Aalto University
School of Science

Master's programme in Mathematics and Operations Research

Modeling supply contracts in production planning with disjunctive programming

Thomas-Roy Samuel Holt

Master's Thesis
2023

Aalto University
School of Science

Author Thomas-Roy Samuel Holt

Title Modeling supply contracts in production planning with disjunctive programming

Degree programme Mathematics and Operations Research

Major Operations Research

Supervisor Prof. Fabricio Oliveira

Advisor Nikita Belyak

Date 9 February 2023 **Number of pages** 40+1 **Language** English

Abstract

This thesis aims to demonstrate how disjunctive programming can be used to model raw material procurement contracts in the context of a two-stage stochastic mixed integer linear programming (MILP) production planning problem. We will see that disjunctive programming provides an efficient framework for defining the constraints needed to model raw material procurement contracts. More specifically, we will analyze a simple two-stage stochastic production planning model where first-stage variables represent decisions regarding contracts and second-stage variables correspond to production planning decisions. We conclude the advantage of implementing the stochastic model is not as great as we initially thought. Namely, the Value of the Stochastic Solution (VSS) was lower then expected. Aside from this, the literature review in this area suggests most existing works are concerned with industry specific applications. Hence, our other goal is to introduce an example application that is easier to understand than the more sophisticated application-specific works available in the literature. The model is implemented using Julia programming language and makes use of the DisjunctiveProgramming.jl library. Overall, we concluded that disjunctive programming is a useful framework that can simplify and expedite the modeling process by providing a higher level of abstraction. However, there is still quite a bit of work to be done on scalability and performance if these types of models are to be deployed in the industry.

Keywords linear programming, MILP, disjunctive programming, supply chain optimisation

Contents

Chapter 1

Introduction

In this thesis, we investigate a two-stage stochastic production planning model. First-stage decisions are given by the result of deciding a supply contract for raw materials in advance, while second-stage decisions arise from the production planning problem itself. We assume raw material prices and customer demand (e.g. how much demand there is for the products we sell per customer) to be the stochastic parameters of our model. In practise, this means having multiple scenarios in the model which will be averaged over in the objective function. The importance of contracts in our model is that they allow us to more greatly exploit certain demand situations (for example, by buying in bulk at a discount if demand is very high). One interesting part of the model is that we model the structure of the contracts via disjunctive programming. Disjunctive programming involves modeling constraints using the logical disjunction operator $\vee$. While the constraints used could certainly be modeled without the framework of disjunctive programming, we found it highly useful during the modeling process as it allowed us to operate at a higher level of abstraction and think about the modeling more clearly.

With the exception of the disjunctive programming constraints, the model implemented is fairly generic. We used terminology common to the forestry industry such as "paper machine" for describing the singular unit or node where production occurs and "paper mill" to describe groups of machines with common characteristics (e.g. distance to a customer, shared inventory space etc.). However, we did not attempt to model any specific process in the paper industry and the model implemented could likely apply in a number of different cases. Also in all of our numerical experiments there is only ever a single paper machine per paper mill. We never explored the avenue of having multiple paper machines grouped under a single mill (although the formulation of our model could support this if desired).

In terms of our findings, the focus of this thesis was verifying existing models and consolidating the work done by others into a simpler form. Due to the inability to find an open-source implementation of disjunctive constraints for a non-trivial optimization model, we have implemented an optimisation model inspired by the ones already existing in the literature (see [1] and [12] for example). We hope it will be useful for those seeking to understand disjunctive programming to have a simple but non-trivial example available that is written in a modern programming

language like Julia. We also believe that disjunctive programming is useful because it abstracts otherwise relatively opaque-looking mixed-integer linear program (MILP) constraints into a form that is more tractable to explain to less technical personal such as senior management or auditors. In practise, this is likely the main gain of using the disjunctive programming approach. The reader can think of it as a black box for which higher-level disjunctive constraints are given as input and traditional MILP constraints are given as output – not entirely different from how a compiler or interpreter takes a high-level language as input and produces machine code as output.

The remainder of this thesis is structured as follows:

- Section 2 gives a brief introduction to disjunctive programming. We also give a brief overview of modeling contracts and give a tutorial introduction of how it can be used to create the disjunctions used in our model. A brief overview of the literature is also given.
- Section 3 gives descriptions of the optimisation model itself and defines needed notation.
- Section 4 describes the process we used to synthesize input data for the model. We then describe the results of several numerical experiments for both a deterministic version of the model as well as the full two-stage stochastic program.
- Section 5 concludes the work and addresses various difficulties and limitations that were encountered during the development of the thesis. Avenues for future work are also discussed.

Chapter 2

Method

2.1 Disjunctive programming

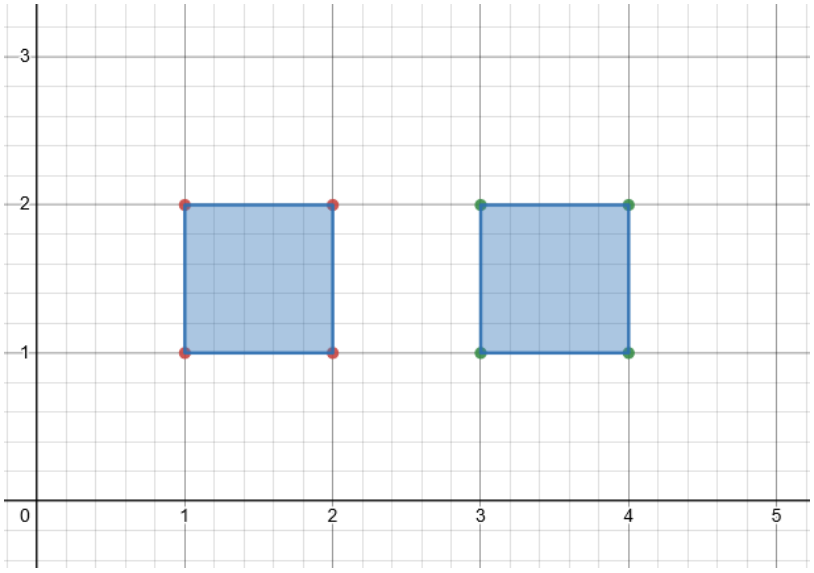


Figure 2.1: Example of a feasible region for a disjunctive programming problem.

In this section, we give a brief summary of disjunctive programming and how it can be used to model various decisions. We assume the reader has a basic familiarity with MILPs and common terminology used in the field of optimization. Our main reference when writing this section was [4]. We also recommend [1] which gives a more detailed theoretical introduction. Disjunctive programming is concerned with modeling constraints defined using the following logical operators: $\vee$ (or), $\wedge$ (and), and $\neg$ (negation). We will focus on $\vee$ since this is all what is needed to model the contracts used in our model. Before investigating how to model contracts, we will start with a simple example. Consider a generic maximization problem

$$P : \max_{x \in S} f(x)$$

where f is a real valued convex function $f : \mathbb{R}^2 \to \mathbb{R}$ and $S \subseteq \mathbb{R}^2$. If S is a square

region then it is straightforward to model S with linear constraints. However, suppose that $S = S_1 \cup S_2$ where S_1 and S_2 are two disjoint square regions (see Figure 2.1). In this case $x \in \mathbb{R}^2$ is a feasible solution if it lies in either S_1 or S_2. We can then no longer model S using standard linear programming constraints.

We can, however, model the region S using disjunctive constraints. Let $x = (x_1, x_2) \in \mathbb{R}^2$ and $y_1, y_2 \in \{0, 1\}$. Then S is determined by the following constraints:

$$\begin{bmatrix} y_1 \\ 1 \leq x_1 \leq 2 \\ 1 \leq x_2 \leq 2 \end{bmatrix} \vee \begin{bmatrix} y_2 \\ 3 \leq x_1 \leq 4 \\ 1 \leq x_2 \leq 2 \end{bmatrix}$$

where $y_1 + y_2 = 1$. Disjunctive programming allows one to transform these logical constraints into traditional 0-1 MILP problems that can be solved using standard algorithms such as branch and bound. There are two main approaches to doing this transformation: the big-M method and the convex hull method. We will perform the reformulation for both cases.

The simpler reformulation option is the big M-method. In this method, the following reformulation is produced of the disjunctive constraints:

$$\begin{aligned} 1 - M(1 - y_1) \leq x_1 \leq 2 + M(1 - y_1) \\ 1 - M(1 - y_1) \leq x_2 \leq 2 + M(1 - y_1) \\ 3 - M(1 - y_2) \leq x_1 \leq 4 + M(1 - y_2) \\ 1 - M(1 - y_2) \leq x_2 \leq 2 + M(1 - y_2) \end{aligned}$$

where M is a suitably large real number. The convex hull method is more complicated but yields a tighter relaxation. For each x_i introduce two new variables x_{i1}, x_{i2} and impose $x_i = x_{i1} + x_{i2}$. Using the current example, the full set of constraints using the convex hull method is then:

$$x_{11}^L y_1 \leq x_{11} \leq x_{11}^U y_1 \tag{2.1}$$
$$x_{21}^L y_1 \leq x_{21} \leq x_{21}^U y_1 \tag{2.2}$$
$$x_{12}^L y_2 \leq x_{12} \leq x_{12}^U y_2 \tag{2.3}$$
$$x_{22}^L y_2 \leq x_{22} \leq x_{22}^U y_2 \tag{2.4}$$
$$y_1 \leq x_{11} \leq 2y_1 \tag{2.5}$$
$$y_1 \leq x_{21} \leq 2y_1 \tag{2.6}$$
$$3y_2 \leq x_{12} \leq 4y_2 \tag{2.7}$$
$$y_2 \leq x_{22} \leq 2y_2 \tag{2.8}$$

where the x_{ij}^U and x_{ij}^L are upper and lower bounds respectively for each component of the variable decomposition.

Intuitively speaking, we decompose each dimension of $x \in \mathbb{R}^2$ to be responsible for fulfilling a certain disjunction. For example x_{12} is the part of x_1 that will fulfill the second disjunction's constraints. And, constraints (1) - (4) imply that if one component of x_1 is fulfilling the second disjunction then necessarily the other component will

be constrained to zero. So that x_1 as a whole will fulfil the constraints of the second disjunction. The linear programming relaxation using the convex hull method is tighter at the cost of having a larger number of variables. There are theoretical justifications (in [1]) which show that the convex hull formulation is "optimal" in a sense (e.g. under certain conditions the resulting relaxation yields the closed convex hull of the polyhedrons determined by each disjunction constraint). These considerations are outside the scope of our work. Also, in practise the reformulation of the disjunctions should be something handled by an external software package such as DisjunctiveProgramming.jl for Julia or Pyomo for Python. In the practical workflow, the modeler would not need to write down the formulations themselves; rather, they would specify only the disjunctions and not need to be aware of the underlying MILP constraints. In our work, due to some issues using DisjunctiveProgramming.jl, we were only able to use the big M reformulation option. The other reformulation options seemed to be in less mature stages of development and caused issues with other constraints in our model.

2.2 The contract selection problem

Here we briefly clarify what is meant by "contract" in this thesis. A contract is an agreement between the manufacturer and the raw material supplier to provide raw material r at a price p in time periods $t_1, \ldots, t_n$ for some arbitrary positive integer n. There will also be specific logical conditions that must be fulfilled in order for the discount to be achieved. In this thesis, we will model four different types of contracts which are commonly analyzed in the contract modeling literature. We will model these mathematically, but at a high level they are:

- *Fixed price contract* (FIXED): the supplier agrees to sell raw material r to the manufacturer at the spot market price at time t.
- *Discount after a certain amount* (DACA): the supplier agrees to sell raw material r to the manufacturer for price p_1 up to a certain quantity sold q and then a lower price p_2 after this threshold is broken. We will often refer to the distinct parts of this contract as first-stage price (p_1) and second-stage price (p_2) respectively. We refer to the specific volumes required to obtain a discount as the discount limits of the contract.
- *Bulk discount* (BULK): the supplier agrees to sell raw material r to the manufacturer for price p given that the quantity sold is larger then some discount limit q. In contrast to DACA type contracts, the manufacturer must buy the entire quantity to receive any discount and the entire quantity will be discounted.
- *Fixed duration contract* (FD): the supplier agrees to sell raw material r to the manufacturer at a fixed prices p_1, p_2, or p_3 for 1, 2 or 3 months respectively provided the quantity purchased exceeds the discount limit for each length of time. For example, a raw material X might be sold at a fixed price of 10 euros per ton for 2 months provided the purchaser buys at least 5 tons of the material

per ton. FD contracts are necessarily more complicated decisions than the other contract types. If the solver chooses to buy a raw material using FD contracts, it must then also choose which length of time (1, 2 or 3 months) is the most optimal.

The *contract selection problem* as defined in [3] consists of choosing which of the above contracts to engage in at each time period of the desired planning horizon. The major part of this thesis is concerned with using disjunctive programming to model each of the above contract types and produce a MILP which gives an optimal solution to the contract selection problem. We will use the abbreviations in parentheses to refer to each type of contract.

2.3 Modeling contracts with disjunctive programming

In this section, we will show how each of the contract types described in Section 2.2 can be modeled using disjunctive programming. We will also use this chapter as an opportunity to introduce the reader to some of the notation that will be used in later sections. To begin with, we will start by modeling DACA contracts.

To model a DACA contract, we require that c_{mrt}^{DACA}, the total cost of procuring raw material type r at time t under DACA contract by paper mill m to be

$$c_{mrt}^{\text{DACA}} = \phi_{rt}^{\text{DACA},1} r_{mrt}^{\text{DACA},1} + \phi_{rt}^{\text{DACA},2} r_{mrt}^{\text{DACA},2} \quad (\forall m \in M, \forall r \in R, \forall t \in T) \tag{2.9}$$

where $r_{mrt}^{\text{DACA},1}$ is the amount procured without any discount and $r_{mrt}^{\text{DACA},2}$ is the amount procured with discount. The prices $\phi_{rt}^{\text{DACA},1}$ and $\phi_{rt}^{\text{DACA},2}$ are the prices guaranteed when the amount purchased is below or above the discount limit $\sigma_{rt}^{\text{DACA}}$. Thus, $\phi_{rt}^{\text{DACA},2}$ will generally be lower to make the contract attractive. As was done in the first disjunctive programming example, we will also need to introduce binary variables $s_{mrt}^{\text{DACA},1}, s_{mrt}^{\text{DACA},2}$ indicating if the contract has been fulfilled (e.g. the amount of the raw material the buyer has bought exceeds the threshold needed to get a discount). We can then form the following disjunction constraints:

$$\begin{bmatrix} s_{mrt}^{\text{DACA},1} \\ 0 \le r_{mrt}^{\text{DACA},1} \le \sigma_{rt}^{\text{DACA}} \\ r_{mrt}^{\text{DACA},2} = 0 \end{bmatrix} \vee \begin{bmatrix} s_{mrt}^{\text{DACA},2} \\ r_{mrt}^{\text{DACA},1} = \sigma_{rt}^{\text{DACA}} \\ r_{mrt}^{\text{DACA},2} \ge 0 \end{bmatrix} \quad (\forall m \in M, \forall r \in R, \forall t \in T) \tag{2.10}$$

In words, this says that if the buyer chooses a DACA contract at time t, then the first $r_{mrt}^{\text{DACA},1}$ will be bought at price $\phi_{rt}^{\text{DACA},1}$ while any excess over $\sigma_{rt}^{\text{DACA}}$ will be bought at the discounted price $\phi_{rt}^{\text{DACA},2}$. This highlights how disjunctive programming can be used to efficiently model the various clauses and stipulations of a typical procurement contract.

The next contract type, BULK, is similar to DACA but now the discount applies uniformly across the entire purchased quantity provided the amount purchased exceeds some pre-defined threshold. Let c_{mrt}^{DACA} denote the cost of procuring raw material r at

time t under the BULK contract type. Similar to the DACA contract type, we have a threshold parameter $\sigma_{rt}^{\text{BULK}}$ which is the amount required to be bought in bulk in order for the buyer to get a discount on the entire purchase. There are also two indicator variables $s_{mrt}^{\text{BULK},1}, s_{mrt}^{\text{BULK},2}$ indicating if the contract has been fulfilled. We then have the disjunctive constraints:

$$\begin{bmatrix} s_{mrt}^{\text{BULK},1} \\ \mathrm{c}_{mrt}^{\text{BULK}} = \phi_{rt}^{\text{BULK},1} r_{mrt}^{\text{BULK}} \\ 0 \leq r_{mrt}^{\text{BULK}} \leq \sigma_{rt}^{\text{BULK}} \end{bmatrix} \vee \begin{bmatrix} s_{mrt}^{\text{BULK},2} \\ c_{mrt}^{\text{BULK}} = \phi_{rt}^{\text{BULK},2} r_{mrt}^{\text{BULK}} \\ r_{mrt}^{\text{BULK}} \geq \sigma_{rt}^{\text{BULK}} \end{bmatrix} \quad (\forall m \in M, \forall r \in R, \forall t \in T) \tag{2.11}$$

where $\phi_{rt}^{\text{BULK},1}, \phi_{rt}^{\text{BULK},2}$ represent the price of the raw material with and without the bulk discount respectively.

Lastly, we describe the FD-type contracts with disjunctive programming. This example is of particular interest in that it shows how more than one disjunction can be used to encode fairly complicated logical conditions in a compact manner. Recall that FD-type contracts give the buyer the option to buy raw materials at a fixed price in a given time period provided they agree to buy a minimum amount in each period.

Let $\mathrm{c}_{mrt}^{\text{FD}}$ denote cost of procuring raw material r at time t under the FD contract type, let $\phi_{rt}^{\text{FD},k}$, $k \in \{1, 2, 3\}$ denote the fixed price agreed to purchase raw material r at for the next k months, let r_{mrt}^{FD} denote the amount of raw material r purchased at time t under FD contract. Finally let $\sigma_{rt}^{\text{FD},k}$ denote the k-th stage discount limit for the contract. This is the minimum the buyer must procure for the next k months if they seek a fixed price. We can then form the disjunction constraints

$$\begin{bmatrix} s_{mrt}^{\text{FD},1} \\ c_{mrt}^{\text{FD}} = \phi_{rt}^{\text{FD},1} r_{mrt}^{\text{FD}} \\ r_{mrt}^{\text{FD}} \geq \sigma_{rt}^{\text{FD},1} \end{bmatrix} \vee \begin{bmatrix} s_{mrt}^{\text{FD},2} \\ \mathrm{c}_{mrt}^{\text{FD}} = \phi_{mrt}^{\text{FD},2} r_{mrt}^{\text{FD}} \\ \mathrm{c}_{mr(t+1)}^{\text{FD}} = \phi_{rt}^{\text{FD},2} r_{r(t+1)}^{\text{FD}} \\ r_{mrt}^{\text{FD}} \geq \sigma_{rt}^{\text{FD},2} \\ r_{mr(t+1)}^{\text{FD}} \geq \sigma_{rt}^{\text{FD},2} \end{bmatrix} \vee \begin{bmatrix} s_{mrt}^{\text{FD},3} \\ r_{mrt}^{\text{FD}} = \phi_{mrt}^{\text{FD}} r_{mrt}^{\text{FD}} \\ r_{mr(t+1)}^{\text{FD}} = \phi_{rt}^{\text{FD},2} r_{mr(t+1)}^{\text{FD}} \\ r_{mr(t+2)}^{\text{FD}} = \phi_{rt}^{\text{FD}} r_{mr(t+2)}^{\text{FD}} \\ r_{mrt}^{\text{FD}} \geq \sigma_{rt}^{\text{FD},3} \\ r_{mr(t+1)}^{\text{FD}} \geq \sigma_{rt}^{\text{FD},3} \\ r_{mr(t+2)}^{\text{FD}} \geq \sigma_{rt}^{\text{FD},3} \end{bmatrix} \tag{2.12}$$

for all $m \in M$, $r \in R$, $t \in T$. Here the $s_{mrt}^{\text{FD},1}, s_{mrt}^{\text{FD},2}$, and $s_{mrt}^{\text{FD},3}$ are the usual indicator variables controlling if the price will be fixed for one, two or three months respectively.

While the above logics could certainly be encoded in an optimisation model without the help of disjunctive programming, we believe disjunctive programming provides an excellent high-level language for modeling complex business ideas such as contracts in optimization models. Existing packages such as DisjunctiveProgramming.jl can then be used to reformulate the disjunctive constraints into MILP constraints that may be provided to off-the-shelf solvers. More practically, when presenting models to business-users and management, the logically equivalent disjunction constraints can be presented and explained over the less straightforward MILP constraints.

2.4 Literature Review

At its core, the problem addressed in this thesis belongs to the field of supply chain management (SCM). The precise definition of SCM is not exactly straightforward with many interpretations existing in the literature. For example, see [10] for an in-depth review of this question from a business perspective. However, for our purposes, it is enough to have a pragmatic understanding. We directly quote [17]: *a supply chain is two or more parties linked by a flow of goods, information, and funds*. In our problem, there are three parties between which goods flow: the raw material suppliers, the manufacturer and the customers. The manufacturer has prior information about the customers in the form of historical data and the resulting forecasts.

Since firms often must hedge against future uncertainty in demand and prices, making a decision regarding raw material procurement contracts is an integral step in the SCM process. We believe the global uncertainty in recent years and its resulting impact on raw material prices suggests that contract decisions must be integrated directly into a firm's decision support tools and not left in the hands of heuristics or manual processes. In [17], a foundation is given on which most modern contract modeling papers are built upon. The authors discuss the different aspects of common procurement contracts in SCM and give various modeling approaches. However, the focus is less on the optimal selection and more on clarifying the different types of contracts. While there is no mention of disjunctive programming it is likely that the different contract types used in our paper (which were originally used in [12] and [3]) originally stem from the work done in [17].

The literature on contract modeling using disjunctive programming is relatively small with [3], [12] and [16] being the main examples. There could be a few reasons for this, one being that modern optimization libraries (e.g. Julia's JuMP and Python's Pyomo) have only recently begun to produce mature libraries that automate the process of converting disjunctive programming formulations to MILP constraints (see DisjunctiveProgramming.jl or PyOMO.GDP for example). Without these libraries, the process of manually calculating the MILP constraints corresponding to a disjunctive program can be tedious and seem ad-hoc to someone without prior knowledge in modeling and optimisation. Another reason could be the large increase in solution time that comes with introducing different types of contracts which we saw first hand in our work.

Both [3] and [12] use disjunctive programming to model the contract selection problem in the context of chemical process networks. The former work's model is stochastic while the latter uses a deterministic formulation. The model used in [3] is reasonably similar to the one in this thesis, however, we opted for a simpler model and did not make use of a demand-response function (i.e. manufactured goods price as a function of demand). From our knowledge, we believe [3] and [16] to be the only references using disjunctive programming to model contracts in problems with uncertainty in the context of supply chain optimization. However, [16] makes the reliability of supplier deliveries uncertain which is unique. The method of using disjunctive programming also differs in [16] compared to our own work. While our model consists of single sets of disjunctions with indicator variables (one for each contract type), [16] have a single disjunction which selects between three different

similarly structured contracts with different numerical parameters (e.g. discount or minimum quantity requirements). Two of the contract types studied by [16] are nearly identical to the so-called DACA contract we defined earlier with the third being a more complex type involving interest rates. Interestingly, they also make use of generalized disjunctive programming in their model while we are restricted to classical disjunctive programming constraints. The generalized disjunctive programming can consist of a larger set of logical operators such as negation ($\neg$) or and ($\wedge$) as well as having multiple disjunctions nested inside one another.

On the other hand, there are several works modeling contracts without the explicit use of disjunctive programming. In [8] the authors study the optimal contract selection problem in the case of liquefied natural gas (LNG) contracts. They model a single contract type with an MILP choosing the optimal selection of contract parameters. An MILP model for optimal procurement of wood supply is studied in [14]. The authors focus on the case of flexibility in contracts. Specifically, the buyer has the ability to buy more or less than the agreed amount. Each contract in the model has different parameters for flexibility and the model will choose whichever is optimal. The model is a traditional MILP without disjunctive constraints. They also modeled in greater detail the process of transporting raw materials by introducing transportation costs for initial raw material delivery as well as transportation between the manufacturer's owned storage sites. Individual suppliers are also modeled and can have different features and offer different types of contracts (differing from our own model where suppliers are abstracted away and only different types of contracts can be selected from). The same authors generalized this model in [15] to the case of stochastic programming. Similar to our own work, it is a two-stage stochastic program where first-stage decisions concern the contract types and second-stage decisions are associated actual production plan.

The aforementioned works [8], [14] and [15] highlight an important point in our view. Namely, in the contract selection literature, there seems to be two different problems that are both grouped under the label of the contract selection problem. First, there is the problem of choosing between contract types with differing logical structure and second, there is the problem of selecting between contracts of differing parameters but shared logical structure. Examples of the former include our own work as well as [3] and [12] while [8], [14] and [15] give examples of the latter. We believe it is an important distinction to make since the formalism of disjunctive programming could seem unnecessary and overly complicated to use if the problem targets selecting between contracts of differing parameters but with a shared logical structure. For example, [14] uses the following (we have simplified for the example) constraint to impose the logic of the contract on the optimization model:

$$(F_g^+ + 1)Q_g z_g \geq x_g \tag{2.13}$$

$$(F_g^- + 1)Q_g z_g \leq x_g \tag{2.14}$$

where F_g^+ (F_g^-) is a multiplier indicating how far one can deviate above the nominal value Q_g of the contract (e.g. we request more (less) raw material then initially agreed), z_g a binary variable indicating if contract g is used and lastly x_g the amount actually

procured under contract g. It would be unnecessary to introduce the abstraction of disjunctive programming in this case. In general, disjunctive programming is most useful when the business problem under consideration consists of several different alternatives (in our case contracts) that have a distinct internal logical structure.

From reviewing the literature we conclude that there is, in general, not much work done on using disjunctive programming in the context of supply chain management and optimisation. There is also some ambiguity on what exactly is meant by the contract selection problem which we attempted to clarify in the previous discussion. The remainder of the thesis is an attempt to contribute to the existing literature by providing a simple example of disjunctive programming in the context of supply chain management. The source code is freely available at the author's GitHub repository (see [7]) and could hopefully be used as a model for future works in the literature.

Chapter 3

Modelling approach

In this section, we describe the two-stage stochastic programming optimisation model we will analyze. The problem of deciding what contracts will be undertaken at each planning period is understood as something that must be decided "in advance" and will thus be independent of the stochastic scenarios in the model. The second-stage decisions consist of deciding the production plan itself and will be scenario-dependent.

3.1 Contract decisions

Here we primarily describe the parts of the model that are related to the modeling and selection of contracts.

Sets

$A = \{\text{FIXED}, \text{DACA}, \text{BULK}, \text{FD}\}$	set of contract types
$S = \{\text{S0}, \ldots, \text{S}n\}$	set of scenarios

Parameters

$\phi_{rt}^{a,i}$	$a \in A, t \in T, r \in R, i \in \{1,2,3\}$	i-th stage price of raw material r at time t under contract type a
$\sigma_{rt}^{a,i}$	$a \in A, t \in T, r \in R, i \in \{1,2,3\}$	i-th stage discount limit for material r at time t under contract type a

Decision variables

Binary decision variables

z^{a}_{mrt}	$a \in A, m \in M, r \in R, t \in T$	contract type a is used by mill M to procure raw material r in period t
$s^{a,i}_{mrt}$	$a \in A, m \in M, r \in R, t \in T, i \in \{1,2,3\}$	stage i of contract type a is used by mill m to procure raw material r at time t
l^{21}_{mrt}	$m \in M, r \in R, t \in T$	time period t is the second month of a two-month FD contract used by mill M to procure raw material r
l^{31}_{mrt}	$m \in M, r \in R, t \in T$	time period t is the second month of a three-month FD contract used by mill M to procure raw material r
l^{32}_{mrt}	$m \in M, r \in R, t \in T$	time period t is the third month of a three-month FD contract used by mill M to procure raw material r

Continuous decision variables

$r^{a,i}_{mrt}$	$m \in M, r \in R, t \in T, i \in \{1,2\}$	tons of raw material r purchased by mill M under contract type a at i-th stage price under DACA contract at time t
c^{a}_{mrt}	$a \in A, m \in M, r \in R, t \in T$	cost of raw material r purchased by mill M with contract type a at time t

Constraints

The total cost for materials procured via FIXED type contracts is simply the market rate multiplied by the amount purchased.

$$c^{\text{FIXED}}_{mrt} = \phi^{\text{FIXED}}_{rt} r^{\text{FIXED}}_{mrt} \tag{3.1}$$

The disjunctions for DACA contracts are given by:

$$\begin{bmatrix} s^{\text{DACA},1}_{mrt} \\ 0 \le r^{\text{DACA},1}_{mrt} \le \sigma^{\text{DACA}}_{rt} \\ r^{\text{DACA},2}_{mrt} = 0 \end{bmatrix} \vee \begin{bmatrix} s^{\text{DACA},2}_{mrt} \\ r^{\text{DACA},1}_{mrt} = \sigma^{\text{DACA}}_{rt} \\ r^{\text{DACA},2}_{mrt} \ge 0 \end{bmatrix} \quad (\forall m \in M, \forall r \in R, \forall t \in T) \tag{3.2}$$

The sum of non-discounted and discounted volumes for DACA contracts make the total.

$$r^{\text{DACA}}_{mrt} = r^{\text{DACA},1}_{mrt} + r^{\text{DACA},2}_{mrt} \quad (\forall m \in M, \forall t \in T, \forall r \in R) \tag{3.3}$$

The total cost of DACA type contracts is given by summing the costs of the non-discounted and discounted volumes.

$$\mathrm{c}^{\text{DACA}}_{mrt} = \phi^{\text{DACA},1}_{rt} r^{\text{DACA},1}_{mrt} + \phi^{\text{DACA},2}_{rt} r^{\text{DACA},2}_{mrt} \quad (\forall m \in M, \forall r \in R, \forall t \in T) \tag{3.4}$$

The disjunctions for BULK contracts are given by:

$$\begin{bmatrix} s^{\text{BULK},1}_{mrt} \\ c^{\text{BULK}}_{mrt} = \phi^{\text{BULK},1}_{rt} r^{\text{BULK}}_{mrt} \\ 0 \leq r^{\text{BULK}}_{mrt} \leq \sigma^{\text{BULK}}_{rt} \end{bmatrix} \vee \begin{bmatrix} s^{\text{BULK},2}_{mrt} \\ c^{\text{BULK}}_{mrt} = \phi^{\text{BULK},2}_{rt} r^{\text{BULK}}_{mrt} \\ r^{\text{BULK}}_{mrt} \geq \sigma^{\text{BULK}}_{rt} \end{bmatrix} \quad (\forall m \in M, \forall r \in R, \forall t \in T) \tag{3.5}$$

The disjunctions for FD contracts are given by:

$$\begin{bmatrix} s^{\text{FD},1}_{mrt} \\ c^{\text{FD}}_{mrt} = \phi^{\text{FD},1}_{rt} r^{\text{FD}}_{mrt} \\ r^{\text{FD}}_{mrt} \geq \sigma^{\text{FD},1}_{rt} \end{bmatrix} \vee \begin{bmatrix} s^{\text{FD},2}_{mrt} \\ c^{\text{FD}}_{mrt} = \phi^{\text{FD},2}_{mrt} r^{\text{FD}}_{mrt} \\ c^{\text{FD}}_{mr(t+1)} = \phi^{\text{FD},2}_{rt} r^{\text{FD}}_{r(t+1)} \\ r^{\text{FD}}_{mrt} \geq \sigma^{\text{FD},2}_{rt} \\ r^{\text{FD}}_{mr(t+1)} \geq \sigma^{\text{FD},2}_{rt} \end{bmatrix} \vee \begin{bmatrix} s^{\text{FD},3}_{mrt} \\ r^{\text{FD}}_{mrt} = \phi^{\text{FD}}_{mrt} r^{\text{FD}}_{mrt} \\ r^{\text{FD}}_{mr(t+1)} = \phi^{\text{FD},2}_{rt} r^{\text{FD}}_{mr(t+1)} \\ r^{\text{FD}}_{mr(t+2)} = \phi^{\text{FD}}_{rt} r^{\text{FD}}_{mr(t+2)} \\ r^{\text{FD}}_{mrt} \geq \sigma^{\text{FD},3}_{rt} \\ r^{\text{FD}}_{mr(t+1)} \geq \sigma^{\text{FD},3}_{rt} \\ r^{\text{FD}}_{mr(t+2)} \geq \sigma^{\text{FD},3}_{rt} \end{bmatrix} \tag{3.6}$$

for all $m \in M$, $r \in R$, $t \in T$

We can only procure raw materials using active contracts:

$$r^{a}_{mrt} \leq M z^{a}_{mrt} \quad (\forall a \in A, \forall m \in M, \forall t \in T, \forall r \in R) \tag{3.7}$$

for large $M \in \mathbb{R}$.

For each mill, raw material type, we can only choose one type of contract in a single time period:

$$\sum_{a \in A} z^{a}_{mrt} \leq 1 \quad (\forall m \in M, \forall r \in R, \forall t \in T) \tag{3.8}$$

Contract stages are mutually exclusive in DACA and BULK contract types:

$$z^{a}_{mrt} = s^{a,1}_{mrt} + s^{a,2}_{mrt} \quad (\forall m \in M, \forall r \in R, \forall t \in T) \tag{3.9}$$

where $a \in \{\text{BULK}, \text{DACA}\}$. Contract stages are also mutually exclusive in FD contracts. However, we need some extra variables to ensure the contract is binding in future time periods if selected:

$$z^{a}_{mrt} = l^{21}_{mrt} + l^{31}_{mrt} + l^{32}_{mrt} + \sum_{i \in \{1,2,3\}} s^{\text{FD},i}_{mrt} \quad (\forall m \in M, \forall r \in R, \forall t \in T) \tag{3.10}$$

To make Equation (3.10) work we also need the following:

$$l^{21}_{mrt} = s^{\text{FD},2}_{mr(t-1)} \quad (\forall m \in M, \forall r \in R, \forall t \in T) \tag{3.11}$$

$$l^{31}_{mrt} = s^{\text{FD},3}_{mr(t-1)} \quad (\forall m \in M, \forall r \in R, \forall t \in T) \tag{3.12}$$

$$l^{32}_{mrt} = s^{\text{FD},3}_{mr(t-2)} \quad (\forall m \in M, \forall r \in R, \forall t \in T) \tag{3.13}$$

Constraints (3.11) - (3.13) ensure that when the model selects an FD contract at period t then it will be forced to select it again in the next one or two periods (depending on

which stage was selected). For example, suppose the model decided to take a 3-month FD contract at time t'. Then constraints (3.12) and (3.13) ensure $l^{31}_{mr(t'+1)}$ and $l^{32}_{mr(t'+2)}$ will be set to 1. This means that the mutual exclusivity constraint (3.10) will force FD to be chosen in periods $t'+1$ and $t'+2$ as we would expect for a 3-month binding contract.

There are a few technicalities regarding this notation. For FIXED type contracts $\phi^{a,i}_{rt}$ has no dependency on i since FIXED contracts use only the market spot price with no discount possible. Hence we will drop i when using this variable for FIXED contracts. Similarly, DACA and BULK contracts only have a single discount limit. So $\sigma^{a,i}_{rt}$ has no dependency on i for $a \in \{\text{DACA}, \text{BULK}\}$. Thus we will also drop the i when using this variable in the context of DACA and BULK contracts. In variables that depend on $i \in \{1, 2, 3\}$, it is possible the variable is undefined for $i = 3$. For example, DACA contracts do not have a third stage so $s^{\text{DACA},3}_{mrt}$ will be undefined. In all these cases the reader should assume the variable will be constrained to 0 for $i = 3$. Lastly, FIXED contract material prices are stochastic and implicitly depend on $s \in S$. We decided not to make it explicit here for simplicity.

3.2 Production planning model

In this section, we describe the parts of the model dealing with production planning.

Sets

The customers in C are buying products produced by the paper machines in set U. Note that the elements of M are subsets of U so that each paper machine can belong to the same mill. The set T is the number of planning periods. In each planning period, the complete production schedule must be designed and a contract type decided for each raw material in R needed to produce the products of P that are in demand.

U	set of paper machines
$M \subseteq 2^U$	set of paper mills
P	set of paper products
R	set of raw materials
C	set of customers
T	set of time periods

Parameters

α_{pr}	$p \in P, r \in R$	conversion factor to convert 1 ton of product p into required raw material r
δ^s_{cpt}	$p \in P, t \in T, c \in C, s \in S$	demand in time period t for product p from customer c in scenario s
κ_u	$u \in U$	monthly production capacity of paper machine u.
λ_{cu}	$c \in C, m \in M$	logistics costs for transporting products from paper machine u to customer c
θ_m	$m \in M$	storage costs per ton of product or raw material stored at mill m
ψ_{pt}	$p \in P, t \in T$	price per ton of product p at time t
π^s	$s \in S$	probability of scenario s
γ_m	$m \in M$	maximum inventory space for mill m

Decision variables

Cont. decision variables

x^s_{cput}	$c \in C, p \in P, u \in U, t \in T, s \in S$	tons of paper product p produced by paper machine u in time t for customer c in scenario s
d^s_{cput}	$p \in P, t \in T, c \in C, s \in S$	demand in time period t for product p from customer c allocated to machine u in scenario s
w^s_{cmpt}	$c \in C, m \in M, p \in P, t \in T, s \in S$	tons of paper product p in storage at mill m at time t for customer c in scenario s
y^s_{mrt}	$m \in M, r \in R, t \in T, s \in S$	tons of raw material r in storage at mill m at time t in scenario s
q^s_{cput}	$p \in P, u \in U, c \in C, t \in T, s \in S$	tons of unfulfilled demand from customer c for product p by paper mill u at time t in scenario s

Objective function

The objective function is

$$\text{EXP. PROFIT} = \sum_{s \in S} \pi^s (\text{SALES}_s - \text{ICOSTS}_s - \text{RCOST} - \text{LCOST}_s) \qquad (3.14)$$

where

$$\text{SALES}_s = \sum_{p\in P}\sum_{t\in T}\sum_{u\in U}\psi_{pt}(x^s_{cput} - q^s_{cput}) \tag{3.15}$$

$$\text{RCOST} = \sum_{r\in R}\sum_{t\in T}\sum_{a\in A} c^a_{rt} \tag{3.16}$$

$$\text{ICOST}_s = \sum_{m\in M}\sum_{t\in T}\theta_m(\sum_{p\in P}\sum_{c\in C} w^s_{cmpt} + \sum_{r\in R} y^s_{mrt}) \tag{3.17}$$

$$\text{LCOST}_s = \sum_{t\in T}\sum_{u\in U}\sum_{c\in C}\sum_{p\in P}\lambda_{cu}x^s_{cput} \tag{3.18}$$

Specifically, SALES_s is the revenue received from selling products to customers. The variable q^s_{cput} acts as a penalty for when demand is not completely fulfilled. The variable RCOST is the costs of procuring raw materials and ICOST_s is the costs of storing raw materials and finished products in inventory. Lastly, LCOST_s is costs of logistics of transporting product p to customer c.

Constraints

Customer demand is distributed to each paper machine to maximise the profit.

$$\sum_{u\in U} d^s_{cput} = \delta^s_{cpt} \tag{3.19}$$

Demand should be balanced taking into account existing inventory and currently produced products. If it is not feasible to satisfy demand, q^s_{cput} will supply the shortfall which is deducted from the objective function.

$$q^s_{cput} + x^s_{cput} + w^s_{cm'p(t-1)} = d^s_{cput} + w^s_{cm'pt} \quad (\forall t \in T, \forall p \in P, \forall c \in C, \forall u \in U, \forall s \in S) \tag{3.20}$$

where m' is the mill containing the paper machine u. Products can only be produced if the necessary materials have been purchased or already exist in inventory.

$$y^s_{mr(t-1)} + \sum_{a\in A} r^a_{mrt} = y^s_{mrt} + \sum_{c\in C}\sum_{u\in U}\alpha_{pr}x^s_{cput} \quad (\forall t \in T, \forall p \in P, \forall r \in R, \forall m \in M, \forall s \in S) \tag{3.21}$$

Each paper mill (e.g. grouping of paper machines) has a fixed size inventory. In this inventory we can store a mixture of raw materials and products. The total quantity stored cannot exceed a fixed amount.

$$\sum_{p\in P}\sum_{c\in C} w^s_{cmpt} + \sum_{r\in R} y^s_{mrt} \le \gamma_m \quad (\forall m \in M, \forall t \in T, \forall s \in S) \tag{3.22}$$

Similarly, each paper machine has a maximum capacity of paper it can produce in a single month.

$$\sum_{p\in P}\sum_{c\in C} x^s_{cput} \le \kappa_u \quad (\forall t \in T, \forall u \in U, \forall s \in S) \tag{3.23}$$

Lastly, we have non-negativity: all continuous decision variables are non-negative.

Chapter 4

Numerical results

4.1 Data generation

As mentioned, our model is a two-stage stochastic MILP with raw material prices and demand being the sources of uncertainty. Our data largely comes from synthetic data creation processes. Namely, our demand data is generated by using bootstrap re-sampling on historic time-series of wood pulp price data sourced from [9]. Bootstrap sampling consists of re-sampling a single sample of data in an attempt to approximate the underlying sampling distribution (see [11] for a more detailed introduction). In our case, we start from a time series of prices

$$p_1, \ldots, p_n.$$

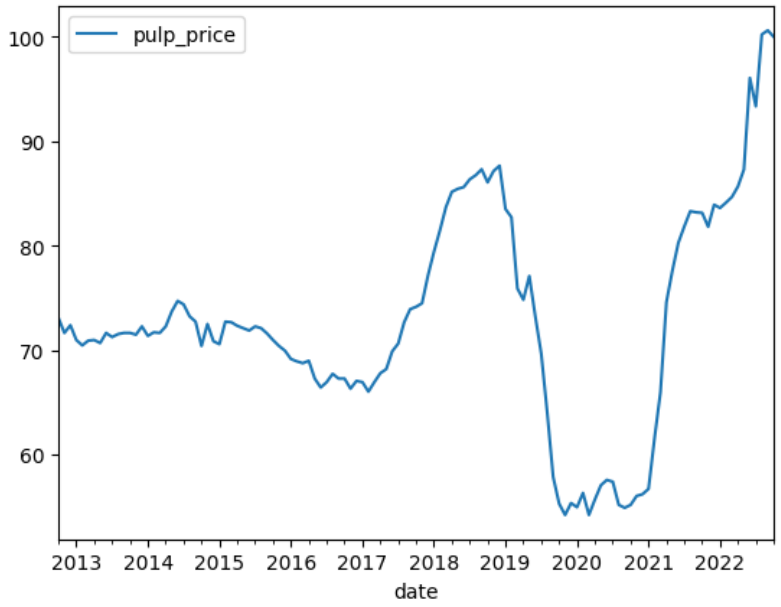


Figure 4.1: Sample of pulp prices used as input to the data generation process for our model.

We would like to generate new time series data that is somehow similar to this original sample. Traditionally, bootstrap re-sampling consists of taking random selections with replacements from the original sample to create a new sample. For example,

$$p_4, p_1, p_1, p_2, \ldots, p_4$$

could be a valid bootstrap re-sampling where the number of terms in the sample is still n which is the same as the original sample. However, in the case of time-series data, we traditionally expect some degree of autocorrelation to exist in the data. Hence, we need to modify our re-sampling procedure to take this into account. One way to do this is called block bootstrapping (thoroughly described and analyzed in [6]). This method allows us to create new time-series data from a single sample while also taking into account the local qualities of the original sample. Briefly, this process is as follows:

1. Given time-series data $T : p_1, \ldots p_n$, form l blocks $B_1, \ldots, B_l$ of length k where $B_1 = \{p_1, p_2, \ldots, p_k\}$, $B_2 = \{p_{k+1}, p_{k+2} \ldots, p_{2k}\}$ and so on so that the union of the B_i is equal to T (assuming for simplicity that the number of data points in the original sample is divisible by l).

2. Next, instead of re-sampling from the original time series with replacement, we re-sample l sets from the collection $\{B_1, \ldots, B_l\}$ with replacement.

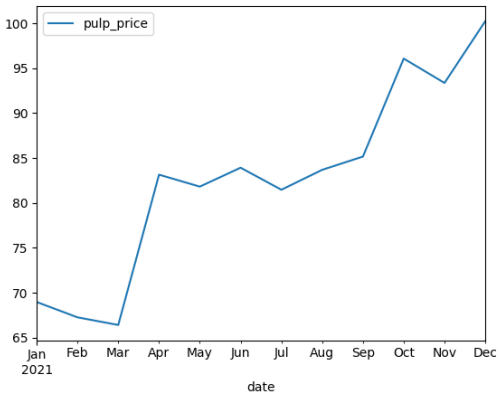
3. Lastly, form a new time series T' by taking the union of the l sets we randomly selected from the previous step.

In this way, provided k is not too small, we preserve the local dependency of the original sample. There are various theoretical results summarized in [6] on the statistical proprieties relating T and T'. However, for our purposes, we take a pragmatic approach and simply use bootstrap re-sampling as a means to efficiently generate more data. For examples of the output of bootstrap re-sampling, refer to Figures 4.2a to 4.2d. In general, we believe this process was successful in generating a diverse number of scenarios. Qualitatively, we can see fairly realistic market scenarios of sudden increases and decreases as well as periods of relative stability.

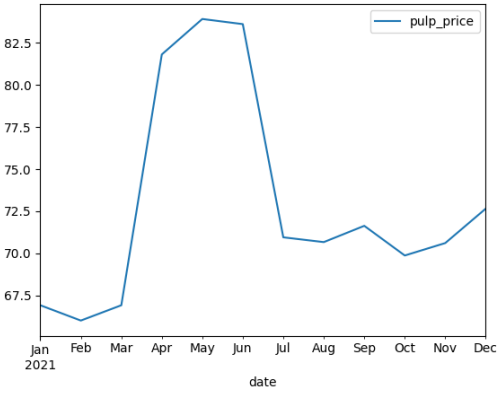
In terms of how this data will be used in the model, each type of raw material will have several scenarios of pricing data that will be generated by using bootstrap re-sampling. In order to get more diverse trends for each raw material, we decided to also change the dates from which we sample from the original time series in Figure 4.1. For example, one raw material might have its data generated by re-sampling from the year 2013 while another from the year 2020 and so on. In this way, each raw material will have its own distinct statistical characteristics and typical trends.

Lastly, we also have a simple data-generating process for the customer demand data. This process is purely synthetic and does not rely on external data sources. Specifically we assume that for a product i, time period t, customer c and scenario s the demand is

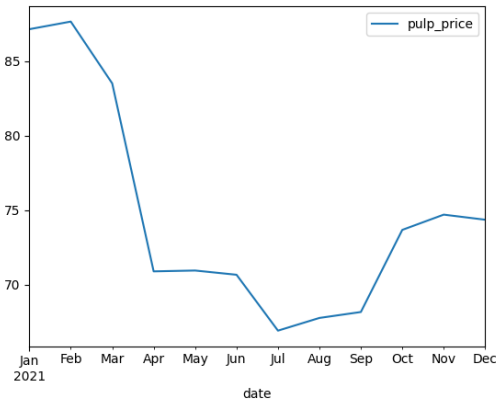
$$D(i, t, c, s) = a \sin(Y_{ic}t + Z_c + X) + aX + W_{ic} \tag{4.1}$$



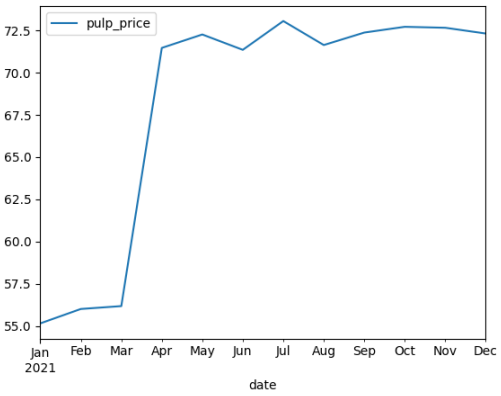
(a) Sample 1



(b) Sample 2



(c) Sample 3



(d) Sample 4

Figure 4.2: Various samples from the block bootstrap process using one year (2021 Jan - 2021 Dec) of the time series in Figure 4.1 as input.

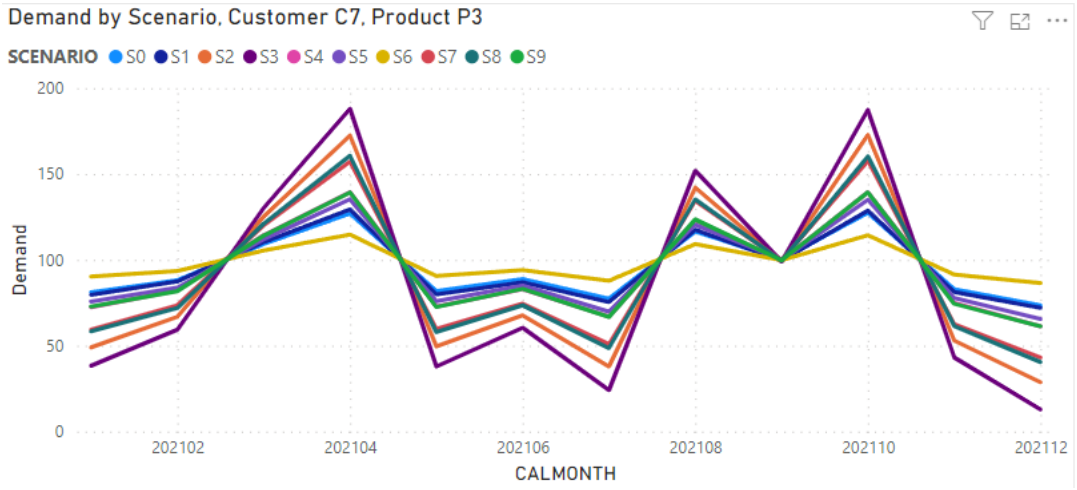


Figure 4.3: Example demand generated by our data generation process for a specific customer and product.

where the X_s, Y_i, Z_c and W_s are uniformly distributed random variables as follows:

$$\begin{aligned} X &\sim U(0,1) \\ Y_{ic} &\sim U(1,2) \\ Z_c &\sim U(0,5) \\ W_{ic} &\sim U(0,1) \end{aligned}$$

and $a > 0$ is a positive real number. The above parameters were purely chosen for convenience and to limit the time spent searching for data. An example of how the demand data looks as a result of this data generation process can be found in Figure 4.3.

In general, we believe there is opportunity for improvement in our data generation methodology. We found the problem of generating realistic data to be fairly challenging. This again reflects the fact that we were mainly focused on understanding the implementation details of disjunctive programming and less on the actual realism of the underlying data in this project. The Python script that performs both main data generation tasks mentioned in this section can be found in `data_generation/gen_input_file.py` located at [7]. This script generates all the parameters necessary to formulate the optimization model described in the next sections.

4.2 Technical information

The model in this thesis was implemented using the Julia programming language along with the DisjunctiveProgramming.jl package (see [13]). Preparation of some of the input data was also performed in Python. The Gurobi solver (see [5]) was used to solve all instance of our model with `MIPGap` set to 0.05%. Computational experiments were performed on a Windows 10 Professional computer with an AMD Ryzen 5 PRO

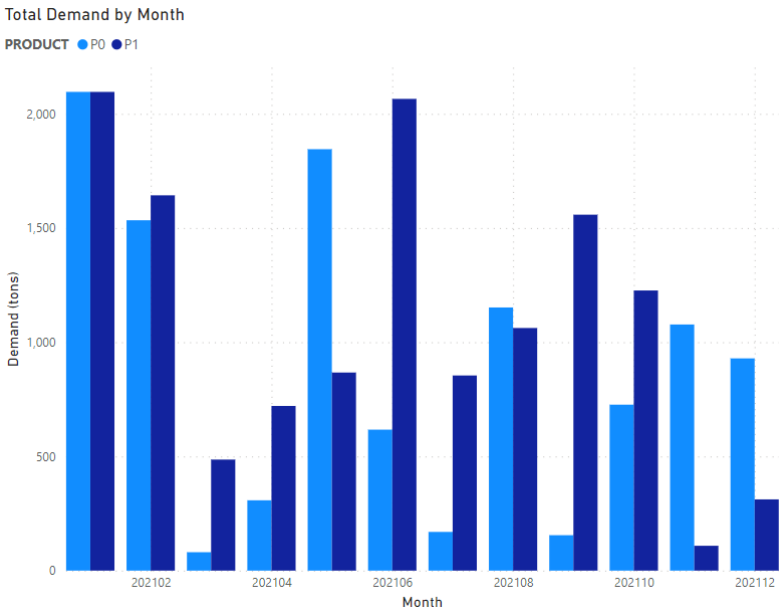


Figure 4.4: Customer demand from problem instance 0 of the deterministic model. Data can be found in CustomerDemand tab of `paper_det_examples/high_demand/RESULT_0.xlsx` located in [7].

2500U 2.00 GHz processor and 16 GB of RAM. Source code and input data used can be found at [7].

4.3 Deterministic model

In this section, we will analyze the model performance considering only one scenario (e.g. without any stochasticity in the demand or raw material market prices). See Table 4.2 for the list of parameters used in this example. The deterministic model is useful as introducing stochasticity gives a significant performance overhead when solving the model to optimality. Hence, the deterministic setting allows us to scale up the model to more closely resemble reality while having acceptable solution times. However, the scale of our parameters are quite small in relation to reality. The structure of this section is as follows: first, we will visualise the input data used as well as the general structure of the problem; second, we will solve the problem considering several instances of randomized input data; lastly, we will analyze the numerical results and indicate some basic conclusions. We also perform a rudimentary cost-benefit analysis to estimate how profitable introducing the complexity of modeling the contracts is. The input data used for all problem instances in this section can be found in the `paper_det_examples` folder of [7].

As mentioned earlier, we will analyze the model performance considering several randomized instances of input data (an instance meaning one specific result of the data generation process described in Section 4.1). However, we will keep the parameters of the model constant to those given in Table 4.2 unless otherwise stated. Figure 4.7 gives a high-level schematic for the model. Raw materials flow from suppliers

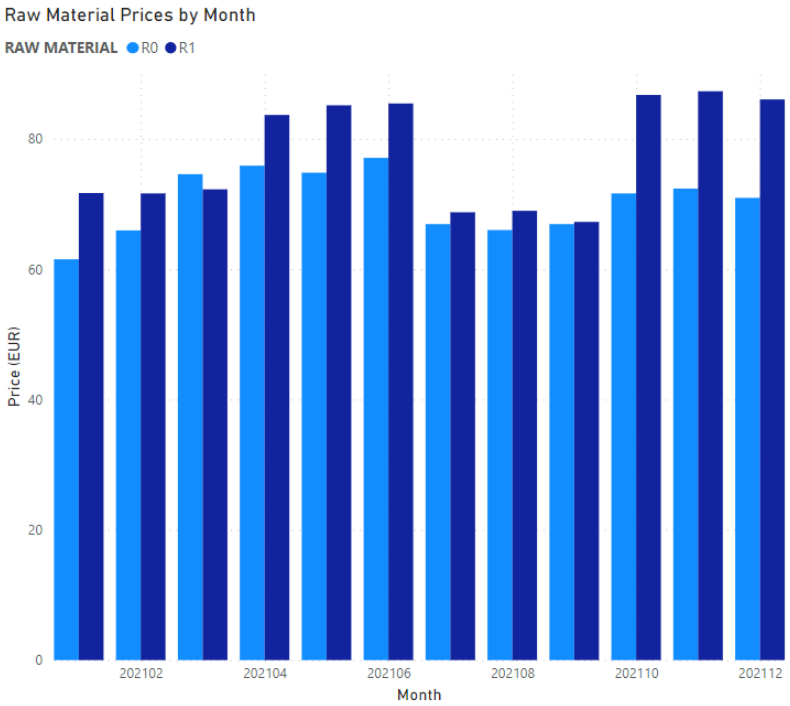


Figure 4.5: Raw material prices (per ton) from problem instance 0 of the deterministic model. Data can be found in CustomerDemand tab of `paper_det_examples/high_demand/RESULT_0.xlsx` located in [7].

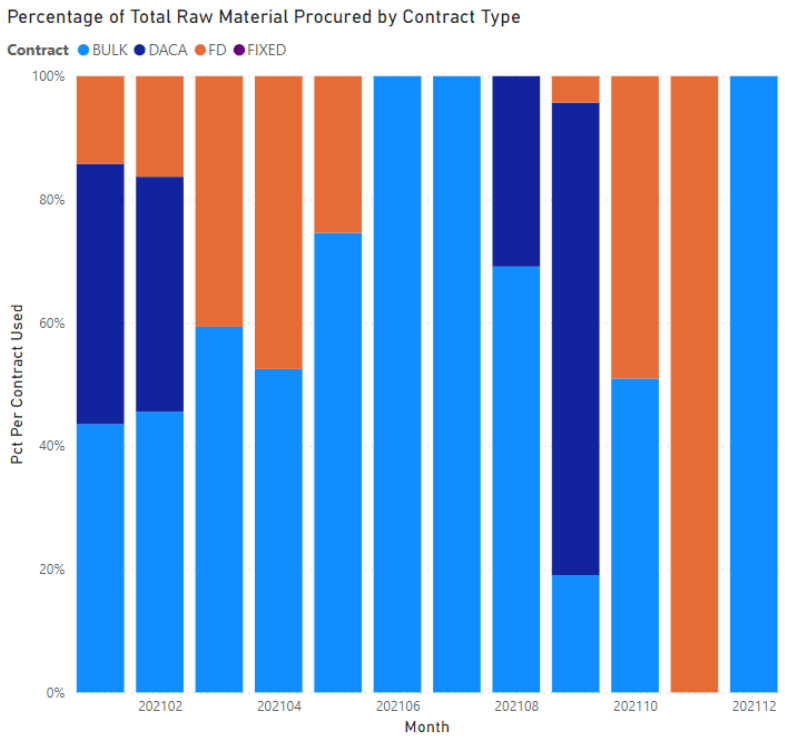


Figure 4.6: Contract selections per each period as a percentage of the total amount of raw material produced. Data can be found in RawMaterialContract tab of `paper_det_examples/high_demand/RESULT_0.xlsx` located in [7].

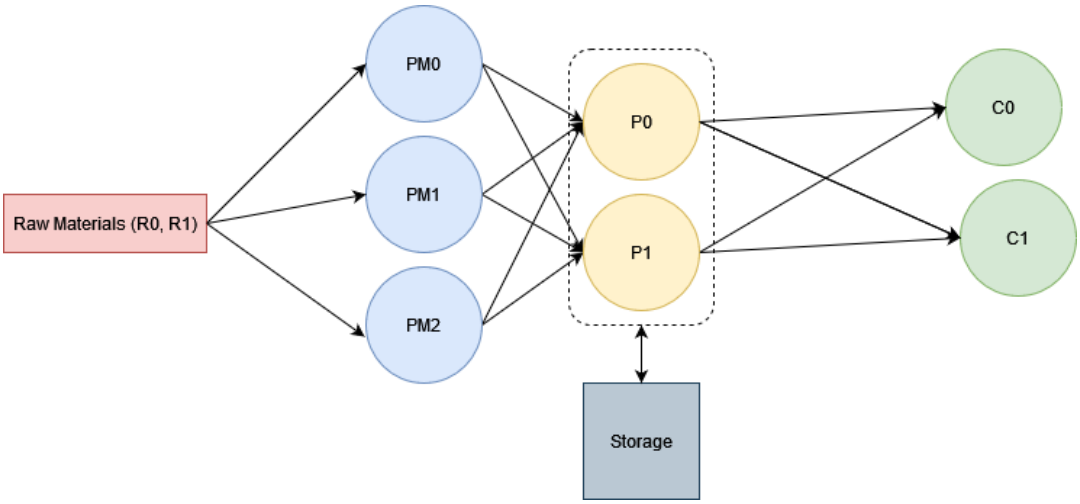


Figure 4.7: High-level schematic of the deterministic model.

to three production units $PM0$, $PM1$ and $PM2$ which in turn produce products $P0$ and $P1$ taking into account the demand of customers $C0$ and $C1$. It is also possible for excess raw material or excess product to be stored in inventory. See Figures 4.4 and 4.5 for an idea of how the demand and raw material prices look for a typical instance of the deterministic model. Lastly, see Figure 4.6 for an example of how the different contract types are used in solving the problem instances. Now that the reader is more accustomed to the data of the problem, we will move on to actually solving the problem instances.

The input data of the problem instances are randomly generated using the process described in Section 4.1. Because of the random nature of the data generation process, we will solve the problem several times considering different instances of the data generation process created by the process described in Section 4.5. This will provide a sensitivity analysis to see if small to medium perturbations in the input data result in different contract selections and also to evaluate the stability of the objective value. The results of generating and solving ten problem instances can be found in Table 4.1 and the input data used for each scenario can be found in the folder `paper_det_examples/high_demand/` at [7]. The columns BULK, DACA, FD and FIXED indicate what percentage of the total purchased raw materials was purchased using the respective contract type. Looking at the Table 4.1, there are a few things that one could point out. First, on average the model obtains an objective value of approximately 14 million EUR with the exception of instances 1, 4 and 7 which have a noticeable jump to around 20 million EUR. This is due to larger amounts of demand in these cases in comparison to the other instances. In all cases, the BULK contract type was used to purchase the largest percentage of the raw materials (see the BULK column of Table 4.1). There is also no instance which uses the FIXED contract type. This is because the FIXED contract type uses the market rate to purchase a particular material. The other contract types are all tied to the market rate so FIXED contracts would only be applicable in the case of high inventory costs (so that the model cannot group small demands together) and low demands (so that discounts will not apply).

Instance	Obj. value	Runtime (min)	BULK	DACA	FD	FIXED
0	13023671	3.6	52.7	27.8	19.6	0
1	23027574	20.7	63	8.2	28.9	0
2	14979594	1.2	56.2	0	43.8	0
3	10486945	1.2	65.4	15	19.6	0
4	21541664	3	68.7	13.5	17.9	0
5	17983085	1.2	79.6	20.4	0	0
6	11985918	1	54.1	24.6	21.3	0
7	20671034	11.4	70.5	0	29.5	0
8	15261114	2	85.2	5.5	9.2	0
9	15510437	1.4	72.7	10.3	17	0

Table 4.1: Results from running ten problem instances (high demand, $a = 500$). Data files for these instances are placed in `paper_det_examples/high_demand/` located in [7].

For example, in Table 4.3 we ran a similar experiment after reducing the amplitude a in Equation 4.1 from the data generation process. In the numerical experiments in Table 4.3 we also set all parameters to 1 to try to speed up performance. In our experience this did not impact the resulting contract portfolio. That is, with all parameters set to 1 and $a = 500$, we see a similar contract portfolio as Table 4.1. In Table 4.3, we can see BULK is preferred 100 percent of the time in almost all cases. This is because the model tends to buy in bulk at a single or small number of early periods and store the rest in inventory. For example in Instance 0, almost all of the raw material was bought in the second time period (February, 2021) and then stored in inventory to be used in later periods. In general, it appears the contract portfolio given by the model is highly dependent on demand. This is to be expected considering all of the contract definitions have logic that yields a more favorable discount after a certain amount purchased. On the other hand, if inventory costs are sufficiently high, then we will see the FIXED contract being used more dominantly as can be seen in Table 4.4.

Parameter	Size
Number of PMs	3
Number of Customers	2
Number of Products	2
Number of Raw Material	1
Number of Scenarios	1
Number of Periods	12

Table 4.2: Parameters used for deterministic model.

Overall, we found these results to be as expected. The contracts have discounts which allow the manufacturer to take advantage of economies of scale. Thus, it is

Instance	Obj. value	Runtime (min)	BULK	DACA	FD	FIXED
0	198896	0.13	95.3	0	0	4.7
1	89291	0.01	100	0	0	0
2	152759	0.01	100	0	0	0
3	148501	0.01	98.8	0	0	1.2
4	250150	0	99.8	0	0	0.2
5	105777	0.01	100	0	0	0
6	133699	0.01	100	0	0	0
7	85195	0.01	100	0	0	0
8	103553	0.01	100	0	0	0
9	212911	0.01	100	0	0	0

Table 4.3: Results from running ten problem instances (low demand, $a = 20$, all parameters listed in Table 4.2 set to 1 to improve performance). See the input files in `paper_det_examples/low_demand` at [7].

Instance	Obj. value	Runtime (min)	BULK	DACA	FD	FIXED
0	130324	1.7	80	14	0	6
1	135923	0.13	25	0	0	75
2	115933	0.13	49	0	40	10
3	93007	0.13	0	0	0	100
4	268055	0.13	68	0	19	13
5	61641	0.13	56	0	0	44
6	104276	0.13	23	0	39	38
7	184437	0.13	36	10	0	53
8	118130	0.13	13	0	39	47
9	80156	0.13	02	20	51	8

Table 4.4: Results from running ten problem instances (low demand, $a = 20$, all parameters listed in Table 4.2 set to 1 to improve performance). We also increased inventory costs significantly which caused FIXED contracts to be used more frequently. See the input files in `paper_det_examples/low_demand_high_storage_costs` at [7].

not surprising when high demand results in contracts with discounts being used and more pessimistic situations result in purchasing by the market rate. The results of the numerical experiment for the deterministic model confirmed to us that the model is working correctly and gives sane results. One interesting conclusion is that it seems the choice of contracts are relatively robust against random changes in demand and raw material price fluctuations. Despite randomly generating the data for each instance, there were similarities in the resulting contract choices. For example, in the high demand case illustrated in Table 4.1, the demand time series for Instance 1 and Instance 5 are fairly different (see Figure 4.8) yet both have a similar distribution of contracts with BULK being the dominant one.

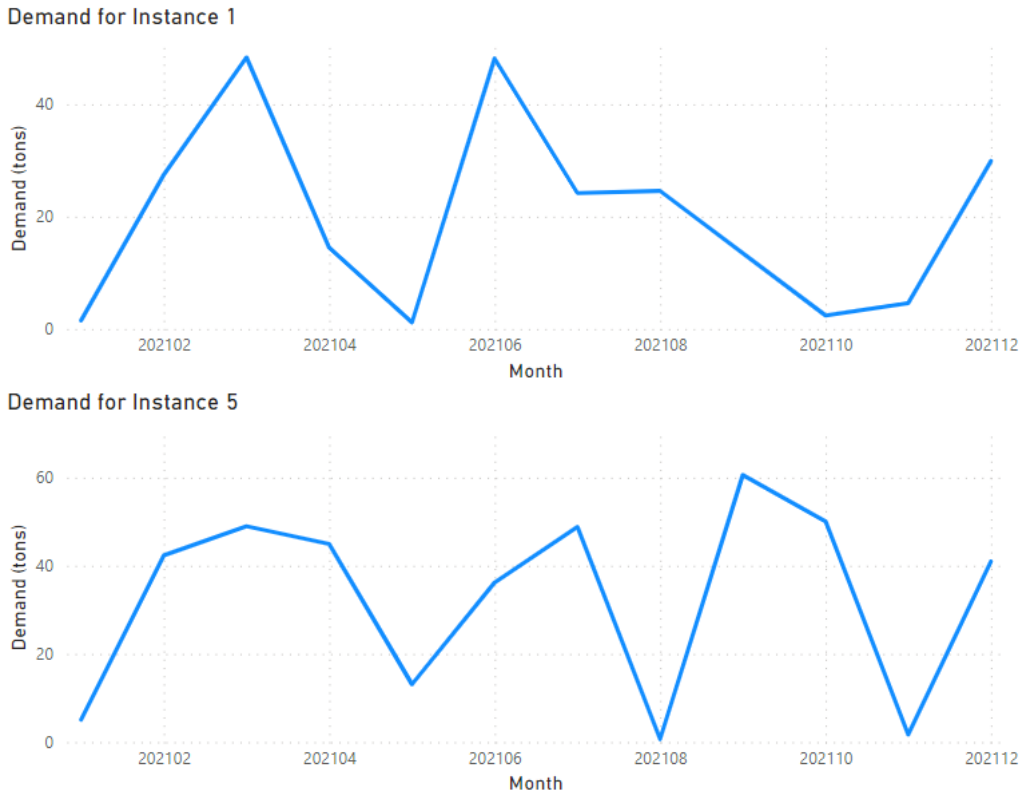


Figure 4.8: Demand developments for Instance 1 and Instance 5 in high demand case from Table 4.1.

This is also easily interpretable. The definitions of the DACA, BULK and FD contracts are purely dependent on the total amount of demand and not on how the demand changes over time. Provided that in total there is enough demand to fulfill the discount limit of each contract and that inventory is available and sufficiently cheap, DACA, BULK or FD will always used by the model since a discount will be applied and the objective value will increase. Overall, this gave us a feeling that implementing a stochastic formulation with multiple scenarios might not give as strong improvement to the objective function as we initially though. We will investigate this more in the next section.

Lastly, we conclude with a simplistic cost-benefit analysis. Intuitively, the presence of contracts in the model allows us to take advantage of economies of scale and significantly reduce costs. The cost of modeling contract types is that we transform what was a production planning problem that did not distinguish between contract types into a more complicated MILP. To estimate the potential advantage of modeling different contract types, we have computed objective values considering several input data instances. The input data can be found in `paper_det_examples/high_demand_simplified` at [7]. For each instance, we solved the problem with and without contracts. The results can be seen in Table 4.5. In general, there was a reasonable benefit in implementing the MILP constraints to model different contract types with an average 3.8 percent increase in the objective value. Of course, in practise the benefit will depend on market

conditions. If demand is high and contracts exist with significant discounts, we could see even larger improvements. On the other hand, if demand is low and discount rates are not favorable then there is little benefit in introducing different contracts.

In conclusion, this section was a basic analysis of the deterministic case of our model. That is, we did not introduce stochasticity into demand or raw material market prices. We conducted an overview of the input data as well as performed several numerical experiments that validated our intuitions about the model behavior. It was noted that contract selections appear to be robust against random changes in the input data and that this could potentially affect the value of implementing a stochastic model. Lastly, we quantitatively estimated the potential benefit of modeling different contract types in our planning problem and concluded that there was a reasonable increase in objective values as exhibited in 4.5; however, the magnitude of the increase will necessarily depend on the market conditions and industry practises.

Instance	Obj. value (contracts)	Obj. value (no contracts)	Absolute chg	Pct. chg
0	2696720	2549349	147371	5,8
1	4801842	4659040	142802	3,1
2	5880741	5649311	231430	4,1
3	4390298	4220882	169416	4
4	3587040	3450037	137003	4
5	2324270	2281367	42903	1,9
6	3975660	3855006	120654	3,1
7	6487264	6287059	200205	3,2
8	4522111	4366880	155231	3,6
9	3394824	3230683	164141	5,1

Table 4.5: Objective value comparison with and without contracts. As before all parameters listed in Table 4.2 were set to 1 and the input data from `paper_det_examples/high_demand_simplified` was used located in [7]. The column Obj. value (contracts) contain the objective value of solving the model with modeling contract types. Similarly Obj. value (no contracts) contain the objective value of solving the model without modeling contract types. Lastly, Absolute chg and Pct. chg give the difference and percent difference between the objective values respectively.

4.4 Stochastic model

In the previous section, we analyzed the model in the deterministic case. For an individual instance, there was no stochasticity in demand or raw material prices. We will now introduce randomness in raw material prices and demand so that the model becomes a two-stage stochastic programming problem. The first-stage decisions will be choosing which contract types to use at each time period. The second stage decisions will be from the production planning model itself. Specifically these second-stage decisions are the decision variables that were defined in Section 3.2. Similar to

Parameter	Size
Number of PMs	1
Number of Customers	1
Number of Products	1
Number of Raw Material	1
Number of Scenarios	20
Number of Periods	12

Table 4.6: Parameters used for stochastic model.

the previous section, we will generate multiple problem instances using our data generation process. However, in this case, there is now randomness in demand and raw material prices. This means that for a single problem instance, there will be several scenarios for demand and raw material prices instead of just a single scenario as was done in the previous section. Naturally, introducing more scenarios, increases the number of variables in the model and the resulting solution time. In order to keep solve times manageable, we use the parameters given in Table 4.6. These are much more conservative then the previously used parameters (see Table 4.2) with the exception of Number of Scenarios which was increased to 20 since we now have stochasticity in demand and raw materials. This is a significant simplification but we believe it is permissible since our main interest is on understanding the contracts selected and not so much on the realism of the underlying production planning model. Of course, this is certainly a limitation of our work and should be kept in mind when interpreting our results. Lastly, we also deviated slightly from the data generation process originally presented in Section 4.1. Instead of the sinusoidal-based method of generating demand, we opted for a slightly more complicated approach that simulates a process with autocorrelation for each scenario. The full version of the code can be found in `data_generation/gen_input_file.py` (see [7]) and Figure 4.9 should convey the general idea of having forecasts that gradually become more uncertain into the future. The demand generation process used to generate data in the deterministic formulation was essentially a randomized sine wave. This is enough for the deterministic case because there is only a single fixed scenario under consideration. However, for the stochastic formulation, there are multiple scenarios for customer demand and as time evolves their should be more uncertainty in the customer demand which is why we switched from the previous randomized sine wave approach. As can be seen in Figure 4.9, scenarios will "fan out" over time reflecting increased uncertain in the true demand.

With the above remarks in mind, this section will largely revolve around understanding two metrics relevant to stochastic programming models. First, there is the Value of the Stochastic Solution (VSS). The VSS provides a measure of the value of implementing the stochastic model (the definition we use is aligned with [3]). Formally, VSS is given by

$$VSS = RP - EV \tag{4.2}$$

where RP is the objective value of the so-called *recourse problem* and EV is the objective value of the *expected value* problem. The recourse problem is the standard stochastic programming formulation without modification. The expected value problem is a completely deterministic formulation where we have replaced parameters that vary with scenario by their expected value (hence a problem with multiple scenarios is reduced to the case of one). The RP problem is solved first and then the first stage decisions of the EV problem are fixed to the same values decided by the RP problem. In this way, we can make a reasonable comparison between the stochastic and deterministic formulations of the model. If the VSS is large, then the stochastic formulation has some value over the deterministic form and might be worth considering for implementation. However, if the VSS is small or zero, this suggests little benefit is given by the stochastic formulation. That is, in some sense, stochasticity does not play a strong role in further increasing the objective value of the problem.

The second metric is the Expected Value of Perfect Information (EVPI) as introduced in Chapter 4 of [2]. In the context of stochastic programming, EVPI measures the maximum amount of improvement to the objective function that could be made if we could remove uncertainty from the model. In our case, that means we would know the raw material prices and demand exactly so that introducing uncertainty is not needed. Mathematically, EVPI is defined as

$$EVPI = WS - RP \tag{4.3}$$

where RP is as defined as before and WS (known as the wait-and-see solution) is the probability-weighted sum of objectives from solving the problem for each scenario under consideration. The difference between EVPI and VSS is that VSS measures the gain from changing a fundamental aspect of the model formulation: going from a deterministic model to a stochastic model. On the other hand, EVPI is less about the formulation of the model and more a measure of how much we should pay to reduce uncertainty. For example a company might want to understand how much value could come from improving the certainty of the estimates used for the demand scenarios. EVPI rather than VSS would be the metric to use in that case.

Using these two metrics, we can analyse the results in Table 4.7. The table is similar to the tables used during the discussion of the deterministic model. Each row corresponds to a solved problem instance of the model. The input data used can be found in `paper_stoc_examples` at [7]. The columns are as follows: RP is the objective value of the recourse problem, EV is the objective value of the expected value problem, VSS is the value of the stochastic solution, VSS Pct. Chg is given by $100 \times (RP - EV)/EV$ and EVPI is the expected value of perfect information. We can see our stochastic model's VSS is generally quite a bit smaller than the 10 per cent increase the authors of [3] obtained. We believe this is mainly due to the fact that [3] includes product selling prices in first-stage decisions in addition to contract-type decisions. It seems plausible this would ultimately allow the model to increase profitability and more greatly exploit certain demand patterns (e.g. set higher prices for high demand). Because this introduces non-linearity into the problem, we opted to only include contract decisions in our model and not have selling prices for our products also as decision variables. However, there is a reasonably large VSS for

some problem instances such as 7 and 8. We will analyze these in detail to see what is driving the larger VSS.

Instance	RP	EV	VSS	VSS Pct. Chg	EVPI
0	706400	701082	5319	0.76	11646
1	1229996	1214886	15111	1.24	8408
2	274598	264214	10384	3.93	6812
3	623924	619063	4861	0.79	6584
4	1173028	1164411	8616	0.74	5997
5	942539	942303	236	0.03	5159
6	898363	877338	21025	2.4	7663
7	337913	314924	22990	7.3	4499
8	264331	250689	13642	5.44	7667
9	1122632	1110775	11858	1.07	7821

Table 4.7: Results for running the stochastic model 10 times.

In general, as we suggested in the previous section, the stochastic model behaves quite differently for scenarios with low and high demand. For cases where demand is generally lower across scenarios, the VSS is higher. For cases where demand is higher across scenarios, the VSS is lower. This seems to be the main phenomenon driving the difference between VSS in Instances 5 and 7 (these two instances represent the extremes with instance 5 having low VSS and instance 7 having high VSS). Figure 4.9 provides a visualisation of the difference in the demand scenarios for the two problem instances with Instance 7 clearly having less demand in all its demand scenarios. We can also see in Table 4.9 that, as we would expect, Instance 5 has almost no difference in contract choices when comparing the optimal contract portfolios between the RP and EV problems. On the other hand, Instance 7 is the opposite. The EV problem made substantial use of the BULK contract type in the later periods of the model while the RP problem only used it twice.

In conclusion, we believe that there is evidence that would support the use of the stochastic model. Namely, we found that VSS was positive in all cases and in some specific instances such as problem instances 7 and 8 there is a material increase in the objective function when using the stochastic model relative to the expected value formulation. However, the benefit of the stochastic model appears to be heavily dependent on the overall level of demand in the dataset. For cases where demand is low (e.g. close to or below the discount limits of the contracts), it seems the stochastic model provides a good improvement in increasing the objective function. However, in cases where all scenarios have demand above the discount limits, first-stage decisions appear to be quite robust in the face of uncertainty. Namely, we found that the relative difference between the RP and EV objective function values was smaller. Thus, it is was not possible to conclude that the stochastic model is superior in all cases.

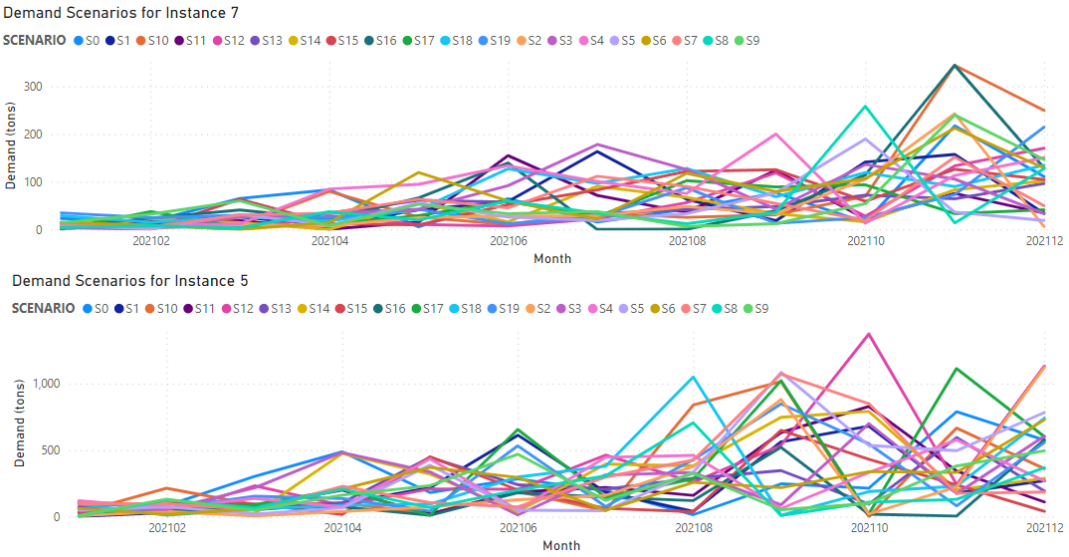


Figure 4.9: Demand scenarios for Instance 7 (top) and Instance (5).

Period	Inst. 5, RP	Inst. 5, EV	Inst. 7, RP	Inst. 7, EV
202101	BULK	BULK	FIXED	FIXED
202102	FD	FD	DACA	DACA
202103	FD	FD	FIXED	FIXED
202104	FD	FD	FIXED	FIXED
202105	BULK	BULK	DACA	DACA
202106	BULK	BULK	FIXED	FD
202107	BULK	BULK	FIXED	FD
202108	BULK	BULK	BULK	BULK
202109	BULK	BULK	BULK	BULK
202110	DACA	DACA	FD	BULK
202111	DACA	BULK	FD	BULK
202112	BULK	BULK	FD	BULK

Table 4.8: Differences in contract decisions between the two instances. Inst. 5, RP and Inst. 5 EV are the contract decisions made in the regular stochastic problem and the deterministic formulation respectively.

Chapter 5

Conclusion

In this thesis, we implemented a two-stage stochastic MILP production planning model where supply contracts for raw materials were modeled using disjunctive programming. Julia was used to implement the model in practise and it was solved using the Gurobi solver. Overall, we found that the framework of disjunctive programming seems to provide an expressive language for modeling contracts. We conclude that the approach of using higher-level logical operators to describe the contracts rather than traditional MILP constraints allowed us to focus more on the big picture of the problem and less on the technicalities of implementing the contracts-related constraints. However, it does not mean introducing disjunctive programming is always helpful in modeling contracts. As seen in the literature review, some problems may be structured so that the usage of disjunctive programming would only complicate things unnecessarily.

Regarding our results, we analyzed two models with the first being a deterministic version and the second being the full stochastic formulation. The stochastic formulation introduced stochasticity (e.g. multiple scenarios) into the raw material price data and product demand data. The deterministic analysis was very simplistic and it drew our attention to the fact that simulating data for optimization problems is quite a challenging task at times. One needs to generate data that is not biased (e.g. artificially generating interesting behavior) but at the same time interesting enough so that the more complicated aspects of the model have a chance to shine. We concluded that the deterministic model appeared to work consistently with our expectations. A variety of contracts from all types (DACA, BULK, FD and FIXED) were selected depending on characteristics of the data. Specifically, we found that the overall level of product demand to be a driving factor in the resulting contract selections. In cases where demand was low, the model would group purchases together and only use the BULK contract type as was seen in Table 4.3. On the other hand, pairing low demand with high inventory costs resulted in increased use of the FIXED contract type as evidenced in Table 4.4.

The results of the stochastic model were less straightforward to analyze. The two metrics we calculated Value of the Stochastic Solution (VSS) and Expected Value of Perfect Information (EVPI) were relatively low. A low VSS indicates that the work of adding stochasticity to the formulation does not necessarily provide a significant increase in the objective function. On the other hand, a lower EVPI indicates

that the value (e.g. corresponding increase in objective value) in procuring more exact information about demand and raw material prices is small (assuming they are uncertain). The largest percentage change in objective we observed was only around 7 per cent upon moving to the stochastic model. As observed in the deterministic case, the behavior of the contract choices in the stochastic model depended heavily on the level of demand in the scenarios. Problem instances with mostly high demand scenarios (relative to the discount limits of each contract type) resulted in solutions with small VSS. However, instances with lower demand scenarios had a higher VSS. When VSS was high, there was a reasonable difference in contract choices between stochastic formulation and deterministic expected value formulation. Thus, we believe our work suggests the stochastic model could be worth implementing in some cases but we cannot provide a uniform recommendation.

For future work, there are quite a few avenues which should be explored. In our view, the greatest weakness of the current work is the crude data generation process rather than the model itself. Synthesizing the input data was consistently the hardest part of our work. Going forward, we should revisit this process and try to incorporate more real-world data. The parts that are randomly created should be done in a more statistically rigorous way. Specifically, the demand data generation process as well as the raw material price generation process were implemented in a very ad-hoc manner that was mainly for convenience rather then being based on existing literature. The model also has limitations that can be addressed in the future. Firstly, we struggled to obtain reasonable solution times when integer variables and constraints were introduced. We believe one cause of this is the underlying production planning model. There are likely too many variables than needed and more clever formulations could improve performance. Next, as it currently stands, the small number of customers and products we have used in our examples means it would not be feasible to deploy the model in practise where these parameters are typically in the hundreds. It would be interesting to obtain access to more enterprise-grade hardware and run the model on a larger scale. This is also something quite lacking in the literature as a whole with many disjunctive programming works running the model on consumer-grade desktops.

In conclusion, disjunctive programming is an effective modeling paradigm that narrows the gap between how humans naturally think about business logic and the corresponding description in formal language. Similar to how high-level programming languages such as C, Java and Python abstracted assembly language and fundamentally changed software development, perhaps concepts such as disjunctive programming will fundamentally change mathematical modeling by allowing users with high levels of business knowledge but comparatively less technical knowledge to formulate models easier. One could imagine AI-assisted user-friendly tools that allow the creation of disjunctive constraints in plain language that are then "compiled" down into MILP constraints. The users themselves would not need to know the existence of the MILP constraints or understand them similar to how most software developers do not understand the assembly code produced by compiling the programs they write using high-level languages.

Acknowledgement

Getting this done was a battle. I am very thankful to my supervisor and advisor for the constant support despite things taking longer than we all would have liked. Lastly, I am most thankful to my wife Honoka as well as my grandparents. It would not have been finished without them as my source of motivation. While I am not entirely proud of the results of this thesis; I am proud of how much I learned along the way. The countless wrong paths I explored and the mistakes I made taught me lessons I will remember long into the future.

Tom Holt
September, 2023

Bibliography

[1] Egon Balas. *Disjunctive programming*. Springer, 2018.

[2] John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[3] Bruno A Calfa and Ignacio E Grossmann. "Optimal procurement contract selection with price optimization under uncertainty for process networks". In: *Computers & Chemical Engineering* 82 (2015), pp. 330–343.

[4] Pedro M Castro and Ignacio E Grossmann. "Generalized disjunctive programming as a systematic modeling framework to derive scheduling formulations". In: *Industrial & Engineering Chemistry Research* 51.16 (2012), pp. 5781–5792.

[5] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2023. URL: https://www.gurobi.com.

[6] Wolfgang Härdle, Joel Horowitz, and Jens-Peter Kreiss. "Bootstrap Methods for Time Series". In: *International Statistical Review* 71.2 (2003), pp. 435–459.

[7] Thomas-Roy S. Holt. *Thesis*. https://github.com/th28/thesis. 2023.

[8] Rajab Khalilpour and IA Karimi. "Selection of liquefied natural gas (LNG) contracts for minimizing procurement cost". In: *Industrial & engineering chemistry research* 50.17 (2011), pp. 10298–10312.

[9] U.S. Bureau of Labor Statistics. *Producer Price Index by Commodity: Pulp, Paper, and Allied Products: Wood Pulp — fred.stlouisfed.org*. https://fred.stlouisfed.org/series/WPU0911. [Accessed 28-Jun-2023].

[10] John T Mentzer et al. "Defining supply chain management". In: *Journal of Business logistics* 22.2 (2001), pp. 1–25.

[11] David S Moore and George P McCabe. *Introduction to the Practice of Statistics*. WH Freeman/Times Books/Henry Holt & Co, 1989.

[12] Minhwan Park et al. "Modeling of purchase and sales contracts in supply chain optimization". In: *Industrial & Engineering Chemistry Research* 45.14 (2006), pp. 5013–5026.

[13] Hector D Perez, Shivank Joshi, and Ignacio E Grossmann. "DisjunctiveProgramming. jl: Generalized Disjunctive Programming Models and Algorithms for JuMP". In: *arXiv preprint arXiv:2304.10492* (2023).

[14] Ali Rahimi et al. "An optimization model for selecting wood supply contracts". In: *Canadian Journal of Forest Research* 50.4 (2020), pp. 399–412.

[15] Ali Rahimi et al. “Selecting wood supply contracts under uncertainty using stochastic programming”. In: *INFOR: Information Systems and Operational Research* 59.2 (2021), pp. 191–211.

[16] Maria Analia Rodriguez and Aldo Vecchietti. “Logical and generalized disjunctive programming for supplier and contract selection under provision uncertainty”. In: *Industrial & engineering chemistry research* 48.11 (2009), pp. 5506–5521.

[17] Andy A Tsay, Steven Nahmias, and Narendra Agrawal. “Modeling supply chain contracts: A review”. In: *Quantitative models for supply chain management* (1999), pp. 299–336.

Notation guide

The reader might find Table .01 - .03 useful. Symbols as they appear in the code are mapped to the corresponding symbol in the model as it is described in LaTeX.

Model	Code
$\phi_{tr}^{\text{DACA},i}$	`dacapr`
ϕ_{tr}^{FIXED}	`RP`
$\phi_{tr}^{\text{FD},i}$	`fdpr`
$\phi_{tr}^{\text{BULK},i}$	`bulkpr`
$\sigma_{rt}^{\text{DACA}}$	`dacalim`
$\sigma_{rt}^{\text{BULK}}$	`bulklim`
$\sigma_{rt}^{\text{FD},i}$	`fdlim`
α_{pr}	`a`
δ_{cpt}^{s}	`D`
κ_{u}	`PC`
λ_{cu}	`L`
θ_{m}	`SC`
γ_{m}	`ST`
ψ_{pt}	`PR`
π^{s}	`Prb`

Table .01: Mapping between LaTeX notation and code for parameters.

Model	Code
A	`A`
M	`M`
U	`PM`
P	`P`
R	`R`
C	`C`
T	`T`
S	`Scn`

Table .02: Mapping between LaTeX notation and code for sets.

Model	Code
z^{a}_{mrt}	`z`
l^{21}_{mrt}	`slack2_1`
l^{31}_{mrt}	`slack3_1`
l^{32}_{mrt}	`slack3_2`
c^{a}_{mrt}	`RC`
$r^{\text{DACA},i}_{mrt}$	`daca`
c^{BULK}_{mrt}	`rcost_bulk`
y^{s}_{mrt}	`RI`
x^{s}_{cput}	`x`
d^{s}_{cput}	`d`
w^{s}_{cmpt}	`I`
q^{s}_{cput}	`demand_slack`
c^{FD}_{mrt}	`rcost_fd`
c^{FIXED}_{mrt}	`rcost_f`
c^{DACA}_{mrt}	`rcost_daca`
SALES_{s}	`sales`
RCOST_{s}	`rcost`
ICOST_{s}	`icost`
LCOST_{s}	`lcost`

Table .03: Mapping between LaTeX notation and code for variables.