm	m–stage assembly flow shop
Q	Parallel machines (uniform)	J	Job shop
R	Parallel machines (unrelated)	FJ	Flexible job shop
Fm	m–stage flow shop	O	Open shop

Table 2.1: Shop type (α field)

β

Notation	Description	Notation	Description
ST_{si}	Sequence–independent setup time	SC_{sdf}	Sequence–dependent family setup cost
SC_{sd}	Sequence–dependent setup cost	ST_{psd}	Past–sequence–dependent setup time
ST_{sd}	Sequence–dependent setup time	$PREC$	Precedent constraints
ST_{sif}	Sequence–independent family setup time	r_j	Non–zero release date
ST_{sdf}	Sequence–dependent family setup time		

Table 2.2: Shop characteristics (β field)

γ

Notation	Description	Notation	Description
C_{max}	Makespan	$\sum T_j$	Total tardiness
E_{max}	Maximum earliness	$\sum W_j$	Total waiting time
L_{max}	Maximum lateness	$\sum U_j$	Number of tardy (late) jobs
T_{max}	Maximum tardiness	$\sum w_j F_j$	Total weighted flow time
D_{max}	Maximum delivery time	$\sum w_j F_j$	Total weighted flow time
TSC	Total setup/changeover cost	$\sum w_j U_j$	Total weighted number of tardy jobs
TST	Total setup/changeover time	$\sum w_j E_j$	Total weighted earliness
TNS	Total number of setups	$\sum w_j T_j$	Total weighted tardiness
$\sum F_j$	Total flow time	$\sum w_j W_j$	Total weighted waiting time
$\sum C_j$	Total completion time	$\sum h(E_j)$	Total earliness penalties
$\sum E_j$	Total earliness	$\sum h(T_j)$	Total tardiness penalties
$TADC$	Total absolute differences in completion times		

Table 2.3: Performance (γ field)

This classification is also proposed in a similar way by Pinedo [69]. The α describes the machine environment (Table 2.1), the second, β, indicates the characteristics associated with jobs (Table 2.2), and the third, γ, defines the performance measure (Table 2.3) [69]. For example, a single machine Flow Shop Scheduling Problem (FSP) for minimizing the maximum makespan with batch sequence dependent setup times will be noted as $1|ST_{sd}|C_{max}$.

The optimization criterion is defined in terms of one or several possible measures of performance. The most common performance measures in scheduling problems are, according to [7]:

- Minimize makespan (C_j) – The makespan represents the date of the end of the last operation to be processed. It is a criterion commonly used to measure the level of utilization of the machines.It can be measured by the difference between the instant the last job finishes its processing and the instant that the first job begins. Considering $C_{max} = \max C_j$.
  Thus, if C_j is the processing end time for job j, with $j = 1, \ldots, n$, then:
  $$C_{max} = \max_{j=1,\ldots,n} \{C_j\},$$
  and the objective function to use is:
  $$\min C_{max}$$

- Minimize flow time (F_j) – The flow time is equal to the sum of the end times, C_j, of all jobs. The objective function to use is:
  $$\min \sum_{j=1}^{n} C_j$$

- Minimize tardiness (T_j) – Tardiness, designate T_{max}, corresponds to the job with the largest difference, T_j, between the end time C_j and the due date d_j, when the job is completed after the due date. Considering $T_{max} = \max T_j$. In this context:
  $$T_{max} \geq T_j \quad j = 1, \ldots, n$$
  $$T_j \geq C_j - d_j \quad j = 1, \ldots, n$$
  $$T_j \geq 0$$
  that is,
  $$T_j = \max\{0; C_j - d_j\}$$
  Thus, T_j cannot be less than zero, as this would mean advance rather than delay. The objective function to use is:
  $$\min T_{max}$$

- Minimize earliness (E_j) – Earliness, in opposition to tardiness, is the amount of time that job j finished prior to its due date. The objective function to use is:
  $$E_j = \max_{j=1,\ldots,n} \{0; d_j - C_j\}$$

- Minimize lateness (L_j) – Lateness, L_j, is the difference between the end date of a job j and the end time. If positive indicates a delay in delivery and if negative means an anticipation of the delivery time. Considering $L_{max} = \max L_j$, the maximum lateness, the problem formulation can be presented as follows:

$$\min L_{max}$$

$$L_{max} \geq L_j \quad j = 1, \ldots, n$$

$$L_j = C_j - d_j \quad j = 1, \ldots, n$$

Note that the difference between lateness L_j and tardiness T_j is that lateness can be positive or negative indicating a delay or anticipation, where T_j cannot be negative, i.e. either delayed or not delayed.

- Minimize number of tardy jobs – Consists of determining the smallest sum of the delays. The objective function to use is:

$$\min \sum_{j=1}^{n} T_j$$

Pinedo [69] evidences that in all scheduling problems considered, the number of jobs and the number of machines are assumed to be finite. The number of jobs are denoted by n and the number of machines by m. Subscript i and j refer to specific job and machine, respectively, where the pair (i, j) refers to the processing step of job j $(j = 1, \ldots, n)$ on machine i $(i = 1, \ldots, m)$. The notations introduced in Table 2.4 are considered by Pinedo [69] as extremely important when talking about production scheduling.

Notation	**Description**	**Definition**
p_{ij}	Processing time	Represents the processing time of job j on machine i. The subscript i is omitted if the processing time of job j does not depend on the machine or if job j is only to be processed on one given machine.
r_j	Release date	This is the instant that job j can start its processing.
d_j	Deadline	Represents the instant that the job j should be completed. Completion of a job after its due date is allowed, but a penalty is incurred.
w_j	Weight	It is a priority factor denoting the importance of job j relative to the other jobs in the system.
$S_{jj'}$	Sequence–dependent setup times	Represents the sequence–dependent setup time that is incurred between the processing of jobs j and j'. Particularly, $S_{0j'}$ denotes the setup time for job j' if job j' is first in the sequence and S_{j0} the clean–up time after job j if job j is last in the sequence. If the setup time between jobs j and j' depends on the machine i, then the subscript is included, i.e., $s_{ijj'}$.

Table 2.4: Notations.

A widely used approach to real-world scheduling, where problems are often characterised by a highly complex and dynamic environment, are dispatching rules [68]. Dispatching rules consist of a well defined set of rules for deciding which job (or set of jobs) to process on a machine next. Advantages of dispatching rules include to be executed quickly, the ability to react to dynamic changes, their simple and intuitive nature, their ease of implementation within practical settings, and their flexibility to incorporate domain knowledge and expertise [68]. Jayamohan and Rajendran [45] classified the dispatching rules into five categories:

- Rules involving process times;
- Rules involving due–dates;
- Simple rules involving neither process times nor due–dates;
- Rules utilizing shop floor conditions;
- Rules involving two or more of the first four classes.

The amount and nature of this type of rules is quite high, so the main rules adapted in Holthaus and Rajendran [43], Slack *et al.* [82], Pinedo [69] can be defined as being:

- First In First Out (FIFO) – This is the simplest process scheduling in which, priority is given according to the sequence jobs arrive at the system, i.e. the first allocated job is executed first. This rule is often used as a bench–mark rule and it is easy to implement. [43].
- Earliest Due Date (EDD) – This is the earliest due date based rule in which, jobs are sequenced according to their due dates, i.e., priority is given to the execution of the most urgent orders in terms of due date. The goal is to reduce delays. Generally it gives good results under light load conditions, although the performance of this rule deteriorates under high loading levels.
- Shortest Processing Time (SPT) – This rule ignores the due date information of jobs. Priority is given according to the shortest total processing time, and is commonly used to minimize mean fl ow time and percentage of tardy jobs.
- Critical Ratio (CR) – Priority is given to the lowest critical ratio (time to due date divided by total remaining production time) among the waiting jobs. This is a dynamic rule that seeks to combine EDD with SPT, which only considers processing time.

Although these rules have been highly used with good results [20], none of these can guaranty the optimal solution [45].

2.2.1 Scheduling Classifications

The scheduling of production activities is accomplished through different strategies, and their application leads to processes with different structures and characteristics. Literature presents several scheduling classifications, according to the number of machines, jobs and operations. Graves [37] covering the general characteristics of both scheduling theory and scheduling practice, classifies production scheduling problems in three dimensions:

- Requirements generation – *open shop*, when requirements are generated directly by customers' orders or *closed shop*, when requirements are generated indirectly by inventory replenishment decisions.
- Processing complexity – *one stage–one processor* (one machine problem) when all jobs require one processing step which must be done on the production facility; *one stage–parallel processors*, which is similar to the previous except that each job requires a single processing step which may be performed on any of the parallel processors; *multistage–flow shop*, when all jobs are to be processed on the same set of facilities with an identical precedence ordering of the processing steps; *multistage–job shop*, when there are no constraints on the processing steps for a job, and alternative routings for a job are allowed.
- Scheduling criteria – *scheduling costs*, when concerned with fixed costs associated with production setups or changeovers, inventory holding costs; *scheduling performance*, such as the percentage of late jobs, the average or maximum tardiness for a set of jobs.

Furthermore, scheduling problems can be classified by production environments into static and dynamic problems in which the system parameters may be either deterministic or stochastic [37]. Static problem consist in determining a solution to a scheduling problem instance, i.e., characterizing the starting times of a known set of activities subject to deterministic resource and precedence constraints. While dynamic problem consist on reacting to some changes of the initial problem characteristics that generally occur in real time, either by performing a global rescheduling each time such a change occurs, or by adapting an initial static schedule to each disruption.

The two most important scheduling problem types are the job shop and the flow shop [69]. Figure 2.3 illustrates a comparison between the flow shop strategy with the job shop strategy. In the flow shop strategy all products run the same type of route (in the same direction), however there is no obligation to go through all machines. On the other hand, in job shop there is a route that has the opposite direction, i.e., routes are customized to products.

The Job Shop Scheduling Problem (JSP) may be described as given a set of n jobs, each composed of several operations that must be processed on m machines, to find the best schedule of the operations on the machines, taking account the precedence constraints. The objective of JSP is to minimize the makespan value C_{max} [32]. Commonly, it is assumed that: jobs are independent; machines and jobs are all continuously available; the setup times can be ignored or included into processing times; each machine can process at most one operation at a time and once an operation initiates processing on a given machine it must complete processing on that machine without interruption; the operations of a given job have to be processed in a given order.

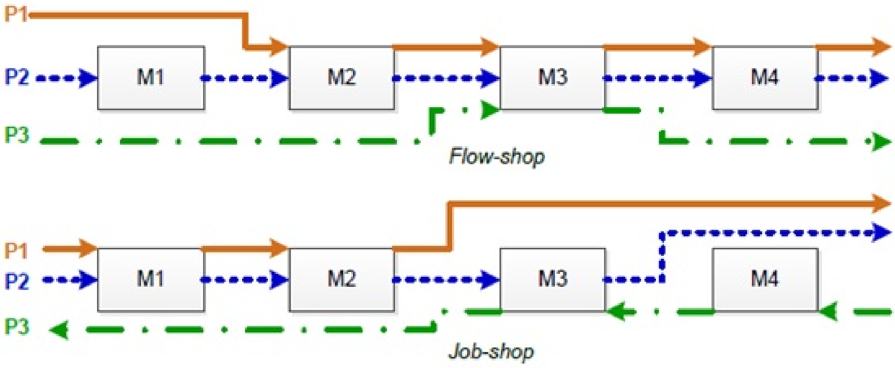


Figure 2.3: Flow Shop Scheduling *vs* Job Shop Scheduling

The Flow Shop Scheduling Problem (FSP) is a particular case of JSP which consists on scheduling n jobs with the same order and given processing times on m machines [15]. When the sequence of tasks is the same on all machines, it is knowed as a permutational flow shop. The classical FSP has as criterion the makespan minimization. Considering p_{ij} as the processing time of job j $(j = 1, \ldots, n)$ on machine i $(i = 1, \ldots, m)$, the goal is to find the desired schedule in which these n jobs should be processed on each of the m machines in order to minimize a well defined measure. This problem is known to be NP–Complete in the strong sense when $m \geq 3$ [69]. Bachman and Janiak [9] deal with the single machine maximum lateness minimization problem where the job processing times are given as a non–decreasing linear function. The authors consider that the processing time of job is a function dependent on its position in a sequence.

Gupta and Stafford Jr. [38] enumerates 21 assumptions for the traditional FSP originally conceived in Johnson paper in 1954 [46], concerning with characteristics of jobs, machines and operating polices of the shop. The assumptions concerning operation jobs are the following:

(J1) Each job is released to the shop at the beginning of the scheduling period.

(J2) Each job may have its own due date which is fixed and is not subject to change.

(J3) Each job is independent of each other.

(J4) Each job consists of specified operations, each of which is performed by only one machine.

(J5) Each job has a prescribed technological order which is the same for all jobs and is fixed.

(J6) Each job requires a known and finite processing time to be processed by various machines. This processing time includes transportation and setup times, if any, and is independent of preceding and succeeding jobs.

(J7) Each job is processed no more than once on any machine.

(J8) Each job may have to wait between machines and thus in process inventory is allowed.

The assumptions concerning machines are the following:

(M1) Each machine center consists of only one machine, i.e., the shop has only one machine of each type.

(M2) Each machine is initially idle at the beginning of the scheduling period.

(M3) Each machine in the shop operates independently of other machines and thus is capable of operating at its own maximum output rate.

(M4) Each machine can process at most one job at a time. This eliminates those machines that are designed to process several jobs simultaneously, like multi–spindle drill.

(M5) Each machine is continuously available for processing jobs throughout the scheduling period and there are no interruptions due to breakdowns, maintenance or other such causes.

The assumptions concerning operating policies are the following:

(P1) Each job is processed as early as possible. Thus, there is no intentional job waiting or machine idle time.

(P2) Each job is considered an indivisible entity even though it may be composed of a number of individual units.

(P3) Each job, once accepted, is processed to completion, i.e., no cancellation of jobs is permitted.

(P4) Each job, once started on a machine, is completed to its completion before another job can start on that machine, i.e., no preemptive priorities are assigned.

(P5) Each job is processed on no more than one machine at a time. (This is a result of assumptions J5 and P2).

(P6) Each machine is provided with adequate waiting space for allowing jobs to wait before starting their processing.

(P7) Each machine is fully allocated to the jobs under consideration for the entire scheduling period, i.e., machines are not used for any other purpose throughout the scheduling period.

(P8) Each machine processes jobs in the same sequence, i.e., no passing or overtaking of jobs is allowed.

The literature on FSP is extensive. Recent reviews are those by Hejazi and Saghafian [74], Minella *et al.* [60], Xu and Zhou [95], Sun *et al.* [85].

Gupta and Stafford [38] gived a summarized version of the evolution of this research over the period from 1955 to 2005. The current FSP that are being solved and the approaches taken to solve (optimally or approximately) are presented. The authors consider that although there are significant progress in FSP theory, it is not appreciable to solve practical FSP optimally and efficiently.

2.2.2 Scheduling with Setup Times/Costs

According to Allahverdi *et al.* [5], the first studies assume setup times as negligible or as part of job processing time. For the same authors, these assumptions simplify the problem analysis, but ultimately affect the quality of the solutions of many of the scheduling problems that need to explicitly address setup times. In fact, scheduling with setup times/costs plays a crucial role in industry. Allahverdi [3] remarks that the interest in scheduling problems with setup times/costs is growing, however, research on this problems is still less than 10 percent of the available research on scheduling problems.

A formal definition of setup times/costs is provided by Allahverdi and Soroush [6]. Namely, setup time can be defined as the time required to prepare the necessary resource (people, machines) to perform a task (operation, job). Setup cost is the cost to set up any resource used prior to the execution of a job. These authors present a comprehensive classification on scheduling with setup time/cost, and divide the scheduling problems with setup times/costs into two broad groups: Batch and Non–Batch. The big difference is that, in the first production is done in batches. So, the problem to analyze and solved is influenced by factors such as the existent number of batches, the size of each batch, or the number of batches jobs per batch produced. The second group (Non–Batch) can be regarded as a particular case of the first group in which batches have dimension one.

Batch and Non–Batch scheduling are also divided into two problem sub–types, depending on whether or not setup times are influenced by the sequence of jobs. Ulungu *et al.* [89] also divide setup time into two categories: sequence–independent and sequence–dependent. While in sequence–independent setup time is only dependent on the job that is being processed, in sequence–dependent the setup time depends on the job to be processed and on the one already processed. From these two categories the most difficult occurs when setup times are dependent on sequence, specifically when setup times are unsymmetrical. This means that the setup time from job j to j' is different when compared to the setup time from job j' to j.

The importance of explicitly treating setups times/costs in production scheduling has been addressed several times in literature. Allahverdi *et al.* [4] in 1999 make a comprehensive review of the literature on static scheduling problems involving setup times/costs, from mid–1960s to mid–1988s, covering about 200 papers. The authors also classify scheduling problems into batch and non–batch, sequence–independent and sequence–dependent setup. In addition, they categorize literature according to the shop environments on single machine, parallel machines, flow shops, and job shops. Gaps on existing research, including in the underlying theoretical underpinnings and in the treatment of multiple objectives are identified. Further, the author emphasize that few papers have treated the many industrial applications having sequence–dependent setups.

Allahverdi *et al.* [5] in 2008 also address scheduling problems with setup times/costs, covered about 300 papers published from mid–1998s to mid–2006s. The authors, further categorize scheduling literature on models with setup times/costs according to shop environments, including single machine, parallel machines, flow shop, no–wait flow shop, flexible flow shop, job shop, open shop, and others. The common solution methods that are used are branch and bound algorithms, dynamic programming algorithms, mathematical programming formulations, heuristics and meta–heuristics.

In 2015, Allahverdi [3], once again conducts a survey on these problems, involving static, dynamic, deterministic and stochastic problems for different shop environments with setup times/costs, covering about 500 papers published from mid–2006 to the end of 2014. It further classifies the problems as family and non–family, where each family is a set of jobs that has similar characteristics in terms of setups, tooling and operation sequence. There is a negligible setup time/cost to change from one job to another with the same family but a major setup time/cost is needed between job families.

2.2.3 Schematic Representation of Scheduling Problems

The Gantt chart has been one of the most important tools for visual schematic representation of schedules in industrial settings [67]. The horizontal axis of the Gantt chart represents the time and the vertical axis represents the processing units. A rectangle on the chart stands for a task. Its location and length indicate when the task is started and how long the task is executed.

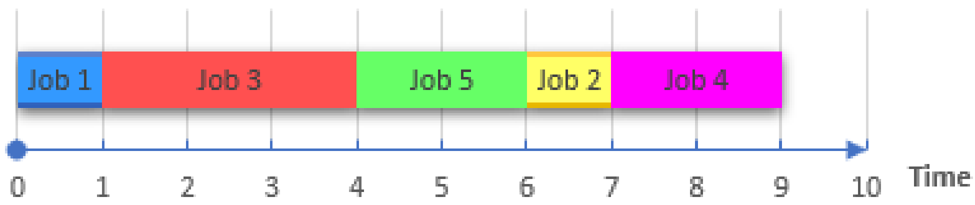


Figure 2.4: Illustrative example of a Gantt chart.

A Gantt chart is pictured in Figure 2.4, for one machine, where each order is represented by a rectangle that is labeled with the job number. For example, Job 1 is processed on machine followed by jobs Job 3, Job 5, Job 2 and Job 4.

Panwalkar and Koulamas [67] point others schematics to depict a schedule, namely: cartesian coordinates, disjunctive graph model and the string analog schematic using strings, weights and pulleys for scheduling and sequencing problems involving general precedence relations. However, Gantt chart are by far the most used schematic representation of scheduling solutions.

2.2.4 Modeling and Resolution of Scheduling Problem

Optimization refers to the process of finding optimal values for the decision variables of a given system from all the feasible values which maximize or minimize the output, and mathematical optimization is the process of finding the best solution to a problem from a set of available alternatives [64]. In literature there are several classifications on optimization problems [48]. Nguyen *et al.* [64] present several aspects that need to be considered during the optimization:

- Number of decision variables – Optimization problems can be classified as *one–dimensional* or *multi–dimensional* optimization, depending on the number of decision variables.
- Nature of the decision variables – Decision variables can be independent or mutually dependent. Optimization problems can be stated as *static* or *dynamic* if decision variables are independent, and *deterministic* if decision variables are subject to small uncertainty or have no uncertainty. Opposite, optimization problems can be seen as probabilistic optimization if decision variables are subject to uncertainty.
- Types of decision variables – Variables can be *continuous* if it can assume any real value in a range, and *discrete* if it can assume only integer values or discrete values, or both. The latter is referred to as MILP.
- Number of objective functions – Problems can be classified as *single–objective* if only one objective is considered, or *multi–objective* if more than one objective is considered.
- Natures of the objective function – Different optimization techniques can be established depending on whether the objective function is linear or nonlinear, convex or non–convex, uni–modal or multi–modal, differentiable or non–differentiable, continuous or discontinuous, and computationally expensive or in–expensive. These result in *linear* and *nonlinear programming*, *convex* and *non-convex optimization*, *derivative–based* and *derivative–free optimization* methods, *heuristic* and *meta–heuristic optimization* methods, *simulation–based* and *surrogate–based optimization.*
- Presence of constraints and constraint nature – Optimization can be classified as *constrained* or *unconstrained problems* based on the presence of constraints which define the set of feasible solutions within a larger search-space. Two major types of constraints are equality or inequality. A constraints function may have similar attributes to those of objective functions, and can be separable or inseparable.
- Problem domains – *Multi-disciplinary optimization* or *single-domain optimization.*

Scheduling problems are very challenging from the mathematical point of view, and are known to be very difficult to model and solve in an efficient way, being that Mathematical programming (MP) methods are the most widely reported methods for scheduling industrial problems [83]. MP involves the development of mathematical models that represent real world situations and can be used to determine the optimal [94].

Optimization problems are generally classified according to the complexity of the functions involved in the problem. The most relevant subareas are as follows:

- Nonlinear Programming (NLP) – In this kinds of problems one or more of the constraints and/or the objective function are nonlinear functions. When the production processes under consideration possess significant nonlinear characteristics as often in real time production scheduling or in control problems, it is sometimes necessary to make linear approximations to the nonlinear functions [83].

- Linear Programming (LP) – This is the simplest case, with the easiest resolution. The objective function and all constraints are linear functions. The exact form of these constraints may differ from one problem to another. A problem LP can be transformed into the following standard form [54]:

$$\min \quad c_1 x_1 + c_2 x_2 + ... + c_n x_n \tag{2.1}$$

$$\text{s.t.} \quad a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \tag{2.2}$$

$$a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2 \tag{2.3}$$

$$\vdots \qquad \vdots \tag{2.4}$$

$$a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n = b_m \tag{2.5}$$

$$x_1, x_2, ..., x_n \geq 0. \tag{2.6}$$

  where the b_i's, c_i's and a_{ij}'s are fixed real constants with $b_i \geq 0$, and the x_i's are real numbers to be determined.

- Mixed Integer Linear Programming (MILP) – Involves real variables usually related to operation and discrete variables (integers or binary), such as decision variables, allocation and decision, or any other quantity that requires modeling with discrete variables. Scheduling problems usually involve discrete decisions such as equipment assignment and task allocation over time and hence MILP formulation is used [83].

- Mixed Integer Nonlinear Programming (MINLP) – This is the area of optimization that addresses nonlinear problems with continuous and integer variables, which have characteristics of the MIP and NLP problems already described. MINLP has proven to be a powerful tool for modeling. At the same time, it combines algorithmic design challenges from combinatorial and nonlinear optimization [79].

Three different very common ways of formulating the scheduling problems are next presented, distinguished by the type of binary variables used. The three types of variables are the proposed by Wagner [92], Bowman [13] or Manne [56]. These variables are called precedence variables [10].

Wagner [92], whose work focused on minimizing makespan in three machine FSP, presents the following key binary variables:

- $x_{jki} \in \{0, 1\}$, that is 1 if job j is scheduled at position k for being processing on machine i, and 0 otherwise.

Bowman [13], whose work focused on minimizing makespan in the JSP, the key binary variables are:

- $x_{jit} \in \{0,1\}$, being 1 if job j is processed by machine i at discrete time interval t, and 0 otherwise.

Manne [56], who also focused on the makespan problem for the JSP, presented the binary variables:

- $x_{jj'i} \in \{0,1\}$, 1 if job j precedes order j' on machine i (not necessarily immediately before), and 0 otherwise.

Many algorithms for solving optimization problems involve a large number of decision choices and algorithms' specific parameters that need to be carefully set to reach their best performance. Maximizing the performance of these algorithms may involve the proper setting of tens to hundreds of parameters. A good introduction to OR models for scheduling is given by Pinedo [69] where a review on different models and solution approaches is presented.

Next several approaches for solving and modelling scheduling problems are described. According to the nature of the solution approach, these can be classified as: exact algorithms, heuristics, hybrid approaches and simulation/decision support system (DSS) [75] (see Figure 2.5).

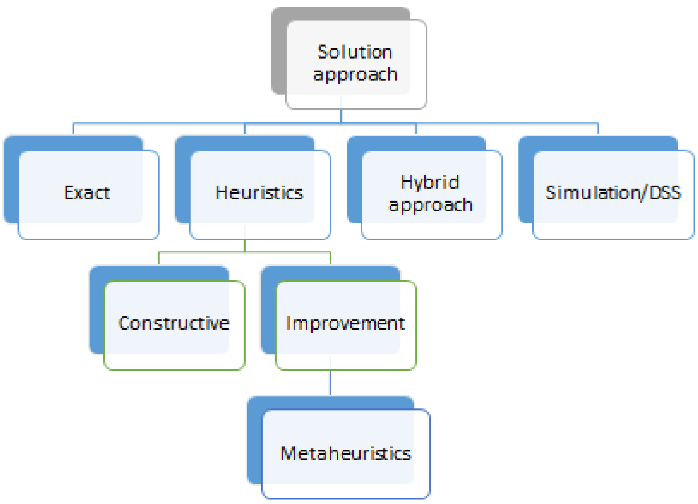


Figure 2.5: Classification proposed according to the solution approach (Adapted from [75]).

Exact algorithms propose a method to obtain the optimal solution. The heuristics can be classified either as constructive or improvement heuristics, in which the first builds a feasible schedule from scratch, step by step, and the latter try to improve a previously generated schedule by normally applying some form of specific problem knowledge [76]. In the improvement procedures category there are the metaheuristic methodologies. Hybrid approaches propose a procedure that combines two or more approaches to obtain a solution. The simulation/DSS approach analyse the performance of systems restricted by certain rules or constraints.

Exact Approaches

Exact algorithms have been developed with the objective of optimizing certain well defined criterion find for problems involving a limited number of jobs and machines [84]. Branch and Bound (B&B) is the most common technique to solve FSP and differ in terms of the lower bounds used and the branching strategies. McMahon and Burton [58] applied the B&B technique to three machines FSP with the objective to minimize makespan.

Lee and Kim [52] proposed a B&B algorithm for the problem with two stages and any number of identical parallel machines at the second stage, with the objective the minimization of total tardiness. Choi and Lee [16] have studied a two–stage problem with multiple identical parallel machines at each stage for the minimization of tardy jobs. The authors propose a B&B method as well as some ad–hoc heuristics.

Rabadi *et al.* [70] addressed a single machine early/tardy (E/T) scheduling problem with setup times and sequence–dependent, whose objective minimize the total amount of earliness and tardiness. B&B algorithm is developed to obtain optimal solutions for the problem and will be shown that this algorithm solved problems three times larger than what has been reported in the literature.

Exact approaches have demonstrating limited results for medium or large problems and are specially sensible to the number of jobs and/or machines as can be observed in the work of Ladhari and Haouari [50] and Ruiz and Maroto [76]. However, with the development of computational resources and algorithms, competitive exact approaches are already begin to verified. Recently Nesello *et al.* [63] proposed an exact algorithm based on the iterative solution of three alternative arc–time–indexed models to solve the single machine scheduling problem with periodic maintenance and sequence–dependent setup times with aim of minimizing the makespan.

Wang and Ye [93] presented an order acceptance and scheduling problem on unrelated parallel machines, where a B&B based formulation has been developed. The branching determines how many jobs should be accepted followed by addressing of the unrelated parallel machine scheduling sub problem with the minimization of the total weighted tardiness. Two Mixed Integer Programming (MIP) models were formulated and the results demonstrate the efficiency of the enhancement techniques for the formulations, as well as the effectiveness and efficiency of the B&B based formulation algorithm.

Heuristic Approaches

A major drawback of the above techniques, that has been pointed in literature, is that there is a limit on the size of the problems that can be solved using such techniques in a reasonable time. On the other hand, it is argued that heuristic algorithms can solve large FSP instances and are able to find high quality solutions in short computation time. Several such algorithms have been proposed and can be classified into the following categories: (i) construction methods; (ii) improvement heuristics; (iii) metaheuristics [44].

Construction methods build a feasible schedule from scratch [61]. The final sequence is obtained by integrating jobs (usually in an iterative manner) on an incomplete sequence [24]. An example is Johnson's algorithm[46]. Among the existing heuristics, the insertion method of Nawaz *et al.* in 1983 [62], referred to as NEH (Nawaz–Enscore–Ham) is considered one of the best heuristics for the FSP (see work Ruiz and Maroto [76]). In according Ribas *et al.* [75] the NEH procedure can be divided into two steps: the generation of an initial order for the jobs applying the largest processing time rule and the iterative insertion of jobs in a partial sequence in accordance with the initial order obtained in the first step. More recent methods, such as those of Dong *et al.* [18], Kalczynski and Kamburowski [47], Rad *et al.* [71] have demonstrated NEH outperforming algorithms.

Improvement heuristics are usually composed of two phases. The first is a construction phase where a complete sequence is computed [24]. In the second phase is improved the previously generated schedule, typically by specific knowledge of the problem [41, 84].

Metaheuristics are defined as search methodologies or algorithms of a solution space for a near optimal solution [86]. The initial solution for the metaheuristics is provided by constructive and/or improvement heuristics, because it requires a fast initial solution to start the procedure [76]. The quality of this initial solution can influence the performance of the metaheuristic [24]. This type of algorithm basically combines heuristic methods in higher level frameworks. These methods were first introduced by Glover [31]. An advantage of metaheuristics over simple heuristics is solution robustness. However metaheuristics are usually more difficult to implement and adapt since in order to obtain good results special information about the problem must be provided [102]. Nearly all metaheuristic algorithms share the same following characteristics: are nature–inspired; make use of random variables; do not require substantial gradient information; and have several parameters that need to be fitted to the problem. Factors that affect the quality of the search result include the type of problem model, applied metaheuristic algorithm, size of the search space, time allocated for the search, and computer capacity. According Boussaïd *et. al* [12] metaheuristics can be divided in single point search working only on one solution at a time and population based working on multiple solutions at the same time.

Exact algorithms did not prove efficient in solving real life instances of this problem until recently, in way more papers on heuristic and metaheuristic approaches can be found in the literature [22]. Examples of matheuristics are the algorithms of Simulated Annealing (SA), Tabu Search (TS), Genetic Algorithms (GA), Ant Colony Algorithms (ACO), Particle Swarm Optimization (PSO) and Greedy Randomized Adaptive Search Procedure (GRASP).

Simulated Annealing (SA) is an evolutionary metaheuristic approach which is inspired from the physical process of annealing, in which a solid is heated to a liquid state and, when cooled sufficiently slowly, takes up the configuration with minimal inner energy. The principles of SA is first to change a current solution randomly to obtain other various solutions (diversification phase). An example of SA algorithms is the are presented by Osman and Potts [66]. This algorithm is fairly simple in the sense that it uses a constant temperature and an insertion neighborhood along with a FIFO dispatching rule initialization procedure. Ying *et al.* [98] point out that the basic SA algorithm has a major weakness, the number of neighborhood solutions generated grows enormously as the number of jobs increases, meaning that finding an efficient solution requires a large computational burden. In this view the authors proposes a restricted simulated annealing, which demonstrate superior results with respect to the existing literature at the moment. A paralleled version of a SA is presented in [25].

The Tabu Search (TS) was originally developed by Glover in 1989 [31]. This technique aims to guide the search by exploring the solution space of a problem beyond local optimality. Ben–Daya and Al–Fawzan in 1998 [11] implemented a TS algorithm with some extra features such as intensification and diversification schemes that provides better moves in the TS process. Grabowski and Wodecki [33] deals with a classic FSP with makespan criterion, and to propose a new very fast local search procedure based on a tabu search approach. The results indicate that the algorithm proposed solves the flow shop instances with high accuracy in a very short time.

Genetic Algorithms (GA) are adaptive methods based on the genetic process of biological organisms, since the original idea proposed by Holland in 1975 [42]. Unlike the conventional optimization methods, a GA maintains for each generation a set of potential solutions a population and encodes the factors of a problem by chromosomes, where each gene represents a feature. The combinations of genes evolved through genetic operators so that the chromosomes would approach the optimal solution from generation to generation. There are three genetic operators: reproduction, crossover, and mutation. Reproduction is a process which forms a new population of strings by selecting strings in the old population based on their fitness values. Crossover is used as the main genetic operator on which the performance of a GA is extremely dependent. The mutation operator plays a very substantial role in GA and helps to maintain the diversity in the population to prevent premature convergence. Chen *et al.* [14] generated a GA based heuristic for FSP with makespan as criterion. Ruiz and Maroto [77] employed a GA to minimize makespan on a m–stage problem with unrelated parallel machines, sequence–dependent setup times and machine eligibility. Zhang *et al.* [100] proposed an hybrid GA for permutation FSP, with total flow time minimization. GA for the unrelated parallel machine scheduling problem in which machine and job sequence–dependent setup times, presented by Vallada and Ruiz [90] was able to give even smaller deviations from best known solutions. Another GA for this problem was proposed by Yilmaz Eroglu *et al.* [97].

The Ant Colony Algorithms (ACO) algorithm described is competitive and presents advantage for larger problems, when compared with a GA, a SA, a local search method and a branch and bound algorithm. Ying and Lin [99] proposed an ACO metaheuristic for the multiprocessor task problem with precedence relationships. While, Rajendran and Ziegler [72] investigate scheduling in permutation flow shops by using two ACO algorithms, with the objective of minimizing the makespan.
Particle Swarm Optimization (PSO) was first proposed by Kennedy and Eberhart in 1995 [49]. This is one of the most recent and hopeful evolutionary metaheuristics, which is inspired by adaptation of a natural system based on the metaphor of social communication and interaction [44]. The challenge has

been to employ the algorithm to combinatorial problems, thus new PSO variants have been developed. A PSO metaheuristic was proposed by Tseng and Liao [88] for the scheduling of multiprocessor tasks in a multistage hybrid flow shop with makespan minimization. The proposed PSO algorithm has a new encoding scheme, an implementation of the best velocity equation and neighbourhood topology among several different variants, and an effective incorporation of local search.
Feo and Resende [23] proposed Greedy Randomized Adaptive Search Procedure (GRASP). This is a memory–less multi–start metaheuristic for combinatorial optimization problems. There are two phases within each GRASP iteration: (i) construction and (ii) local search [12]. The construction step builds a feasible solution using a randomized greedy heuristic. In the second phase, this solution is used as the initial solution for a local search procedure. After a given number of iterations, the GRASP algorithm terminates and the best solution found is returned. Resende *et al.* [73] devise several versions of an hybrid algorithm based on GRASP and path relinking methodologies for the max–min diversity problem.

2.2.5 Scheduling in the Aluminium Industry

This last subsection aims at referencing scheduling papers that have been developed for the aluminium industry.

Duman *et al.* [19] presents a study of a problem scheduling casting lines of an aluminium casting and processing plant in a parallel machine environment with sequence–dependent setups. A mathematical formulation for scheduling jobs to minimize the total setup time while achieving workload balance between the production lines is presented. A four–step algorithm for finding good solutions in a reasonable amount of time is proposed. In this process, a set of asymmetric travelling salesman problems is followed by a pairwise exchange heuristic. The solutions presented are better compared to those verified by the company in study.

Gravel *et al.* [35] addresses a GA for the solution of an industrial scheduling problem of n orders with sequence–dependent setup times on m parallel machines in aluminium foundry. Multiple criteria are considered, namely: meeting due dates, number and duration of required setups and metal flow. The algorithm presents provides a better solution than the manual process.

Schwindt and Trautmann [80] address a scheduling problem arising in the context of a rolling ingots production, that consists of computing a feasible production schedule with minimal makespan. A B&B solution procedure is presented in a two–phase approach. In the first phase, the resource requirements of the changeover operations are disregard and, once a feasible schedule for the latter relaxation has been found, the job sequences on the resources which are changed over between consecutive jobs are fixed. In the second phase, a feasible schedule are determined by resuming the branching process of the first phase and resolving the conflicts arising from the resource requirements of the changeover operations. The algorithm has been implemented as a beam search heuristic enumerating alternative sets of precedence relationships.

Gagne *et al.* [30] proposed an ACO algorithm for the problem non–batch sequence–dependent setup times in a casting centre and their objective is to minimize total tardiness. The authors showed that it performs competitively with the best results of Tan *et al.* [87], in terms of solution quality while taking much less computational time, presenting a certain advantage for larger problems.

Xu *et al.* [96] focused on minimizing the energy consumption and have developed a hybrid PSO for a LP of aluminium industry continuous casting and rolling production. Simulation results demonstrate the effectiveness of both the models and the optimization method, and that the optimal scheduling scheme with minimum energy consumption can be obtained efficiently.

Esteban and Penya [21] applying a set of GA, SA and backtracking algorithms to the foundry scheduling problem. The objective is to decrease the energy spend of foundries. The scheduling of the orders and the optimisation of the foundry process it self is carried out based on expert's knowledge acquired over years of experience, neglecting issues such as, energy costs and hourly electricity prices.

Gravel *et al.* [36] present an ACO metaheuristic fin order to treat the multiobjective industrial problem of n orders, with sequence-dependent setup times on one machine, taking into account the technological and logical constraints on equipment and the management of the supply of liquid metal. The objectives are the minimization of unused capacity, minimization the total tardiness of the set of orders, minimization of the total number of drainings required when changing alloys, and minimization the total unused vehicle capacity. The results could not be compared with an ex ante situation because was a new continuous casting facility with no previous work history, however the solutions presented of a better overall quality, than those found by single objective optimization.

Chapter 3

The Company under Study

This section describes the company where the present work was developed. A brief presentation is made and the production process is described. The company provided us with two datasets and they are presented. The first dataset with the extrusion process to first half of 2018 and the second concerned the information the order of its customers for the month of May 2018.

3.1 Company Characterization and Problem Description

The present work is motivated by an industrial challenge proposed by ADLA – Aluminium Extrusion, S.A., a company that operates in the aluminium market, located in the region of Celorico de Basto. The company's core business is the extrusion and treatment of aluminium profiles (Figure 3.1). Innovation, quality, technology and environment are at its core values. The company considers internationalization a vital aspect for its survival. Investments in the production of new goods and services or significant improvements in production, contemplating concerns with principles of ecodesign, efficient use of resources, and durability of materials and products.

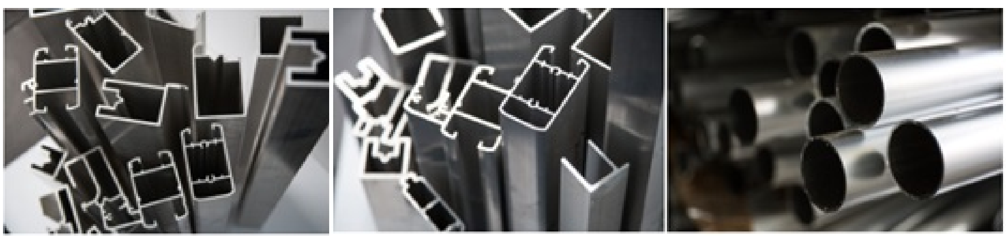


Figure 3.1: Type of aluminium profiles production in company. Source: [1]

At the company, as mentioned before, aluminium profiles are produced through direct extrusion (Figure 2.1). The raw material used is supplied in the form of aluminium alloy cylinders, called a billet. In a phase prior to extrusion, the billet, in order to have the necessary plasticity, is pre-heated gradually in an oven (Figure 3.3, left) and then placed in the container.

The extrusion press receives the billet on one side and the die opposite side, which is also preheated to avoid thermal shocks. Next, in the process of extrusion by the action of the hydraulic force of the press, the billet is pressed against the die, causing the change of the aluminium form and the formation of the profile. The dies (Figure 3.2) are steel blocks with one or more holes, solid and resistant, whose inscribed configuration gives the profile the desired shape. If the profile has to be changed or the die has already reached its maximum use time, the process is interrupted and the die is changed.

Figure 3.2: Type of dies. Source: [1]

Because extrusion is a continuous process, a large automated circular saw cuts the billet to the required length as it is produced. After the pressing operation, profile is quickly cooled (with air or water) along a lane, the tip of the profile being held by a puller, which pulls it to the end of the lane. It is then subjected to a cold stretching work to relieve any tension and obtain the desired straightness and cut (Figure 3.3, right).

Finally, the profile under goes a heat treatment of tempering to give it the specified mechanical properties. In the company under study only thermal finishing is carried out, and the surface treatment is outsourcing, with an average time of five days to be ready. The surface treatment can be anodizing or lacquering, according to the customers' requirements.

Figure 3.3: Production of aluminium profiles in the company. Source: [1].

Summarising, aluminium extrusion is a continuous process and is essentially divided into four moments: (i) pressing operation – extrusion, (ii) stretching, (iii) cutting and (iv) heat treatment. Figure 3.4 presents the entire process.

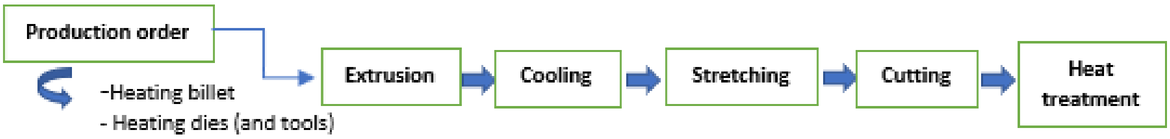


Figure 3.4: Production process in company.

The customers order are quite diversified. According characteristics, these orders are scheduled based on the alloy type, the die, the order quantity, the delivery date, and others. The production is planning every week. The focus of the current study is to reduce scrap at the moment of extrusion, through minimization of setup times, occurred by die change. Thus, it is intended to optimize production scheduling in order to increase productivity, maintaining quality and reducing delivery times.

According to Saha [78], productivity is defined by the quantity of good extrusion produced per unit time. There are a number of ways to maximize productivity, namely: (i) minimize the amount of avoidable scrap per each billet run for the same press speed, billet size, and runout length; (ii) maximize the billet length to best utilize press capacity and runout table length; (iii) optimize the speed, predicting/controlling the variation of the exit temperature, that rise during extrusion.

Process scrap, can originate from high purity and specification requirements, quality problems, defects, over–ordering, mismatches between batch and order volumes, subtractive processing, scalping and trimming during processing and start–up losses [39]. Reducing the generation of process scrap can provide both energy and economic benefits.

Both the company under study and the literature point out that die changes generates scrap [39]. In Figure 3.4 the production process in company can be seen. The die changes occurs in the extrusion stage. These changes occur whenever the type of profile to be produced changes, or when the die exceeds its use time. Thus, a good production scheduling that minimizes die changes will allow significant scrap reduction. This is where our study is centered.

3.2 Datasets

The company provided two datasets. The first dataset, named Production process dataset, concerns the extrusion process and contains information collected between January 1st and June 30th, 2018. The second dataset, named Customers' orders dataset, concerns information of the customers' orders for the month of May 2018, with a total of 498 orders. Other information concerning production was also provided by the company.

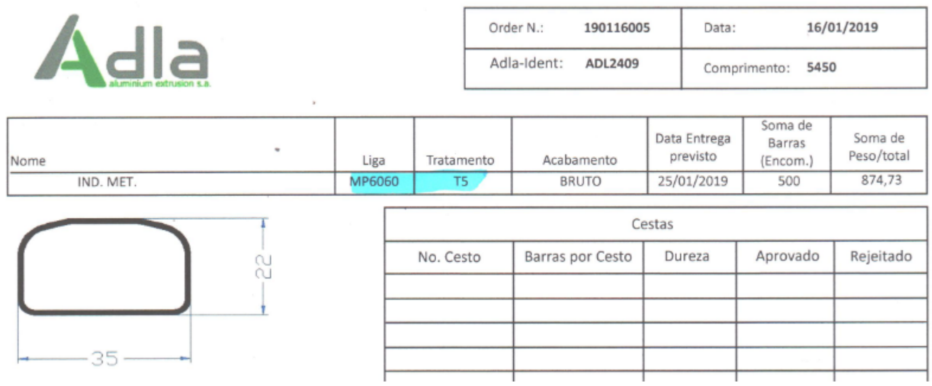
The dataset contains 42 827 observations, each corresponding to an extruded billet, and is also used in work of Ferrás *et al.* [26]. This dataset contains 65 variables, 62 of which are quantitative (discrete and continuous) variables and three are nominal qualitative. These variables can be divided into three categories: pre–extrusion (PR), extrusion (E) and post–extrusion (PO), depending on the phase of the process in which they are included, except the variable order and the production date. In the present study, and according to the literature review in Section 2.1, only some of the extrusion and post–extrusion variables are considered for the developed scheduling model. However, all interfere with the quality of the extruded profile. The variables considered for the development of model presented in Section 4.2 are depicted in Table 3.1.

	Acronym	Description	Classification
Extrusion	E_D	Die	Qualitative nominal
	E_To	Tool	Qualitative nominal
	E_NH	Number of holes	Quantitative discrete
	E_t	Extrusion time (s)	Quantitative continuous
Post_Extrusion	PO_LB	Length bars (mm)	Quantitative continuous
	PO_NB	Number of bars	Quantitative discrete
	PO_WB	Weight bars (kg)	Quantitative continuous

Table 3.1: Extrusion process dataset variables used.

The E_D variable corresponds to the die used, and E_To represents the tool that was used. The E_NH variable indicates the number of holes in the die, and it is through these holes that the amount of kg of extruded product is determined. Finally, the E_t variable indicates the time each billet took to be extruded. For the post–extrusion variables, the PO_LB indicates the length of each bar, PO_NB the number of bars, and PO_WB the total weight of the bar of each extruded billet.

The variables E_t and PO_WB, will be used for estimate the processing time for each order (see Section 4.3.2).


Adla aluminium extrusion s.a.

Order N.:	190116005	Data:	16/01/2019
Adla-Ident:	ADL2409	Comprimento:	5450

Nome	Liga	Tratamento	Acabamento	Data Entrega previsto	Soma de Barras (Encom.)	Soma de Peso/total
IND. MET.	MP6060	T5	BRUTO	25/01/2019	500	874,73

Cestas				
No. Cesto	Barras por Cesto	Dureza	Aprovado	Rejeitado

Figure 3.5: Example of a customers' order [1].

The Customers' orders dataset, contains information on each customers' order (see Figure 3.5) and includes the following specifications:

- *Order N* – code that the company uses for identifying each order.
- *Data* – day that the company has received the order.
- *Adla-Ident* – the identification of the used die – E_D.
- *Comprimento* – length of the bars to be produced, in mm – LB.
- *Nome* – Name of customer which is omitted for confidentiality reasons.
- *Liga* – alloy that is used.
- *Tratamento* – finishing treatment that is used.
- *Acabamento* – description of the type of finishing that is used and can be surface or thermal. As mentioned in Section 3.1, the company only carries out thermal treatment and therefore it has to recur to external companies for surface finishing.
- *Data Entrega previsto* – date of delivery of the order, i.e. deadline of the order.
- *Soma de barras (Encom.)* – total number of bars to be produced – NB.
- *Soma de Peso/total* – total weight of the bars, in kg – c_j.
- Drawing of the profile geometry, including the measurements.

From the data provided by the company, further information was computed, namely the release date and the deadline, whose calculation is presented in Section 4.3.1. In Figure 3.6 it is shown an example of an order received on 02/05/2018. The order has a deadline of 44 days to be delivered to the customer.

The order characteristics are given, namely, the desired profile type the die to be used has the code ADL4581, the order consists of 150 bars with a length of 4 meters, whose total weight is 208.8 kg.

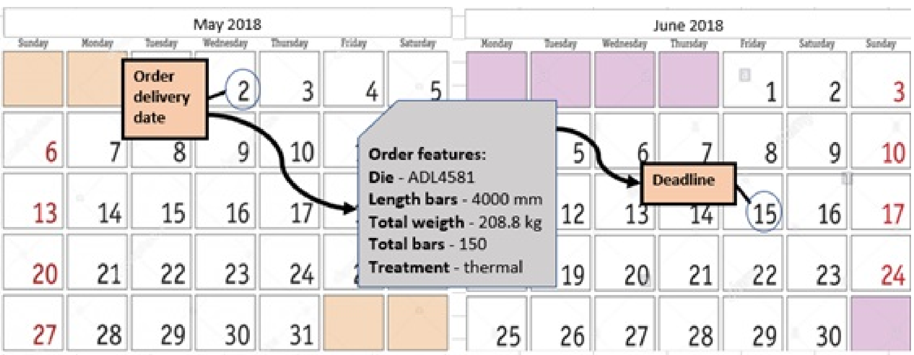


Figure 3.6: Example of an order received on 02/05/2018.

Furthermore, for performing the production scheduling more information was provided by the production manager, namely:

- Raw material dimensions – maximum 7 m.
- When to lubricate the material – every 30 billets, on average.
- Number of dies and tools available – more than 1 500.
- Maximum and minimum allowed usage time for each die – less then 25 hours but depends on the nutrition (steel hardening process).
- Maximum processing weight (kg) with a given die – 30 000 kg.
- Minimum processing weight (kg) with die – 250 kg.
- Time to exchange a die – 1.5 minutes, on average.
- Maximum length of the bars after extrusion and before cutting – 56 m.
- Time for surface treatment – 5 days.
- Time specified by the company for the order to reach production – 60 minutes.
- Time specified by the company to perform other tasks such as, cutting the bars, packing, cleaning, surface treatment and thermal treatment – 1 day, with to 5 extra days if surface treatment.
- Expected stop periods – Sundays, holidays and scheduled maintenance.

To know the exact number of copies (E_To) of the dies used for producing the orders of May, information from the Production process dataset was used.

These datasets will be study in detail in the next section.

Chapter 4

Research Methodologies

In this chapter the research methodologies are presented. More specifically, a general data analysis of the two data sets that were provided by the company (presented in Section 3.2) is made. The developed optimization model is presented, namely the parameters, decision variables, objective function and their constraints, in Section 4.2. Based on historical data of the extrusion process, some unknown parameters of the model are computed and others are estimated, in Section 4.3.

4.1 General Datasets Analysis

4.1.1 Production Process Dataset

Regarding the Production process dataset it can be seen from the variable (E_D), which represents the die used for the extrusion of each billet, that a total of 3 101 orders were produced, where 764 different dies were used. Analyzing the tools of each die (E_To), in the dataset can be seen that the tools can range from 1 to 27. According to company and confirmed in the dataset the maximum of number of holes (E_NH) is 12. The extrusion time (E_t) of each billet ranges from 56 to 1 200 seconds, and it can be seen that 75% of cases do not have a time greater than 195 seconds (can see in Table 4.1).

Variable	Min	Q1	Median	Mean	Q3	Max
E_t	56.00	143.00	167.00	173.93	195.00	1 200.00
PO_LB	2 050.00	6 000.00	6 500.00	6 542.00	6 500.00	12 050.00
PO_NB	0.00	5.00	14.00	17.24	20.00	228.00
PO_WB	0.00	59.00	66.00	63.47	72.00	88.00

Table 4.1: Statistics of the extrusion variables present in Table 3.1.

The length bars obtained in the extrusion of each billet (PO_LB) has bellows between 2 050 and 12 250 with mean value 6 542, the corresponding number of bars produced is between 0 and 228, and bars weight PO_WB is between 0 and 88 kg.

4.1.2 Customers' Orders Dataset

In this section a characterization of the Customers' orders dataset is presented.

A large diversity in the order of customers is observed (Table 4.2). In fact, the lengths of each bar to be produced (LB) ranges from 1 875 mm to 10 050 mm, and half of the ordered bars have lengths bellow 6 500 mm. Also in terms of the number of bars to be produced (NB) there is a large variability, ranging from 14 to 20 165 bars, while half of the orders correspond to 135 bars or less. The total length of bars (TLB) of each order varies from 60 to 120 969.80 meters and the total weight (c_j) of the order varies between 30.75 kg and 30 000.52 kg. Half of orders have 750 meters or less, and a total weight bellow 488.87 kg.

Variable	Min	1st Q	Median	Mean	3rd Q	Max
LB	1 875.00	5 594.00	6 500.00	5 903.00	6 500.00	10 050.00
NB	14.00	53.88	135.00	278.17	299.50	20 165.00
TLB	60.00	343.33	750.00	1 626.80	1 749.00	120 969.80
c_j	30.75	254.59	488.87	908.95	999.50	30 000.52

Table 4.2: Descriptive statistics of the customers' orders.

The variable TLB stands for the total length of bars, in meters, to each order, by the equation (4.1):

$$TLB = NB \times LB \tag{4.1}$$

The month of May was characterized by a total of 498 orders from 67 distinct customers. A total of 305 different profiles are to be produced. Therefore, there is a wide variety of profiles to produce, which translates into a great complexity in the production planning. Figure 4.1 shows that the day with less orders is Friday (67 orders, or 13%). On the other hand, Thursday is the day with more orders, in a total of 134 (27%).

Upon the reception of an order, a delivery period is established which is expressed by the week of the year (between week 1 and 52). For the purposes of the present analysis the day of delivery is Friday, and consequently the deadline (last day of delivery time) can be computed.

The number of days to produce the orders, i.e., the delivery time, varies between 5 and 98 days, 50% of which have between 38 and 44 days for delivery. Only 25% of orders have delivery deadlines exceeding 44 days. Given that, on average the company has approximately 40 days to produce the orders, lead us to conclude that the deadlines are relatively small. This information can be observed in Table 4.3.

Analysing the medians of delivery time per day of the week (Table 4.3), can be concluded that the day of week that presents smaller delivery time is Thursday while Friday is the one with the highest value. For average values the minimum occurs on Tuesday and the maximum on Friday.

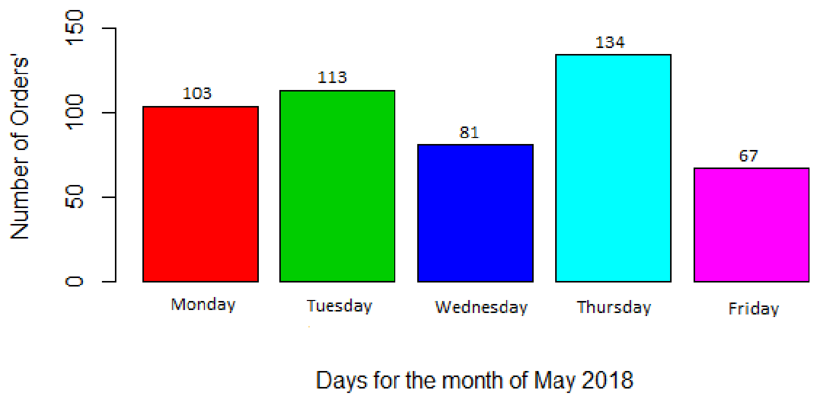


Figure 4.1: Total number of orders' per day of week.

Delivery time	Min	1st Q	Median	Mean	3rd Q	Max
All days of week	5.00	38.00	40.00	39.77	44.00	98.00
Monday	5.00	41.00	41.00	38.96	41.00	53.00
Tuesday	16.00	34.00	40.00	37.66	46.00	76.00
Wednesday	33.00	39.00	39.00	41.74	45.00	51.00
Thursday	8.00	38.00	38.00	39.61	38.00	98.00
Friday	19.00	43.00	43.00	42.46	49.00	49.00

Table 4.3: Delivery time of the orders statistics of May 2018.

The violin plot in Figure 4.2 suggests that most of the orders around 40 days. The weekday with the highest variability is the Thursday. The day with the smallest variability is the Wednesday.

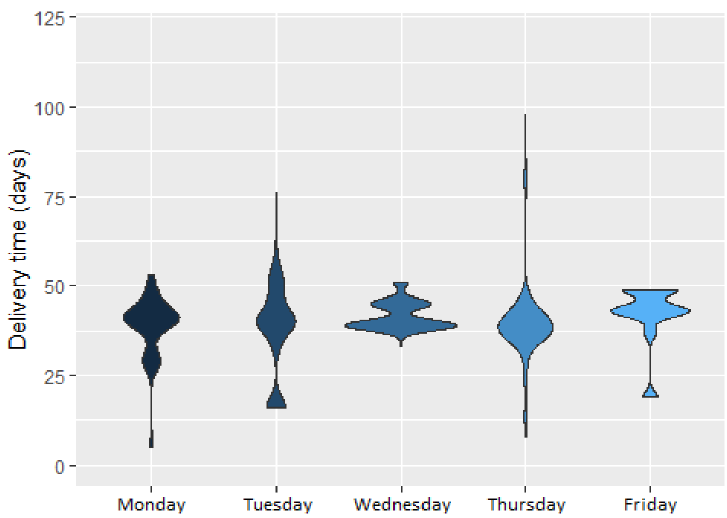


Figure 4.2: Delivery time per order day of week.

In order to verify if there is a significant difference between the delivery time per day of the week some statistical tests were conducted.

In order to compare the mean delivery times per day of the week, the parametric statistical test Analysis of Variance (ANOVA) can be used. Before using ANOVA it is necessary to verified normality, homogeneity of variances and independence of groups. The normality is reject since the p–values of the test Shapiro-–Wilk presents a value less than 5%. However, since the sample dimension is higher than 30, by the Central Limit Theorem, the normality can be assumed.

The Levene test is used to test the homogeneity of the variances. Their p–value is approximately zero ($p = 1.291e^{-05}$), so, $H0$ (group variances are equal) at significance level $\alpha = 0.05$ is rejected. Thus, there is heterogeneity of variances, therefore ANOVA cannot be used and the Kruskal Wallis (KW) for equality for medians is used. Its p–value ($3.395e^{-12}$) is less than 0.05, thus for a significance level of 5%, the null hypothesis is rejected, i.e., there are significant differences between medians delivery times on different days of the week.

Next comparisons between the medians delivery times on two different days of the week is made using the Wilcoxon test. Table 4.4 shows the p–values. It can verified that there are significant differences between the delivery times for all pairs of weekdays except for Wednesday – Monday, and Wednesday – Tuesday.

	Monday	Tuesday	Wednesday	Thursday
Tuesday	0.02566	-	-	-
Wednesday	0.69536	0.69536	-	-
Thursday	0.00104	0.00497	$2.0\ e^{-16}$	-
Friday	$8.2\ e^{-08}$	0.00097	0.03645	$6.4\ e^{-07}$

Table 4.4: P-values for Wilcoxon test.

4.2 The Developed Optimization Model

This work addresses a deterministic scheduling problem with sequence dependent job setup times, where the set of jobs to process is known before their scheduling. Note that in this problem the data is deterministic and known in advance. The objective is to minimize the setups and therefore contribute to the reduction of scrap produced.

In this section the formulation of the production scheduling for aluminium extrusion as an optimization problem is presented.

The scheduling problem studied in this work is a flowshop. In this kind of problems each job j is processed in several machines. However, in accordance with goal of this work, that is to minimize the amount of scrap produced, the present mathematical formulation considers only one machine, the extruder, with several dies. This situation can be interpreted as the existence of several machines, that cannot work simultaneously.

The is evidence that if the setup times at the extrusion are minimized, the will be a decrease in scrap production. These setup times occur whenever die changes are performed. Die changes when there is a profile change in the production or when the time/weight limit of the utilization of the die has been achieved.

The proposed model will is intended to allow for short term production scheduling. Moreover sufficient raw materials are assumed and there is no limitation on storage area. In this deterministic problem the demands is known *a priori* and maximum inventory level is assumed.

4.2.1 Parameters

The parameters of the model concern: orders, dies and setups. The following is a description of the parameters used in the model.

- $j \in Orders$ are the orders, with $\#Orders = n$.
- M_j is the set of dies[1] that can be used for processing order j.
- $d \in Dies$ are all the available dies, with $\#Dies = m$.
- p_j is the extrusion processing time of order j.
  This parameter is not available, therefore it is estimated using linear regression models as explained in Section 4.3.2. This time depend on the quantity, profile geometry, number of holes, among other factors.
- r_j is the release date of order j, in minutes.
  The computation of this parameter is explained in Section 4.3.1.
- d_j is the deadline of order j, in minutes.
  The computation of this parameter is explained in Section 4.3.1.
- c_j is the weight, in kg, to extrude of order j.
- $DMax_d$ is the maximum allowed processing extrusion weight in kg of die d.
- $DMin_d$ is the minimum allowed processing extrusion weight in kg of die d.
- α is the time for changing the die, and is given in Section 3.2.
- $S_{jj'}$ is the setup time that is such that:

$$\begin{cases} \alpha \text{ if from order } j \text{ to } j' \text{ the die must be changed} \\ 0 \text{ otherwise} \end{cases}$$

[1] In fact, this is the combination of die E_D and tool E_To.

- t_{prod} is the time specified by the company, for the order to reach production, in minutes. This is the average time that the company estimates it takes to get the order to the production manager, including the time of evaluating if the die is good for production (see Section 3.2).

- t_{after} is the time specified by the company that takes to perform other tasks such as, cutting the bars, packing, cleaning, surface treatment and thermal treatment (see Section 3.2).

- M represents a sufficiently large positive number.
  This scalar is introduced in such a way that the constraints in which it is used can be made redundant when necessary.

4.2.2 Decision Variables

Three sets of decision variables are used, namely:

- $s_j \geq 0$ is the starting time of extrusion process of order $j \in \{1, ..., n\}$.

- $x_{jj'} \in \{0, 1\}$, and is 1 if order j' immediately follows j, $\forall\, j \neq j'$, and is 0 otherwise.

- $m_j^d \in \{0, 1\}$, is 1 if die d is used for processing order j, and is 0 otherwise.

The decision variables, s_j, are continuous. The other decision variables are binary and are related to the ordering of the orders and die assignment.

4.2.3 Mathematical Model

In this section a Mixed Integer Linear Programming (MILP) model is developed. In the formulation of the model the notation of Areanales [7] is used. Taking into account the characteristics of the problem under study, and the definitions of the previous section, the objective function and constraints are presented next.

Objective Function

Literature presents several objective functions. However none of these will be used, since the goal is to minimize the amount of scrap produced. As peak production amount of scrap is observed in the production process dataset at instances when die change occurs, the minimization of setup times will be used, since this minimization will imply the minimization of die changes.

The objective is to find a schedule that minimizes the sum of setup times:

$$\min \sum_{j \neq j' \in Orders} S_{jj'} x_{jj'} \tag{4.2}$$

Constraints

$$s_j \geq r_j + t_{prod}, \qquad \forall\, j \in Orders \tag{4.3}$$

$$s_{j'} \geq s_j - M + p_j + Mx_{jj'} + S_{jj'}x_{jj'}, \qquad \forall\, j \neq j' \in Orders \tag{4.4}$$

$$\sum_{d \in M_j} m_j^d = 1, \qquad \forall\, j \in Orders \tag{4.5}$$

$$\sum_{d \notin M_j} m_j^d = 0, \qquad \forall\, j \in Orders \tag{4.6}$$

$$s_j + p_j + t_{after} \leq d_j, \qquad \forall\, j \in Orders \tag{4.7}$$

$$\sum_{j \in Orders} m_j^d\, c_j \leq DMax_d, \qquad \forall\, d \in Dies \tag{4.8}$$

$$\sum_{j \in Orders} m_j^d\, c_j \geq DMin_d, \qquad \forall\, d \in Dies \tag{4.9}$$

$$x_{jj'} + x_{j'j} \leq 1, \qquad \forall\, j \neq j' \in Orders \tag{4.10}$$

$$\sum_{j' \in Orders \setminus \{j\}} x_{jj'} = 1, \qquad \forall\, j \in \{0\} \cup Orders \tag{4.11}$$

$$\sum_{j \in \{\{0\} \cup Orders \setminus \{j'\}\}} x_{jj'} = 1, \qquad \forall\, j' \in \{0\} \cup Orders \tag{4.12}$$

$$s_j \geq 0, \qquad \forall\, j \in Orders \tag{4.13}$$

$$x_{jj'} \in \{0, 1\}, \qquad \forall\, j \neq j' \in Orders \tag{4.14}$$

$$m_j^d \in \{0, 1\}, \qquad \forall\, j \in Orders, \quad d \in Dies \tag{4.15}$$

Constraints (4.3) guarantee that each order j can only be processed after all procedures prior to extrusion production are executed. Constraints (4.4) ensures that if a order j precedes order j' (i.e., $x_{jj'} = 1$) then the starting time of order j, s_j, plus its the processing time, p_j, and setup time of j, $S_{jj'}$, is less than the starting time of order j', $s_{j'}$. Otherwise, that is, if $x_{jj'} = 0$, where M is a sufficiently high constant, the respective constraint becomes redundant. Constraints (4.5) require that each order j is only assigned to exactly one die d, among those that can be used. Conversely, constraints (4.6) guarantee that the order j cannot be processed by an die d not in the M_j set. Constraints (4.7) force each order j to be ready for delivery to the customers before the deadline, including the time to be extruded and the time it takes to perform other tasks such as, cutting the bars, packing, cleaning, treatment and finishing or thermic, whose time is given by t_{after}. Constraints (4.8) and (4.9) define the maximum and minimum of extruded weight that can be extruded by each die d. Constraints (4.10) guarantees that either order j precedes (not immediately) order j' or order j' precedes order j. Constraints (4.11) defines that if every order j' has a predecessor order j that may be the initial dummy product. Constraints (4.12) are similar to the previous. Finally, constraints (4.13), (4.14) and (4.15) define the domain of the decision variables.

This model has a total of $n(n+m)$ decision variables and $2n^2+4n+2m+2$ constraints. The $n(n+m)$ decision variables are distributed between (4.13), (4.14) and (4.15), with n, $n(n-1)$ and nm, respectively. While (4.3), (4.5), (4.6) and (4.7) correspond to n constraints each, (4.4) and (4.10) correspond to $n(n-1)$

constraints each, (4.11) and (4.12) correspond to $n+1$ constraints each, (4.8) and (4.9) correspond to m constraints each.

The model here proposed assumes that the weights of the individual orders satisfy the limits set by the production manager. If this does not occur, i.e., if the order's weight is larger than allowed the order is subdivided in several smaller orders in a preliminary step prior to scheduling.

4.3 Parameters Estimation and Computation

In Section 3.2 a brief description of the two datasets used for the development of the optimization model presented in Section 4.2 was made. However, information that is not available is necessary for being included in the model. In this section the calculations to determine the releasing date (r_j) and the deadline (d_j) are presented. Furthermore, the methods used for estimating the processing time (p_j) are presented.

4.3.1 Release Date and Deadline Computation

In the Customers' orders dataset there is information of the date in which each order is received. However, for the development of the proposed model it was necessary to make some adjustments. The dataset starts on 02/05/2018, which was considered the reference date, i.e., $t = 0$.

The release date of order j, r_j, was calculated as the minutes from the reference date until the date in which the order is received, excluding Sundays. For example, $r_j = 0$ corresponds to the release date of the order j that arrived at the company on 02/05/2018. Since production works 24 hours per day (in continuous) the release date of an order that arrives at the company on 03/05/2018 is 24 hours $\times 60$ minutes $\times 1$ day $= 1440$ minutes. An order that arrives at the company on 07/05/2018 is 24 hours $\times 60$ minutes $\times 4$ days $= 5760$ minutes, since there are four working days from 02/05/2018 to 07/05/2018[2], as illustrated in Figure 4.3.

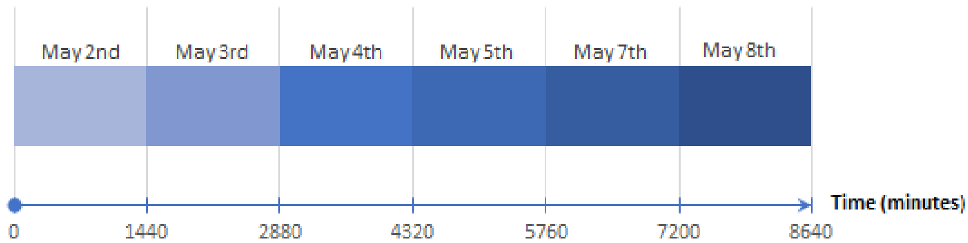


Figure 4.3: Release date illustration.

The deadline of order j, d_j, corresponds to the limit moment for delivery of the order j, and was calculated as the minutes from the reference date until the date the delivery of the order j, excluding

[2]Note that 06/05/2018 was a Sunday, i.e., a non working day.

Sundays. That is, d_j is equal to r_j plus the delivery time of the order j. For example the order $j = 36$, that arrives on 03/05/2018, thus has a release date of $r_{36} = 1\ 440$ minutes, and must be delivery until 15/06/2018 with a delivery time of 38 working days, corresponds to a deadline of $d_{36} = 54\ 720$ minutes.

4.3.2 Processing Time Estimation

In classical deterministic scheduling, the job processing times are generally assumed as constants. However, there are situations in which the real processing times may vary due to the learning, aging or deteriorating effect.

The processing time of each order, p_j, is not available, therefore it is necessary to estimate it. For this purpose, for each die (in a total of 764), a linear regression models was estimated from historical data from the extrusion process.

From the Production process dataset a new dataset was created aggregating all billets of each produced order j in a single row. The variables in this dataset correspond to the total processing time ($\sum E\_t$) and total weight ($\sum PO\_WB$) for each order j.

Variable	Min	1st Q	Median	Mean	3rd Q	Max
Total processing time	113.0	693.0	1 218.0	2 405.0	2 450.0	49 508.0
Total weight	19.0	244.0	439.0	878.8	922.0	20 865.0

Table 4.5: Statistics for total processing time and weight of each order.

The main statistics for these new variables are presented in Table 4.5. A large variability in terms of total processing times and total weights can be observed. The total time of each order is between 113 and 49 508 seconds, and the total weight variable is between 19 and 20 865 kg. Note that 75% of orders present total processing time less than 2 450 seconds and a total weight bellow 922 kg. Thus the majority of the orders are small. Therefore, assuming that the weight limits are respected in a realistic assumption. Figure 4.4 suggests a linear relation between total processing time and total weight.

For exemplification, Figure 4.5 shows the behavior of the two most used dies for which we have historical data on extrusion, that is dies in the Production process dataset. For these two dies the same linear relation between the processing time and weight, can be observed.

The dispersion diagrams in Figures 4.4 and 4.5 show, as expected, a positive correlation between the variables, i.e., on average, an increase in weight leads to an increased processing time. From the graph on the left, for Die A, the processing time is expected two increase 2.7752 seconds per each additional kg, starting with a default time of 211.30939 seconds. For Die B, the ratio of increase per kg is different from the one of Die A. In fact, the graph on the right, shows expected default processing time of 109.68341 seconds, that increases 2.97825 seconds per each additional kg. In both cases, the coefficient of determination, R^2, which is used as a measure of fit quality, is approximately 0.97. Therefore, in both the linear regression models, 97% of the total processing time is explained by the total weight.

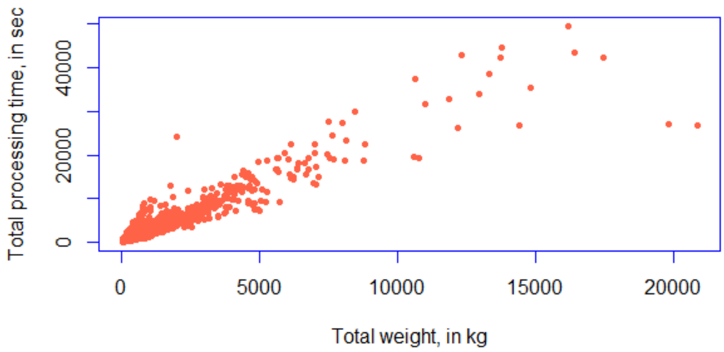


Figure 4.4: Total processing time *vs.* total weight, for all orders.

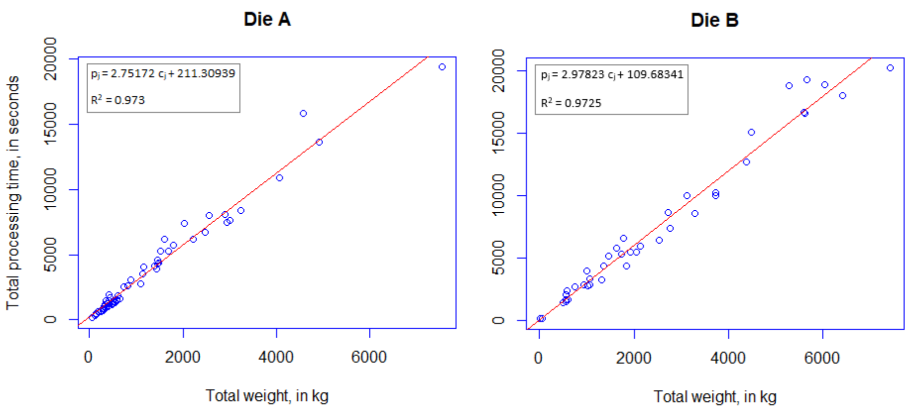


Figure 4.5: Linear regression models.

To estimate the linear regression models the software R, version 3.6.1, was used. For an order j processed using die d, the processing time will be estimated based on following equation:

$$p_j = m^d \, c_j + b^d \tag{4.16}$$

where,

- p_j is the extrusion processing time of job j, in seconds.
- c_j is the quantity produced, in kg.
- b^d is the intercept.
- m^d is the slope.

Three situations where this process cannot be applied are verified:

- The historical data just have one order j for the die d and is not possible estimate the linear model;
- The die d is new and not available in the historical data, because the dies were not used in production from January to June 2018;
- The linear regression estimation resulted in a negative slope, which is not realistic. This occurs in 9 orders.

In the first case the slope was computed using the following:

$$m^d = \frac{\text{Total processing time}^d}{\text{Total weight}^d} \tag{4.17}$$

In the second and third cases the processing time was calculated similarly to the previous one, but using average values of the entire Production Process Dataset (presented in Table 4.1), i.e.:

$$m^d = \frac{173.93}{63.47} \tag{4.18}$$

For all previous cases the intercept, b^d,is set to zero.

There was one case for which the estimated processing time was negative. For this cases the procedure used was the same used in the second and third cases.

Chapter 5

Case Study Results

In the previous chapter, a model that aims at answering the company's specifications was developed. In this chapter, instances extracted from the Customers' orders dataset (see Section 3.2), provided by the company, as well as the computational results obtained for these instances are presented. Since there is no information of the company's scheduling procedure for the dies being schedule, it is not possible to make a direct comparison of the results of the model and the company's reality.

To overcome this situation, the average die changes per week was computed, from the historical production process data. Furthermore, the FIFO dispatching rule (c.f. Section 2.2) very frequently used in practice, was also used with the instanced tested. In this rule, it is selected the first job that arrived in the queue.

Finally, in order to perceive the improvements obtained with the developed model, in Section 5.6 the results obtained with the MILP model are compared to the average die changes per week of company and to the results of the dispatching rule FIFO.

The MILP model (4.2)–(4.10) was implemented using AMPL[1] (A Modeling Programming Language) modeling system and the real instances were solved using the Gurobi solver, available at the NEOS[2] (Network Enabled Optimization System) platform.

AMPL is one of the most used modeling languages for mathematical programming [28]. Due to the generality of its syntax and the similarity of its expressions with algebraic notations, which enables to easily specify and understand the objective function, the constraints, as well as the logical relations between variables. NEOS is an environment for remotely solving optimization problems [17]. It is supported by the Optimization Technology Center of Northwestern University & Argonne National Laboratory and is a free service, requiring no registration, downloads or installations. It supports a wide variety of problem formats (e.g., AMPL, GAMS, etc) and features an easy to use interface.

[1] http://www.ampl.com

[2] https://neos-server.org/neos/solvers/

For obtaining the numerical results of the instances, a single mod file is used. This file presents the specification of the model without any values attributed to the parameters. Also, a single run file containing solver and display specification is used. While, several data files (one per instance), were created (see Figure 5.1).

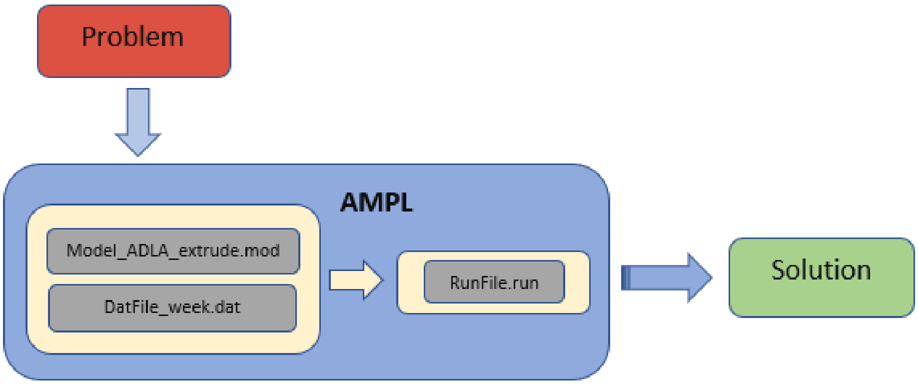


Figure 5.1: Implementation of the model in AMPL

The model is given in the `Model_ADLA_extrude.mod` file and includes the codification of all the parameters, decision variables, objective, constraints. This model is used to test all the instances. For writing the dat files for each instance, the file `WriteDatFiles.R` was created. This R file reads the information from the Customers' orders dataset and writes a dat file for each instance with the syntax used by AMPL. Finally, `RunFile.run` contains the code for the selection of the solver and the formatting information of the output that is produced, i.e., the optimal solution.

5.1 The Procedure of the Company

There is no information on the company's scheduling procedure for the dies being schedule, therefore it is not possible to make a direct comparison of the results of the model and the company's reality.

However, a characterization of the orders from the historical production process data was made and an average die changes, per week, was computed. Analyzing the historical production data provided in the Production process dataset it could be observed that between between January 1st and June 30th a total of 3 101 orders were produced and 2 046 die changes were made. That gives an average of approximately 119 orders and 79 die exchanges, per week.

Figure 5.2 indicates the number of the orders produced and the number of die changes made in each month. This figure shows differences for different months. Specially may is the month with the highest number of produced orders, in a total of 664, but a number of die changes of 368, which is similar to the ones observed in the other months.

A more detailed analysis is performed to verify the ratio of die changes per produced orders, in each month. In Table 5.1 the percentage of produced orders that have die changes is presented. March is the

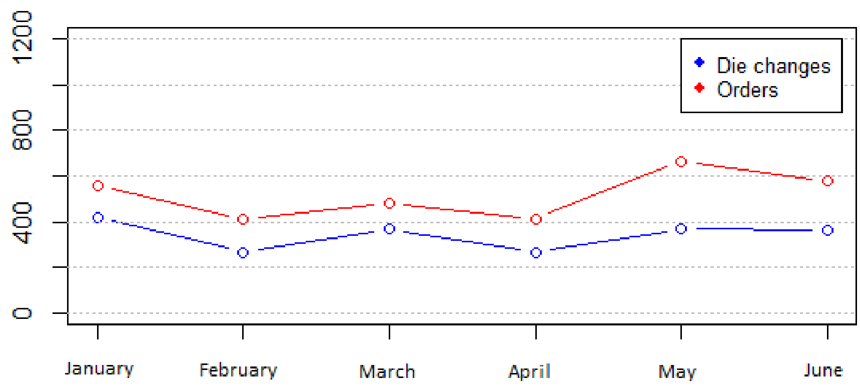


Figure 5.2: Illustration of the number of orders produced and number of die changes, per month, between January 1st and June 30th, 2018.

	Orders	Die changes	Ratio
January	556	416	74.82%
February	409	266	65.04%
March	478	368	76.99%
April	414	265	64.01%
May	664	368	55.42%
June	580	363	62.59%

Table 5.1: Percentage ratio of die changes for the orders produced in each month.

month which presented a highest percentage, approximately 77% , of die changes. On the other hand, May presented the lowest percentage, approximately 55%. Overall, there was an average of approximately 67% of produced orders that have die changes. Furthermore, there was, on average, 79 die changes per week. Note that these values will serve as a reference to compare the numerical results obtained using the model developed.

In the next subsection, the selection of instances, as well as the computational results are presented and discussed.

5.2 Instances Selection

The real data provided by the company concerning the customers' orders for the month of May 2018, presented in Section 4.1.2, was used to construct the instances for which the numerical results were obtained. Taking into account the expected stop periods, specified in Section 3.2, a total of 26 working days and 498 orders were considered.

Instances were selected to test the scheduling of incoming customers' orders. The objective was to consider instances that present the business environment reality, depict the recurring volatility of the markets. These results will allow to validate the model proposed in Section 4.2 which aims to minimize die changes by minimizing the setups times, that consequently will contribute to the reduction of scrap production in extrusion process.

For each instance, the orders relative to a week with six working days are considered. The selection of considering instances with six working days also had into account the statistics results obtained in Table 4.3 (see Section 4.1.2), i.e., the existence of orders that have five days to delivery time.

Information from the company concerning if there are specific days to schedule the production of orders was not provided. Therefore, our selection began with the first working day. In fact, since Wednesday, 01/05/2018, is an holiday, therefore, the weeks are considered from Wednesday to Tuesday, namely:

- Week 1: corresponds the orders from May 2nd to 8th.
- Week 2: corresponds the orders from May 9th to 15th.
- Week 3: corresponds the orders from May 16th to 22nd.
- Week 4: corresponds the orders from May 23rd to 29th.

Note that following this procedure the last days of May, i.e. 30th and 31st, were not considered. Figure 5.3 presents the characterization of the four weeks, relatively to the number of orders received, the number of customers and the different dies used.

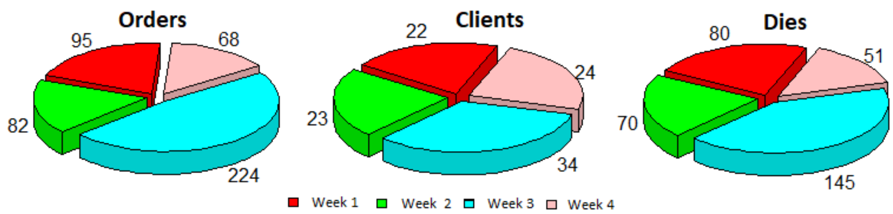


Figure 5.3: Characterization of the weeks considered. From left to right: number of orders, number of customers and number of dies.

In Figure 5.3 it can be seen that in the third week, from May 16th to 22nd, the total number of orders received was 224. This presents approximately 45% of the total number of orders of entire month of May. For the sake of brevity, only 50% of the orders received in the month of May 2018 will be used to test the model. In order to analyze three different scenarios, the selection of instances to test the scheduling of order production relapsed on the period from May 2nd to 8th (Week 1), 16th to 22nd (Week 2) and 23rd to 29th (Week 4).

Table 5.2 describes the instances to be tested, as well the total number of orders in each. In the following sections the results and discussion for the three instances are presented in detail.

Days	Total Orders
May 2nd to 8th	95
May 9th to 15th	82
May 23rd to 29th	68

Table 5.2: Characterization of the real instances used to test the developed model.

5.3 May 2nd to 8th

The first instance corresponds to orders from May 2nd to 8th, and includes the scheduling of 95 orders from 22 different customers (see Figure 5.3). The release date this instance starts at $t = 0$ minutes and finishes at $t = 7\ 200$. Table 5.3 presents information concerning the total weight, the processing time and the deadline of these orders.

	Min	Q1	Median	Mean	Q3	Max
Total weight (c_j)	200.50	252.00	440.70	1 075.10	1 001.90	14 051.40
Processing time (p_j)	7.45	11.83	18.37	52.68	48.13	570.19
Deadline (d_j)	12 960.00	56 160.00	56 160.00	62 435.00	73 440.00	73 440.00

Table 5.3: Total weight, the processing time and the deadline of the orders received from May 2nd to 8th.

In Table 5.3 it can be observed a large variability in the total weight of each order, with orders with weights from 200.50 kg to 14 051.40 kg. However, 75% of orders have a total weight less or equal than to 1 001.90 kg. In the orders produced, 50% have processing times less or equal than to 18.37 minutes and the maximum was of 570.19 minutes. Deadline average is 62 435.00 minutes, ranging from 12 960.00 to 73 440.00 minutes.

5.3.1 Results of FIFO Dispatching Rule

In this subsection the results from May 2nd to 8th obtained using the dispatching rule FIFO are presented. This rule gives priority to the order of arrival, i.e., the first order that arrives is the first to be produced. For this purpose, the data from Customers' orders dataset was sorted by order of arrival and whenever the die that must be used in a given order is different from the previous one a setup time of 1.5 minutes[3] is added. Note that before each order is processed there is a prior time to consider since its arrival, i.e., the average time that the company estimates it takes to get the order to the production manager, including the time of evaluating if the die is good for production. Considering the information provided by the company this was set to 60 minutes.

[3]This value was provided by the company

Applying the dispatching rule FIFO from May 2nd to 8th, a total of 85 die changes is obtained, with a total setup time equal 127.50 minutes. The first order started to be produced in the minute 60.00 and the last order finished in the minute 8 257.50. The time to process all orders is approximately 5 005.00 minutes. Table 5.3 indicates that the minimum order deadline verified for these instance is 12 960.00 minutes, with is quite below the processing time. Therefore, it is guaranteed that all orders were produced on time, i.e., before the deadline.

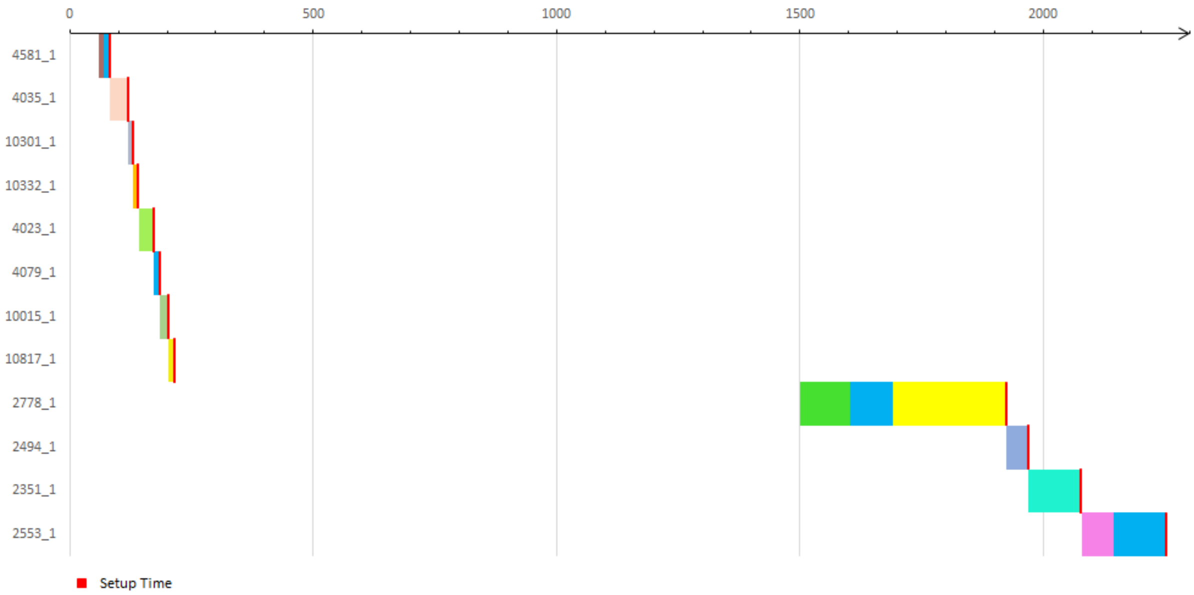


Figure 5.4: Description, per die, of the orders production scheduling obtained by the dispatching rule FIFO on the 16 first orders for the May 2nd to 8th instance.

Figure 5.4 shows the sequence of the 16 first orders and corresponding setup times for each die. The solution for the complete instance is presented in Figure A.1, in Appendix. The 16 first orders for the week 1 instance, can be described as:

- The first order starts to be produced at minute $t = 60$ and die 4581_1 is used, with a processing time of 10.66 minutes. The second order also uses the same die and the processing time was also the same. As the third order does not use the same die, a setup time of 1.5 minutes is considered.
- The next orders to be produced use different dies, namely dies 4035_1, 10301_1, 10332_1, 4023_1, 4079_1, 10015_1 and 10817_1. They have different processing times, and at the end of each production order a setup time of 1.5 minutes is verified.
- The next order only arrived at the minute 1440, so it can only be produced from there, taking into account the 60 minutes needed to be able to be produced. Die 2778_1 is used to produced the next three orders. The first one starts at 1 500.00 minutes and the third one ends at minute 1 921.76. Next, the die is changed, generating a setup time of 1.5 minutes.

- Order 13 and 14 use different dies. The first start at minute 1 923.26 and the second starting at minute 1 968.36. Between the two orders, there is a 1.5 minute setup time resulting from the die exchange.

- Lastly, the 2553_1 die is used for the production of the last two orders, where the first one starts at minute 2 077.97 and has a processing time of 65.57 minutes. The second has a processing time of 108.06 minutes and ends at minute 2 251.60.

5.3.2 Results of the Developed Model

The optimal solution found for the instance of May 2nd to 8th, presents and objective function value of 118.50. Therefore, in total the optimal solution has a minimum total setup times of 118.50 minutes, which corresponds to 79 die changes. The optimization problem presented a total of 9 251 variables, the large majority of which are binary and only 95 are linear, 17 860 inequality constraints and 207 equality constraints (see Table 5.9).

Figure 5.6 presents the result of the scheduling of this instance using the model developed in Section 4.2. Analogously to the FIFO dispatching rule, the first order could only start to be produced 60.00 minutes after arriving to the company, and when die is changed a setup time of 1.50 minute is added.

The first order started to be produced at minute 1 500.00 and the last finished your production at minute 54 735.78. Figure 5.6, shows that that none of the orders produced by the same die overlap. The existence of three dead times in the scheduling of orders was verified (see Figure 5.6), i.e., time intervals in which no production is being made. In fact, the developed model is only minimizing setup times and there is no guarantee that dead times will not be verified.

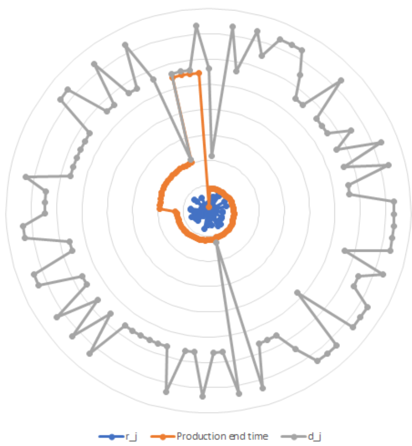


Figure 5.5: Representation of release date, production end time and deadline for each order from May 2nd to 8th.

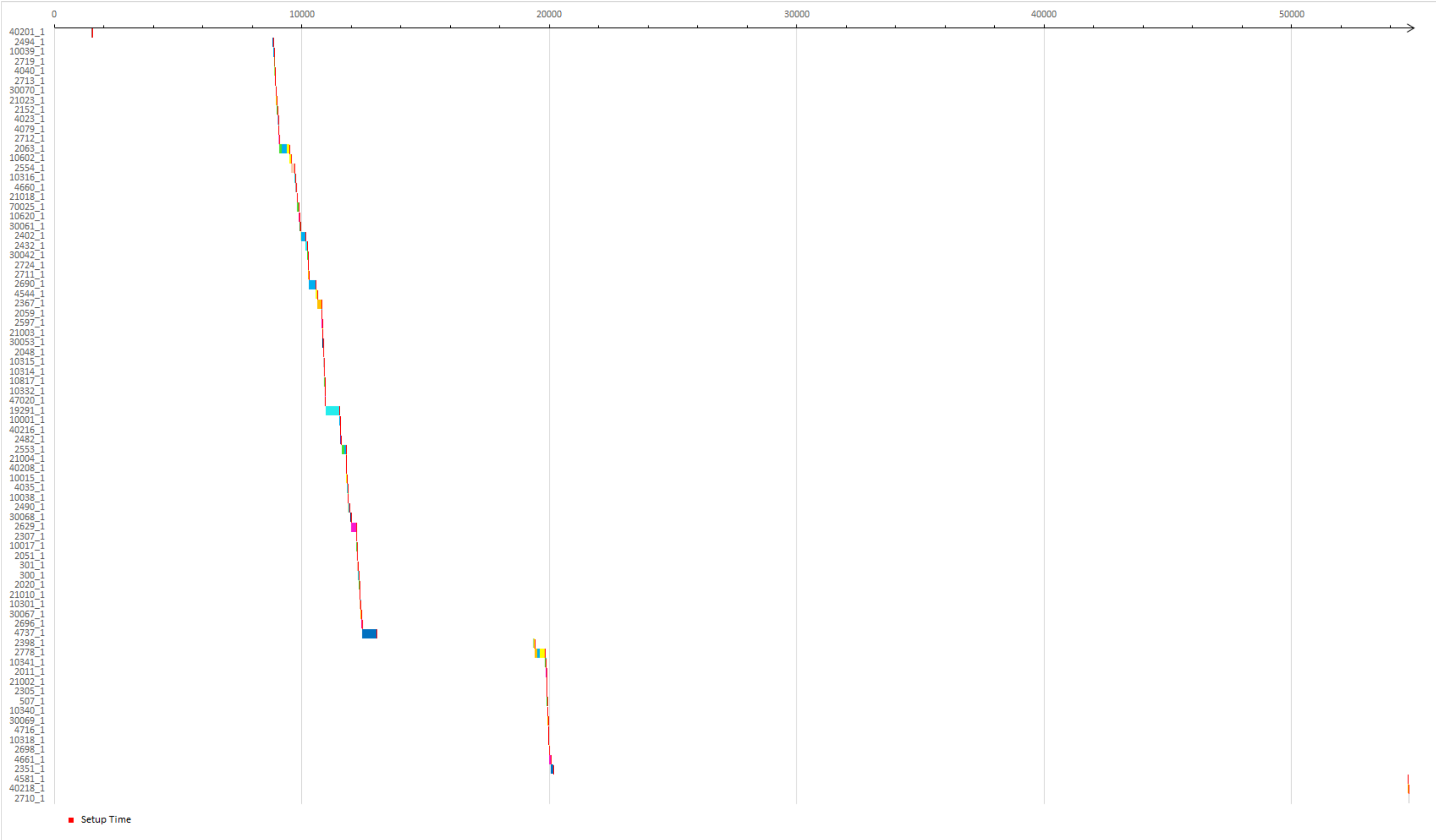


Figure 5.6: Solution, per die, for the orders from May 2nd to 8th.

On the other hand, from Figure 5.5 it is possible to verify that the time constraints are respected, i.e., orders are only processed after release date and finish before the deadline, including processing time and the time to perform other tasks such as, cutting the bars, packing, cleaning, surface treatment and thermal treatment. Therefore, all constraints imposed in the model are being guaranteed.

In order to clarify the results obtained with the developed model, the scheduling solution for a six orders of this instance is presented in Figure 5.7.

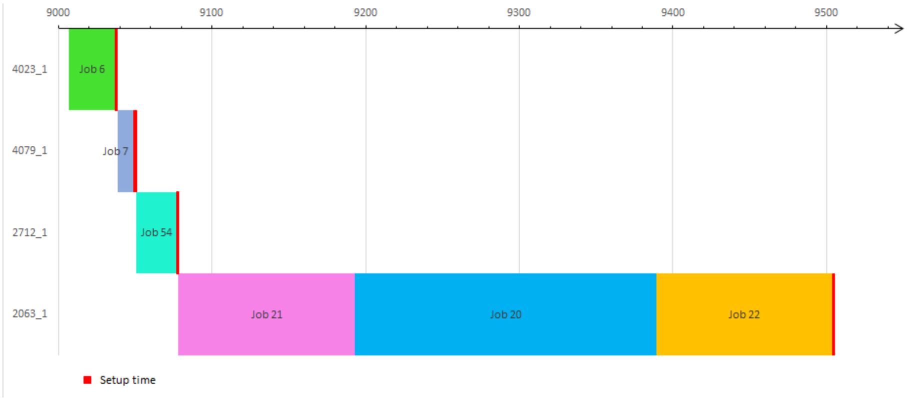


Figure 5.7: Solution, per die, from 11st to 16th produced order, from May 2nd to 8th.

The first order illustrated, Job 6, corresponds to the order was produced in 11st place, and the last order, Job 22, was produced in 16th place. This period is selected since it depicts two distinct cases that may occur, i.e., the die produced only one order, or a die produced multiple orders. The scheduling solution was:

- Job 6 was produced with die 4023_1 and had a processing time of 30.26 minutes. The release date is at minute 0 and started production at minute 9 006.62. This order has a deadline of 56 160.00 minutes to be delivered to the customer. As the next order will use a new die, a setup time of 1.50 minutes is verified.
- The next order to be produced is number 7, Job 7, which also arrived at time 0 and started production at minute 9038.38. The die used is 4079_1. With a processing time of 10.99 minutes per minute, a setup time of 1.50 minutes occurs at minute 9 049.37. The deadline is 56 160.00 minutes.
- Job 54 arrived at the minute 2 880.00 and began to be produced at minute 9 050.87, on the die 2712_1. It has a processing time of 25.95 minutes and in the end die exchange occurs. The deadline is 73 440.00 minutes.
- Jobs 21, 20 and 22 are produced with the die 2063_1. These orders arrived at minute 1 440 and began to be produced at minute 9 078.32, minute 9 193.06 and minute 9 389.13, respectively. At the end a setup time of 1.50 minutes is verified, which denote that the next order will use a different die. The deadline for each order is 56 160.00 minutes.

The optimization model was developed based on the information provided by the company. However, a computational test considering the parameter t_{prod} equal to minute 1 440 was made, i.e., the orders can only start being produced the next day your arrival. This parameter, t_{prod}, is the time specified by the company, for the order to reach production, and in the developed model is considered to minute 60. The results obtained yield an objective function value 118.50 minutes, i.e., there were 79 exchanges. This was also the number of die exchanges found previously. The first order started production at minute 8 640 and the last order finished at minute 13 763.20. Deadline fulfillment was verified and all orders were completed before the delivery date to the customer and all constraints imposed were verified. The results of this model can be seen in Figure A.2, in Appendix.

5.3.3 Performance Comparison

Table 5.4 presents the results obtained by dispatching rule FIFO and the developed model, to a t_{prod} equal to 60 or 1 440 minutes, for the instance of May 2nd to 8th. The results are relative to to the total number of die changes, the total of setup times, and the beginning and end of the first and last order produced. Total processing time is also showed.

	Setup	Die	First order		Last order		Processing
	times	changes	Start time	End time	Start time	End time	time
FIFO	127.50	85	60.00	70.66	8 235.48	8 257.50	5 004.69
OPT	118.50	79	1 500.00	1 544.85	54 721.50	54 735.78	5 004.69
OPT[1]	118.50	79	8 640.00	8 662.63	13 749.00	13 763.20	5 004.69

Table 5.4: Comparison of results for the instance of May 2nd to 8th.

Regarding the results obtained, it can be seen that if the company applies the FIFO rule, 85 die changes will be made, corresponding to 127.50 minutes of setup times. On the other hand, the developed model presents lower values, i.e., for the same period there are 79 die changes with a total 118.50 minutes of setup times. The results refer to a total orders processing time of 5 004.69 minutes.

Considering the purpose of this study, the minimization of the scrap produced, trough the minimizing of die changes, the developed model presents the most advantageous solution for the company when compared to the solution found with the FIFO rule. In relation to the beginning and end of the production of orders, it is verified by the FIFO method the orders start and finish earlier. The results of the developed model obtained for this instance, show that the first order begins production the day after its release date, and the last order finishes at minute 54 735.78. However, few orders are being produced so late and meet the deadline (see Figure 5.6). Comparing the results on the developed model for a $t_{prod} = 60$ (company reference) and a $t_{prod} = 1\ 440$, the setup times and exchanges are the same, however in the

[1]Developed model for a $t_{prod} = 1\ 440$.

second case a more favorable starting time schedule is verified. Orders start production later but finish earlier, with no production stoppages (see Figure A.2).

5.4 May 9th to 15th

The second instance corresponds to orders from May 9th to 15th, whose release date corresponds to minute $t = 8\ 640$ and to minute $t = 15\ 840$, respectively. This instance includes the production scheduling of 82 orders from 23 different customers (see Figure 5.3). Table 5.5 presents information on the total weight, the processing time and the deadline of these orders.

	Min	Q1	Median	Mean	Q3	Max
Total weight (c_j)	99.45	413.57	701.13	1 270.54	1 589.48	8 421.20
Processing time (p_j)	7.33	18.71	30.27	56.14	58.67	440.99
Deadline (d_j)	23 040.00	43 200.00	73 440.00	66 679.00	83 520.00	144 000.00

Table 5.5: Total weight, processing time and deadline of the orders received from May 9th to 15th.

From Table 5.5 it can be observed that the total weight per order ranged from 99.45 kg to 8 421.20 kg, and the processing time ranged from 7.33 minutes to 440.99 minutes. The deadline varies between 23 040.00 and 144 000.00 minutes.

5.4.1 Results of FIFO Dispatching Rule

Applying the dispatching rule FIFO for the instance from May 9th to 15th, a total of 71 die changes is obtained, with a total setup time equal 106.50 minutes. The first order was produced at minute 8 700.00 and the last order to minute 17 572.74 with a processing time of 11.65 minutes. The minimum deadline observed in Table 5.5 is 23 040.00 minutes, so deadlines are met. The solution for this instance is presented in Figure A.3, in Appendix. The orders received per day are not enough for the company to produce all day, and down times can verified.

5.4.2 Results of the Developed Model

The optimization problem solved for the instance from May 9th to 15th has a total of 6 939 variables, the large majority of which are binary and only 82 are linear. The problem has 13 284 inequality constraints and 189 equality constraints, and the objective function is 103.50.

Figure A.4, in Appendix, present the result of the scheduling of this instance using the model developed in Section 4.2. During this period 69 die changes were made, the first order to be produced started at minute 11 580.00 and the last order at minute 71 999.99. In Table 5.5 can see that 50% of orders have deadlines of less than 73 440 minutes, so, is necessary ensure deadlines are being met. Note that, in the developed model, after the production the orders take 1 440 minutes (one day) to be delivered. The time of surface treatment, in case to need, it is a previous procedure done by the manager, and it is already included in the deadline.

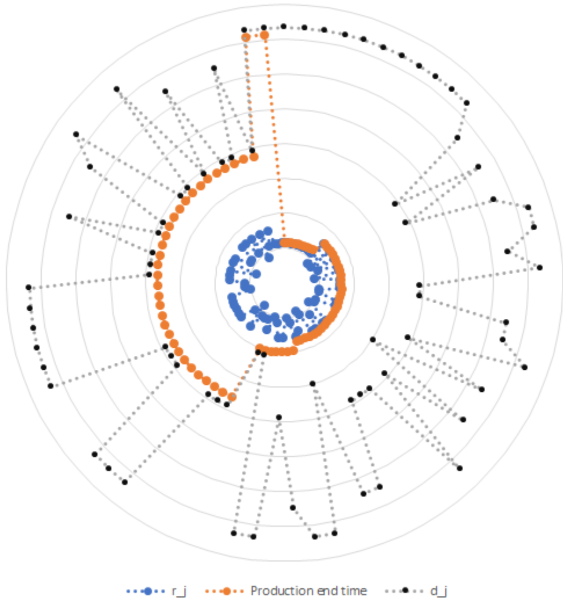


Figure 5.8: Graphic representation of release date, production end time and deadline for each order from May 9th to 15th.

Figure 5.8 illustrates the releasing date, production end time and deadline of each order, and one can verify that none order were starting produced before the release date and ended after the deadline. Only orders with a deadline equal or less to 73 440 minutes are presented, in the others the deadline is guaranteed.

5.4.3 Performance Comparison

Table 5.6 presents information relative to the total number of die changes, the total of setup times and time to produced first and last order, for the instance 9th to 15th, for dispatching rule FIFO and for the developed model.

	Setup times	Die changes	First order		Last order		Processing time
			Start time	End time	Start time	End time	
FIFO	106.50	71	8 700.00	8 735.79	17 572.74	17 584.39	4 538.07
OPT	103.50	69	11 580.00	11 634.06	71 559.00	71 999.99	4 538.07

Table 5.6: Comparison of results for the instance from May 9th to 15th.

In Table 5.6 less die changes with the developed model are verified. Analogous to the results for the previous instance, by the FIFO rule orders start and finish earlier. However in the developed model, although the time discrepancy, the deadlines are fulfilled. The model developed for this instance is more favorable for company, because the exchanges are minimized, which consequently reduces the scrap, are concluded.

5.5 May 23rd to 29th

The last instance corresponds the orders from May 23rd to 29th. Orders arrived between minute $t = 99\ 360.00$ and the minute $t = 90\ 720.00$. This instance includes the production scheduling of 68 orders from 24 different customers (see Figure 5.3).

	Min	Q1	Median	Mean	Q3	Max
Total weight (c_j)	30.75	256.56	383.66	630.81	772.68	2 566.20
Processing time (p_j)	1.81	12.03	22.01	36.82	43.94	225.21
Deadline (d_j)	47 520.00	90 720.00	95 040.00	94 405.00	108 000.00	116 640.00

Table 5.7: Total weight, the processing time and the deadline of the orders received from May 23rd to 29th.

In Table 5.7 a large variability in the total weight and processing times is observed. However, with the verified values one can assume that the orders are relatively small. This variability also can be seen in deadline, 47 520.00 to 116 640.00 minutes, being that 75% of orders have deadlines between 90 720.00 to 116 640.00 minutes, considerably high.

5.5.1 Results of FIFO Dispatching Rule

Applying the dispatching rule FIFO for the instance from May 23rd to 29th, yield a total of 53 die changes, with a total setup time equal 79.50 minutes. The processing time of all the orders is 2 503.77 minutes. The first order was produced at minute 25 980.00 and the last order was minute 34 960.72, with a processing time of 11.86 minutes. Since deadlines are higher then 47 520.00 minutes (see Table 5.7), delivery time are guaranteed.

The solution for the complete instance is presented in Appendix, in Figure A.5. Similar to what happened in the previous instance, the orders do not have enough quantities to produce all day, so down times are verified.

5.5.2 Results of the Developed Model

The optimization problem solved for the instance from May 23rd to 29th has a total of 4 805 variables, the large majority of which are binary and only 68 are linear. The problem has 9 112 inequality constraints and 157 equality constraints, and the objective function is 75.00.

Figure A.6, in Appendix, presents the result of the scheduling of this instance using the model developed in Section 4.2. A total of 50 die changes are verified to a total processing time of 2 503.77 minutes.

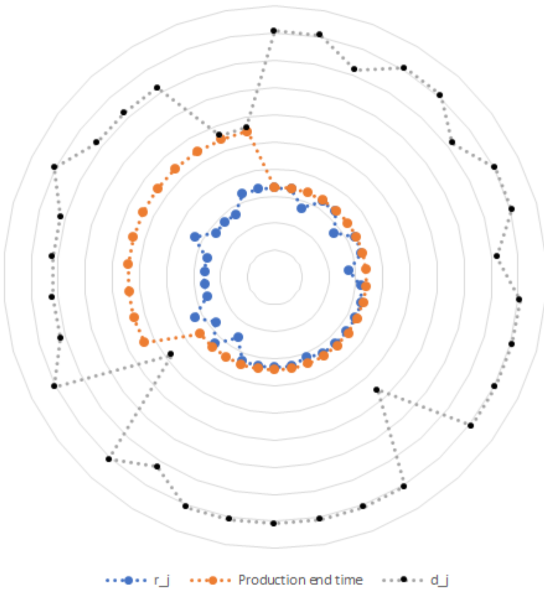


Figure 5.9: Representation of release date, production end time and deadline for each order from May 23rd to 29th.

Deadlines are fulfilled, as you can see in Figure 5.9. Since the last order to be produced ended at minute 97 919.97, we only need to verify orders with a deadline shorter than this one. In Figure 5.9 it is possible to see that we have a large number of orders where the production end time is very close to the release date.

To better understand the schedule made in this instance, a brief explanation of 11 orders produced will be made (see Figure 5.10):

- Job 411, 410, 409 and 408 are produced by die 2391_1, started at 53 961.30 and ended at minute 54287.24. In total 2 198.20 kg were produced over 325.94 minutes.Both orders arrived the company at the minute 25 920.00.
- Job 455 arrived at the company in minute 33 120.00 and started production in minute 54 288.80. The weight total of order is 308.74 kg and it takes 9.35 minutes to produce, in die 4109_1. This order has a deadline of 90 720.00 minutes. Next orders use different die, so a setup time at 1.50 minute are existed.
- Job 414 and 413 whose release date is minute 27 360.00, began production on die 2649_1 in minute 54 299.60 and ended on minute 54 333.78, and a setup time at 1.5 minute are verified. During 34.16 minutes 998.95 kg were produced. Orders have a deadline of 82 080.00 minutes.
- Job 446 arrived at the company in minute 33 120.00 and started production in minute 54 335.30. The weight total of order is 1 164.00 kg and it takes 53.16 minutes to produce, in die 2723_1. This

order has a deadline of 108 000.00 minutes. Next orders use different die, so a setup time at 1.50 minute is existed.

- Job 441, 440 and 418, produced by die 2001_1, started at minute 54 390.00 and ended at minute 54 563.11. These orders have different release dates, i.e., Job 418 arrived at the company at minute 27 360.00, and the others at minute 33 120.00. The deadlines are also different, 82 080.00 and 108 000.00, respectively. A total 3 544.65 kg were produced for 173.13 minutes. In the end there is a setup time, which indicates that the next orders to be produced use different dies.

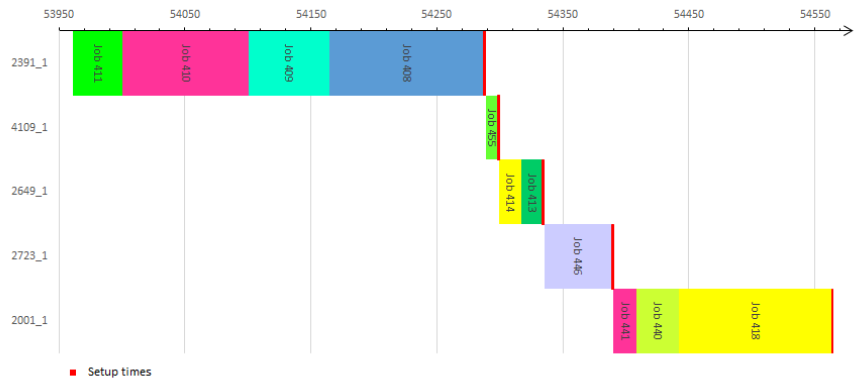


Figure 5.10: Solution, per die, from 11 produced order, from May 23rd to 29th.

5.5.3 Performance Comparison

Table 5.8 presents information relative to the total number of die changes, the total of setup times and time to produced first and last order, for the instance 23rd to 29th, for dispatching rule FIFO and developed model.

	Setup times	Die changes	First order Start time	First order End time	Last order Start time	Last order End time	Processing time
FIFO	79.50	53	25 980.00	26 018.40	34 960.72	34 972.58	2 503.77
OPT	75.00	50	33 180.00	33 185.72	97 876.30	97 919.97	2 503.77

Table 5.8: Comparison of results from instance May 23rd to 29th.

Applying the FIFO rule for the last instance there are 53 die changes with a total time of 79.50 minutes of setup times. Applying the developed model, there are 50 die changes with a total time of 75.00 minutes of setup times. In this instance, the developed model presents better results than the dispatching rule FIFO. Regarding production times, the same as in previous instances occurs.

5.6 Final Considerations

A total of three instances were solved. The results are summarized in Table 5.9. For all instances, the optimal solution was found in a small computational time. The setup times varied from 75.00 to 118.50. The problems presented between 4 805 and 9 251 variables binary and linear decision variables, mostly binary. The constraints range from 9 269 to 18 067, mostly inequalities.

	May 2nd to 8th	May 9nd to 29th	May 23rd to 29th
Objective function	118.50	103.50	75.00
Number of variables	9 251	6 939	4 805
binary	9156	6 857	4 737
linear	95	82	68
Number of constraints	18 067	13 473	9 269
equality	207	189	157
inequality	17860	13284	9 112

Table 5.9: Results and dimension of the developed model obtained for each instance.

Table 5.10 presents the results obtained for the instances, applying the FIFO method and the developed model.

Instance	Method	Setup	Die	First order		Last order	
		times	changes	Start time	End time	Start time	End time
May 2nd	FIFO	127.50	85	60.00	70.66	8 235.48	8 257.50
to 8th	OPT	118.50	79	1 500.00	1 544.85	54 721.50	54 735.78
	OPT[1]	118.50	79	8 640.00	8 662.63	13 749.00	13 763.20
May 9th	FIFO	106.50	71	8 700.00	8 735.79	17 572.74	17 584.39
to 15th	OPT	103.50	69	11 580.00	11 634.06	71 559.00	71 999.99
May 23rd	FIFO	79.50	53	25 980.00	26 018.40	34 960.72	34 972.58
to 29th	OPT	75.00	50	33 180.00	33 185.72	97 876.30	97 919.97

Table 5.10: Comparison of results from all instances.

The developed model, compared to the rule FIFO for instances considered, presents a smaller number of die changes and consequently a minor total setup time. The developed model only takes into consideration the minimization of setup times and does not require orders to be produced the closest possible to the release date. This is an improvement that could be implemented in the model by adding minimizing starting times to the objective function. However, this does not invalidate the results obtained, because all constraints are being met.

[1]Developed model for a $t_{prod} = 1\ 440$

In the first instance an extra comparison was made, i.e., to the developed model we changed a parameter provided by the company.

It was considered that each order could only be produced after one day of arriving at the company. The results in terms of number of die exchanges were the same as if it was consider that they can be produced 60 minutes after their arrival. However there was a better schedule on production times, and orders were produced continuously without interruption. This situation needed further study, because it could be an improvement in production planning for both the model and the company.

As already mentioned, it was not possible to make a direct comparison of the results obtained in this work and those practiced in the company. In Section 5.1 an average of orders and exchanges of die per week, considering the historical production data provided in the Production process dataset are computed. An average of 119 orders were produced and 79 die changes per week were made.

With the developed model, in any of the weeks it was exceeded 79 exchanges of die, per week. In the first instance, where the highest number of orders was produced, close to the estimated average, 79 die exchanges were made. In other instances lower values were verified.

In conclusion, the developed model presents a viable solution for production planning, in the order to decrease the level of scrap produced. It has the advantages of optimizing the use of dies and being an automatic tool. With this solver the production manager can more quickly set up production sequencing for incoming orders, and the responsiveness to the customer is better. Assuming that the production manager in a pre–processing phase, already includes in the deadline all conditions needed, such as treatment type, available dies, alloy type, etc.

Also in a pre–processing phase it is necessary to verified whether there are orders with weights above the set limit, and subdivided into weights below. The model presented is an advantageous solution for the company to sequence orders automatically, ensuring deadlines, minimizing die changes, and consequently reducing scrap.

Chapter 6

Conclusion and Future Work

Production scheduling is one of the major component for efficient management of production. This class of problems is among the most studied in Operational Research (OR) and presents applications in several areas [29]. Scheduling refers to the allocation of resources to jobs over a set of given time periods, optimizing one or more criteria [69], and are strongly NP-hard.

Although studied by many researchers, the majority of the literature ignores setup times/costs. However, in many applications it is crucial to consider setups, since they are essential due to scheduling decisions are made in order to increase productivity, eliminate waste, improve resource utilization, meet deadlines, increase customer satisfaction [3].

In this work, the challenge proposed by a Portuguese aluminium extrusion and treatment company was presented. The main objective of the company was to optimize the production sequencing in order to reduce scrap, while maintaining the quality standards of the extruded aluminium and respecting the deadline of delivery to the customers. Since, whenever a die exchange is made, a setup time occurs and scrap is produced. Therefore, an direct approach for minimizing the amount of scrap produced in the extrusion process is to minimize the die exchanges. For this reason and since there is no estimation on the amount of scrap production, this work focuses on minimizing the setup times.

This work, starts with a literature review on the industrial extrusion process and on production planning and scheduling. The problem proposed by the company and the information necessary for developing the mathematical model are also discussed. The characteristics of the production scenario implies that the dies must be changed for producing the aluminium profiles order by the costumers. These profiles have different geometries and sizes, and it is essential to consider setup times.

The company provided two datasets. The first dataset concerns the extrusion production process for a period of six months, while the second dataset concerns information of the customers' orders for one month.

A mathematical model, more precisely a MILP model, was developed. The objective was to minimize the sum of setup times, and therefore minimize the exchanges die, which consequently decrease the scrap produced. To this end, three real instances, taken from the Customers' orders dataset provided by the company regarding orders received in May 2018, were used to test the model.

Since the scheduling of orders practiced by the company for these instances is not known, a direct comparison is not possible. This represents a limitation of the present study. For having a comparison measure, the results obtained with the developed model were compared the results obtained using the dispatching rule FIFO.

It may be concluded that, for the real instances considered in this work, the developed model presents smaller setup times, and consequently a smaller number of die changes. Furthermore, the results obtained guarantee that the orders' are proceeded only with a single die on a set of allowed ones, all order in are produced, the minimum and maximum weight usage of each die is not exceeded.

As already mentioned it was not possible to make a direct comparison of the results found in the present work and the company's practice. However, by using the six month of historical production data in the production process dataset, provided by the company it was possible to observe an average of the number of orders and die exchanges, per week, that are made by the company. On average, on the company there are, per week, 119 orders and 79 die changes. In the results obtained with the developed model it was possible to verified that in any of the weeks the 79 exchanges of die were observed.

The approach presented in this paper has several advantages over the current procedure used in the company. Currently, scheduling is performed by the production manager by hand and the time it takes to find an admissible sequence for the orders' production is time consuming. In fact, it sequences the customers' orders automatically, ensuring deadlines, minimizing die changes, and consequently reducing scrap. For each instance, the necessary parameters, related to the order characteristics (i.e. identification of the order, the set of dies that can be used to produce the order, its weight and processing time, the release and deadline dates) are automatically extracted from an excel an written in a format that can be interpret by the solver. The optimal results of each instance are obtained automatically in a computational time that is by far smaller than the time that the production manager takes to find a solution by hand. Therefore, the approach here presented presents itself as an useful tool for decision support in the aluminum industry.

However, the model is far from being perfect. In fact, a closer analysis of the results obtained with the developed model showed the existence of dead times. This occurs because the model does not include in this formulation the minimization of dead times. There are several possibilities to overcame this limitation, such as, modifying the objective function such that it included a term for guaranteeing minimization of starting times. In this case it would be necessary to balance the contribution of the setup times and the starting times in the objective function, since they present quite different orders of magnitude. Another possibility is to consider a bi-objective model for minimizing the setups and starting times. Yet another different approach would be to consider additional constrains to the problem that obliges that the difference between the starting time of the last order, S_{max}, and the starting time of the first order, S_{min}, do not exceed a given percentage of the total processing time, i.e. $S_{max} - S_{min} \leq (1 + \lambda)\sum_{j=1}^{n} p_j$, for which the λ threshold must be tuned. Due to time limitations it was not possible to test the above approaches, but the work has already been started.

Another interesting extension of this work would be to study the problem presented in this work in the context of robust stochastic programming and robust optimization.

Appendix A

Gantt Charts

Figure A.1: Description, per die, of the orders production scheduling obtained by the dispatching rule FIFO from instance May 2nd to 8th.

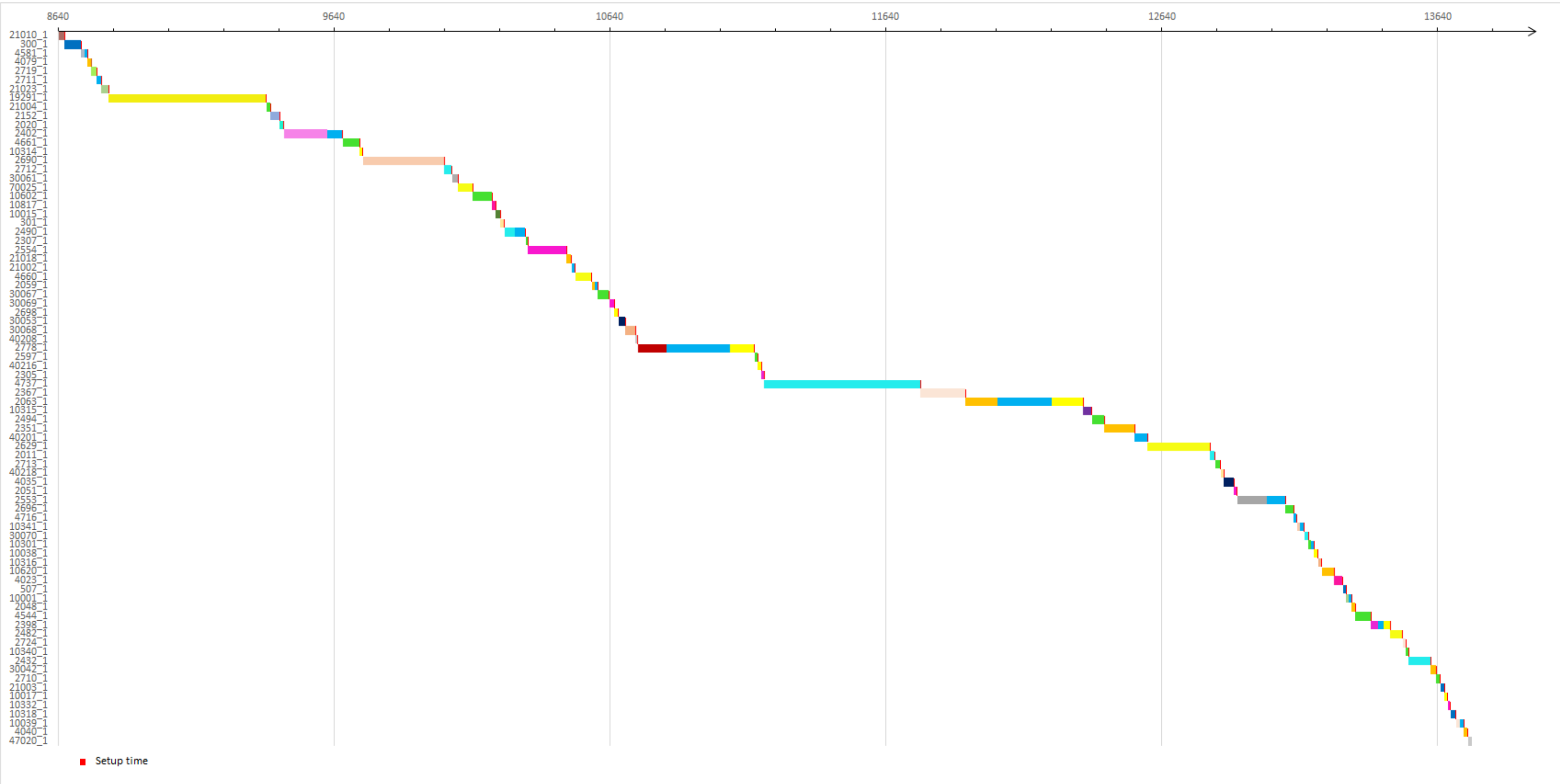


Figure A.2: Solution, per die, for the orders of the May 2nd to 8th, with β equal 1 440 minutes.

Figure A.3: Description, per die, of the orders production scheduling obtained by the dispatching rule FIFO from instance May 9th to 15th.

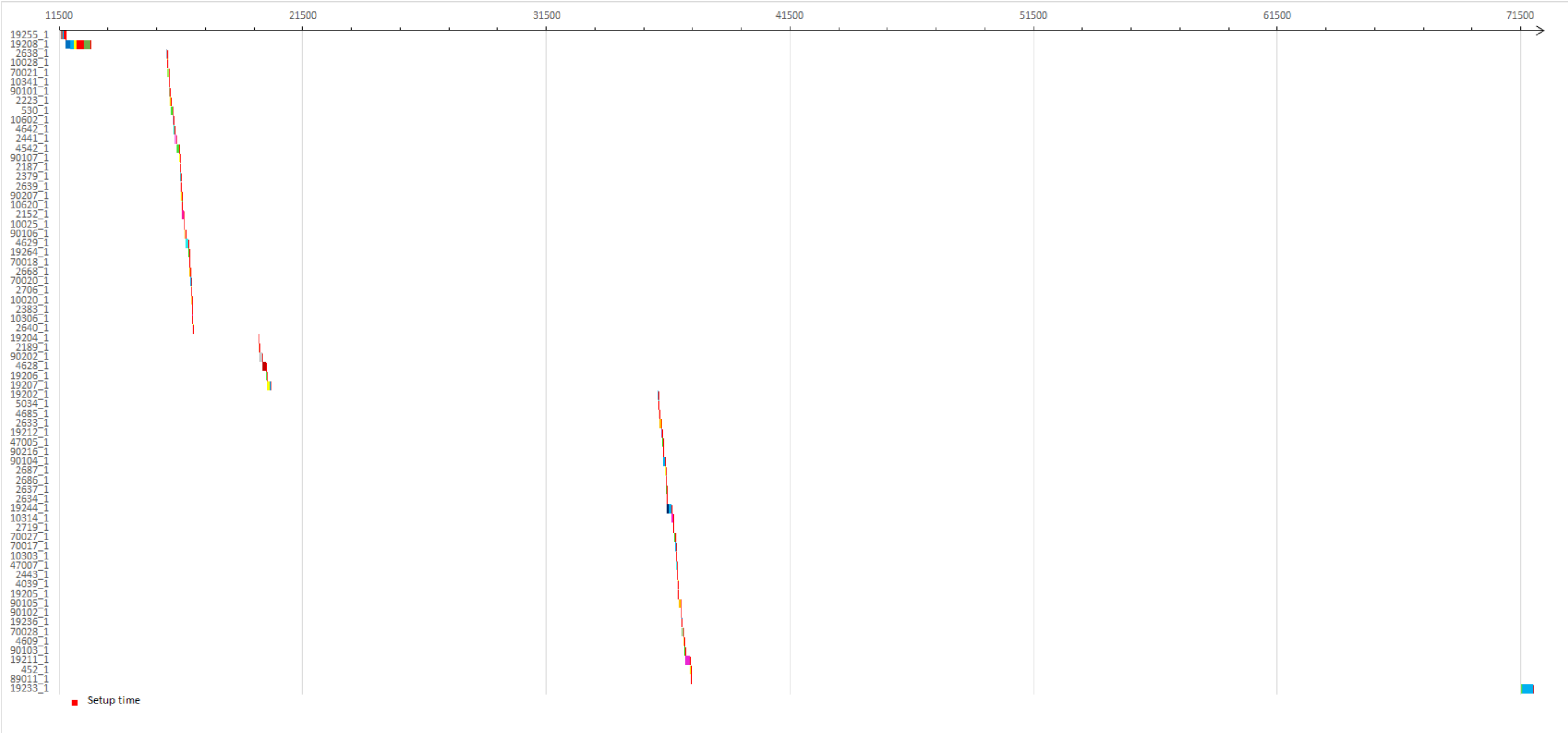


Figure A.4: Solution, per die, for the orders of the May 9th to 15th.

Figure A.5: Description, per die, of the orders production scheduling obtained by the dispatching rule FIFO from instance May 23rd to 29th.

Figure A.6: Solution, per die, for the orders of the May 23rd to 29th.

```
# - * - * - * - * - * - * - * - * - * - * -- * - * - * - * - * - * - * - * - * -
#                                    APPENDIX B
# - * - * - * - * - * - * - * - * - * - * -- * - * - * - * - * - * - * - * - * -

# - * - * - * - * - * - * - * - * - * - * -- * - * - * - * - * - * - * - * - * -
#                                    MODEL AMPL
# - * - * - * - * - * - * - * - * - * - * -

# - * - * - * - * - * - * - * - * - * - * -
#                 SETS
# - * - * - * - * - * - * - * - * - * - * -

# Set of orders
set Orders;

# Set of Dies that can be used to extrude order j
set M_j {Orders};

# Set of Die available
set Die:= union {j in Orders} M_j[j];

# - * - * - * - * - * - * - * - * - * - * -
#               PARAMETERS
# - * - * - * - * - * - * - * - * - * - * -

# Processing time of order j
param p_j {j in Orders};

# release date of order j, in minutes since 02/05/2018
param r_j {j in Orders};

# deadline of  order j, in minutes since 02/05/2018
param d_j {j in Orders};

# quantity, in kg, to extrude in order j
param c_j {j in Orders};

# maximum processing kg or time with die d
param DMax_d := 30000; #{d in Die};

# minimum processing kg or time with die d
param DMin_d :=  0; #{d in Die};

# time specified by the company for the order to each production, in min
param beta := 60;
param otherprocess := 24*60;
param MGrande := 10^10;

# tempo do setup
param S_jj{Orders, Orders};

# - * - * - * - * - * - * - * - * - * - * -
#            DECISION VARIABLES
# - * - * - * - * - * - * - * - * - * - * -
```

```
# is the starting time of order $j$
var s_j {j in Orders} >= 0;

# 1 j precede jl # <> -> significa diferente
var x_jj {j in (Orders union {0}), jl in (Orders union {0}):  j<>jl  } binary;

# 1 j precede jl # <> -> significa diferente
var x_jjd {j in Orders, jl in Orders, d in Die:  j<>jl } binary;

# 1 if die $d$ is used for order $j$
var m_jd {j in Orders, d in Die}  binary;

# - * - * - * - * - * - * - * - * - * - * -
#            OBJECTIVE FUNCTION
# - * - * - * - * - * - * - * - * - * - * -

#  Minimizing Setup Times

minimize f_obj:
        sum {j in Orders, jl in Orders:  j<>jl}  S_jj[j,jl]*x_jj[j,jl];

# - * - * - * - * - * - * - * - * - * - * -
#               CONSTRAINTS
# - * - * - * - * - * - * - * - * - * - * -

#1  Each Order can only be processed after "arriving"
s.t. Arrival {j in Orders}:
                                                        s_j[j] >= r_j[j]+beta;

#2
s.t. startingtime {j in Orders, jl in Orders: j<>jl }:
                        s_j[jl] >= s_j[j]-MGrande+ p_j[j]+MGrande*x_jj[j,jl]+
S_jj[j,jl]*x_jj[j,jl];

#3 each order is processed by a single die from among those that can be used
s.t. rest1 {j in Orders}:
                                        sum{d in M_j[j]} m_jd[j,d] = 1;

# the order cannot be processed by a matrix that is not in M_j
s.t. rest_nova {j in Orders}:
                        sum{d in (Die diff M_j[j])} m_jd[j,d] = 0;

#6
s.t. startingtime2 {j in Orders}:
                                        s_j[j]+p_j[j]+otherprocess <= d_j[j];


#
s.t. MaxMinWeight {d in Die}:
                                        DMin_d <= (sum {j in Orders:d in M_j[j]}
m_jd[j,d]*c_j[j]) <= DMax_d;

#5
```

```
s.t. precedencia {j in Orders, jl in Orders: j<>jl}:
                        x_jj[j,jl] + x_jj[jl,j]<=1;

s.t. resty {j in (Orders union {0})}:
                        (sum{jl in (Orders union {0}): j<>jl} x_jj[j,jl]) = 1;

s.t. resty2 {jl in (Orders union {0})}:
                        (sum{j in (Orders union {0}): j<>jl} x_jj[j,jl]) = 1;


s.t. resty2 {jl in Orders}:
                        (sum{j in Orders: j<>jl} x_jj[j,jl]) = 1;


s.t. resty {d in Die, j in Orders}:
                        (sum{ jl in Orders: j<>jl  and card(M_j[j] inter
M_j[jl])<>0} x_jj[j,jl]) = m_jd[j,d];

s.t. resty2 {d in Die, jl in Orders}:
                        (sum{ j in Orders: j<>jl}       x_jj[j,jl]) =
m_jd[jl,d];

#4
#s.t. rest2 {d in Die, j in Orders}:
#                       (sum{ jl in Orders: j<>jl}      x_jjd[j,jl,d]) <=
m_jd[j,d];

                #s.t. precedencia {j in Orders}:
#                       sum {jl in Orders, d in Die: j<>jl} x_jjd[j,jl,d] =1;
#

# - * - * - * - * - * - * - * - * - * - * -- * - * - * - * - * - * - * - * - * -
#                       DATA Instance 1 (May 2nd to 8th)
# - * - * - * - * - * - * - * - * - * - * -

set Orders :=  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95  ;

set M_j[1] := 4581_1  ;
set M_j[2] := 4581_1  ;
set M_j[3] := 4035_1  ;
set M_j[4] := 10301_1  ;
set M_j[5] := 10332_1  ;
set M_j[6] := 4023_1  ;
set M_j[7] := 4079_1  ;
set M_j[8] := 10015_1  ;
set M_j[9] := 10817_1  ;
set M_j[10] := 2778_1  ;
set M_j[11] := 2778_1  ;
set M_j[12] := 2778_1  ;
set M_j[13] := 2494_1  ;
set M_j[14] := 2351_1 2351_2  ;
```

```
set M_j[15] := 2553_1 2553_2  ;
set M_j[16] := 2553_1 2553_2  ;
set M_j[17] := 2554_1  ;
set M_j[18] := 2490_1  ;
set M_j[19] := 2490_1  ;
set M_j[20] := 2063_1  ;
set M_j[21] := 2063_1  ;
set M_j[22] := 2063_1  ;
set M_j[23] := 2629_1 2629_2  ;
set M_j[24] := 70025_1  ;
set M_j[25] := 40201_1 40201_2  ;
set M_j[26] := 40208_1  ;
set M_j[27] := 40218_1  ;
set M_j[28] := 40216_1  ;
set M_j[29] := 10314_1  ;
set M_j[30] := 10340_1  ;
set M_j[31] := 2724_1  ;
set M_j[32] := 10017_1  ;
set M_j[33] := 10301_1  ;
set M_j[34] := 10341_1  ;
set M_j[35] := 10039_1  ;
set M_j[36] := 10038_1  ;
set M_j[37] := 10001_1  ;
set M_j[38] := 10318_1  ;
set M_j[39] := 4660_1  ;
set M_j[40] := 4661_1  ;
set M_j[41] := 4716_1  ;
set M_j[42] := 2011_1  ;
set M_j[43] := 10620_1  ;
set M_j[44] := 2710_1  ;
set M_j[45] := 30068_1  ;
set M_j[46] := 30042_1  ;
set M_j[47] := 30053_1  ;
set M_j[48] := 30069_1 30069_2  ;
set M_j[49] := 30061_1  ;
set M_j[50] := 30067_1  ;
set M_j[51] := 30070_1  ;
set M_j[52] := 47020_1  ;
set M_j[53] := 2711_1 2711_2  ;
set M_j[54] := 2712_1  ;
set M_j[55] := 2713_1  ;
set M_j[56] := 10001_1  ;
set M_j[57] := 10039_1  ;
set M_j[58] := 2020_1  ;
set M_j[59] := 10316_1  ;
set M_j[60] := 2398_1 2398_2 2398_3 2398_4  ;
set M_j[61] := 2432_1  ;
set M_j[62] := 507_1  ;
set M_j[63] := 10341_1  ;
set M_j[64] := 19291_1  ;
set M_j[65] := 300_1  ;
set M_j[66] := 301_1  ;
set M_j[67] := 4544_1  ;
set M_j[68] := 2402_1 2402_2  ;
```

```
set M_j[69] := 2696_1  ;
set M_j[70] := 2698_1  ;
set M_j[71] := 2048_1  ;
set M_j[72] := 2402_1 2402_2  ;
set M_j[73] := 2307_1  ;
set M_j[74] := 2051_1  ;
set M_j[75] := 4737_1  ;
set M_j[76] := 2690_1  ;
set M_j[77] := 2719_1  ;
set M_j[78] := 2482_1  ;
set M_j[79] := 2367_1 2367_2  ;
set M_j[80] := 2597_1  ;
set M_j[81] := 4040_1  ;
set M_j[82] := 2059_1  ;
set M_j[83] := 2059_1  ;
set M_j[84] := 2305_1  ;
set M_j[85] := 2152_1 2152_2  ;
set M_j[86] := 10602_1  ;
set M_j[87] := 10315_1 10315_2  ;
set M_j[88] := 21010_1  ;
set M_j[89] := 21003_1  ;
set M_j[90] := 21004_1  ;
set M_j[91] := 21002_1  ;
set M_j[92] := 21023_1  ;
set M_j[93] := 21018_1  ;
set M_j[94] := 2398_1 2398_2 2398_3 2398_4  ;
set M_j[95] := 2398_1 2398_2 2398_3 2398_4  ;

param p_j :=
1   10.66294
2   10.66294
3   34.3367
4   10.53665
5   8.789786
6   30.25577
7   10.99138
8   14.61625
9   11.46727
10   104.1959
11   87.40049
12   230.1611
13   43.60334
14   108.1048
15   65.56749
16   108.0649
17   137.6118
18   35.50596
19   39.88648
20   196.0681
21   114.7443
22   114.7625
23   225.2053
24   51.40086
25   44.85341
```

26	7.947351
27	10.97595
28	13.04914
29	10.14951
30	8.907717
31	9.631804
32	10.02448
33	9.174276
34	12.09708
35	11.94924
36	13.31032
37	9.13314
38	18.2222
39	58.69931
40	59.14022
41	10.05259
42	16.54125
43	43.42484
44	14.2814
45	34.03441
46	18.58305
47	23.67932
48	17.17474
49	21.45786
50	39.3711
51	12.45729
52	14.19982
53	14.7993
54	25.9515
55	18.36969
56	8.319319
57	14.21624
58	13.09651
59	11.70876
60	25.08295
61	77.90976
62	10.43946
63	13.66471
64	570.191
65	58.06731
66	13.46324
67	55.0539
68	53.485
69	27.90257
70	13.83541
71	12.21496
72	158.724
73	7.446766
74	11.35283
75	565.1711
76	294.5054
77	19.88905
78	42.46876
79	161.264

```
80   8.727331
81   13.78486
82   9.911346
83   9.904068
84   8.590687
85   32.92109
86   68.27741
87   29.50234
88   22.63333
89   13.56596
90   13.22459
91   12.96265
92   24.15
93   16.76479
94   22.02403
95   22.02403
 ;
param r_j :=
1   0
2   0
3   0
4   0
5   0
6   0
7   0
8   0
9   0
10   1440
11   1440
12   1440
13   1440
14   1440
15   1440
16   1440
17   1440
18   1440
19   1440
20   1440
21   1440
22   1440
23   1440
24   1440
25   1440
26   1440
27   1440
28   1440
29   1440
30   1440
31   1440
32   1440
33   1440
34   1440
35   1440
36   1440
```

37	1440
38	1440
39	1440
40	1440
41	1440
42	2880
43	2880
44	2880
45	2880
46	2880
47	2880
48	2880
49	2880
50	2880
51	2880
52	2880
53	2880
54	2880
55	2880
56	2880
57	2880
58	2880
59	2880
60	2880
61	2880
62	2880
63	2880
64	5760
65	5760
66	5760
67	5760
68	5760
69	5760
70	5760
71	5760
72	5760
73	5760
74	5760
75	5760
76	5760
77	5760
78	5760
79	5760
80	5760
81	5760
82	5760
83	5760
84	5760
85	7200
86	7200
87	7200
88	7200
89	7200
90	7200

```
91   7200
92   7200
93   7200
94   7200
95   7200
 ;
param d_j :=
1   56160
2   56160
3   56160
4   56160
5   56160
6   56160
7   56160
8   56160
9   56160
10   56160
11   56160
12   56160
13   21600
14   21600
15   56160
16   56160
17   56160
18   56160
19   56160
20   56160
21   56160
22   56160
23   56160
24   56160
25   56160
26   56160
27   56160
28   56160
29   56160
30   56160
31   56160
32   56160
33   56160
34   56160
35   56160
36   56160
37   56160
38   56160
39   56160
40   56160
41   56160
42   56160
43   56160
44   73440
45   73440
46   73440
47   73440
```

```
48   73440
49   73440
50   73440
51   73440
52   73440
53   73440
54   73440
55   73440
56   73440
57   73440
58   73440
59   73440
60   73440
61   73440
62   56160
63   56160
64   12960
65   56160
66   56160
67   64800
68   73440
69   73440
70   73440
71   73440
72   73440
73   73440
74   73440
75   64800
76   64800
77   73440
78   73440
79   73440
80   47520
81   64800
82   73440
83   73440
84   73440
85   64800
86   64800
87   64800
88   73440
89   73440
90   73440
91   73440
92   73440
93   73440
94   64800
95   64800
 ;
param c_j :=
1   208.8
2   208.8
3   993.86
4   300.15
```

5	301.35
6	494.165
7	304.017
8	300.636
9	251.076
10	2278.2
11	1898.5
12	5125.95
13	1002.005
14	2498.514
15	1500.734
16	2501.018
17	2998.8
18	699.4933
19	799.771
20	3561.67
21	1780.264
22	1780.663
23	2500.588
24	300.196
25	1001.702
26	248.729
27	252.252
28	250.2825
29	249.3855
30	250.8155
31	248.4755
32	251.68
33	250.4125
34	301.3465
35	248.56
36	298.584
37	250.107
38	396.175
39	1156.272
40	683.1552
41	260.304
42	250.0225
43	1600.95
44	266.5
45	979.875
46	440.7
47	489.45
48	448.5
49	479.05
50	1064.7
51	292.5
52	408.85
53	488.15
54	722.215
55	616.2
56	227.37
57	310.7
58	265.85

```
59   294.775
60   566.8
61   1976
62   249.444
63   350.2135
64   9413.674
65   1830
66   939.6576
67   1205.406
68   500.1772
69   499.2988
70   249.429
71   254.2806
72   1499.839
73   249.555
74   249.4947
75   14051.4
76   6448.2
77   867.4575
78   1510.044
79   2817.72
80   250.0085
81   481.4145
82   250.2324
83   250
84   250
85   200.46
86   478.4
87   589.68
88   247
89   314.6
90   295.75
91   319.8
92   330
93   311.4
94   499.9176
95   499.9176
 ;
param S_jj : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 :=
1    0  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
2    0  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
```

```
3    1.5  1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
4    1.5  1.5  1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
5    1.5  1.5  1.5  1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
6    1.5  1.5  1.5  1.5  1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
7    1.5  1.5  1.5  1.5  1.5  1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
8    1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
9    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
10    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  0  0  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
11    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  0  0  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
```

12 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0 0 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
13 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
14 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
15 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
16 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
17 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
18 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
19 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
20 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

21 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
22 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
23 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
24 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
25 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
26 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
27 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
28 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
29 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

30 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
31 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
32 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
33 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
34 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
35 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
36 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
37 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
38 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

39 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
40 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
41 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
42 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
43 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
44 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
45 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
46 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
47 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

48 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
49 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
50 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
51 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
52 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
53 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
54 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
55 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
56 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

57 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
58 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
59 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
60 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0
61 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
62 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
63 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
64 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
65 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

66 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
67 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
68 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
69 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
70 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
71 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
72 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
73 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
74 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

75 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
76 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
77 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
78 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
79 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
80 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
81 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
82 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
83 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

84 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
85 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
86 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
87 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
88 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5
89 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5
90 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5
91 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5
92 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5

```
93    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  1.5  1.5
94    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  0
95    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  0
 ;

# - * - * - * - * - * - * - * - * - * - * -
#       SOLUTION AND VISUALIZATION
# - * - * - * - * - * - * - * - * - * - * -

#option solver gurobi;

solve;
option omit_zero_rows 1;
display f_obj;
display Orders;
display c_j, r_j, s_j, p_j, d_j;
display x_jj;
display m_jd;

# - * - * - * - * - * - * - * - * - * - * -- * - * - * - * - * - * - * - * - * -
#                           Instance 1 (May 2nd to 8th)
# - * - * - * - * - * - * - * - * - * - * -

Presolve eliminates 458 constraints and 8799 variables.
Adjusted problem:
9251 variables:
        9156 binary variables
        95 linear variables
18067 constraints, all linear; 62926 nonzeros
        207 equality constraints
        17860 inequality constraints
1 linear objective; 8894 nonzeros.

Gurobi 8.1.0: threads=4
Gurobi 8.1.0: optimal solution; objective 118.5
254963 simplex iterations
1067 branch-and-cut nodes
plus 91 simplex iterations for intbasis
f_obj = 118.5
```

```
set Orders :=
1    7    13   19   25   31   37   43   49   55   61   67   73   79   85   91
2    8    14   20   26   32   38   44   50   56   62   68   74   80   86   92
3    9    15   21   27   33   39   45   51   57   63   69   75   81   87   93
4    10   16   22   28   34   40   46   52   58   64   70   76   82   88   94
5    11   17   23   29   35   41   47   53   59   65   71   77   83   89   95
6    12   18   24   30   36   42   48   54   60   66   72   78   84   90;

:       c_j      r_j     s_j        p_j       d_j      :=
1       208.8      0    54696.9     10.6629   56160
2       208.8      0    54686.2     10.6629   56160
3       993.86     0    11814.4     34.3367   56160
4       300.15     0    12359.9     10.5366   56160
5       301.35     0    10923.8      8.78979  56160
6       494.165    0     9006.62    30.2558   56160
7       304.017    0     9038.38    10.9914   56160
8       300.636    0    11798.3     14.6163   56160
9       251.076    0    10910.9     11.4673   56160
10     2278.2   1440    19410.5    104.196    56160
11     1898.5   1440    19514.7     87.4005   56160
12     5125.95  1440    19602.1    230.161    56160
13     1002     1440     8803.28    43.6033   21600
14     2498.51  1440    20051.9    108.105    21600
15     1500.73  1440    11707       65.5675   56160
16     2501.02  1440    11599      108.065    56160
17     2998.8   1440     9575.17   137.612    56160
18      699.493 1440    11904.9     35.506    56160
19      799.771 1440    11865       39.8865   56160
20     3561.67  1440     9193.06   196.068    56160
21     1780.26  1440     9078.32   114.744    56160
22     1780.66  1440     9389.13   114.763    56160
23     2500.59  1440    11977.5    225.205    56160
24      300.196 1440     9805.96    51.4009   56160
25     1001.7   1440     1500       44.8534   56160
26      248.729 1440    11788.8      7.94735  56160
27      252.252 1440    54709       10.9759   56160
28      250.282 1440    11540.5     13.0491   56160
29      249.386 1440    10899.2     10.1495   56160
30      250.815 1440    19915.6      8.90772  56160
31      248.476 1440    10239.9      9.6318   56160
32      251.68  1440    12213.1     10.0245   56160
33      250.412 1440    12350.8      9.17428  56160
34      301.346 1440    19847.4     12.0971   56160
35      248.56  1440     8862.6     11.9492   56160
36      298.584 1440    11850.2     13.3103   56160
37      250.107 1440    11529.8      9.13314  56160
38      396.175 1440    19956.2     18.2222   56160
39     1156.27  1440     9727.49    58.6993   56160
40      683.155 1440    19991.3     59.1402   56160
41      260.304 1440    19944.6     10.0526   56160
42      250.023 2880    19861       16.5413   56160
43     1600.95  2880     9858.86    43.4248   56160
44      266.5   2880    54721.5     14.2814   73440
```

```
45     979.875   2880   11941.9     34.0344    73440
46     440.7     2880   10219.9     18.5831    73440
47     489.45    2880   10829.3     23.6793    73440
48     448.5     2880   19926       17.1747    73440
49     479.05    2880    9903.78    21.4579    73440
50    1064.7     2880   12372       39.3711    73440
51     292.5     2880    8932.59    12.4573    73440
52     408.85    2880   10934.1     14.1998    73440
53     488.15    2880   10251.1     14.7993    73440
54     722.215   2880    9050.87    25.9515    73440
55     616.2     2880    8912.72    18.3697    73440
56     227.37    2880   11521.5      8.31932   73440
57     310.7     2880    8848.38    14.2162    73440
58     265.85    2880   12312       13.0965    73440
59     294.775   2880    9714.28    11.7088    73440
60     566.8     2880   19383.9     25.083     73440
61    1976       2880   10140.5     77.9098    73440
62     249.444   2880   19903.6     10.4395    56160
63     350.214   2880   19833.8     13.6647    56160
64    9413.67    5760   10949.8    570.191     12960
65    1830       5760   12252.5     58.0673    56160
66     939.658   5760   12237.5     13.4632    56160
67    1205.41    5760   10563.4     55.0539    64800
68     500.177   5760    9926.74    53.485     73440
69     499.299   5760   12412.8     27.9026    73440
70     249.429   5760   19975.9     13.8354    73440
71     254.281   5760   10854.5     12.215     73440
72    1499.84    5760    9980.23   158.724     73440
73     249.555   5760   12204.2      7.44677   73440
74     249.495   5760   12224.6     11.3528    73440
75   14051.4     5760   12442.2    565.171     64800
76    6448.2     5760   10267.4    294.505     64800
77     867.457   5760    8876.05    19.8891    73440
78    1510.04    5760   11555       42.4688    73440
79    2817.72    5760   10619.9    161.264     73440
80     250.008   5760   10804        8.72733   47520
81     481.414   5760    8897.44    13.7849    64800
82     250.232   5760   10792.6      9.91135   73440
83     250       5760   10782.7      9.90407   73440
84     250       5760   19893.5      8.59069   73440
85     200.46    7200    8972.2     32.9211    64800
86     478.4     7200    9505.4     68.2774    64800
87     589.68    7200   10868.2     29.5023    64800
88     247       7200   12326.6     22.6333    73440
89     314.6     7200   10814.2     13.566     73440
90     295.75    7200   11774.1     13.2246    73440
91     319.8     7200   19879.1     12.9627    73440
92     330       7200    8946.55    24.15      73440
93     311.4     7200    9787.69    16.7648    73440
94     499.918   7200   19361.9     22.024     64800
95     499.918   7200   19339.9     22.024     64800
;

x_jj [*,*]
```

:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
:=																			
2	0	1	.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	.	1	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	1	0	0	0	0	.	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	1	0	0	0	.	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	.	1	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	.	1	0	0	0	0	0	0
14	0	0	1	0	0	0	0	0	0	0	0	0	0	0	.	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	.	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
25	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
26	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
33	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
44	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
85	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
86	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

:	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
:=																			
0	0	0	0	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	1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
20	0	.	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	1	.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.	0
37	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	.
45	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
54	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
62	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
73	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
87	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
88	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
90	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
93	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

:	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
:=																			
4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
18	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

34	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	1	0	0	.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	.	0	0	0	0	0	1	0	0	0	0	0	0	0
48	0	0	0	1	0	0	0	0	0	0	.	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	.	0
59	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
61	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
70	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
89	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

:	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
:=																			
12	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
13	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
49	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
52	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
65	0	1	0	0	0	0	0	0	.	0	0	0	0	0	0	0	0	0	0
66	0	0	0	0	0	0	0	0	1	.	0	0	0	0	0	0	0	0	0
68	0	0	0	0	0	0	0	0	0	0	0	.	0	0	0	1	0	0	0
69	0	0	0	0	0	0	0	0	0	0	0	0	.	0	0	0	0	0	1
72	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	.	0	0	0
74	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	.	0
76	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
84	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
94	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

:	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94
:=																			
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
28	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
53	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
58	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
67	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
71	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
77	0	.	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
79	0	0	0	.	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	.	0	0	0	0	0	0	0	0	1	0	0	0	0	0
82	0	0	0	0	1	0	.	0	0	0	0	0	0	0	0	0	0	0	0
83	0	0	0	0	0	0	1	.	0	0	0	0	0	0	0	0	0	0	0
91	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	.	0	0	0
92	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	.	0	0

```
95   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1

:   95    :=
75   1
;

m_jd [*,*]
# $1 = 10001_1
# $2 = 10015_1
# $3 = 10017_1
# $4 = 10038_1
# $5 = 10039_1
# $6 = 10301_1
# $7 = 10314_1
# $8 = 10315_1
# $9 = 10315_2
# $10 = 10316_1
# $11 = 10318_1
# $12 = 10332_1
# $13 = 10340_1
# $14 = 10341_1
# $15 = 10602_1
# $16 = 10620_1
# $17 = 10817_1
# $18 = 19291_1
# $19 = 2011_1
:   $1  $2  $3  $4  $5  $6  $7  $8  $9 $10 $11 $12 $13 $14 $15 $16 $17 $18 $19
:=
4    0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
5    0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
8    0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
9    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
29   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
30   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
32   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
33   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
34   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
35   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
36   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
37   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
38   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
42   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
43   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
56   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
57   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
59   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
63   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
64   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
86   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
87   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0

# $1 = 2020_1
# $2 = 2048_1
# $3 = 2051_1
```

```
# $4 = 2059_1
# $5 = 2063_1
# $6 = 21002_1
# $7 = 21003_1
# $8 = 21004_1
# $9 = 21010_1
# $10 = 21018_1
# $11 = 21023_1
# $12 = 2152_1
# $13 = 2152_2
# $14 = 2305_1
# $15 = 2307_1
# $16 = 2351_1
# $17 = 2351_2
# $18 = 2367_1
# $19 = 2367_2
:   $1  $2  $3  $4  $5  $6  $7  $8  $9 $10 $11 $12 $13 $14 $15 $16 $17 $18 $19
:=
14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
20   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
21   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
22   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
58   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
71   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
73   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
74   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
79   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
82   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
83   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
84   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
85   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
88   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
89   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
90   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
91   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
92   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
93   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0

# $1 = 2398_1
# $2 = 2398_2
# $3 = 2398_3
# $4 = 2398_4
# $5 = 2402_1
# $6 = 2402_2
# $7 = 2432_1
# $8 = 2482_1
# $9 = 2490_1
# $10 = 2494_1
# $11 = 2553_1
# $12 = 2553_2
# $13 = 2554_1
# $14 = 2597_1
# $15 = 2629_1
# $16 = 2629_2
```

```
# $17 = 2690_1
# $18 = 2696_1
# $19 = 2698_1
:   $1  $2  $3  $4  $5  $6  $7  $8  $9 $10 $11 $12 $13 $14 $15 $16 $17 $18 $19
:=
13   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
15   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
16   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
17   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
18   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
19   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
23   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
60   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
61   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
68   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
69   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
70   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
72   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
76   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
78   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
80   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
94   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
95   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

# $1 = 2710_1
# $2 = 2711_1
# $3 = 2711_2
# $4 = 2712_1
# $5 = 2713_1
# $6 = 2719_1
# $7 = 2724_1
# $8 = 2778_1
# $9 = 30042_1
# $10 = 30053_1
# $11 = 30061_1
# $12 = 30067_1
# $13 = 30068_1
# $14 = 30069_1
# $15 = 30069_2
# $16 = 30070_1
# $17 = 300_1
# $18 = 301_1
# $19 = 40201_1
:   $1  $2  $3  $4  $5  $6  $7  $8  $9 $10 $11 $12 $13 $14 $15 $16 $17 $18 $19
:=
10   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
11   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
12   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
25   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
31   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
44   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
45   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
46   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
47   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
```

```
48   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
49   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
50   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
51   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
53   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
54   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
55   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
65   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
66   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
77   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0

# $1 = 40201_2
# $2 = 40208_1
# $3 = 40216_1
# $4 = 40218_1
# $7 = 4040_1
# $8 = 4079_1
# $9 = 4544_1
# $10 = 4581_1
# $11 = 4660_1
# $12 = 4661_1
# $13 = 47020_1
# $14 = 4716_1
# $15 = 4737_1
# $17 = 70025_1
:   $1  $2  $3  $4 4023_1 4035_1  $7  $8  $9 $10 $11 $12 $13 $14 $15 507_1 $17
:=
1    0   0   0   0    0      0     0   0   0   1   0   0   0   0   0    0    0
2    0   0   0   0    0      0     0   0   0   1   0   0   0   0   0    0    0
3    0   0   0   0    0      1     0   0   0   0   0   0   0   0   0    0    0
6    0   0   0   0    1      0     0   0   0   0   0   0   0   0   0    0    0
7    0   0   0   0    0      0     0   1   0   0   0   0   0   0   0    0    0
24   0   0   0   0    0      0     0   0   0   0   0   0   0   0   0    0    1
26   0   1   0   0    0      0     0   0   0   0   0   0   0   0   0    0    0
27   0   0   0   1    0      0     0   0   0   0   0   0   0   0   0    0    0
28   0   0   1   0    0      0     0   0   0   0   0   0   0   0   0    0    0
39   0   0   0   0    0      0     0   0   0   0   1   0   0   0   0    0    0
40   0   0   0   0    0      0     0   0   0   0   0   1   0   0   0    0    0
41   0   0   0   0    0      0     0   0   0   0   0   0   0   1   0    0    0
52   0   0   0   0    0      0     0   0   0   0   0   0   1   0   0    0    0
62   0   0   0   0    0      0     0   0   0   0   0   0   0   0   0    1    0
67   0   0   0   0    0      0     0   0   1   0   0   0   0   0   0    0    0
75   0   0   0   0    0      0     0   0   0   0   0   0   0   0   1    0    0
81   0   0   0   0    0      0     1   0   0   0   0   0   0   0   0    0    0
;

# - * - * - * - * - * - * - * - * - * - * -- * - * - * - * - * - * - * - * - * -
# - * - * - * - * - * - * - * - * - * - * -- * - * - * - * - * - * - * - * - * -
#                       DATA Instance 2 (May 9th to 15th)
# - * - * - * - * - * - * - * - * - * - * -

set Orders :=  96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112
113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132
133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152
```

```
153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172
173 174 175 176 177  ;

set M_j[96] := 4642_1  ;
set M_j[97] := 10303_1 10303_2  ;
set M_j[98] := 10341_1  ;
set M_j[99] := 10025_1  ;
set M_j[100] := 10028_1 10028_2  ;
set M_j[101] := 2383_1  ;
set M_j[102] := 2633_1  ;
set M_j[103] := 4609_1  ;
set M_j[104] := 19207_1  ;
set M_j[105] := 2152_1 2152_2  ;
set M_j[106] := 10620_1  ;
set M_j[107] := 2187_1  ;
set M_j[108] := 2223_1  ;
set M_j[109] := 2189_1  ;
set M_j[110] := 2379_1  ;
set M_j[111] := 2634_1  ;
set M_j[112] := 10020_1 10020_2  ;
set M_j[113] := 19202_1 19202_2  ;
set M_j[114] := 19202_1 19202_2  ;
set M_j[115] := 19211_1  ;
set M_j[116] := 5034_1  ;
set M_j[117] := 19204_1  ;
set M_j[118] := 19205_1  ;
set M_j[119] := 19206_1  ;
set M_j[120] := 19212_1  ;
set M_j[121] := 19233_1 19233_2  ;
set M_j[122] := 19233_1 19233_2  ;
set M_j[123] := 19236_1  ;
set M_j[124] := 19244_1  ;
set M_j[125] := 19244_1  ;
set M_j[126] := 19255_1 19255_2  ;
set M_j[127] := 19255_1 19255_2  ;
set M_j[128] := 19255_1 19255_2  ;
set M_j[129] := 19255_1 19255_2  ;
set M_j[130] := 19255_1 19255_2  ;
set M_j[131] := 19208_1 19208_2 19208_3  ;
set M_j[132] := 19208_1 19208_2 19208_3  ;
set M_j[133] := 19208_1 19208_2 19208_3  ;
set M_j[134] := 19208_1 19208_2 19208_3  ;
set M_j[135] := 19264_1  ;
set M_j[136] := 19208_1 19208_2 19208_3  ;
set M_j[137] := 89011_1  ;
set M_j[138] := 4039_1  ;
set M_j[139] := 19207_1  ;
set M_j[140] := 10306_1  ;
set M_j[141] := 2687_1  ;
set M_j[142] := 2686_1  ;
set M_j[143] := 2706_1  ;
set M_j[144] := 2719_1  ;
set M_j[145] := 2668_1  ;
set M_j[146] := 10314_1  ;
```

```
set M_j[147] := 4685_1  ;
set M_j[148] := 10602_1  ;
set M_j[149] := 2441_1  ;
set M_j[150] := 4628_1  ;
set M_j[151] := 4629_1  ;
set M_j[152] := 2443_1  ;
set M_j[153] := 47007_1  ;
set M_j[154] := 47005_1  ;
set M_j[155] := 70017_1  ;
set M_j[156] := 70018_1 70018_2  ;
set M_j[157] := 70028_1  ;
set M_j[158] := 70027_1 70027_2  ;
set M_j[159] := 70021_1 70021_2  ;
set M_j[160] := 70020_1 70020_2  ;
set M_j[161] := 90101_1  ;
set M_j[162] := 90102_1  ;
set M_j[163] := 90103_1  ;
set M_j[164] := 90104_1  ;
set M_j[165] := 90105_1  ;
set M_j[166] := 90106_1  ;
set M_j[167] := 90107_1  ;
set M_j[168] := 90202_1 90202_2  ;
set M_j[169] := 90207_1  ;
set M_j[170] := 90216_1  ;
set M_j[171] := 452_1  ;
set M_j[172] := 530_1  ;
set M_j[173] := 4542_1  ;
set M_j[174] := 2637_1  ;
set M_j[175] := 2638_1  ;
set M_j[176] := 2639_1  ;
set M_j[177] := 2640_1  ;

param p_j :=
96   35.7947
97   13.86693
98   18.58867
99   16.65896
100   16.89655
101   13.25166
102   88.68128
103   47.24854
104   114.8753
105   49.77128
106   18.92034
107   26.1545
108   15.81302
109   29.86206
110   16.28231
111   25.65468
112   19.26041
113   7.331263
114   41.40627
115   192.7507
116   14.20616
```

117	20.87502
118	22.24842
119	52.15229
120	35.72907
121	24.87886
122	440.9867
123	23.53082
124	95.53954
125	78.74459
126	51.64094
127	54.05556
128	9.521329
129	29.36398
130	31.59707
131	283.7264
132	183.9778
133	137.3561
134	137.3561
135	34.0647
136	283.7264
137	13.00493
138	12.44395
139	18.66251
140	9.220029
141	26.39253
142	18.85519
143	9.535127
144	12.76163
145	7.99733
146	83.11176
147	13.53641
148	35.1599
149	45.65583
150	148.1162
151	106.1596
152	25.75153
153	22.5908
154	12.17489
155	16.28157
156	16.12046
157	50.52824
158	65.37407
159	59.37108
160	51.27824
161	43.5852
162	29.44251
163	47.0337
164	56.56225
165	81.15084
166	70.7052
167	16.56541
168	104.1056
169	27.29003
170	30.68519

```
171   27.18896
172   94.44909
173   131.2079
174   11.00786
175   10.04344
176   10.96737
177   11.65062
 ;
param r_j :=
96   8640
97   8640
98   8640
99   8640
100   8640
101   8640
102   8640
103   8640
104   10080
105   10080
106   10080
107   10080
108   10080
109   10080
110   10080
111   10080
112   10080
113   11520
114   11520
115   11520
116   11520
117   11520
118   11520
119   11520
120   11520
121   11520
122   11520
123   11520
124   11520
125   11520
126   11520
127   11520
128   11520
129   11520
130   11520
131   11520
132   11520
133   11520
134   11520
135   11520
136   11520
137   11520
138   11520
139   14400
140   14400
```

```
141   15840
142   15840
143   15840
144   15840
145   15840
146   15840
147   15840
148   15840
149   15840
150   15840
151   15840
152   15840
153   15840
154   15840
155   15840
156   15840
157   15840
158   15840
159   15840
160   15840
161   15840
162   15840
163   15840
164   15840
165   15840
166   15840
167   15840
168   15840
169   15840
170   15840
171   15840
172   15840
173   15840
174   15840
175   15840
176   15840
177   15840
 ;
param d_j :=
96   64800
97   64800
98   64800
99   64800
100   64800
101   64800
102   73440
103   64800
104   21600
105   30240
106   64800
107   64800
108   64800
109   64800
110   64800
```

111	73440
112	64800
113	38880
114	38880
115	38880
116	38880
117	73440
118	73440
119	73440
120	73440
121	73440
122	73440
123	73440
124	73440
125	73440
126	73440
127	73440
128	73440
129	73440
130	73440
131	73440
132	73440
133	73440
134	73440
135	38880
136	73440
137	38880
138	64800
139	21600
140	30240
141	73440
142	73440
143	125280
144	82080
145	82080
146	38880
147	73440
148	73440
149	73440
150	73440
151	73440
152	38880
153	38880
154	38880
155	38880
156	38880
157	38880
158	38880
159	38880
160	38880
161	38880
162	38880
163	38880
164	38880

```
165   38880
166   38880
167   38880
168   38880
169   38880
170   38880
171   64800
172   73440
173   38880
174   73440
175   73440
176   73440
177   73440
 ;
param c_j :=
96   999.096
97   398.208
98   503.706
99   498.96
100   502.26
101   300
102   2020.95
103   1376.55
104   2550.528
105   300.69
106   822.642
107   700.28
108   295.85
109   454.45
110   334.768
111   1001.25
112   421.889
113   215.1028
114   1348.81
115   4166.01
116   445.284
117   531.2006
118   576.1822
119   871.2704
120   1001.123
121   455.4396
122   8421.197
123   638.88
124   2226.217
125   1752.3
126   1378.416
127   1451.736
128   99.45
129   701.974
130   769.782
131   5679.18
132   3669.624
133   2730.375
134   2730.375
```

```
135   919.524
136   5679.18
137   267.54
138   504.0693
139   431.3088
140   196.746
141   608.4
142   488.28
143   280
144   315.7164
145   264.384
146   2001.348
147   398.736
148   249.665
149   999.635
150   4870
151   2865.432
152   622.05
153   425.49
154   486.85
155   435.5
156   410.8
157   1148.55
158   1263.6
159   1248
160   1084.2
161   1210.625
162   856.7
163   1635.4
164   1664
165   1838.2
166   1826.5
167   362.7
168   2496
169   716.3
170   422.5
171   253.536
172   2998.198
173   2872.8
174   308.22
175   300.162
176   306.84
177   308.256
 ;
param S_jj : 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113
114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133
134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153
154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173
174 175 176 177 :=
96    0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
```

1.5 1.5 1.5
97 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
98 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
99 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
100 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
101 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
102 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
103 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
104 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
105 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

```
1.5  1.5  1.5
106    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5
107    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5
108    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5
109    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5
110    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5
111    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5
112    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5
113    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  0  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5
114    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  0  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
```

1.5 1.5
115   1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
116   1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
117   1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
118   1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
119   1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
120   1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
121   1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0 1.5 1.5 1.5 1.5 1.5
 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5
122   1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0 1.5 1.5 1.5 1.5 1.5
 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5
123   1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

```
1.5  1.5  1.5
124    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  0  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5
125    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  0  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5
126    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  0
 0  0  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5
127    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  0
 0  0  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5
128    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  0
 0  0  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5
129    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  0
 0  0  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5
130    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  0
 0  0  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5
131    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  0  0  0  0  1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5
132    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  0  0  0  0  1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
```

1.5
133 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 0 0 0 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5
134 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 0 0 0 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5
135 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
136 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 0 0 0 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5
137 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
138 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
139 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
140 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
141 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

1.5 1.5 1.5
142 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
143 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
144 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
145 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
146 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
147 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
148 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
149 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
150 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

1.5 1.5 1.5
151 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
152 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
153 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
154 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
155 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
156 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
157 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
158 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
159 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

1.5 1.5 1.5
160 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
161 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
162 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
163 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
164 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
165 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
166 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
167 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
168 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5

1.5 1.5 1.5
169 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
170 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5
1.5 1.5 1.5
171 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5
1.5 1.5 1.5
172 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5
1.5 1.5 1.5
173 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5
1.5 1.5 1.5
174 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0
1.5 1.5 1.5
175 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
0 1.5 1.5
176 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 0 1.5
177 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

```
1.5  1.5  0
 ;

# - * - * - * - * - * - * - * - * - * - * -
#       SOLUTION AND VISUALIZATION
# - * - * - * - * - * - * - * - * - * - * -

#option solver gurobi;

solve;
option omit_zero_rows 1;
display f_obj;
display Orders;
display c_j, r_j, s_j, p_j, d_j;
display x_jj;
display m_jd;

# - * - * - * - * - * - * - * - * - * - * -- * - * - * - * - * - * - * - * - * -
#                          Instance 2 (May 9th to 15th)
# - * - * - * - * - * - * - * - * - * - * -

Presolve eliminates 389 constraints and 6837 variables.
Adjusted problem:
6939 variables:
        6857 binary variables
        82 linear variables
13473 constraints, all linear; 46873 nonzeros
        189 equality constraints
        13284 inequality constraints
1 linear objective; 6594 nonzeros.

Gurobi 8.1.0: threads=4
Gurobi 8.1.0: optimal solution; objective 103.5
5022516 simplex iterations
235147 branch-and-cut nodes
plus 80 simplex iterations for intbasis
f_obj = 103.5

set Orders :=
96    103   110   117   124   131   138   145   152   159   166   173
97    104   111   118   125   132   139   146   153   160   167   174
98    105   112   119   126   133   140   147   154   161   168   175
99    106   113   120   127   134   141   148   155   162   169   176
100   107   114   121   128   135   142   149   156   163   170   177
101   108   115   122   129   136   143   150   157   164   171
102   109   116   123   130   137   144   151   158   165   172;

:       c_j       r_j     s_j        p_j       d_j      :=
96     999.096    8640   16205.9    35.7947    64800
97     398.208    8640   36811.7    13.8669    64800
98     503.706    8640   15990.8    18.5887    64800
99     498.96     8640   16599.5    16.659     64800
100    502.26     8640   15911.5    16.8966    64800
101    300        8640   16944.8    13.2517    64800
```

102	2020.95	8640	36133.2	88.6813	73440
103	1376.55	8640	37106.8	47.2485	64800
104	2550.53	10080	20026.5	114.875	21600
105	300.69	10080	16548.2	49.7713	30240
106	822.642	10080	16527.8	18.9203	64800
107	700.28	10080	16441.1	26.1545	64800
108	295.85	10080	16056	15.813	64800
109	454.45	10080	19686.2	29.8621	64800
110	334.768	10080	16468.8	16.2823	64800
111	1001.25	10080	36425.3	25.6547	73440
112	421.889	10080	16924	19.2604	64800
113	215.103	11520	36052.2	7.33126	38880
114	1348.81	11520	36059.5	41.4063	38880
115	4166.01	11520	37204.1	192.751	38880
116	445.284	11520	36102.4	14.2062	38880
117	531.201	11520	19663.9	20.875	73440
118	576.182	11520	36892.4	22.2484	73440
119	871.27	11520	19972.8	52.1523	73440
120	1001.12	11520	36223.3	35.7291	73440
121	455.44	11520	71534.1	24.8789	73440
122	8421.2	11520	71559	440.987	73440
123	638.88	11520	37029.7	23.5308	73440
124	2226.22	11520	36531.2	95.5395	73440
125	1752.3	11520	36452.4	78.7446	73440
126	1378.42	11520	11695	51.6409	73440
127	1451.74	11520	11580	54.0556	73440
128	99.45	11520	11746.7	9.52133	73440
129	701.974	11520	11665.7	29.364	73440
130	769.782	11520	11634.1	31.5971	73440
131	5679.18	11520	12216.4	283.726	73440
132	3669.62	11520	11757.7	183.978	73440
133	2730.38	11520	11941.7	137.356	73440
134	2730.38	11520	12079	137.356	73440
135	919.524	11520	16797.5	34.0647	38880
136	5679.18	11520	12500.1	283.726	73440
137	267.54	11520	37427	13.0049	38880
138	504.069	11520	36878.4	12.4439	64800
139	431.309	14400	20141.3	18.6625	21600
140	196.746	14400	16959.5	9.22003	30240
141	608.4	15840	36364.5	26.3925	73440
142	488.28	15840	36392.4	18.8552	73440
143	280	15840	16913	9.53513	125280
144	315.716	15840	36712.8	12.7616	82080
145	264.384	15840	16850.7	7.99733	82080
146	2001.35	15840	36628.2	83.1118	38880
147	398.736	15840	36118.1	13.5364	73440
148	249.665	15840	16169.2	35.1599	73440
149	999.635	15840	16243.2	45.6558	73440
150	4870	15840	19823.2	148.116	73440
151	2865.43	15840	16689.9	106.16	73440
152	622.05	15840	36851.2	25.7515	38880
153	425.49	15840	36827.1	22.5908	38880
154	486.85	15840	36260.6	12.1749	38880
155	435.5	15840	36793.9	16.2816	38880

```
156    410.8     15840   16833.1    16.1205     38880
157   1148.55    15840   37054.7    50.5282     38880
158   1263.6     15840   36727.1    65.3741     38880
159   1248       15840   15929.9    59.3711     38880
160   1084.2     15840   16860.2    51.2782     38880
161   1210.62    15840   16010.9    43.5852     38880
162    856.7     15840   36998.8    29.4425     38880
163   1635.4     15840   37155.5    47.0337     38880
164   1664       15840   36306.4    56.5622     38880
165   1838.2     15840   36916.1    81.1508     38880
166   1826.5     15840   16617.7    70.7052     38880
167    362.7     15840   16423.1    16.5654     38880
168   2496       15840   19717.6   104.106      38880
169    716.3     15840   16499      27.29       38880
170    422.5     15840   36274.2    30.6852     38880
171    253.536   15840   37398.3    27.189      64800
172   2998.2     15840   16073.3    94.4491     73440
173   2872.8     15840   16290.4   131.208      38880
174    308.22    15840   36412.7    11.0079     73440
175    300.162   15840   15900      10.0434     73440
176    306.84    15840   16486.6    10.9674     73440
177    308.256   15840   16970.3    11.6506     73440
;

x_jj [*,*]
:     0  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113
:=
105   0   0   0   0   1   0   0   0   0   0   .   0   0   0   0   0   0   0   0
106   0   0   0   0   0   0   0   0   0   0   1   .   0   0   0   0   0   0   0
107   0   0   0   0   0   0   0   0   0   0   0   0   .   0   0   1   0   0   0
112   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   .   0
117   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
119   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
122   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
139   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
143   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
147   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
148   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
155   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
157   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
159   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
161   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
167   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
169   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
174   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
175   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0

:   114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132
:=
0     0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
102   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
111   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
113   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
114   .   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

121	0	0	0	0	0	0	0	.	1	0	0	0	0	0	0	0	0	0	0
125	0	0	0	0	0	0	0	0	0	0	1	.	0	0	0	0	0	0	0
126	0	0	0	0	0	0	0	0	0	0	0	0	.	0	1	0	0	0	0
127	0	0	0	0	0	0	0	0	0	0	0	0	0	.	0	0	1	0	0
128	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.	0	0	0	1
129	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	.	0	0	0
130	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	.	0	0
134	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
137	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
138	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
150	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
162	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
163	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
177	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

:	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151
:=																			
96	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
101	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
104	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
116	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
124	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
131	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
132	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
133	.	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
141	0	0	0	0	0	0	0	0	.	1	0	0	0	0	0	0	0	0	0
146	0	0	0	0	0	0	0	0	0	0	0	1	0	.	0	0	0	0	0
151	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.
152	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
156	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
160	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
164	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
166	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
168	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
171	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
172	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

:	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170
:=																			
97	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
98	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
100	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
103	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
109	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
118	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
120	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
123	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
135	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
144	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
145	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
153	1	.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
154	0	0	.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
158	0	0	0	1	0	0	.	0	0	0	0	0	0	0	0	0	0	0	0

```
165    0   0   0   0   0   0   0   0   0   0   1   0   0   .   0   0   0   0   0
170    0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   .
173    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
176    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0

:   171 172 173 174 175 176 177     :=
108    0   1   0   0   0   0   0
110    0   0   0   0   0   1   0
115    1   0   0   0   0   0   0
136    0   0   0   0   1   0   0
140    0   0   0   0   0   0   1
142    0   0   0   1   0   0   0
149    0   0   1   0   0   0   0
;

m_jd [*,*] (tr)
:         96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113
:=
10020_1    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
10025_1    0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
10028_1    0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
10303_1    0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
10341_1    0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
10620_1    0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
19202_1    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
19207_1    0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
2152_1     0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
2187_1     0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
2189_1     0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
2223_1     0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
2379_1     0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
2383_1     0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
2633_1     0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
2634_1     0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
4609_1     0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
4642_1     1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

:        114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131
:=
19202_1    1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
19204_1    0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
19205_1    0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
19206_1    0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
19208_1    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
19211_1    0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
19212_1    0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
19233_1    0   0   0   0   0   0   0   1   1   0   0   0   0   0   0   0   0   0
19236_1    0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
19244_1    0   0   0   0   0   0   0   0   0   0   1   1   0   0   0   0   0   0
19255_1    0   0   0   0   0   0   0   0   0   0   0   0   1   1   1   1   1   0
5034_1     0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

:        132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149
:=
```

```
10306_1   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
10314_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
10602_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
19207_1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
19208_1   1   1   1   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
19264_1   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
2441_1    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
2668_1    0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
2686_1    0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
2687_1    0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
2706_1    0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
2719_1    0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
4039_1    0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
4685_1    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
89011_1   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0

:       150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167
:=
2443_1    0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
4628_1    1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
4629_1    0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
47005_1   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
47007_1   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
70017_1   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
70018_1   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
70020_1   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
70021_1   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
70027_1   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
70028_1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
90101_1   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
90102_1   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
90103_1   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
90104_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
90105_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
90106_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
90107_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1

:       168 169 170 171 172 173 174 175 176 177     :=
2637_1    0   0   0   0   0   0   1   0   0   0
2638_1    0   0   0   0   0   0   0   1   0   0
2639_1    0   0   0   0   0   0   0   0   1   0
2640_1    0   0   0   0   0   0   0   0   0   1
452_1     0   0   0   1   0   0   0   0   0   0
4542_1    0   0   0   0   0   1   0   0   0   0
530_1     0   0   0   0   1   0   0   0   0   0
90202_1   1   0   0   0   0   0   0   0   0   0
90207_1   0   1   0   0   0   0   0   0   0   0
90216_1   0   0   1   0   0   0   0   0   0   0
;
# - * - * - * - * - * - * - * - * - * - * -- * - * - * - * - * - * - * - * - * -
# - * - * - * - * - * - * - * - * - * - * -- * - * - * - * - * - * - * - * - * -
#                       DATA Instance 3 (May 23rd to 29th)
# - * - * - * - * - * - * - * - * - * - * -
```

```
set Orders :=  402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417
418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437
438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457
458 459 460 461 462 463 464 465 466 467 468 469  ;

set M_j[402] := 2328_1  ;
set M_j[403] := 2328_1  ;
set M_j[404] := 2329_1  ;
set M_j[405] := 2329_1  ;
set M_j[406] := 2328_1  ;
set M_j[407] := 2329_1  ;
set M_j[408] := 2391_1 2391_2  ;
set M_j[409] := 2391_1 2391_2  ;
set M_j[410] := 2391_1 2391_2  ;
set M_j[411] := 2391_1 2391_2  ;
set M_j[412] := 2732_1  ;
set M_j[413] := 2649_1  ;
set M_j[414] := 2649_1  ;
set M_j[415] := 10319_1  ;
set M_j[416] := 85013_1  ;
set M_j[417] := 85011_1  ;
set M_j[418] := 2001_1 2001_2 2001_3  ;
set M_j[419] := 511_1  ;
set M_j[420] := 2228_1  ;
set M_j[421] := 2228_1  ;
set M_j[422] := 2228_1  ;
set M_j[423] := 2228_1  ;
set M_j[424] := 2228_1  ;
set M_j[425] := 2139_1  ;
set M_j[426] := 19270_1  ;
set M_j[427] := 2788_1  ;
set M_j[428] := 2667_1 2667_2 2667_3  ;
set M_j[429] := 2553_1 2553_2  ;
set M_j[430] := 2553_1 2553_2  ;
set M_j[431] := 2554_1  ;
set M_j[432] := 2063_1  ;
set M_j[433] := 2629_1 2629_2  ;
set M_j[434] := 21023_1  ;
set M_j[435] := 21010_1  ;
set M_j[436] := 21013_1  ;
set M_j[437] := 10306_1  ;
set M_j[438] := 2563_1  ;
set M_j[439] := 4685_1  ;
set M_j[440] := 2001_1 2001_2 2001_3  ;
set M_j[441] := 2001_1 2001_2 2001_3  ;
set M_j[442] := 2453_1  ;
set M_j[443] := 2610_1  ;
set M_j[444] := 2610_1  ;
set M_j[445] := 2471_1  ;
set M_j[446] := 2723_1  ;
set M_j[447] := 2316_1  ;
set M_j[448] := 2472_1  ;
set M_j[449] := 2230_1 2230_2 2230_3  ;
set M_j[450] := 2230_1 2230_2 2230_3  ;
```

```
set M_j[451] := 2117_1  ;
set M_j[452] := 2408_1 2408_2  ;
set M_j[453] := 2695_1 2695_2  ;
set M_j[454] := 10827_1 10827_2  ;
set M_j[455] := 4109_1  ;
set M_j[456] := 44_1  ;
set M_j[457] := 10016_1 10016_2  ;
set M_j[458] := 4759_1  ;
set M_j[459] := 10005_1  ;
set M_j[460] := 10001_1  ;
set M_j[461] := 10012_1  ;
set M_j[462] := 10210_1  ;
set M_j[463] := 10351_1  ;
set M_j[464] := 10315_1 10315_2  ;
set M_j[465] := 10345_1  ;
set M_j[466] := 47009_1 47009_2 47009_3  ;
set M_j[467] := 40221_1  ;
set M_j[468] := 10038_1  ;
set M_j[469] := 10347_1  ;

param p_j :=
402   38.40363
403   10.18409
404   19.78328
405   5.062095
406   12.43733
407   6.236595
408   122.52
409   64.24642
410   99.81185
411   39.36341
412   2.985742
413   17.075
414   17.08729
415   13.81941
416   8.632669
417   9.299062
418   121.2127
419   37.05272
420   12.09297
421   6.046487
422   4.837189
423   1.813946
424   10.27903
425   53.5181
426   34.38481
427   13.96482
428   119.3161
429   43.67136
430   108.0649
431   115.0998
432   114.7703
433   225.2053
434   24.15
```

```
435   22.63333
436   12.76546
437   27.90204
438   9.385087
439   9.967096
440   33.30668
441   18.6104
442   18.94657
443   45.90086
444   29.37655
445   32.51882
446   53.16279
447   45.51781
448   38.67104
449   44.73489
450   44.73489
451   45.70585
452   32.9048
453   32.26078
454   121.6434
455   9.354437
456   21.38328
457   13.58439
458   30.26453
459   10.38603
460   14.45428
461   20.30771
462   9.349624
463   27.44171
464   19.45464
465   12.58457
466   31.23212
467   5.724701
468   13.31032
469   11.85876
 ;
param r_j :=
402   25920
403   25920
404   25920
405   25920
406   25920
407   25920
408   25920
409   25920
410   25920
411   25920
412   25920
413   27360
414   27360
415   27360
416   27360
417   27360
418   27360
```

```
419   27360
420   31680
421   31680
422   31680
423   31680
424   31680
425   31680
426   31680
427   33120
428   33120
429   33120
430   33120
431   33120
432   33120
433   33120
434   33120
435   33120
436   33120
437   33120
438   33120
439   33120
440   33120
441   33120
442   33120
443   33120
444   33120
445   33120
446   33120
447   33120
448   33120
449   33120
450   33120
451   33120
452   33120
453   33120
454   33120
455   33120
456   33120
457   33120
458   33120
459   33120
460   33120
461   33120
462   33120
463   33120
464   33120
465   33120
466   33120
467   33120
468   33120
469   33120
 ;
param d_j :=
402   99360
```

403	99360
404	99360
405	99360
406	99360
407	99360
408	82080
409	82080
410	82080
411	82080
412	82080
413	82080
414	82080
415	47520
416	82080
417	82080
418	82080
419	82080
420	108000
421	108000
422	108000
423	108000
424	108000
425	90720
426	90720
427	108000
428	108000
429	99360
430	99360
431	99360
432	99360
433	99360
434	108000
435	108000
436	108000
437	90720
438	90720
439	90720
440	108000
441	108000
442	108000
443	108000
444	108000
445	108000
446	108000
447	108000
448	108000
449	108000
450	108000
451	116640
452	56160
453	56160
454	56160
455	90720
456	99360

```
457   90720
458   90720
459   90720
460   90720
461   90720
462   90720
463   90720
464   90720
465   90720
466   90720
467   90720
468   90720
469   90720
 ;
param c_j :=
402   847.0884
403   216.505
404   146.5428
405   37.497
406   266.855
407   46.197
408   816.928
409   437.2232
410   668.964
411   275.088
412   126
413   499.2328
414   499.7135
415   255.507
416   247.455
417   247.975
418   2566.2
419   250.6002
420   205.03
421   102.515
422   82.012
423   30.7545
424   174.2755
425   1496
426   752.856
427   305.76
428   2156.88
429   985.3536
430   2501.018
431   2501.856
432   1780.835
433   2500.588
434   330
435   247
436   279.5
437   500
438   256.9125
439   295.36
440   649.446
```

```
441   329
442   450
443   1005
444   643.2
445   712
446   1164
447   1029
448   705
449   360
450   360
451   1000.73
452   668.9182
453   772.2365
454   774
455   308.736
456   468.1872
457   400.2375
458   998.634
459   301.086
460   398.772
461   502.398
462   300.672
463   883.025
464   368.55
465   252.72
466   299.4875
467   252.7785
468   298.584
469   302.8935
 ;
param S_jj : 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418
419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438
439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458
459 460 461 462 463 464 465 466 467 468 469 :=
402    0  0  1.5  1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5
403    0  0  1.5  1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5
404    1.5  1.5  0  0  1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5
405    1.5  1.5  0  0  1.5  0  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5
```

406 0 0 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
407 1.5 1.5 0 0 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
408 1.5 1.5 1.5 1.5 1.5 1.5 0 0 0 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
409 1.5 1.5 1.5 1.5 1.5 1.5 0 0 0 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
410 1.5 1.5 1.5 1.5 1.5 1.5 0 0 0 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
411 1.5 1.5 1.5 1.5 1.5 1.5 0 0 0 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
412 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
413 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
414 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
415 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
416 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

1.5 1.5 1.5 1.5 1.5
417 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
418 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
419 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
420 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 0 0 0 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
421 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 0 0 0 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
422 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 0 0 0 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
423 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 0 0 0 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
424 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 0 0 0 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5
425 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
426 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
427 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
428 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
429 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
430 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
431 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
432 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
433 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
434 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
435 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
436 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
437 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
438 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
439 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
440 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
441 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
442 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
443 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
444 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 0 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
445 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
446 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
447 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
448 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
449 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5

1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
450 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
0 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5
451 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
452 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
453 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
454 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
455 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
456 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
457 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
458 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5
459 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 0 1.5 1.5 1.5 1.5 1.5
1.5 1.5 1.5 1.5 1.5

```
460    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5
461    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5
462    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  1.5  1.5
1.5  1.5  1.5  1.5  1.5
463    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0  1.5
1.5  1.5  1.5  1.5  1.5
464    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  0
1.5  1.5  1.5  1.5  1.5
465    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
0  1.5  1.5  1.5  1.5
466    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  0  1.5  1.5  1.5
467    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  0  1.5  1.5
468    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  0  1.5
469    1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
 1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5  1.5
1.5  1.5  1.5  1.5  0
 ;

# - * - * - * - * - * - * - * - * - * - * -
#       SOLUTION AND VISUALIZATION
```

```
# - * - * - * - * - * - * - * - * - * - * -

#option solver gurobi;

solve;
option omit_zero_rows 1;
display f_obj;
display Orders;
display c_j, r_j, s_j, p_j, d_j;
display x_jj;
display m_jd;

# - * - * - * - * - * - * - * - * - * - * -- * - * - * - * - * - * - * - * - * -
#                           Instance 3 (May 23rd to 29th)
# - * - * - * - * - * - * - * - * - * - * -

Presolve eliminates 389 constraints and 6837 variables.
Adjusted problem:
6939 variables:
        6857 binary variables
        82 linear variables
13473 constraints, all linear; 46873 nonzeros
        189 equality constraints
        13284 inequality constraints
1 linear objective; 6594 nonzeros.

Gurobi 8.1.0: threads=4
Gurobi 8.1.0: optimal solution; objective 103.5
5022516 simplex iterations
235147 branch-and-cut nodes
plus 80 simplex iterations for intbasis
f_obj = 103.5

set Orders :=
96    103   110   117   124   131   138   145   152   159   166   173
97    104   111   118   125   132   139   146   153   160   167   174
98    105   112   119   126   133   140   147   154   161   168   175
99    106   113   120   127   134   141   148   155   162   169   176
100   107   114   121   128   135   142   149   156   163   170   177
101   108   115   122   129   136   143   150   157   164   171
102   109   116   123   130   137   144   151   158   165   172;

:        c_j       r_j      s_j        p_j       d_j      :=
96      999.096    8640   16205.9    35.7947    64800
97      398.208    8640   36811.7    13.8669    64800
98      503.706    8640   15990.8    18.5887    64800
99      498.96     8640   16599.5    16.659     64800
100     502.26     8640   15911.5    16.8966    64800
101     300        8640   16944.8    13.2517    64800
102    2020.95     8640   36133.2    88.6813    73440
103    1376.55     8640   37106.8    47.2485    64800
104    2550.53    10080   20026.5   114.875     21600
105     300.69    10080   16548.2    49.7713    30240
106     822.642   10080   16527.8    18.9203    64800
```

107	700.28	10080	16441.1	26.1545	64800
108	295.85	10080	16056	15.813	64800
109	454.45	10080	19686.2	29.8621	64800
110	334.768	10080	16468.8	16.2823	64800
111	1001.25	10080	36425.3	25.6547	73440
112	421.889	10080	16924	19.2604	64800
113	215.103	11520	36052.2	7.33126	38880
114	1348.81	11520	36059.5	41.4063	38880
115	4166.01	11520	37204.1	192.751	38880
116	445.284	11520	36102.4	14.2062	38880
117	531.201	11520	19663.9	20.875	73440
118	576.182	11520	36892.4	22.2484	73440
119	871.27	11520	19972.8	52.1523	73440
120	1001.12	11520	36223.3	35.7291	73440
121	455.44	11520	71534.1	24.8789	73440
122	8421.2	11520	71559	440.987	73440
123	638.88	11520	37029.7	23.5308	73440
124	2226.22	11520	36531.2	95.5395	73440
125	1752.3	11520	36452.4	78.7446	73440
126	1378.42	11520	11695	51.6409	73440
127	1451.74	11520	11580	54.0556	73440
128	99.45	11520	11746.7	9.52133	73440
129	701.974	11520	11665.7	29.364	73440
130	769.782	11520	11634.1	31.5971	73440
131	5679.18	11520	12216.4	283.726	73440
132	3669.62	11520	11757.7	183.978	73440
133	2730.38	11520	11941.7	137.356	73440
134	2730.38	11520	12079	137.356	73440
135	919.524	11520	16797.5	34.0647	38880
136	5679.18	11520	12500.1	283.726	73440
137	267.54	11520	37427	13.0049	38880
138	504.069	11520	36878.4	12.4439	64800
139	431.309	14400	20141.3	18.6625	21600
140	196.746	14400	16959.5	9.22003	30240
141	608.4	15840	36364.5	26.3925	73440
142	488.28	15840	36392.4	18.8552	73440
143	280	15840	16913	9.53513	125280
144	315.716	15840	36712.8	12.7616	82080
145	264.384	15840	16850.7	7.99733	82080
146	2001.35	15840	36628.2	83.1118	38880
147	398.736	15840	36118.1	13.5364	73440
148	249.665	15840	16169.2	35.1599	73440
149	999.635	15840	16243.2	45.6558	73440
150	4870	15840	19823.2	148.116	73440
151	2865.43	15840	16689.9	106.16	73440
152	622.05	15840	36851.2	25.7515	38880
153	425.49	15840	36827.1	22.5908	38880
154	486.85	15840	36260.6	12.1749	38880
155	435.5	15840	36793.9	16.2816	38880
156	410.8	15840	16833.1	16.1205	38880
157	1148.55	15840	37054.7	50.5282	38880
158	1263.6	15840	36727.1	65.3741	38880
159	1248	15840	15929.9	59.3711	38880
160	1084.2	15840	16860.2	51.2782	38880

```
161   1210.62    15840   16010.9    43.5852     38880
162    856.7     15840   36998.8    29.4425     38880
163   1635.4     15840   37155.5    47.0337     38880
164   1664       15840   36306.4    56.5622     38880
165   1838.2     15840   36916.1    81.1508     38880
166   1826.5     15840   16617.7    70.7052     38880
167    362.7     15840   16423.1    16.5654     38880
168   2496       15840   19717.6   104.106      38880
169    716.3     15840   16499      27.29       38880
170    422.5     15840   36274.2    30.6852     38880
171    253.536   15840   37398.3    27.189      64800
172   2998.2     15840   16073.3    94.4491     73440
173   2872.8     15840   16290.4   131.208      38880
174    308.22    15840   36412.7    11.0079     73440
175    300.162   15840   15900      10.0434     73440
176    306.84    15840   16486.6    10.9674     73440
177    308.256   15840   16970.3    11.6506     73440
;

x_jj [*,*]
:     0  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113
:=
105   0   0   0   0   1   0   0   0   0   0   .   0   0   0   0   0   0   0   0
106   0   0   0   0   0   0   0   0   0   0   1   .   0   0   0   0   0   0   0
107   0   0   0   0   0   0   0   0   0   0   0   0   .   0   0   1   0   0   0
112   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   .   0
117   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
119   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
122   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
139   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
143   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
147   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
148   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
155   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
157   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
159   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
161   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
167   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
169   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
174   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
175   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0

:   114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132
:=
0     0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
102   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
111   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
113   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
114   .   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
121   0   0   0   0   0   0   0   .   1   0   0   0   0   0   0   0   0   0   0
125   0   0   0   0   0   0   0   0   0   0   1   .   0   0   0   0   0   0   0
126   0   0   0   0   0   0   0   0   0   0   0   0   .   0   1   0   0   0   0
127   0   0   0   0   0   0   0   0   0   0   0   0   0   .   0   0   1   0   0
128   0   0   0   0   0   0   0   0   0   0   0   0   0   0   .   0   0   0   1
```

```
129    0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   .   0   0   0
130    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   .   0   0
134    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
137    0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
138    0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
150    0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
162    0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
163    0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
177    0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

:    133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151
:=
96     0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
101    0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
104    0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
116    0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
124    0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
131    0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
132    1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
133    .   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
141    0   0   0   0   0   0   0   0   .   1   0   0   0   0   0   0   0   0   0
146    0   0   0   0   0   0   0   0   0   0   0   1   0   .   0   0   0   0   0
151    0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   .
152    0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
156    0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
160    0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
164    0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
166    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
168    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
171    0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
172    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0

:    152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170
:=
97     0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
98     0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
99     0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
100    0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
103    0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
109    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
118    0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
120    0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
123    0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
135    0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
144    0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
145    0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
153    1   .   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
154    0   0   .   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
158    0   0   0   1   0   0   .   0   0   0   0   0   0   0   0   0   0   0   0
165    0   0   0   0   0   0   0   0   0   0   1   0   0   .   0   0   0   0   0
170    0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   .
173    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
176    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
```

```
:   171 172 173 174 175 176 177    :=
108   0   1   0   0   0   0   0
110   0   0   0   0   0   1   0
115   1   0   0   0   0   0   0
136   0   0   0   0   1   0   0
140   0   0   0   0   0   0   1
142   0   0   0   1   0   0   0
149   0   0   1   0   0   0   0
;

m_jd [*,*] (tr)
:         96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113
:=
10020_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
10025_1   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
10028_1   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
10303_1   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
10341_1   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
10620_1   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
19202_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
19207_1   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
2152_1    0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
2187_1    0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
2189_1    0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
2223_1    0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
2379_1    0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
2383_1    0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
2633_1    0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
2634_1    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
4609_1    0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
4642_1    1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

:        114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131
:=
19202_1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
19204_1   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
19205_1   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
19206_1   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
19208_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
19211_1   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
19212_1   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
19233_1   0   0   0   0   0   0   0   1   1   0   0   0   0   0   0   0   0   0
19236_1   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
19244_1   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0   0   0   0
19255_1   0   0   0   0   0   0   0   0   0   0   0   0   1   1   1   1   1   0
5034_1    0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

:        132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149
:=
10306_1   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
10314_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
10602_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
19207_1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
19208_1   1   1   1   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
19264_1   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
2441_1    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
2668_1    0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
2686_1    0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
2687_1    0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
2706_1    0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
2719_1    0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
4039_1    0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
4685_1    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
89011_1   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0

:       150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167
:=
2443_1    0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
4628_1    1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
4629_1    0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
47005_1   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
47007_1   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
70017_1   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
70018_1   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
70020_1   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
70021_1   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
70027_1   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
70028_1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
90101_1   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
90102_1   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
90103_1   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
90104_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
90105_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
90106_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
90107_1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1

:       168 169 170 171 172 173 174 175 176 177     :=
2637_1    0   0   0   0   0   0   1   0   0   0
2638_1    0   0   0   0   0   0   0   1   0   0
2639_1    0   0   0   0   0   0   0   0   1   0
2640_1    0   0   0   0   0   0   0   0   0   1
452_1     0   0   0   1   0   0   0   0   0   0
4542_1    0   0   0   0   0   1   0   0   0   0
530_1     0   0   0   0   1   0   0   0   0   0
90202_1   1   0   0   0   0   0   0   0   0   0
90207_1   0   1   0   0   0   0   0   0   0   0
90216_1   0   0   1   0   0   0   0   0   0   0
;
# - * - * - * - * - * - * - * - * - * - * -- * - * - * - * - * - * - * - * - * -
```

Bibliography

[1] ADLA. Aluminium Extrusion, S.A. Available at: http://adla-aluminium.pt. Accessed on 2019/10/01.

[2] AICEP. Aicep portugal global. Available at: http://portugalglobal.pt/. Accessed on 2019/02/01.

[3] A. Allahverdi. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246(2):345–378, 2015.

[4] A. Allahverdi, J. N. Gupta, and T. Aldowaisan. A review of scheduling research involving setup considerations. *Omega*, 27(2):219–239, 1999.

[5] A. Allahverdi, C. Ng, T. E. Cheng, and M. Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032, 2008.

[6] A. Allahverdi and H. Soroush. The significance of reducing setup times/setup costs. *European Journal of Operational Research*, 187(3):978–984, 2008.

[7] M. Arenales, R. Morabito, V. Armentano, and H. Yanasse. *Pesquisa operacional: para cursos de engenharia*. Elsevier Brasil, 2015.

[8] A. Arif, A. Sheikh, and S. Qamar. A study of die failure mechanisms in aluminum extrusion. *Journal of Materials Processing Technology*, 134(3):318–328, 2003.

[9] A. Bachman and A. Janiak. Scheduling jobs with position-dependent processing times. *Journal of the Operational Research Society*, 55(3):257–264, 2004.

[10] K. R. Baker and B. Keller. Solving the single-machine sequencing problem using integer programming. *Computers & Industrial Engineering*, 59(4):730–735, 2010.

[11] M. Ben-Daya and M. Al-Fawzan. A tabu search approach for the flow shop scheduling problem. *European journal of operational research*, 109(1):88–95, 1998.

[12] I. BoussaïD, J. Lepagnot, and P. Siarry. A survey on optimization metaheuristics. *Information sciences*, 237:82–117, 2013.

[13] E. H. Bowman. The schedule-sequencing problem. *Operations Research*, 7(5):621–624, 1959.

[14] C.-L. Chen, V. S. Vempati, and N. Aljaber. An application of genetic algorithms for flow shop problems. *European Journal of Operational Research*, 80(2):389–396, 1995.

[15] T. E. Cheng, J. N. Gupta, and G. Wang. A review of flowshop scheduling research with setup times. *Production and Operations Management*, 9(3):262–282, 2000.

[16] H.-S. Choi and D.-H. Lee. Scheduling algorithms to minimize the number of tardy jobs in two-stage hybrid flow shops. *Computers & Industrial Engineering*, 56(1):113–120, 2009.

[17] J. Czyzyk, M. P. Mesnier, and J. J. Moré. The neos server. *IEEE Computational Science and Engineering*, 5(3):68–75, 1998.

[18] X. Dong, H. Huang, and P. Chen. An improved neh-based heuristic for the permutation flowshop problem. *Computers & Operations Research*, 35(12):3962–3968, 2008.

[19] E. Duman, M. B. Yildirim, and A. F. Alkaya. Scheduling continuous aluminium casting lines. *International Journal of Production Research*, 46(20):5701–5718, 2008.

[20] M. Durasević and D. Jakobović. A survey of dispatching rules for the dynamic unrelated machines environment. *Expert Systems with Applications*, 113:555–569, 2018.

[21] O. K. Esteban and Y. K. Penya. Energy-aware foundry production scheduling. In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, pages 2827–2832. IEEE, 2012.

[22] L. Fanjul-Peyro, R. Ruiz, and F. Perea. Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. *Computers & Operations Research*, 101:173–182, 2019.

[23] T. A. Feo and M. G. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

[24] V. Fernandez-Viagas, R. Leisten, and J. M. Framinan. A computational evaluation of constructive and improvement heuristics for the blocking flow shop to minimise total flowtime. *Expert Systems with Applications*, 61:290–301, 2016.

[25] A. M. Ferreiro, J. A. García-Rodríguez, C. Vázquez, E. C. e Silva, and A. Correia. Parallel two-phase methods for global optimization on GPU. *Mathematics and Computers in Simulation*, 156:67–90, 2019.

[26] A. F. Ferrás, F. De Almeida, E. Costa e Silva, A. Correia, and F. J. G. Silva. Scrap production of extruded aluminum alloys by direct extrusion. *Procedia Manufacturing*, 2019.

[27] C. A. Floudas and X. Lin. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers & Chemical Engineering*, 28(11):2109–2129, 2004.

[28] R. Fourer, D. M. Gay, and B. W. Kernighan. A modeling language for mathematical programming. *Management Science*, 36(5):519–554, 1990.

[29] H. Y. Fuchigami and S. Rangel. A survey of case studies in production scheduling: Analysis and perspectives. *Journal of Computational Science*, 25:425–436, 2018.

[30] C. Gagné, W. Price, and M. Gravel. Comparing an aco algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times. *Journal of the Operational Research Society*, 53(8):895–906, 2002.

[31] F. Glover and M. Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. Springer, 1998.

[32] J. F. Gonçalves, J. J. de Magalhães Mendes, and M. G. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European journal of operational research*, 167(1):77–95, 2005.

[33] J. Grabowski and M. Wodecki. A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. *Computers & Operations Research*, 31(11):1891–1909, 2004.

[34] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. R. Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics*, volume 5, pages 287–326. Elsevier, 1979.

[35] M. Gravel, W. L. Price, and C. Gagné. Scheduling jobs in an alcan aluminium foundry using a genetic algorithm. *International Journal of Production Research*, 38(13):3031–3041, 2000.

[36] M. Gravel, W. L. Price, and C. Gagné. Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic. *European Journal of Operational Research*, 143(1):218–229, 2002.

[37] S. C. Graves. A review of production scheduling. *Operations research*, 29(4):646–675, 1981.

[38] J. N. Gupta and E. F. Stafford Jr. Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169(3):699–711, 2006.

[39] J. Haraldsson and M. T. Johansson. Review of measures for improved energy efficiency in production-related processes in the aluminium industry–from electrolysis to recycling. *Renewable and Sustainable Energy Reviews*, 93:525–548, 2018.

[40] I. Harjunkoski, C. T. Maravelias, P. Bongers, P. M. Castro, S. Engell, I. E. Grossmann, J. Hooker, C. Méndez, G. Sand, and J. Wassick. Scope for industrial applications of production scheduling models and solution methods. *Computers & Chemical Engineering*, 62:161–193, 2014.

[41] J. C. Ho and Y.-L. Chang. A new heuristic for the n-job, m-machine flow-shop problem. *European Journal of Operational Research*, 52(2):194–202, 1991.

[42] J. H. Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[43] O. Holthaus and C. Rajendran. Efficient dispatching rules for scheduling in a job shop. *International Journal of Production Economics*, 48(1):87–105, 1997.

[44] B. Jarboui, S. Ibrahim, P. Siarry, and A. Rebai. A combinatorial particle swarm optimisation for solving permutation flowshop problems. *Computers & Industrial Engineering*, 54(3):526–538, 2008.

[45] M. Jayamohan and C. Rajendran. New dispatching rules for shop scheduling: a step forward. *International Journal of Production Research*, 38(3):563–586, 2000.

[46] S. M. Johnson. Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, 1(1):61–68, 1954.

[47] P. J. Kalczynski and J. Kamburowski. An empirical analysis of the optimality rate of flow shop heuristics. *European Journal of Operational Research*, 198(1):93–101, 2009.

[48] A. A. Keller. *Mathematical Optimization Terminology: A Comprehensive Glossary of Terms*. Academic Press, 2017.

[49] J. Kennedy. Particle swarm optimization. *Encyclopedia of machine learning*, pages 760–766, 2010.

[50] T. Ladhari and M. Haouari. A computational study of the permutation flow shop problem based on a tight lower bound. *Computers & Operations Research*, 32(7):1831–1847, 2005.

[51] C. Le Hesran, A.-L. Ladier, V. Botta-Genoulaz, and V. Laforest. Operations scheduling for waste minimization: A review. *Journal of cleaner production*, 206:211–226, 2019.

[52] G.-C. Lee and Y. D. Kim. A branch-and-bound algorithm for a two-stage hybrid flowshop scheduling problem minimizing total tardiness. *International Journal of Production Research*, 42(22):4731–4743, 2004.

[53] Q. Li, C. Harris, and M. R. Jolly. Finite element modelling simulation of transverse welding phenomenon in aluminium extrusion process. *Materials & design*, 24(7):493–496, 2003.

[54] D. G. Luenberger, Y. Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

[55] Y. Mahmoodkhani, M. A. Wells, N. Parson, and W. Poole. Numerical modelling of the material flow during extrusion of aluminium alloys and transverse weld formation. *Journal of Materials Processing Technology*, 214(3):688–700, 2014.

[56] A. S. Manne. On the job-shop scheduling problem. *Operations Research*, 8(2):219–223, 1960.

[57] C. T. Maravelias. General framework and modeling approach classification for chemical production scheduling. *AIChE Journal*, 58(6):1812–1828, 2012.

[58] G. McMahon and P. Burton. Flow-shop scheduling with the branch-and-bound method. *Operations Research*, 15(3):473–481, 1967.

[59] C. A. Méndez, J. Cerdá, I. E. Grossmann, I. Harjunkoski, and M. Fahl. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers & chemical engineering*, 30(6-7):913–946, 2006.

[60] G. Minella, R. Ruiz, and M. Ciavotta. A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing*, 20(3):451–471, 2008.

[61] T. Murata, H. Ishibuchi, and H. Tanaka. Genetic algorithms for flowshop scheduling problems. *Computers & Industrial Engineering*, 30(4):1061–1071, 1996.

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

[63] V. Nesello, A. Subramanian, M. Battarra, and G. Laporte. Exact solution of the single-machine scheduling problem with periodic maintenances and sequence-dependent setup times. *European Journal of Operational Research*, 266(2):498–507, 2018.

[64] A.-T. Nguyen, S. Reiter, and P. Rigo. A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy*, 113:1043–1058, 2014.

[65] J. Olhager and J. Wikner. Production planning and control tools. *Production Planning & Control*, 11(3):210–222, 2000.

[66] I. H. Osman and C. Potts. Simulated annealing for permutation flow-shop scheduling. *Omega*, 17(6):551–557, 1989.

[67] S. Panwalkar and C. Koulamas. The evolution of schematic representations of flow shop scheduling problems. *Journal of Scheduling*, pages 1–13, 2019.

[68] C. W. Pickardt, T. Hildebrandt, J. Branke, J. Heger, and B. Scholz-Reiter. Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. *International Journal of Production Economics*, 145(1):67–77, 2013.

[69] M. Pinedo. *Scheduling*. Springer, 2016.

[70] G. Rabadi, M. Mollaghasemi, and G. C. Anagnostopoulos. A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. *Computers & Operations Research*, 31(10):1727–1751, 2004.

[71] S. F. Rad, R. Ruiz, and N. Boroojerdian. New high performing heuristics for minimizing makespan in permutation flowshops. *Omega*, 37(2):331–345, 2009.

[72] C. Rajendran and H. Ziegler. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, 155(2):426–438, 2004.

[73] M. G. Resende, R. Martí, M. Gallego, and A. Duarte. Grasp and path relinking for the max–min diversity problem. *Computers & Operations Research*, 37(3):498–508, 2010.

[74] S. Reza Hejazi and S. Saghafian. Flowshop-scheduling problems with makespan criterion: a review. *International Journal of Production Research*, 43(14):2895–2929, 2005.

[75] I. Ribas, R. Leisten, and J. M. Framiñan. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, 37(8):1439–1454, 2010.

[76] R. Ruiz and C. Maroto. A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2):479–494, 2005.

[77] R. Ruiz and C. Maroto. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, 169(3):781–800, 2006.

[78] P. K. Saha. *Aluminum extrusion technology.* ASM International, 2000.

[79] N. V. Sahinidis. Mixed-integer nonlinear programming 2018, 2019.

[80] C. Schwindt and N. Trautmann. Scheduling the production of rolling ingots: industrial context, model, and solution method. *International Transactions in Operational Research*, 10(6):547–563, 2003.

[81] T. Sheppard. *Extrusion of aluminium alloys.* Springer Science & Business Media, 2013.

[82] N. Slack, S. Chambers, and R. Johnston. *Operations management.* Pearson education, 2010.

[83] H. Stefansson, S. Sigmarsdottir, P. Jensson, and N. Shah. Discrete and continuous time representations and mathematical models for large production scheduling problems: A case study from the pharmaceutical industry. *European Journal of Operational Research*, 215(2):383–392, 2011.

[84] S. Suliman. A two-phase heuristic approach to the permutation flow-shop scheduling problem. *International Journal of production economics*, 64(1-3):143–152, 2000.

[85] Y. Sun, C. Zhang, L. Gao, and X. Wang. Multi-objective optimization algorithms for flow shop scheduling problem: a review and prospects. *The International Journal of Advanced Manufacturing Technology*, 55(5-8):723–739, 2011.

[86] E.-G. Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[87] K.-C. Tan, R. Narasimhan, P. A. Rubin, and G. L. Ragatz. A comparison of four methods for minimizing total tardiness on a single processor with sequence dependent setup times. *Omega*, 28(3):313–326, 2000.

[88] C.-T. Tseng and C.-J. Liao. A particle swarm optimization algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *International Journal of Production Research*, 46(17):4655–4670, 2008.

[89] E. Ulungu, J. Teghem, P. Fortemps, and D. Tuyttens. Mosa method: a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236, 1999.

[90] E. Vallada and R. Ruiz. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 211(3):612–622, 2011.

[91] C. Vercellis. *Business intelligence: data mining and optimization for decision making*. Wiley Online Library, 2009.

[92] H. M. Wagner. An integer linear-programming model for machine scheduling. *Naval Research Logistics Quarterly*, 6(2):131–140, 1959.

[93] S. Wang and B. Ye. Exact methods for order acceptance and scheduling on unrelated parallel machines. *Computers & Operations Research*, 104:159–173, 2019.

[94] W. L. Winston and J. B. Goldberg. *Operations research: applications and algorithms*, volume 3. Thomson/Brooks/Cole Belmontˆ eCalif Calif, 2004.

[95] J. Xu and X. Zhou. A class of multi-objective expected value decision-making model with birandom coefficients and its application to flow shop scheduling problem. *Information Sciences*, 179(17):2997–3017, 2009.

[96] X. Xu, E. Yin, and F. Zou. Researches on optimal scheduling for aluminum industry continuous casting and rolling production. In *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 1, pages 348–352. IEEE, 2010.

[97] D. Yilmaz Eroglu, H. C. Ozmutlu, and S. Ozmutlu. Genetic algorithm with local search for the unrelated parallel machine scheduling problem with sequence-dependent set-up times. *International Journal of Production Research*, 52(19):5841–5856, 2014.

[98] K.-C. Ying, Z.-J. Lee, and S.-W. Lin. Makespan minimization for scheduling unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing*, 23(5):1795–1803, 2012.

[99] K.-C. Ying and S.-W. Lin. Multiprocessor task scheduling in multistage hybrid flow-shops: an ant colony system approach. *International Journal of Production Research*, 44(16):3161–3177, 2006.

[100] Y. Zhang, X. Li, and Q. Wang. Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization. *European Journal of Operational Research*, 196(3):869–876, 2009.

[101] J. Zhou, L.-x. Li, and J. Duszczyk. Computer simulated and experimentally verified isothermal extrusion of 7075 aluminium through continuous ram speed variation. *Journal of materials processing technology*, 146(2):203–212, 2004.

[102] G. Zobolas, C. D. Tarantilis, and G. Ioannou. Exact, heuristic and meta-heuristic algorithms for solving shop scheduling problems. In *Metaheuristics for scheduling in industrial and manufacturing applications*, pages 1–40. Springer, 2008.