STOCHASTIC PROGRAMMING SOLUTIONS TO SUPPLY CHAIN MANAGEMENT

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF
MANAGEMENT SCIENCE AND ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
OPERATIONS RESEARCH

Rene C. Schaub
March 2009

Abstract

We formulate a finite-horizon multistage stochastic linear program model for tactical supply chain planning problems and study the effect of random demands on expected profit, through appropriate material flow and inventory allocation.

The associated deterministic model is a capacitated Leontief substitution system, which itself is an instance of a capacitated Hypergraph. We find a fast algorithm to compute approximate solutions, based on Positive Linear Programming.

We explore different decomposition approaches and suggest enhancements to the Benders decomposition method. Supply chain test cases from the literature are used to generate models with large numbers of scenarios and stages, for which we compute near-optimal solutions.

Acknowledgments

I would like to thank my advisor Prof. Gerd Infanger for his guidance and advice that always turned out to be to the point. He showed much patience for my deliberate branching into areas that I thought merited investigation. I would like to thank Profs. Michael Saunders and Yinyu Ye for their numerous advice and contageous optimism. Profs. Serge Plotkin and Warren Hausman agreed to serve on my defense committee. I could not have hoped for a better matched group of experts to present my findings. My girlfriend of many years Galicia Vaca could not believe that I finally finished this project of mine, and I am grateful for her support through all this time and hard work. Sometimes I think I may not have been able to find the endurance to finish my dissertation without her. I would also like to thank my parents for always offering support to all my endeavours, and the companies that provided me with consulting and work opportunities, providing financial independence and the ability to go where I wanted to be.

Contents

Tables, Figures, and Algorithms

Tables

Figures

Chapter 1

Introduction

Operational or tactical supply chain planning problems (TSCP) have a horizon of less than one year, and have long been the domain of Material Resource Planning (MRP), see [33] and [52] for historical surveys. The decisions to be made concern material flow, stocking, sourcing and timing [17, 39, 58]. Because of the short horizon it is assumed the supply chain network itself has already been designed and built; that is, manufacturing plants, distribution centers, and potential transportation routes already exist.

MRP uses rules to make decisions. When there are many suppliers, a sourcing rule may allocate each one a fixed percentage of business. When an order is confirmed and material released, the planning process takes one step at a time in looking up which materials go into the current assembly, and allocates them at the earliest availability, or creates new requisitions, which in turn are followed upstream in the supply chain until all components have been accounted for and resources allocated.

Kanban is a very simple, 'just-in-time' (JIT) manufacturing planning system popularized by Toyota. Contrary to the above 'push'-based system — meaning orders can result in coordinated material decisions affecting many stages of the chain all at once — Kanban is a 'pull'-based system. Material orders are only released when a certain number of units have been used up (Kanban cards), and only for the immediate upstream stage. That way, material is 'pulled' down the chain. The advantages are that very little coordination is required, implementation is easy, and decisions can be made very quickly. The process of setting the right number of Kanban cards is made in advance and is itself an independent planning process.

What has crystallized in recent years in the commercial space is a focus on more and more integrated multi-echelon planning (enterprise or inter-enterprise wide) as opposed to the traditional decentralized local approaches as it has become clear that there are additional cost savings to centralized inventory planning once uncertainties are made part of the model [55, 36, 19]. This requires that all of the enterprise data be accessible from a central location. The enabler of such integrated data storage in practice are so-called Enterprise Resource Planning (ERP) systems [34].

During the last decade, operations research optimization techniques have been applied to commercial MRP software, typically going under the name of 'Advanced Supply Chain Planning' or MRP II. For early approaches see [4]. One typically sees a combination of linear programming and constraint programming. More recently, some approaches have considered uncertainties in the supply chain. Here we are mainly concerned with demand uncertainties. We propose a model that optimizes over multiple items and multiple stages ('echelons'), globally over a certain number of time units. While it may seem that a multi-echelon view spanning production and distribution would make the model too unwieldy, we build on simple constructs that allow us to take advantage of structural properties. Furthermore we restrict ourselves to continuous variables to make the model computationally tractable. For an overview of modeling approaches and issues, see [16].

Related issues that we do not address here are integral constraints such as used in the lot-sizing problem, or designing the supply chain. The latter can be solved separately and in advance of the TSCP, while the former can be treated as a post-processing step.

1.1 Finite Horizon

We use a finite-horizon formulation. It can be applied for seasonal demand that vanishes after the last stage, or over the life cycle of a family of products. An infinite horizon or static model can be approximated by making the horizon sufficiently long and rolling over the time window after some number of periods.

1.2 Aggregation

The length of a period is typically chosen to be between a day and a month. This implies aggregation of all time-related entities of the supply chain, such as demand amounts, resource capacities, and rounding of production lags to time periods. Clearly, accuracy is lost, and optimization of such a model may require post-processing of the solution to enforce constraints locally at a finer level. In the extreme case, aggregation results in all productions finishing in the same time bucket. The only productions with non-zero lag are the inventory productions, carrying over stock to the next period. Demand is satisfied instantaneously. The only remaining issue is material stocking levels if production cannot meet demand spikes. On the other end, we have a continuous time spectrum with individual orders and exact production lags. A compromise is struck to balance computational and other effort with accuracy.

1.3 Continuous Variables

Our model is a continuous one, with decision variables taking values on the real line. We don't include binary or integer variable constraints, which result in problems that in general cannot be solved to optimality (*NP hard*). The main victims are setup costs and lot sizing, which, however, become progressively less important at higher levels of aggregation. One possibility for approximately modeling setup costs is to introduce multiple bills of materials with different lag times and capacities, corresponding to the length of production runs. Each bill will have the setup cost added as part of the total cost. The item output of the BOM can also be spread out evenly. The longer the lag, the lower the per-unit setup cost. It is not an entirely accurate way to measure setup costs, because the total amount being built is a continuous variable and may be less than capacity. The idea is that committing for a long run has a much higher risk of producing too much or too little, if the long lag is used to reduce setup costs only. As our model is capable of optimizing such profit risks, it will select shorter runs with higher setup cost if responsiveness to changes in demand is important.

Chapter 2

Stochastic Programming for Supply Chain Management

2.1 Productions as a Directed Hypergraph Model

Let us start by looking at a simple bill of materials (BOM), which describes the composition of an item made from raw materials or intermediate items.

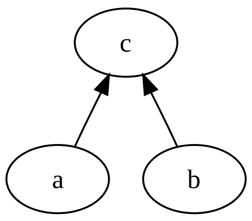

FIGURE 2.1 *A simple bill of materials*

We assume that Figure 2.1 means that to make one unit of item c, we need one unit of item b and one unit of item c.

We can express this relationship as a hyperedge $e \in E$ in a hypergraph (E, V), with the items as nodes V. We adopt here the notation in [8]. A hyperedge e differs from an edge in a regular graph in that it can connect more than two nodes. e is a set $\{v_1, \ldots, v_k\} \in V$ of the nodes. We are interested in directed Hypergraphs that have at most one destination node — that is, one produced item — denoted by h_e (head of edge). The set of source nodes is denoted by $\mathcal{T}_e$ (tail of edge). Then the hyperedge e is the pair $(\mathcal{T}_e, h_e)$. In our example, the bill of materials is hyperedge e, the produced item is $h_e = \{c\}$, and the components are $\mathcal{T}_e = \{a, b\}$. As the components used may not be exactly one unit each, we have the usage $p(v, e)$ indicating the amount. The number of time units to finish production is indicated by l_e. Costs associated with unit production of h_e are denoted by $c(e)$. Given these parameters, we now introduce a decision variable x_e for each production in a given time period to indicate how many units are being produced. To summarize,

(E, V)	Productions $e \in E$ are hyper-arcs between items $v \in V$
$\mathcal{T}_e$	Set of component items consumed by production e
h_e	Item being produced by e
$p(v, e)$	Amount of item $v \in \mathcal{T}_e \cup h_e$ required (≤ 0) or produced (≥ 0) in production e
l_e	Time units to complete production e (lag)
$c(e)$	Unit production cost (≤ 0) or revenue (≥ 0)
x_e	Decision variable of how many units to produce

What is interesting about our production construction is that while mainly intended to model BOMs, it actually suffices to model the whole supply chain. This implies that a solution is simply a flow on a directed hypergraph. When written as linear program equations, the corresponding production matrix is a Leontief substitution system, which has at most one positive entry in each column.

We now show what we mean by the statement that we only use productions. Let us start with an inventory type equation, which keeps track of the net inventory position of item v at a given time, which we shall call I_v:

$$I_v = \sum_{v \in h_e} x_e - \sum_{v \in \mathcal{T}_e} p(v, e) x_e.$$

Note that if we allow $p(v, e)$ to be negative or positive according to v corresponding to a component item or the item produced, we only need one summation term with $v \in h_e \cup \mathcal{T}_e$:

$$I_v = \sum_{e \in E(v)} p(v, e) x_e,$$

where $E(v) \subset E$ is the set of hyperarcs that contain v. We further assume that we have a distinct item v for each time period $1, \ldots, T$; that is, v only exists for that time period. In other words, $V = V_0 \times \{1, \ldots, \mathcal{T}\}$ for some initial timeless item set V_0, and similarly for E. We don't signal the time period with an index on v, rather we use $t(v)$ and $t(e)$ to retrieve the item's location in time, and the production's starting period, respectively. What is needed now clearly is linking the inventories I_v for consecutive time periods, as otherwise any positive inventory I_v simply expires. Thus we have

$$I_v = \sum_{e \in E(v)} p(v, e) x_e + I_{\mathrm{pr}(v)},$$

where $\mathrm{pr}(v)$ is the item instantiation during the previous time period, so that $t(v) = t(\mathrm{pr}(v)) + 1$. Similarly, let $\mathrm{nx}(v)$ be the item in the next period. Now it is obvious that I_v is simply another production e_a, with $\mathcal{T}_{e_\mathrm{a}} = \{v\}$, $h_{e_\mathrm{a}} = \{\mathrm{nx}(v)\}$, $p(v, e_\mathrm{a}) = -1$, $p(\mathrm{nx}(v), e_\mathrm{a}) = 1$, $t(e_\mathrm{a}) = t(v)$. x_{e_a} then indicates the inventory at time $t(v)$.

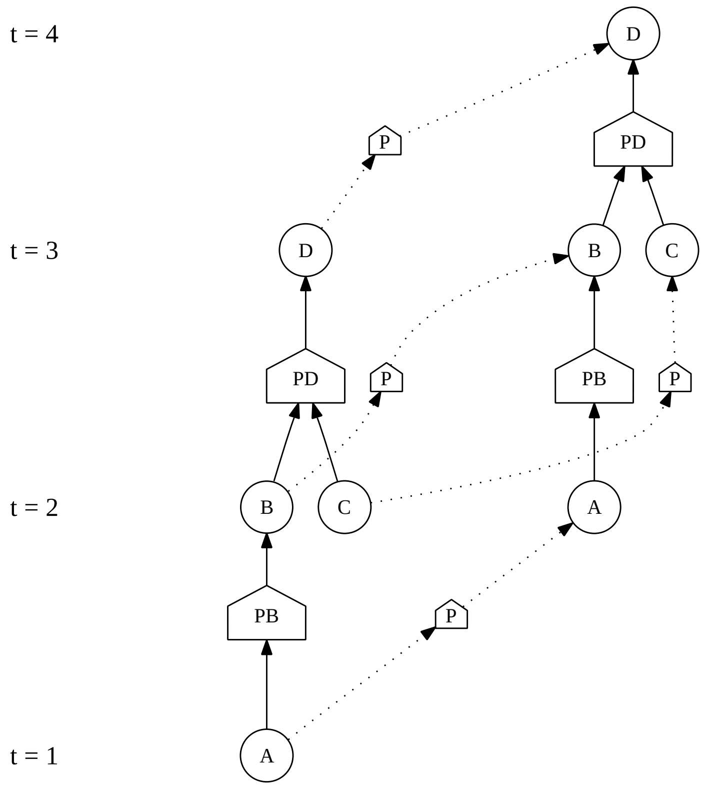

FIGURE 2.2 *Productions over time. The angled icons represent unit lag hyperedge productions, linking components and output items over time periods*

Our inventory (or flow conservation) equation now takes the simple final form

$$\sum_{e \in E(v)} p(v, e) x_e \geq 0 \tag{2.1}$$

using only productions. See Fig. 2.2 for an example bill of materials tracing each item for two time periods. For several reasons, we use greater than instead of equality in (2.1). This says that we can drop any inventory at any given time. While this would be very unlikely to occur in any optimal feasible solution, it is very useful for feasible, non-optimal solutions encountered during problem decomposition. It prevents subproblems from ever becoming infeasible, thus saving us from introducing relief inventory productions, or dealing with feasibility cuts. It is also natural in the case of resource productions, as a time resource not used is a missed opportunity, but there is no point in forcing its usage, and it cannot be recovered by carrying it over to the next period.

Each production variable can have an upper bound $\bar{u}(e)$, and is always non-negative. In general, this corresponds to capacity constraints on processes modeled by productions. In particular though, it also allows us to model demand, for which we appropriately introduce demand productions e_{d}, with $h_{e_{\mathrm{d}}} = \emptyset$, $\mathcal{T}_{e_{\mathrm{d}}} = \{v\}$, $p(v, e_{\mathrm{d}}) = -1$, and $\bar{u}(e_{\mathrm{d}}) = d$, where the capacity bound d indicates how much demand exists during the time period. Of course the incentive to satisfy demand, that is invoke the demand production decision variable $x_{e_{\mathrm{d}}}$, is missing. We address this by adding the profit objective,

$$\sum_{e \in E} c(e) x_e, \tag{2.2}$$

where $c(e)$ is the positive cost or negative revenue, so that (2.2) is to be minimized. For demand productions, $c(e_{\mathrm{d}})$ is always negative to indicate revenue opportunity.

2.2 Expressiveness of the Production Model

Now we are ready to show that the remaining essential constructs of a supply chain can also be modeled by productions. Standardized definitions can be found in the APICS dictionary [10].

Safety Stock Safety stock is just the inventory production e_{a}. The production amount indicates how much surplus was carried over to the next period. (Safety stock trades off lead time. If we always keep enough safety stock at a location to satisfy actual local demand, the effective lead time is zero).

External Supply Raw material or starting components have to be sourced from somewhere. These productions e_{s} don't use any other items, that is, $\mathcal{T}_{e_{\mathrm{s}}} = \emptyset$.

Resource In addition to using common items, productions can use common resources such as work hours available, or a shared machine. Like external supply, the resource production has $\mathcal{T}_{e_{\mathrm{r}}} = \emptyset$. In addition, there is no inventory: we drop the carry over productions e_{a} corresponding to the resource 'item'. Either the hours of work available are used, or they expire. Of course every resource has a limiting upper bound capacity $\bar{u}(e_{\mathrm{r}})$. To indicate use of a common resource e_{r}, any production e can add the resource item $h_{e_{\mathrm{r}}}$ to its components $\mathcal{T}_e$ with appropriate usage $p(h_{e_{\mathrm{r}}}, e,)$.

Warehouse Capacity and Cost If shared, limited storage availability can be modeled as a resource used by the inventory productions e_{a}. If limited space is allocated for an item individually, we can simply set the capacity of the corresponding inventory production. The cost is charged to the respective resource or inventory production.

Warehousing costs are also incurred for items in the production pipeline, proportional to the amount being produced, in which case we add the cost to the production itself.

Optionally, some other concepts can be implemented using productions. Some only become meaningful in the stochastic setting, in particular when timing of decisions is concerned. The cost typically is an increase of part of the problem size by a constant multiple.

Shelf Life Some types of item, for example produce or consumer electronics, expire after a while and become more or less worthless. We assume that after a certain time period since production, the item has no more use and its inventory is scrapped. We model this by introducing a version of the item indicating its current age in $1, \ldots, exp$; that is, we go from v to $\{v_1, \ldots, v_{exp}\}$. Instead of the usual inventory productions $e_{\mathrm{a}v}$ having the same item in the next period as their output, they point to the adjacent item at age $+1$. v_{exp} does not have any inventory carried over, and its stock will have to be used up or expire. In addition, we have to replicate all productions feeding off v in a similar fashion so that they can use any aged version of the item.

Lag It is possible to model lags > 1 using only unit lags. While it may not be clear why we would want to do that right now, we shall indicate the procedure here. Assume the production e takes l_e periods to complete. Then we proceed similarly to the case for shelf life, by introducing aged versions of the output item h_e. Only the last version v_{exp} with $\mathrm{exp} = l_e$ is used by any productions feeding off v.

Backorder We have so far assumed that demand has to be satisfied during the time period in which it materializes. This corresponds to a service time of zero, which we would see for example at a consumer outlet, where the customer expects the item to be stocked and available. Similarly, it could model a situation where an order is taken and expected to be shipped during the next business day. If however we allow for a service time to fulfill the order (not counting shipping), we can model that as a backorder. That is, we allow demand to live on for a certain number of periods, until it expires if not satisfied. To do this, we introduce an external supply that injects a shelf life token item, up to the amount of the demand during that period. This item is then passed on — as far as it hasn't been used by appropriate demand productions — up to the expiration period.

Alternatively, we can approximate a positive service time slack s modeled as *early demand*, see below, without tokens. We use a stage-dependent additive distribution where the demand has a regressive lag equal to the demand slack, and demand times are moved out by s periods.

Early Demand In some situations, precise demand may be assumed known in advance of the actual time of commitment to fulfill the order. This is only a slight variation of the backorder construction. The only difference is that demands may feed off the last token item only, instead of all of them. But we don't need to implement early demand that way, as it is more natural to represent it directly in a VAR demand model, see section 5.5. (Conversely, any positive VAR distribution can be implemented using token flows to carry random outcomes to future stages.)

On the other hand, if early demand means early commitment, then the proper thing to do is to allocate all materials needed at the time of the order. This can be achieved as follows. Let $\mathcal{L}$ be the time allowed to fulfill the order. From the demand item as root, we collect all upstream production trees that have a max total lag of no more than $\mathcal{L}$. Then we roll up the bills of materials so that for each tree we end with a cumulative bill of materials that has no output and as inputs all the leaf components in the tree. Depending on the supply chain and $\mathcal{L}$, there may be too many such trees, in which case we could instead use the uncommitted early demand approach as an approximation.

Another useful case for early demand is when a manufacturer or distributor shares demand data with an upstream supplier. The latter will know in advance what amounts are needed. Conversely, if that information is not shared, the supplier will have to create a demand forecast based on past history and in general carry greater safety stocks, or quote a longer lead time. Withholding demand information from upstream suppliers can result in greater variation of demand, the so-called *Bullwhip effect* [30].

Diminishing Marginal Revenue Diminishing revenue can be modeled by linearizing a concave revenue curve and splitting it into linear parts, each of which becomes a different demand production for the same item, each subsequent one with lower revenue. The same can be used for convex costs.

Supplier Failure and Random Yield AMR Research has reported that businesses indicate supplier failure as their number one risk factor [23]. Supplier failure can be modeled as random capacities for supply productions. While the risk of an individual supplier failing may be small, the overall probability of at least one supplier failing will be much larger in a supply chain with many external components. Some of the risk may lie with supplier stockouts, but more dangerous are orders with long lag time that fail to arrive.

To model disruption, we introduce a dummy item for each period until delivery, and put zero capacity as a possible outcome on some of the carry productions. To model recovery of supplier payments, we add an artificial, parallel demand production at cost, whose capacity is different from zero only if disruption was the outcome.

The cost of this manoeuver is that outcomes with large jumps (such as zero vs non-zero capacities) generally are so different that Benders cuts (as described in chapter 5) derived on one path may not provide much support for other paths with different outcomes, so that more cuts are needed for convergence.

Random yield can be modeled the same way. Alternatively, it can be implemented directly by making the output coefficient of the production matrix random. If serial random correlation is required (we only support random rhs for serial correlation), we can model it indirectly through capacities, as follows. For each outcome, we replicate the production, and set the yield according to the outcome. Randomness is then moved to the production capacities, with only one production having non-zero capacity at each outcome.

Random Variable Lead Time Globalized procurement is one source for greater variability of lead times. The latter can be modeled by sending the supply into a dummy holding item each period, with connecting carry production, up to a maximum lag period. For each holding item, there is a delivery production that moves the dummy to the real item. Its capacity is determined according to the lead time distribution (in parallel, the dummy carry production may be penalized by at least the carry cost so that the supply is pushed into the real item).

We can model any lag distribution by independent binary random capacities on each dummy delivery production.

Cost Risk Cost increases cut into profitability. We model the risk within our existing framework by using several supply productions with different costs. A high cost scenario is characterized by limited availability of the lower cost productions.

Sequential dependency of the capacity distributions makes sense here (section 5.5), otherwise a high cost will most likely revert back by the independence of the stage random variables.

Tax uncertainty from country to country in a global network also translates into cost risk and can be modeled as such.

Overtime We copy the production or resource that is going into overtime and add overtime costs to its objective. Then the overtime production will be selected only when the regular production has been exhausted.

Product Substitution Clearly our model allows for substituting the same item from different sources or productions. Similarly, different items can be modeled this way if they effectively can satisfy the same demand. When items are a little bit more differentiated, we would like to have separate demands, but correlated in some way. For example, an item may be a substitute for another, if the price is brought down sufficiently to make the consumer consider the lower priced item. Similarly, a customer may choose to upgrade to a more expensive item if the price is brought down sufficiently. For effect of pricing on substitutability, see for example [32]. In either case, we achieve demand pooling, thus preventing stockouts in one item to result in a lost sale. This can be easily implemented as follows: in case of the expensive item, let it flow to its original price point demand, as well as to a separate demand where it is the substitution item. The latter will have lower revenue, so the higher cost item will only satisfy that demand after there is no more of its own proper demand, and after the low cost item supply has been exhausted. Similarly for substituting with a lesser item.

The opposite case is when we have items that are not substitutes, but demand is inversely correlated, in that one of them will be much more popular than the other, no matter what pricing points are chosen. Instead of risk pooling, we increase risk by differentiation. Such a case can be modeled with an appropriate, autocorrelated demand distribution. We would expect that as a profit maximizing measure, the model would decide to build sufficient inventory at intermediate assemblies before the items are differentiated.

Expediting To satisfy demand expediently, sometimes it pays to expedite movement of material and products. This can be modeled as alternate transportation routes with smaller lead times and higher cost.

Lastly we describe some concepts that cannot be modeled effectively in the production model and haven't already been mentioned.

Satisfaction Level One customer satisfaction or service metric is to guarantee a minimum percentage of orders filled. While this can be as simple as an upper bound on satisfying demand in a period, even in this case we are changing the model in that subproblems in Benders decomposition (chapter 5) can now be infeasible, requiring infeasibility cuts and introducing other difficulties.

When the percentage is to be over multiple time units, more complications arise, which in any case invalidate our simple productions model.

The good news is that satisfaction levels have been mainly in use to drive stocking levels of a single-echelon system, which makes sense in such a simple model. But in a multi-echelon, multi-item model such as ours, where we can finally begin to talk about expected profit, maximizing the latter is a lot more useful. If we are concerned about customer goodwill, we can simply assign a cost of unserved demand to each order, and add it to revenue. Then the expected lost goodwill can be broken out from the solution and reported. Desired higher order fill rates for distinct item categories (A,B,C [10]) can also be reflected in higher lost order values for the higher priority items.

This seems to be a better way to account for customer satisfaction than strict percentages, which may or may not make economical sense (and certainly are not profit driven).

The other case where satisfaction levels are useful is when cost data is very inaccurate. This is quite a common occurrence in ERP databases, and the culprit usually is that the cost data has not been used in any critical function, e.g. to drive profit. Of course the remedy is to improve accuracy of the costing data.

'Rolling' Inventory A company may decide to store products in truck trailers at the back of the warehouse instead of at the warehouse itself [47].

As this is trying to maximize individual trailer usage, it is better to model this as a MIP and thus falls outside the scope of our model.

2.3 Demand Distributions

The question of what kind of demand distribution (expressed as the values $\bar{u}(e_\mathrm{d})$) we should have needs to be addressed. Demand should be non-negative. Negative demand can be interpreted as customer returns, although realistically returns are a small fraction of demand and would be very unlikely to outnumber total sales during a given period. Models composed of normal random variables can always be negative, albeit the probability can be kept arbitrarily small when the mean is sufficiently large and the variance small. When errors are not normal, more safety stock is usually needed [54]. A variety of standard non-negative distributions lends itself as building blocks to time series construction. We shall not elaborate on conditions for stationarity, which is extensively treated in the literature. A stationary model is easier to estimate. However, even a trend or seasonal adjustment does not hide the fact that variance remains constant over time. We believe it is more natural for the demand variance to grow with time, rather than to assume a fixed underlying trend. The toolbox of standard continuous non-negative distributions includes the exponential, truncated normal, conditional normal [54], lognormal, Weibull, and Gamma distribution. For the positive integers, there are the Poisson and binomial distributions. Simple autoregressive models with non-normal innovation have been analyzed in [20].

Correlation between time series can be captured in a (lagged) covariance matrix. For a vector normal distribution, this completely determines all parameters. By conditioning on positive vector outcomes (i.e. discarding sample vectors with negative coefficients), we obtain a model that may be easy to fit and allows for negative autocorrelation. The probability of a negative outcome should be small however, otherwise conditioning may be hard to achieve. Negative correlation can also be easily realized without allowing the vector to go negative by correlating the positive vector of innovation in a multiplicative or additive vector autoregressive model (fitting the model may have its own challenges, but we ignore that problem here). However we do not see the need to model negative correlation: the main case that comes to mind is cannibalization, which seems to be avoidable when the supply chain in question is for products of the same company. Instead, we would expect the products to be substitutable, in which case a common demand distribution can be found for the product family of substitutes. Similarly, introducing a new product replacement does not need to be left to random chance, a strategy might be to force production capacities for the replaced item to go to zero quickly.

We do not address the question of how to estimate or select among the various models given historical samples and other information.

2.3.1 Unknown Trend Models

For illustrative purposes, we chose a model that exhibits positive autocorrelation but no correlation between demand time series, and has growing variance. (The model can be easily extended to dependencies between time series, for example by sharing some of the underlying variables or series.) one instance of our model exhibits a linearly growing standard deviation using a triangle distribution construction (with some noise thrown in if desired to add another source of variability). We believe this to be a very good approximation of actual uncertainty, as when demand peaks is the most critical unknown given a certain growth trend. We acknowledge this very important source of uncertainty by using a distribution that actually reflects it. To hedge, we have to know what we are hedging against.

In practice the peak will announce itself by flatlining growth close to it, but triangle trends with common growth rates are a useful approximation.

Note. If each trend outcome had a distinct incline rate, then the full triangle could be identified as soon as the gradient became visible during the first few periods. Hence we assume here that all possible trend outcomes have the same incline, so that the curve is identified only after its peak. This would not be very useful for traditional forecasting applications, as we cannot make any reliable forecasts until the peak occurs, but it is very useful indeed to let the stochastic program hedge against all such trend outcomes.

Demand for a given item at stage t is defined to be

$$d(t) = \sum_{j=1}^{t} \varepsilon_{jt} \prod_{k=1}^{t} \mu_k, \tag{2.3}$$

where ε_{jt} and μ_k are independent non-negative random variables, for example truncated normals, and $t = 1, \ldots, T$.

If we let μ_t be binomial random variables, and let ε_{jt} be identical constants, we obtain the 'triangle' distribution. It embodies the unknown peak property, as the first failure outcome $\mu_t = 0$ signals that the peak occurred at $t-1$. Of course ε_{jt} can be any function of j and t, so that the individual peak demand curves can take any shape.

As a second more practical instance, let μ_t be some random variable with

$$\mathrm{E}(\mu_t) > 1, \qquad \mathrm{Var}(\mu_t) < 1. \tag{2.4}$$

We can interpret $\varepsilon_{jt},\ t = j, \ldots, T$ as early demand information that becomes available in period j. Future demand at t is positively correlated with early demand information via $\mu_k,\ k = 1, \ldots, t$. This implements the notion that the amount of early demand is a good predictor of future demand.

2.3.2 Forecast Update Models

Stefanescu et al. [53] look at the problem of estimating a correlated demand model as data becomes available. They assume that a common initial 'shock' vector determines all the individual demand trends. As more data becomes available, the parameter estimation is refined. We shall now explain why this and similar models are not going to be very useful here, even when they are a good fit. The reason is that forecasts are updated as more information becomes available, and production adjusted accordingly, instead of hedging against possible future outcomes. A Bayesian approach seems conceivable, where the conditional distribution model is represented by the posterior. However this is circular, as we would have to simulate a partial outcome from the true distribution to obtain the posterior. Instead of updating a forecast or parameters for a fixed-trend model, the partial demand outcomes should be used only to condition the distribution. If we knew the exact parameters, then the stochastic program would know the trend functions and be able to optimize for the remaining random noise. Since we don't know the exact trends in advance, the trend curves can take various forms that become clear only later on, so it is a mistake to think that the stationary random noise remaining will somehow obscure the trends and lead the stochastic program to hedge against different trend curves: the conditional future distribution is fully known to the stochastic program for any simulated outcome. Any trend that becomes deterministic after some simulated point in time is fully visible for all future periods, and no hedging occurs, except for the remaining yet to be realized noise.

The appropriate analogy for our model of the approach used in [53] is

1. Solve the stochastic program.
2. Implement the first stage decisions.
3. Observe the next stage demand outcome and any other visible variables.
4. Update the demand distribution given the new information.
5. Roll the time horizon window forward one period.

Chapter 3

Leontief Systems and Extensions for Capacities and Profits in the Deterministic Case

Looking at the linear program made up of the minimizing objective term (2.2) and the item constraints (2.1), we see that the matrix $P := \big(p\,(v, e)\big)_{|V|\times|E|}$ is *pre-Leontief* [56]: any production e is represented by a column in P that has at most one positive entry for the output item. All other items v in e are consumed and have negative $p(v, e)$ coefficients. As we allow more than one production to output the same item (that is, we allow multiple sources), the corresponding linear program is a *pre-Leontief substitution system* (LSS). Leontief systems, not surprisingly, have their origins in production planning and economics. For some models that are very similar to ours, see for example [11, 57]. Koehler et al. suggest solving a general LSS with matrix iterative techniques in [29]. Dantzig [11] shows how to solve efficiently the special block triangular structure resulting from a time-phased LSS with the simplex algorithm.

When constructing (V, E) from time-phasing items and productions using lag l_e, the matrix P takes the form

$$\begin{pmatrix} A & & & & & \\ B_1 & A & & & & \\ \vdots & B_1 & \ddots & & & \\ B_L & \vdots & & & & \\ & B_L & & & & \\ & & \ddots & & \ddots & \\ & & & B_L & \dots & B_1 & A \end{pmatrix}, \tag{3.1}$$

where $L = \max_e l_e$, $P_{i\cdot}$ denotes items at time i, and $P_{\cdot j}$ productions starting at time

j. The band block structure is due to ordering the item rows by time $t(v)$, and the production columns by the production starting period $t(e)$. Alternatively we could have ordered the productions by finishing period, that is $t(e) + l_e$, but we need to group variables by time of decision (starting period) for the stochastic version later.

We don't allow cycles in the supply chain. An item may not be fed to one of its component productions. Such occurrences would mainly be seen when reverse supply chains for returns and disassembly are integrated into the main supply chain. While reverse supply chains are certainly important, we don't see a need to integrate them into the global supply chain, as the reverse flows are only a small fraction of the total. It seems sufficient to treat the reverse supply chain independently as a separate problem. We thus consider only acyclic LSS (ALSS). Based on [11], Veinott shows in [57] how ALSS can be solved in a single pass through the (ordered) data.

None of these approaches applies when there are upper bound capacity constraints on the productions (as the problem is not a LSS anymore, except in special cases, see [57]). Cambini et al. [8] propose a network simplex-inspired method for solving flows on capacitated directed hypergraphs, or equivalently, *capacitated Leontief substitution systems* (CLSS). There is a correspondence between spanning hypertrees and bases, but unlike the network simplex method, updating the basis tree potentially involves a quadratic effort. Note that any linear program with upper bounds on all its variables can be translated into a CLSS [8].

The problem that we are interested in solving is a capacitated, acyclic Leontief substitution system (CALSS). This is the linear program

$$\begin{aligned} \min \quad & cx \\ Px &\geq 0 \\ x &\leq \bar{u}, \end{aligned} \tag{3.2}$$

where $\bar{u}$ and c are the production capacity and cost vector, and P is as in (3.1).

3.1 Solving the Capacitated Acyclic Leontief Substitution System

Since the problem matrix P grows linearly with the number of periods, it can become quite large. As deterministic subproblems can be used to solve a stochastic version of (3.2), it is important that the many inner iterations involving (3.2) be solved quickly. In the following we take advantage of the structure of CALSS to develop an approximation algorithm. A hypertree is a subgraph of (V, E) without undirected cycles.

Theorem 3.1. *Every optimal solution to CALSS (3.2) is a sum of at most $|E|$ flows on hypertrees.*

Proof. A feasible solution can have surplus at any given item and time. But since the only productions with negative cost are the terminal demand productions, any feasible solution with discarded surplus can be transformed into a solution without surplus and equal or lower total cost.

Assume optimal solutions x^* don't have surplus. Then $Px = \bar{u}$, which acts as flow conservation constraints, so that x^* is a flow on the hypergraph (V, E). All flow terminates at the demands. Since the hypergraph is acyclic, we can start at any demand production variable e_{d} and trace the flow upstream. At each upstream production, we determine how much has to flow on it to satisfy the downstream flow value. If there are multiple upstream productions providing flow for a component item, then we arbitrarily pick one source e. We potentially reduce the flow if x^*_e is not large enough. In the end, we have traced out a hypertree with the demand production e_{d} as root, and a corresponding flow that feeds a portion or all of the demand production needs. Since the flow conservation constraints have to hold, we can subtract the tree flow from x^* and still have a feasible flow left. Each time we reduce at least one edge of x^* to zero. As there are only a finite number of productions, eventually we will have partitioned x^* into flows on hypertrees with demand productions as roots. □

We now reformulate (3.2) in terms of hypertrees. Assign each possible hypertree δ with some demand root a flow variable $f(\delta)$. The size of f indicates how many units of demand are being satisfied by the tree. The tree is represented by the vector $p(., \delta)$, indicating how many units of capacity the tree needs for each of its productions, to satisfy one unit of demand. For example, if a unit of the demand item can be made by a production using 2 units of capacity, and that production in turn has a component that can be made by another production e with 3 units of capacity, then the usage $p(e, \delta)$ of that production is 6. $d_\delta \in E$ is the tree's demand production, $c(d_\delta)$ is the revenue of the tree's demand production, $c(\delta)$ denotes the cumulative production cost on the tree to satisfy one unit of demand. The resulting problem is

$$\max_f \quad \sum_\delta (c(d_\delta) - c(\delta))\, f(\delta) \qquad \text{s.t.} \quad \sum_{e \in \delta} p(e, \delta) f(\delta) \le \bar{u}(e), \qquad e \in E, \qquad f \ge 0. \tag{3.3}$$

There can be an exponential number of possible trees, and thus solving (3.3) directly may be intractable. But there are only linearly many trees in the solution. A column generation type approach is needed. We need to be able to find a tree quickly given some 'reduced-cost' type criterion.

What is interesting about (3.3) is that $p(e, \delta)$ and $\bar{u}(e)$ are non-negative, and $f \geq 0$. We are thus dealing with all positive constraints, that is with a *positive linear program* (PLP), with only *fractional packing* inequalities ($\leq$). Fast approximation algorithms exist for this problem, for example [61]. A variety of special cases are treated in [18, 40]. PLP also goes by the name of *(mixed) fractional packing and covering* in the Computer Science literature. Similar algorithmic approaches can also be applied when the positive constraints are nonlinear convex and concave functions [27].

What both algorithms by Young [61] and Koenemann [18] have in common is they assign dual weights to the rows and require finding the minimum weight [18] or negative weight [61] rows in an iteration, and maintain weights to remain exponential in the constraint violations.

Koenemann considers a PLP of the form

$$\max_x cx \qquad \text{subject to} \quad Ax \leq b, \quad x \geq 0, \tag{3.4}$$

and defines column j's cost as

$$c(j) := \pi' A_{.j} / c_j \tag{3.5}$$

for some π. We cannot apply (3.3) directly here, as $c(j)$ depends on which tree has been selected for a given demand, and we don't know how to find efficiently a smallest cost tree when the cost involves a divisor that is a function of the tree. Instead, we use a *budget constraint*, as in [40]. The budget constraint

$$\sum_\delta c(\delta) f(\delta) \leq \beta \tag{3.6}$$

separates the cost from the revenue in the profit objective of (3.3), resulting in problem

$$g(\beta) := \max_f \sum_\delta c(d_\delta) f(\delta) \qquad \text{subject to} \quad \begin{aligned} &\sum_{e\in\delta} p(e,\delta) f(\delta) \leq \bar{u}(e), \qquad e \in E, \\ &\sum_\delta c(\delta) f(\delta) \leq \beta, \\ &f \geq 0. \end{aligned} \tag{3.7}$$

The revenue of a tree for a given demand does not depend on the selected tree, and as shown below, we can efficiently find the smallest cost tree. The budget constraint (3.6) sums up the total production cost of all trees, and requires it to be less than

some constant β. If β is too large, it means we are ignoring production costs and only maximize total revenue. If β is set too small, potential revenue is held back. For a given β, total profit is at most total revenue minus β (If the budget is not binding at an optimal solution, it implies that the trees maximizing total revenue are all profitable. Furthermore, the trees maximizing flow also use the cheaper production sources. While this is highly unlikely by chance, it certainly holds when the supply chain problem is posed pre-sourced, that is each item is only made by one production.)[1]
A bisection line search can be applied to maximize

$$g(\beta) - \beta, \tag{3.8}$$

which is a concave function of β.[2]

3.1.1 Minimum Cost Trees

For simplicity assume that $p(h_e, e) = 1$ for all productions. Designate by $w(v)$ the minimum cost of any tree producing item v:

$$w(v_0) = \min_{h_e = v_0} \; \pi_e + c(e)\pi_{\mathrm{b}} + \sum_{v \in \mathcal{T}_e} p(v, e) w(v), \tag{3.9}$$

where π_e and π_{b} are the dual weight assigned to productions and the budget constraint. $w(v)$ can be computed recursively in any order implementing the partial order induced by the acyclic productions. As the total column cost according to Koenemann is defined to be the inner product of the column and the dual weights, divided by the objective coefficient, we can get the final tree column cost by adding π_{d_δ} to $w(\mathcal{T}_{d_\delta})$, and then dividing by $c(d_\delta)$. Then among all demands we simply select the one with the smallest tree cost. We traverse the selected tree a second time to obtain the total usages $p(e, \delta)$ for each production that the tree uses, which constitute the column coefficients of the tree. We also don't forget the tree's demand production, which has a capacity, and a usage of 1 as column coefficient. The effort to determine the minimum cost trees given dual weights π is $O(|E|)$.

Young's algorithm [61] finds a feasible point, if it exists, of mixed PLP equations

$$Kx \le p, \quad Cx \ge c, \quad x \ge 0. \tag{3.10}$$

[1]This does not result in more efficient selection of minimum cost trees, as for different duals traversing the tree for each demand will cover all productions, otherwise the production would not be included. In the time-phased case, it does not apply at all as we always have the 'sourcing' option of taking from inventory vs. from the item production.

[2]Problem (3.7) is solved for different values of β a logarithmic number of times

To bring (3.3) into conformity here, we move the profit objective into the constraints, requiring a minimum profit of p:

$$\sum_{\delta} \left(c(d_\delta) - c(\delta)\right) f(\delta) \geq p. \tag{3.11}$$

Note that (3.11) has all positive coefficients. This is because if a tree has negative profit (costs exceeding revenue), we do not need to include it at all, as it will not take part in an optimal solution. The reason we can include profit directly here (in contrast to Koenemann's algorithm) is that there is no objective divisor, and the cost of a column j is defined as

$$c(j) := \pi'_{\mathrm{K}} K_{.j} - \pi'_{\mathrm{C}} C_{.j}. \tag{3.12}$$

Of course we still have to do bisection to find the maximum profit p. For given p, the problem then is finding a feasible flow f to

$$\begin{aligned} \sum_{\delta} \left(c(d_\delta) - c(\delta)\right) f(\delta) &\geq p, \\ \sum_{e \in \delta} p(e,\delta) f(\delta) &\leq \bar{u}(e), \quad e \in E, \\ f &\geq 0. \end{aligned} \tag{3.13}$$

The minimum cost trees can be found by using the same recursion function as before (3.9). The only difference is that instead of dividing by revenue, we subtract

$$\pi_{\mathrm{p}} c(d_\delta) \tag{3.14}$$

from the tree cost, where π_{p} is the dual weight for the profit constraint (3.11). The mixed PLP algorithm [61] proceeds in phases during which flow increments are added to all trees with negative cost. As before, to find these trees, essentially a pass through the production matrix is sufficient, requiring $O(|E|)$ operations.

3.1.2 Running Times

First we state a result about the mixed PLP feasible point problem.

Corollary 3.2. *The time to obtain an approximately feasible solution to* (3.13) *is*

$$O(\varepsilon^{-2} F \, |E| \, \log |E|),$$

where F designates the number of potential bottleneck capacities (in a time-phased formulation, both F and $|E|$ grow with the number of periods).

Proof. According to Young [61, Corollary 1], to solve (3.10) takes $m \log m$ iterations, each of which requires finding a negative cost tree, which we have shown to be $O(|E|)$. m is the number of constraints, which in our case corresponds to the number of productions that can become bottlenecks in some demand tree, in the sense described in the proof of Theorem 3.1. □

Whether a production capacity can be a bottleneck can be estimated roughly, for example by looking at the highest profit demand trees and removing any capacities that are more than three times the minimum capacity productions in those trees, with respect to one unit of demand. Naturally, if a production has no capacity restriction, it does not qualify as a bottleneck.

While the worst case is $O(\varepsilon^{-2}|E|^2 \log|E|)$, we can usually cut corners calculating the least cost trees (3.9). If the undirected hypergraph does not contain any cycles involving productions on the previous least cost tree, we need to update only nodes on that tree. Furthermore, the length of an efficient tree will not span all stages, as carry costs will disqualify trees that are too long. Lastly, as there are many demands (usually at least one per stage), many of them will have least cost trees that are negative, so that more than one increment can be made during an iteration.

Note that the faster 'phased' approach in [61] can not be applied here: the proof of [61, Corollary 6] relies on having all negative cost trees available during a phase, while (3.9) only yields the least cost tree for each demand.

Note. One of the main characteristics of the PLP approach is that the sparsity of the original problem is maintained. We mention some other approaches that also have this property: a simple cyclic projection algorithm with a twist (going under the name of *row-action methods*) by Dykstra[3], for linear programs with a quadratic objective [15, 6]. For an interesting overview, see Dax [13]. While convergence may be much slower (no explicit guarantees) than for PLP, the quadratic objective enables one to consider augmented Lagrangian alternatives to the bundle method approach. Similarly, interior methods for LP allow a quadratic objective, and inner iterations to solve the normal equations can employ a conjugate gradient method, which also maintain sparsity.

The deterministic problem (3.3) can be used in the aggregate scenario decomposition method in chapter 4.

---

[3]This Dykstra is from Statistics, not the same Dijkstra of Computer Science fame, and the algorithm is not related to Dijkstra's algorithm for shortest paths.

Chapter 4

Stochastic Supply Chains using Positive Programming and Scenario Aggregation

So far we have only considered a deterministic view of supply chains and solution approaches. We have done this with the requirement in mind that the ideas introduced will be of use in a stochastic formulation. We now introduce demand uncertainties into the model.

$$\min\ E\big(cx^{\omega}\big) \qquad \text{subject to}$$

$$\begin{pmatrix} A &&&&&& \\ B_1 & A &&&&& \\ \vdots & B_1 & \ddots &&&& \\ B_L & \vdots &&&&& \\ & B_L &&&&& \\ && \ddots &&& \ddots & \\ &&& B_L & \dots & B_1 & A \end{pmatrix} \begin{pmatrix} x_1^{\omega} \\ x_2^{\omega} \\ \vdots \\ \vdots \\ \\ \vdots \\ x_T^{\omega} \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ \\ \vdots \\ 0 \end{pmatrix}, \text{ for all } \omega \in \Omega \tag{4.1a}$$

$$x^{\omega} \leq \bar{u}^{\omega} \qquad \text{(random demand)} \tag{4.1b}$$

$$x_i^{\omega} = E\big(x_i^{\omega} \mid x_1^{\omega}, \dots, x_{i-1}^{\omega}\big) \qquad \text{(non-anticipativity)} \tag{4.1c}$$

Since demand is implemented as demand productions with capacities indicating the amount of demand, we only need random coefficients in part of the right-hand side (4.1b) of the problem. Decisions x^{ω} are now dependent on the random outcome $\omega \in \Omega$. If we don't require (4.1c), we have an independent solution x^{ω} for each outcome ω. Then the expectation in (4.1a) is the *perfect information* or *'wait and see'* solution. Instead, the non-anticipativity constraint (4.1c) forces x^ω to be a *policy*, also referred to as the *'here and now'* solution.

Definition 4.1 (Policy)**.** *A policy prescribes production decisions x^ω such that if the outcomes of $\omega^{(i)}$ and $\omega^{(j)}$ are the same, $\omega_t^{(i)} = \omega_t^{(j)}$, on the initial path $t = 1, \ldots, K \leq T$, then $x_t^{\omega^{(i)}} = x_t^{\omega^{(j)}}$ along that path.*

Given all current and past information, a policy tells us what decisions to make next. At a given time t, we have available

$$\{\bar{u}_1^\omega, \ldots, \bar{u}_t^\omega\}\,, \tag{4.2}$$

and need to decide x_t^ω. The non-anticipativity constraints make sure that for the same information, that is, any ω sharing the same outcome $\{\bar{u}_1^\omega, \ldots, \bar{u}_t^\omega\}$ during the first t periods , x_t^ω is the same (x^ω is a *filtration* in probability speak, e.g. [14]).

The stochastic linear program (4.1a)–(4.1c) can be very large. If the discrete random probability space $(\Omega, \mathcal{F}, P)$ is made up of underlying independent random variables in each period, say with d distinct combined outcomes, then the total number of possible outcomes is d^T for T periods. Even for a moderate number of periods, it will be impossible to compute an optimal solution directly with a linear program solver.

We now step back to review some non-smooth optimization and problem decomposition methods.

4.1 Decomposition

It often happens that a large problem has special structure, in that there are sets of variables that more or less only appear in constraints together, and have very little connection to other variables in other sets. If it wasn't for the pesky connecting constraints between sets, the problem could be split into the smaller sets and each subproblem solved independently (and much faster than the larger problem of all variables). This is the idea behind decomposition. The connecting constraints are used to communicate information between the smaller subproblems. Let us begin with a problem of the form

$$\max_y \; g(y) \qquad \text{subject to} \quad Ay \leq a, \quad By = b, \tag{4.3}$$

where g is concave and separable, $B = [B_1 B_2 \ldots]$ contains connecting constraints, and $A = \operatorname{diag}(A_j)$. That is, A is ordered into blocks of mutually independent constraints.

Taking the Lagrangian dual on the connecting constraints in B results in subproblems

$$\min_x \max_y \; g(y) + \langle x, By - b \rangle \qquad \text{subject to} \quad Ay \le a, \tag{4.4}$$

where x is the vector of Lagrange multipliers, see for example Rockafellar [42]. Let

$$f_j(x) := \max_y \; g_j(y_j) + \langle x, B_j y_j - b_j \rangle \qquad \text{subject to} \quad A_j y_j \le a_j \tag{4.5}$$

be the subproblems and $f(x) = (f_j(x))$. Designating the set of all subgradients of f at x by $\delta f(x)$, called the *subdifferential*, we also have

$$B\hat{y} - b \in \delta f(x), \tag{4.6}$$

where $\hat{y}$ is an optimal solution in the definition of $f(x)$. This directly follows from the definition of $\delta f(x)$:

$$f' \in \delta f(x) \iff f(z) \ge f(x) + \left\langle f', z - x \right\rangle, \quad \text{for all} \quad z, \tag{4.7}$$

and by plugging in f and substituting $B\hat{y} - b$ for f':

$$\begin{aligned} \max_{y:Ay \le a} \; g(y) + \langle z, By - b \rangle &\ge \max_{y:Ay \le a} \; g(y) + \langle x, By - b \rangle + \langle B\hat{y} - b, z - x \rangle \\ &= g(\hat{y}) + \langle x, B\hat{y} - b \rangle + \langle B\hat{y} - b, z - x \rangle \\ &= g(\hat{y}) + \langle z, B\hat{y} - b \rangle . \end{aligned} \tag{4.8}$$

We have in effect transformed the concave maximization problem (4.3) into a convex minimization problem $\min_x f(x)$ in the dual Lagrangian multipliers, by bringing the coupling constraints into the objective, where they become a separable sum of linear terms. Evaluating (4.4) at x means solving the (simpler) inner subproblems, which in addition gives us a subgradient at x (under some constraint qualification criteria, see e.g. [42]). It can be shown that perturbing the objective of a maximization LP is a piecewise convex function of the objective perturbation parameter [3]. In other words, $f(x)$ is non-smooth and hence non-differentiable, and δf usually is a set of subgradients instead of just a single gradient as in the smooth, differentiable case. This takes us into the realm of non-smooth optimization. Note that our stochastic program (4.1a)–(4.1c) can be brought into the form (4.3), where the coupling constraints are the non-anticipativity constraints and the production PLPs for each outcome are the decomposable subproblems. In chapter 5 the same decomposition approach is used recursively, where the connecting constraints are the blocks in the production matrix to the left and right of the current stage. The non-anticipativity constraints in that formulation are modeled implicitly by mapping them onto the decision variables.

4.2 Non-Smooth Optimization Using a Collection of Subgradients

Assume that we can evaluate $f(x)$ and obtain a subgradient $f' \in \delta f(x)$, for any given x of interest. f is convex and non-smooth. No more information about f can be obtained. Looking at the subgradient definition (4.7), we see that each function value and subgradient at x form a lower linear approximation of f, which is exact at x. This gives rise to the piece-wise linear approximation

$$\bar{f}_k(x) = \max_{1 \le i \le k} \{ f(x_i) + \langle f_i', x - x_i \rangle \} \tag{4.9}$$

with $\bar{f} \le f$.

4.2.1 Cutting Plane Method

The cutting plane method [28] obtains the next iterate x_{k+1} by minimizing the current outer linearization of $\bar{f}$:

$$x_{k+1} := \operatorname{argmin} \bar{f}_k(x), \qquad \bar{f}_k(x) = \max_{1 \le i \le k} \{ f(x_i) + \langle f_i', x - x_i \rangle \}. \tag{4.10}$$

Slow convergence is the main problem with (4.10), but if the evaluations of f are relatively cheap, that may not be so important. Benders decomposition (Chapter 5) is based on the cutting plane method.

4.2.2 Bundle Methods

The main issue with the cutting plane approach is that far away from the linear support planes, the error $f(x) - \bar{f}_k(x)$ can be large. It would make sense to stay reasonably close to the support points where the cutting planes were obtained. This is also borne out by slow convergence and big jumps in the iterates x_k for the cutting plane method [37]. Bundle methods try to keep the next iterate close to the current *prox center*, which has its name from the *proximal point method* [44] that also uses the same quadratic term, and is based on a contraction operator proof. The difference is that bundle methods use $\bar{f}$ instead of f. The quadratic term in the bundle approach is actually an approximation of an earlier investigation into using a 'bundle' of second-order approximations

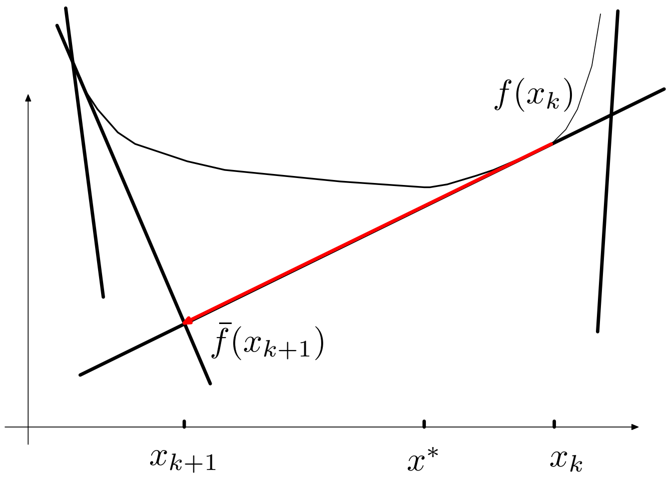

FIGURE 4.1 *Cutting plane iteration*

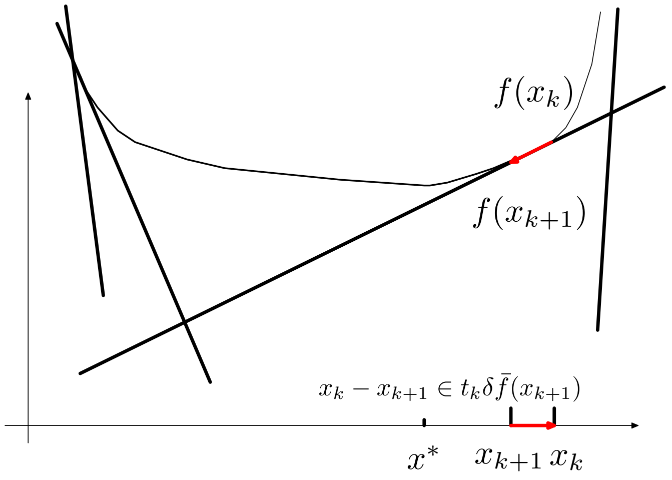

FIGURE 4.2 *Bundle method*

by Lemaréchal, as a historical note. It takes the form

$$\min \bar{f}_k(x) + \frac{1}{2t_k} \|x - x_k\|^2 , \tag{4.11}$$

where t_k is a parameter. A new cut is generated from the subgradient at the solution $\bar{x}$ to (4.11). If $f(\bar{x})$ has sufficiently decreased from $f(x_k)$, a step is taken towards $\bar{x}$, and x_{k+1} is updated. Otherwise, x_{k+1} remains the same until sufficient cuts have been generated around it so that the cutting plane approximation is detailed enough to result in an appreciable decrease of f when a new $\bar{x}$ is calculated from (4.11). In Lemaréchal, Nemirovskii, and Nesterov [31] it is noted that the bundle method derives its benefit mainly from the stabilization property of the quadratic term, and that as such other closeness metrics or *trust regions* may be used.

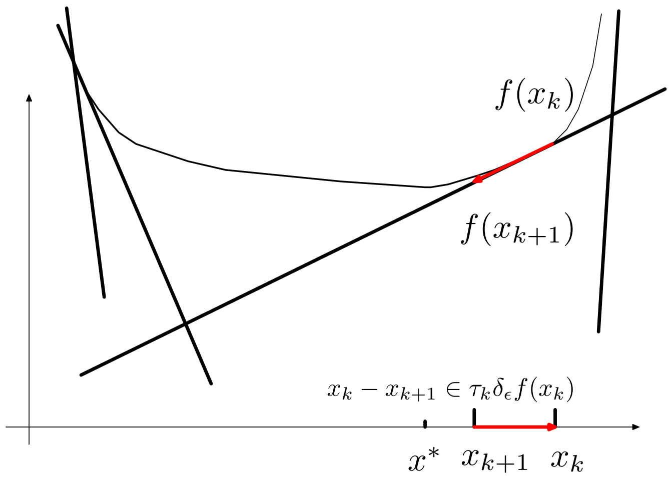

FIGURE 4.3 *Bundle ϵ-descent method*

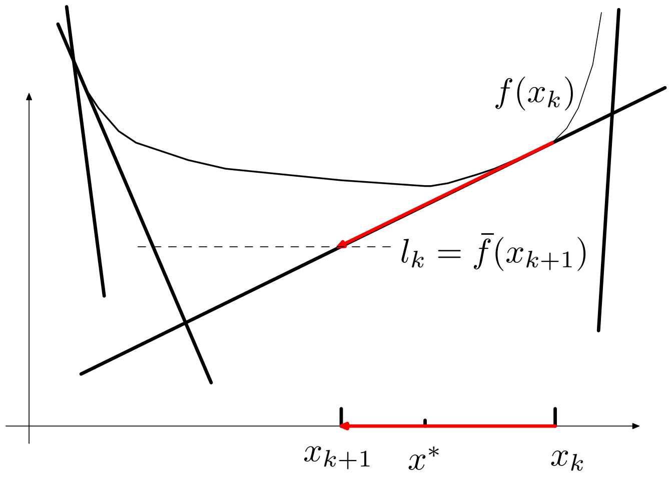

FIGURE 4.4 *Bundle method with level sets iteration*

When seen from the Lagrangian dual viewpoint, (4.11) gives rise to another bundle method, which can be shown [9] to be the same as (4.11) for a certain step size:

$$\min_{\lambda} \left\| \sum \lambda_i f_i' \right\|^2 \quad \text{subject to} \quad \sum \lambda_i f_i' \in \delta_{\epsilon_k} f(x_k), \quad \sum \lambda_i = 1, \quad \lambda \geq 0, \tag{4.12}$$

where x_k is the proximal point at iteration k. ϵ-subgradients $f_\epsilon'(x)$ are defined by

$$f(z) \geq f(x) + \left\langle f_\epsilon'(x), z - x \right\rangle - \epsilon, \quad \text{for all} \quad z, \tag{4.13}$$

and it follows that at any x, f_i' are $\epsilon_i(x)$-subgradients with

$$\epsilon_i(x) = f(x) - f(x_i) + \left\langle f_i', x - x_i \right\rangle, \tag{4.14}$$

x_i again being the point where the ith subgradient cut was obtained. The condition $\sum \lambda_i f'_i \in \delta_\epsilon f(x_k)$ can then be expressed as

$$\sum \lambda_i \epsilon_i(x_k) \leq \epsilon_k. \tag{4.15}$$

The solution λ to (4.12) can be seen as forming an approximate direction of steepest descent

$$d = -\sum \lambda_i f'_i. \tag{4.16}$$

A step is taken in that direction using a line search, if the decrease is sufficient. We stop when d is approximately zero, which means we are approximately optimal.

More recently, Lemaréchal et al. [31] have introduced another variant of bundle methods, based on *level sets* $\bar{f}_k(x) \leq l_k$:

$$\min \; \|x - x_k\|^2, \quad \bar{f}_k(x) \leq l_k. \tag{4.17}$$

That is, we are directly minimizing the distance to the proximal point while keeping $\bar{f}_k(x)$ below some level between $f(x_k)$ and $\min \bar{f}_k(x)$. In other words, x_k is projected onto the level set. This is shown in [31] to converge linearly.

4.3 Some Stochastic Decomposition with Subgradient Optimization Algorithms

The decomposition approach described earlier can be combined with any of the subgradient methods. With multistage Benders decomposition [2] in Chapter 5, we split (4.1) by time periods. Bundle-based scenario aggregation [35, 41] uses the ϵ-descent bundle algorithm, together with a scenario aggregation approach introduced by Rockafellar and Wets [45]. The latter uses an approximation to the *augmented Lagrangian method of multipliers* (AL) [43, 38], which, while linearly convergent, is slower in practice than the bundle-based algorithm, because of the difficulty of selecting a good sequence of weights for the quadratic penalty, according to Robinson [41]. As a side note, the quadratic penalty term in AL is being modified in [45] and essentially becomes the same as the proximal quadratic term in the bundle methods. The difference however is that the augmented quadratic term is added to the Lagrangian and is in the original variables (and hence the subproblems), while the bundle proximity term is in the Lagrangian dual variables, and thus appears only on the outer optimization in terms of the duals. Conversely, there is no explicit outer optimization for AL, as the new dual values are a simple function of the subproblem solution.

Robinson [41] applied the Lagrangian dual (4.4) of the decomposition method to the bundle method (4.12), and remarked on how the latter will actually converge to an optimal primal solution as well. The issue is that even if we know the optimal duals x in (4.4), it does not follow that (4.5) will return a feasible primal solution to the original problem (4.3). All we know is that $0 \in \delta f(x)$, but if $\max_y \; g_i(y_i) + \langle x, B_i y_i - b_i \rangle$ subject to $A_i y_i \leq a_i$ (4.5) has multiple solutions at the optimal x, that is $\delta f(x)$ contains more than only the zero subgradient, then we can't guarantee which one we're getting by evaluating $f(x)$. The bundle method doesn't address this question, and because it doesn't know about the decomposition primal problem underlying $f(x)$, it only assumes some $f'(x)$ can be obtained. Nevertheless it turns out that (in contrast to the cutting plane method) the bundle method terminates also with an approximately feasible primal solution. To see this, we show that approximate optimality of dual x implies approximate feasibility of primal y. The former means $\|d\| \leq \epsilon$ (4.16). Substituting (4.6) for f' in d,

$$\epsilon \geq \sum \lambda_i (B\hat{y}_i - b) = B(\sum \lambda_i \hat{y}_i) - b, \tag{4.18}$$

which means that $\sum \lambda_i \hat{y}_i$ is approximately feasible.

4.4 Degeneracy and the Cutting Plane LP

No treatise involving linear programming is complete without some notes about degeneracy. First we would like to express the opinion that 'degeneracy' is a misnomer. It has the air of implying that we should not have to deal with it. The usual method of avoiding degeneracy is a slight random perturbation of the data. However, usually degeneracy is a structural property of the problem, and to discard that information by random perturbation does not seem to be a necessarily good idea. Furthermore, there exist Simplex-type algorithms that can work directly with degeneracy using a notion of a basic variable set [1]. For an insightful unified approach see [51].

4.5 Scenario Aggregation with Bundle Method

We are now ready to apply bundle decomposition to scenario analysis. We start by applying decomposition to the stochastic problem (4.1). Random scenarios ω correspond to the blocks A_j in the decomposition problem (4.3), and the non-anticipativity constraints (4.1c) to the coupling constraints B between the x^ω, $\omega \in \Omega$, which can be expressed as linear constraints

$$x_t^{\omega_0} = \sum_{\omega \in \mathcal{A}_t(\omega_0)} p(\omega) x_t^{\omega} \Big/ \sum p(\omega) \,, \qquad \text{for all} \quad \omega_0 \in \Omega, \quad t = 1, \ldots, T. \tag{4.19}$$

$\mathcal{A}_t(\omega) \subset \Omega$ here denotes the set of scenarios that share the same path up to period t. It can be readily seen that the expectation operator in (4.1c) is an orthogonal projection onto the feasible policy space, but this is not relevant to our exposition.

The main problem with scenario analysis is that direct sampling schemes are not possible in general, which is an observation omitted by the papers on scenario analysis [45, 41, 35] (the same problem exists in Benders multistage regularization [48]). The issue is the following: for any appreciable number of periods, the number of scenarios is too large to solve (as it grows exponentially), even though the individual problems may be solved rather quickly. By necessity, some form of sampling has to take place. But the probability that any scenario has an overlap with any other sampled scenario in more than the first few stages is close to zero. In other words, the conditional expectation operator will not work, because for a given sampled scenario ω and period t, it most likely is the only one in $\mathcal{A}_t(\omega)$, even though in reality the distribution is (in our case) a tree over the periods. Then the scenarios will be wrongly considered independent.

Consider a modified set

$$\hat{\mathcal{A}}_t(\omega_0) = \{\omega \mid d_t(\omega_0, \omega) \leq r\} \,, \tag{4.20}$$

for some distance metric d. If d is the indicator function of $\mathcal{A}_t(\omega_0)$, we get the original set back. If

$$d = 0, \tag{4.21}$$

all scenarios take part in the conditional expectation, which has the effect of forcing x to be the same for all scenarios (assuming such a solution exists). In between, there is a tradeoff between the ability to delay decisions until an outcome occurs (the normal case), and being a policy. Clearly, if the original $\mathcal{A}_t(\omega_0)$ only contains one element after some period t, the solution to (4.1) is not a policy, as the decisions along the sample path are taken with only that one path under consideration, corresponding to a 'wait and see' approach (as if all future information were available at time t). On the other hand, $d = 0$ will result in a valid policy, if it exists under these conditions. So we would like a middle ground, an approximately valid policy, while not giving away too much of the decision space. d also affects the size of the sample required. In case of (4.21), the usual law of large numbers applies and expectation accuracy grows approximately by the root of the sample size. In the unmodified case, the sample size required is intractable, which was the problem to begin with. In between, we only know it will take more than in case (4.21), depending on which d is chosen. We add samples to the set until almost no more changes in the solution occur, indicating a sufficient sample size has been achieved.

Chapter 5

Stochastic Supply Chains Using Multistage Benders Decomposition

Benders decomposition (BD) [2] has been the main work-horse in two-stage stochastic linear programming. This is despite slow convergence, as the subproblems are easy to solve. What is especially nice about BD is that under certain circumstances, it scales with the number of periods, in that exponential explosion of scenarios is not an issue, unlike scenario analysis. This is due to sharing of cuts, as explained later. It is also receptive to sampling techniques, which extend to multistage problems. See Infanger [24] for importance sampling and other statistical methods applicable to BD. The Benders algorithm is in fact equivalent to *Dantzig-Wolfe decomposition* (DW) [12], but the latter usually is cited in the context of column generation, which is how the cuts appear as when considering the master problem from the dual perspective.

5.1 Benders Decomposition

We can derive BD by using decomposition from section 4.1 combined with the cutting plane method in section 4.2.1. We illustrate the two-stage BD case, which has the form

$$\begin{array}{lrcl} \min & c_1x_1 + c_2x_2 & & \\ \text{subject to} & Ax_1 \quad\quad & \leq & a, \\ & Wx_1 + Bx_2 & \leq & b. \end{array} \tag{5.1}$$

We assume B is a block matrix of the form $\operatorname{diag}(B_j)$. That makes W the connecting constraints matrix, if we switch to the dual viewpoint:

$$\begin{aligned} \max \quad & a'\pi_1 + b'\pi_2 \\ \text{subject to} \quad & A'\pi_1 + W'\pi_2 = c_1, \\ & B'\pi_2 = c_2, \\ & \pi_1, \pi_2 \leq 0. \end{aligned} \tag{5.2}$$

Applying to this the decomposition approach in section 4.1, we get

$$\begin{aligned} f(\hat{x}) = \max\ & a'\pi_1 + b'\pi_2 + \left\langle \hat{x}, (A'\pi_1 + W'\pi_2 - c_1)\right\rangle \\ \text{subject to} \quad & B'\pi_2 = c_2, \quad \pi_1, \pi_2 \leq 0, \end{aligned} \tag{5.3}$$

with subgradient

$$A'\pi_1 + W'\pi_2 - c_1. \tag{5.4}$$

Now we further observe that $f(\hat{x})$ is ∞ unless $A\hat{x} + a \geq 0$. Then $\pi_1 = 0$. Since $\hat{x}$ is not restricted by sign, we can replace it with $x_1 = -\hat{x}$, so that $Ax_1 \leq a$. $f(x_1)$ is still convex (if $f(\hat{x})$ is convex in $\hat{x}$, so is $f(-\hat{x})$), and only the sign of the subgradient changes. Thus we can equivalently minimize $f(x_1)$ subject to $Ax_1 \leq a$, with

$$f(x_1) = \max\ b'\pi_2 - \left\langle x_1, (W'\pi_2 - c_1)\right\rangle \qquad \text{subject to} \quad B'\pi_2 = c_2, \quad \pi_2 \leq 0, \tag{5.5}$$

a subgradient

$$f'(x_1) = -W'\pi_2 + c_1, \tag{5.6}$$

and a new lower bounding cutting plane in the variable x_1:

$$f(x_1) \geq b\pi_2 - \left\langle x_1, (W'\pi_2 - c_1)\right\rangle. \tag{5.7}$$

We now have all the components to apply the cutting plane method. As $\bar{f}_k(x_1)$ takes the maximum over all the cutting planes, we note that the term c_1x_1 is the same for all planes. We can thus formulate the cutting plane problem as

$$\begin{aligned} \min \quad & c_1x_1 + u \\ \text{subject to} \quad & Ax_1 \leq a, \\ & \pi_2'Wx_1 + u \geq \pi_2'b, \qquad \pi_2 \in K, \end{aligned} \tag{5.8}$$

where K contains all the cuts' dual variables. This is Benders decomposition; therefore we have shown that BD is an instance of the cutting plane method applied to decomposition.

5.2 Multistage Benders Decomposition

Benders decomposition needs to be extended for a multiperiod setting and randomness before it can be applied to the multiperiod supply chain (4.1). The resulting algorithm is known as *Multistage Benders* or *Nested Benders*. We refer to [5, 24] for a derivation.

There have been attempts at speeding up the relatively slow convergence of BD in the two-case and multistage settings. In the former, bundle methods and regularization [48, 50] have been successful. However, while Ruszczyński [49] extends his regularization framework to the multistage case, regularization does not generalize to a shared cuts approach with a fast-forward, fast-backward algorithm [60], which does not allow us to stay close to a regularization point (or we would not obtain sufficient coverage). As there is an exponential number of scenarios and thus required cuts if we cannot rely on cut sharing, that approach does not seem promising. For a stage-wise augmented Lagrangian-based decomposition approach for stochastic multistage linear programs, see [46].

Parallelization approaches have been many and are usually straightforward. Most multistage algorithms are highly parallelizable from solving many subproblems for (conditional) expectations, and also because it is not necessary to follow a sequence of stages, so that multiple subproblems can be solved in parallel. As far as generic, non-trivial parallelization methods are concerned, where required, we recommend the classic text by JáJá [26].

5.3 Multiperiod Cuts

Returning to our stochastic supply chain (4.1), we now cast it as a Multistage Benders (MB) problem. Contrary to the way that the non-anticipativity constraints (4.1c) were represented for scenario decomposition, for MB we identify variables with their equivalence class $\mathcal{A}_t(\omega)$. That is, for each such set, we have only one variable in the formulation, so that (4.1c) is automatically satisfied and does not have to be considered further. In particular, this is simple under the assumption that the demand distributions for each $\bar{u}_t^\omega$ are independent (we show later how to, in a certain sense, get past that limitation). That is,

$$\omega \in \Omega_1 \times \Omega_2 \cdots \times \Omega_T \quad \text{and} \quad E\big(X(\omega_i)Y(\omega_j)\big) = E\big(X(\omega_i)\big)E\big(Y(\omega_j)\big), \tag{5.9}$$

for any $i \neq j$ and any random variables $X(\omega_i), Y(\omega_j)$. This allows us to say in particular that the conditional distribution, given information up to stage t, does not depend on that information. A cut obtained at t can be shared for different path histories; that is, it can be made to be always valid, as shown below.

We allow lags greater than one, so that production matrix P in (4.1) is a band matrix, instead of simply staircase. The latter would be required in order to apply MB directly. However, this is not really a problem. We refer to section 2.2 for expressing longer lags in terms of unit lags, at the cost of extraneous dummy production variables and constraints. We show here how to modify Benders to express non-unit lags directly.

Production decisions are assumed to be finalized during the period of the start of productions, that is at the latest possible moment, to allow for as much information as possible to base decisions on. This means that for a given time t, A contains all component requirements of productions with starting time $t(e) = t$, as well as the output items of productions with zero lag, if any. The subproblems are over the variables of these productions, making decisions for the given time period. These decisions affect the next L time periods, during which various of the started productions finish their output item. To the left, there are all the output blocks of productions having started up to L periods earlier, together forming the inventory constraints for period t. The modifications are straightforward. A new cut now is dependent on the earlier path. To see this, let us focus on the constraints at time t,

$$B_L x^{\omega}_{t-L} + \cdots + B_1 x^{\omega}_{t-1} + A x^{\omega}_t \geq 0, \qquad x^{\omega}_t \leq \bar{u}^{\omega}_t, \qquad x^{\omega}_t \geq 0, \tag{5.10}$$

for all $\omega_t \in \Omega_t$, and $\omega_{t-L}, \ldots, \omega_{t-1}$ fixed. The variables corresponding to the latter time periods constitute the Benders master problem (only for purpose of cut generation), while variables associated with t appear in the Benders dual subproblems. Conditioning on the fixed outcome ω up to $t-1$ leaves us with expected cost over $\omega_t \in \Omega_t$. Each ω_t gives rise to a subproblem, which we average in case of sampling, or simply solve for each outcome if there aren't too many. In the usual staircase MBD, a cut does not involve variables from the earlier path history. As we can see, a cut here involves several stages. Let G be the matrix containing the current cut coefficients. To avoid cluttered notation, we do not explicitly mark G's dependency on the subproblem's stage t, or the dual variables' ω and t index.

$$\begin{aligned}
& & \text{minimize} \quad cx_t^\omega + \theta, \qquad \text{subject to} & \qquad (5.11a)\\
\pi_A : & & Ax_t^\omega \geq -(B_L x_{t-L}^\omega + \cdots + B_1 x_{t-1}^\omega), & \qquad (5.11b)\\
\pi_{\bar{u}} : & & x_t^\omega \leq \bar{u}_t^\omega, & \qquad (5.11c)\\
\rho : & & G_0 x_t^\omega + \theta \geq g - (G_L x_{t-L-1}^\omega + \cdots + G_1 x_{t-1}^\omega), & \qquad (5.11d)\\
& & x_t^\omega \geq 0, & \qquad (5.11e)
\end{aligned}$$

where we denote the dual variables resulting from solving subproblem (5.11) for different values of ω_t by $\pi_A \geq 0$, $\pi_{\bar{u}} \leq 0$, $\rho \geq 0$. Given $x_{t-L}^\omega, \ldots, x_{t-1}^\omega$, the new cut is the conditional expectation of

$$\begin{aligned}
& \pi_A' \left(-B_L x_{t-L}^\omega - \cdots - B_1 x_{t-1}^\omega\right) & \qquad (5.12a)\\
&+ \pi_{\bar{u}}' \bar{u}_t^\omega & \qquad (5.12b)\\
&+ \rho' \left(g - G_L x_{t-L-1}^\omega - \cdots - G_1 x_{t-1}^\omega\right), & \qquad (5.12c)
\end{aligned}$$

and will be added to the (G, g) cut block at the previous stage $t' = t - 1$, as a linear function of $x_{t'-L+1}^\omega, \ldots, x_{t'}^\omega$. Note the cut itself is the expression for the conditional expectation over any such path history, for a fixed set of dual solutions. It is also a subgradient of the conditional expectation at the path history where it was derived from. The cut can be used with any ω as usual, as the dual feasible region of (5.11) clearly is not affected. Although the cut stretches over $L-1$ stages, it will only be used at stage $t' = t - 1$, where $x_{t'}^\omega$ is decided, as only then do we have all the variable values over the stages that the cut contains. Other than that, we proceed exactly as for regular MB.

Note (Feasibility Cuts). Withdrawing intermediate items or raw material from inventory or incoming production always occurs at the time the respective production decisions are made. Therefore, the subproblems will never have commitments from earlier periods for inventory withdrawals, which could possibly draw inventory below zero and violate the partial conservation constraints (4.1a). Conversely, if there is a surplus of any item for which there is no carry-forward inventory production or other use, it is simply discarded. Together, these provisions ensure that subproblems are feasible at all times, and thus no feasibility cuts [24] are required.

5.4 Shared Cuts

Having good coverage of future cost cuts is essential. If we had to keep a different set of cuts for each path history, we would require an exponential number of such sets. Instead, if the dual feasible region of (5.11) stays the same, any dual solution will form a valid cut, if the dual objective is updated to reflect the path history. It is a valid cut in the sense that it still provides a lower bound plane for the future expected cost at any given x, but is weak in that it may not be a subgradient at any x.

It is certainly possible to have also matrices B with independent random coefficients. Looking at (5.12a), we can keep accumulating B dependencies (given by π_A from various stages) for each stage. Then we evaluate the B term for the outcome given by the current path to adjust the cut. (Once a B term has been evaluated for a given outcome, it can also be stored for reuse, if there are not too many outcomes.) Along the same line of thought, we can have random coefficients for A. While the dual region evidently is affected by the random coefficients, we are saved by the conditional expectation over the current stage scenarios that goes into a cut.

5.5 Serial Correlation of Demand Capacities

As we noted earlier, our model only has randomness in the right-hand side through (demand) capacities. Infanger and Morton [25] showed that under these circumstances, the random outcomes can depend on previous stages, while still allowing sharing of cuts among outcomes. This is essential for accumulating sufficient coverage of cuts when sampling paths (remember there is an exponential number of paths). We now derive this result and extend it to multiplicative dependencies.

We assume the capacity vector $\bar{u}_t^\omega$ (only the demand capacities are random) is an easily computable function $f_t(e_{t-L}, \ldots, e_t)$, where e_i are independent random vectors. Then (5.12b) becomes

$$\pi_{\bar{u}}' f_t(e_{t-L}, \ldots, e_t). \tag{5.13}$$

$t' = t-1$ is the stage of the new cut. The conditional expectation of (5.13),

$$h_{t'}(e_{t'-L+1}, \ldots, e_{t'}) = E\big(\pi_{\bar{u}}' f_t(e_{t-L}, \ldots, e_t) \bigm| e_{t-L}, \ldots, e_{t-1}\big), \tag{5.14}$$

is a finite weighted sum, assuming the independent distributions e_i are discrete, or the expectation is a sample average statistic. $h_{t'}(e_{t'-L+1}, \ldots, e_{t'})$ is part of the right-hand side of the cut (5.12), which now has a dependency on the previous $L-1$ random outcomes, that will propagate to lower cut stages through ρ (5.12c). Essentially, the expectation operation and multiplication with dual variables results in a sum of f terms in the respective cut right-hand sides.

Note. The crucial property of the conditional expectation (5.14) is that it is always over the same outcomes and probabilities because the current stage variables e_t are independent of the previous stage variables. Updating probabilities on an outcome dependent distribution already would require an exponential effort in the number of stages.

The effort to update all cuts based on a current path is too great without some restrictions on f. The amount of evaluations of f would be exponential in the number of stages (up to the number of arguments of f), with exponential base the number of cuts times the number of outcomes involved in each expectation.

Dependencies on additive terms involving f_t develop directly in (5.11d) and indirectly in (5.11e).

5.5.1 Vector Autoregression and Multiplicative Dependencies

Let f be a composition (to be specified) of a finite number of elementary terms g. Let the set of these terms be $\mathcal{S}$. Each $g(e_{t_1}, \dots, e_{t_2})$ can have a different input range $t_1, \dots, t_2$ and is a function of some independent variable vectors e_t. We introduce a 'version' of g for each $t = 1, \dots, T$, by fixing all random variable parameters for periods greater than t, at designated random outcomes $\omega_{t+1}, \dots, \omega_{t_2}$ that are set arbitrarily in the beginning. Thus g_t is a function of $e_{t_1}, \dots, e_t$, equivalent to g, when parameters for periods $t+1, \dots, t_2$ are fixed at some $\omega_{t+1}, \dots, \omega_{t_2}$. Also let $\mathcal{S}_t$ be the set $\mathcal{S}$ with each element g replaced by its version g_t.

We show that for certain f,

1. right-hand side dependencies of period t cuts remain simple compositions of terms in $\mathcal{S}_t$, and
2. the cost of adjusting a cut is of the order $O(|\mathcal{S}|)$ evaluations of terms in $\mathcal{S}_t$.

Let the class of compositions f be the linear combinations of terms in $\mathcal{S}_t$. Then the cost clearly is of the order claimed. We prove claim 1 by induction, for two different types of f.

Let f_t be a linear function of the independent random variables up to t. This includes any kind of *vector autoregression* (VAR) — in the moving-average representation — not necessarily stationary or even co-integrated (for an introduction to VAR, see e.g. Hamilton [21]). Furthermore, the autoregression coefficients can change with each current period t. This by itself provides for a large modeling space of stochastic processes. For example, it includes random walks, trend, and seasonal modeling.

The elementary terms are all the elements of the independent random vectors e. A version t of a term $g = e_{it'}$ is $e_{it'}$ itself, if $t' \leq t$, and the constant outcome $e^{\omega}_{it'}$ for a fixed ω otherwise.

The base case is creating the first cut from subproblems at some stage t. (5.12b) is a linear combination H of elements in $\mathcal{S}_t$. The conditional expectation of (5.12) takes a weighted sum of H at different outcomes of the elements in H that correspond to random variables of the current period t. It is clear that this weighted sum can be expressed in terms of a linear combination of elements in $\mathcal{S}_{t-1}$, which is the period of the new cut. Let us now proceed with creation of a new cut from subproblems at period t when there are existing cuts. (5.12b) again provides a linear combination of terms. By hypothesis, (5.12c) is a linear combination of terms in $\mathcal{S}_t$, so that by grouping coefficients of the same terms in (5.12), we obtain the sum H again as a linear combination of terms in $\mathcal{S}_t$. Taking the conditional expectation of H over the current stage random variables corresponds to taking a weighted sum of H with different fixed values for the random variables of period t. Grouping the corresponding coefficients again, and correcting for the fixed 'null' values of the constant versions, we see that again the result is a linear combination of the distinct terms in $\mathcal{S}_{t-1}$; that is, of the 'version set' of the cut's period $t' = t - 1$. This completes the proof for f a linear function.

Now let us assume that the elements of f_t are arbitrary polynomials in the independent random variables up to t. This subsumes the previous case, and also allows modeling of multiplicative serial dependencies, for example of *varying coefficient regression* type [22]. The elementary terms are all the monomials occurring in f, namely those of the form

$$(e_{i_1 t_1})^{n_1} (e_{i_2 t_2})^{n_2} \cdots (e_{i_m t_m})^{n_m}. \tag{5.15}$$

A version t of a term g is the shorter, scaled monomial that has the variables corresponding to $t' > t$ replaced by some fixed 'null' outcome. The base case follows from the general case. The multiplication by dual variables results in a polynomial H over the same monomials as are contained in the capacity term (5.12b) and the existing cut dependencies (5.12c), which by hypothesis only have monomials contained in $\mathcal{S}_t$. The conditional expectation affects the dual weights inside H and the monomials that involve current stage variables. For each such term, the submonomial consisting of the current stage random variables and a coefficient made up of dual variables is replaced by its expectation. Clearly, this can again be expressed as a polynomial consisting of terms in $\mathcal{S}_{t-1}$.

To summarize, we can efficiently construct and update the relevant right-hand side dependencies of cuts for a variety of serial correlations on the independent variables. For a closed-form expression for the VAR case, see Infanger and Morton [25].

Note. Even if f_t is identical for each stage, $\mathcal{S}$ will contain all the elements for each f_t, with their corresponding shifted time stamps. In the additive VAR model, since the elements are the individual random variables themselves, there can only be $O(T)$ dependencies on any cut. But for the general multiplicative model, if f_t is made up of $O(T)$ distinct elements, $|\mathcal{S}| = O(T^2)$. Then again, the possibly quadratic effort to update a cut is not as bad as it sounds, as the number of stages that we consider will be 20–40 at most. Typically there are also only very few lags ($\ll T$) in most time series models.

Note. While our framework assumes a *moving-average* representation of VAR and other distributions, which tends to result in polynomials stretching back to the first period, the participating monomials are shared between different periods. Also it is not hard to change the framework to allow full polynomials as basic terms (recursive definition as common in time series).

5.5.2 Positive Demand

Additive, non-monotonic time series such as VAR with normal innovations have the drawback that they can become negative. This would make the demand capacity right-hand side negative as well. Not only is this undesirable, but it will break our model, as productions are already required to be non-negative.

What happens if we allow the demand to go negative? We can remove the positivity constraints for demand productions. It destroys the Leontief structure, but that is not an issue in Benders decomposition. Negative demand then can be interpreted as customers returning goods that they are not satisfied with. Throwing demand and returns into the same distribution however is more of a modeling problem than any numerical concern. First, if demand goes negative, we are saying there are more returns than actual demand in that period, which is extremely unlikely from a practical standpoint (given that typical returns run around 10% of total). Another undesirable property is that once a VAR distribution goes negative, it tends to remain so for some time, which makes even less practical sense. In statistical forecasting, this usually is not a problem even for positive data, as the fitted series tends to remain positive when we forecast only a few periods ahead.

The alternative is to restrict attention to positive distributions. In the additive regressive model, this can be achieved by using positive innovations (typically from an exponential distribution) and positive coefficients. There are known criteria how to render such a distribution stationary, if desired, see e.g. [20]. Multiplicative distributions are also very interesting, particularly because they can have long memory without forcing the distribution to become more or less monotonically increasing.

5.5.3 Correlated Demand

While our modeling space described in section 5.5.1 is large, the restriction that the polynomials remain positive either allows us to model only positive correlation between time series polynomials, or prevents us from using the mainstream statistics tools such as *Principal Component Analysis* (PCA) or *Independent Component Analysis* (ICA) in developing distribution models. On the other hand, a custom approach may be called for anyway because statistical toolsets are geared towards statistical inference or optimal data representation, rather than creating models that are used to simulate processes. The area of statistics research most applicable to us is positive time series, see e.g. [7].

While addition of whole times series is one way to obtain positive correlation, in general we can say that (positive and negative) correlation is achieved by using common or correlated sections of the stage-independent random vectors e_t inside the polynomials for the distinct demand time series. For example, the conditional distribution of one correlated demand with respect to another will only involve expectations over the common components.

Lagged correlation between time series is straightforward. We first create the two timeseries with correlated components without a lag. This results in a series of polynomials $p(t)$ that consist of sums of monomials. A lagged version p_l of the time series simply means $p_l(t) = p(t - l)$, which can be directly implemented in our framework.

Note. Lagged demand, that is demand that only depends on realizations of random variables in earlier periods, is not ‘hidden’ from the model. The stochastic linear program knows about the future demand at the time the random variables are realized. This is because the decision production variables, by implementation of non-anticipativity, are distinct for each outcome of the independent random vectors, even if that outcome does not have an effect on the current stage.

Chapter 6

Computational Results and Comparison

We obtain results for the multistage Benders decomposition algorithm, using modified examples from industry collected and published by Willems [59]. We would like to take the opportunity to thank the author Willems, as we found it almost impossible to obtain any other data sets in the supply chain space. This data appears to be highly guarded by companies, presumably as their supply chains and contracts are regarded as trade secrets. In particular, costing information is absolutely critical for the success of our model, and is not something that firms like to share. Furthermore, the supply chain examples in Willems apparently constitute real problems that the companies providing the data were interested in solving. They are not just simple expansions of bill of materials at the ERP database level, which can easily result in problems with tens of thousands of productions and items and are too spurious to gain much insight from any solution. Rather, the problems are condensed with focus on critical paths and items.

The chains are pre-sourced, so that no substitution decisions or multiple stocking locations for the same item can be made. However, our model certainly allows for making sourcing decisions, and simple extensions to the test chains can bring additional sources into consideration.

6.1 Benders Implementation

As described at the end of section 5.5.1, we implemented a fully recursively specifiable polynomial rhs vector model on top of a multistage Benders decomposition, which can express multiplicative VAR and ARIMA timeseries, as well as other models. Cut rhs are computed as normalized expressions of the demand polynomials. A group of scalar random variables sharing the same stage can assume any joint distribution. It is left to the modeler to ensure that the polynomial timeseries don't have negative outcomes (although it would not be difficult to simulate the series conditioned on positive outcomes).

Cuts are also allowed to span multiple stages (section 5.3), and we permit arbitrary production lags. All cuts are shared cuts. The supply chain is specified using the production model described in section 2.1 and expressed in (4.1).

To solve the LP subproblems, we used the commercial CPLEX solver. The program that reads the Willems supply chain graphs and costing data and generates a corresponding productions model is written in Java. The multistage Benders decomposition program that solves the productions model is written in C. The examples were run on a Windows PC with a single-core AMD Semprom processor at 1.79 GHz and 1.5 GB of RAM.

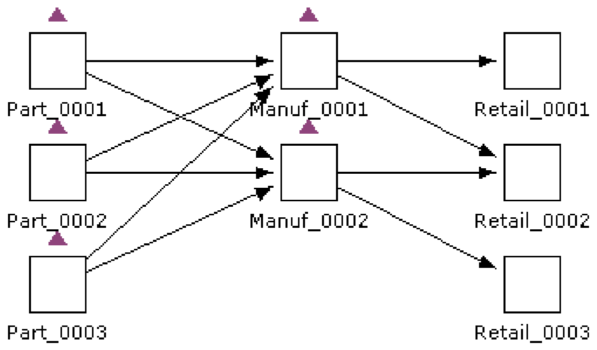

FIGURE 6.1 *Test Case 1*

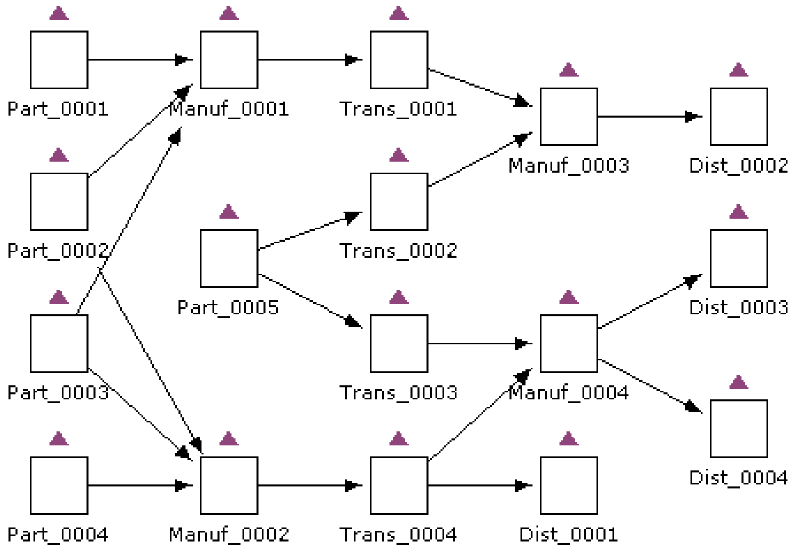

FIGURE 6.2 *Test Case 3*

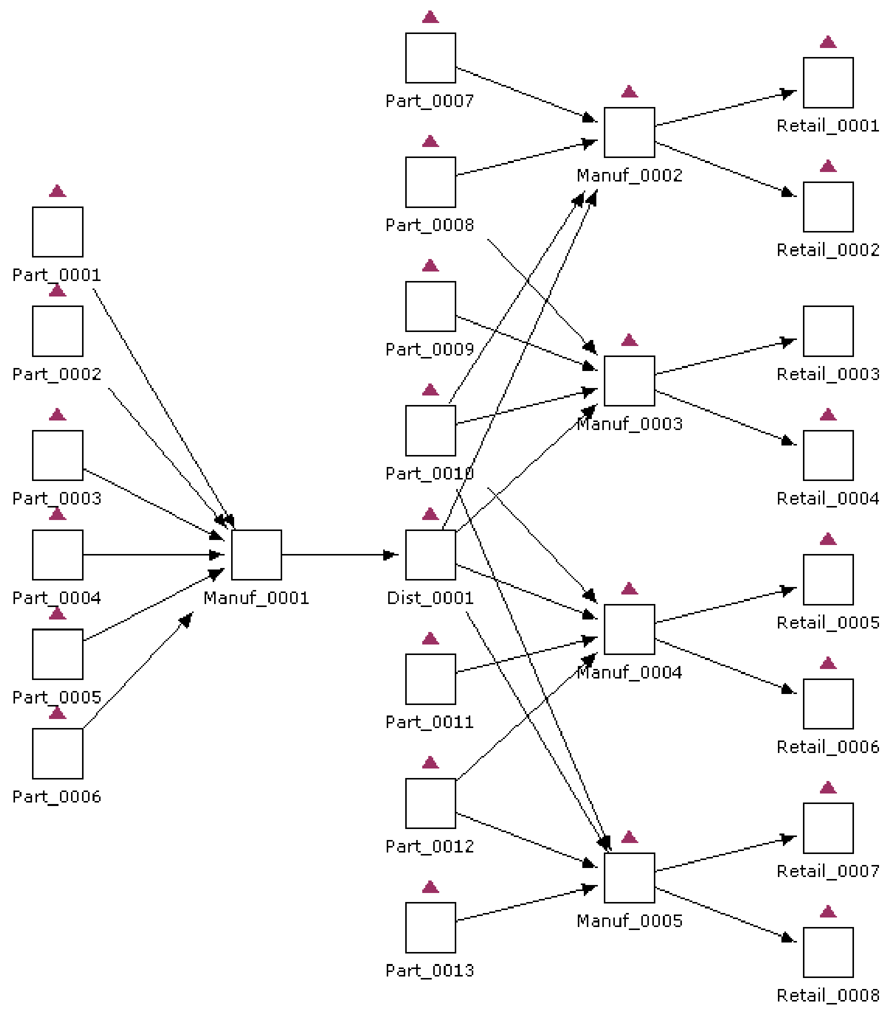

FIGURE 6.3 *Test Case 5*

6.2 Test Supply Chains

The original data is taken from the collection of supply chain examples in [59], which specifies cost and duration for each processing point like manufacturing, distribution, transportation and procurement. We set uniform profit margins at 40% and daily storage costs at 0.5% of total production costs. The numerical results can be found in Table 6.2, with descriptions of the columns in Table 6.3.

6.2.1 Demand Distributions

The distributions are chosen as explained in section 2.3.1 for the multiperiod setting, using the method outlined in section 5.5.1.

The simple ‘triangle’ instance of (2.3) is useful because it has only two outcomes

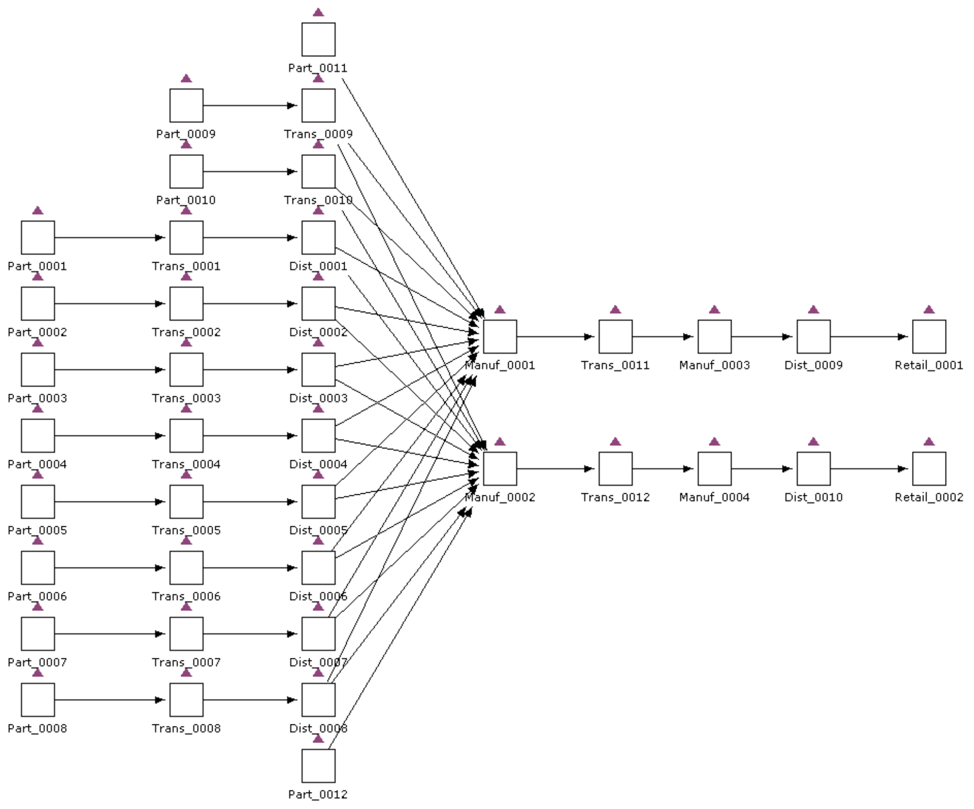

FIGURE 6.4 *Test Case 8*

per demand and per stage, so that when we have a moderate number of demand items within a time period, the Benders expected cuts can be calculated exactly, instead of having to resort to sampling. It follows that the lower bound is exact. The sample upper bound can always be made arbitrarily accurate by increasing the sample size (by the Central Limit theorem), so that we can guarantee near optimality of a solution in our test cases when the bounds have converged.

We also implemented (2.4), which has $O(DT^2)$ independent random variables, where D is the number of demand items. For example, let $D = 5$, $T = 15$, and assume, without loss of generality, a discretized version of each random variable with 8 outcomes, then the total number of scenarios is $8^{5\times16\times15/2} = 2^{1800}$. The equivalent linear program to (4.1) is inconceivable to solve. As we can solve problems of this size in a matter of a couple of hours on a PC using multistage Benders decomposition, this shows the practical value of the approach, as there is no other known method that can solve such problems to any demonstrable degree of optimality.

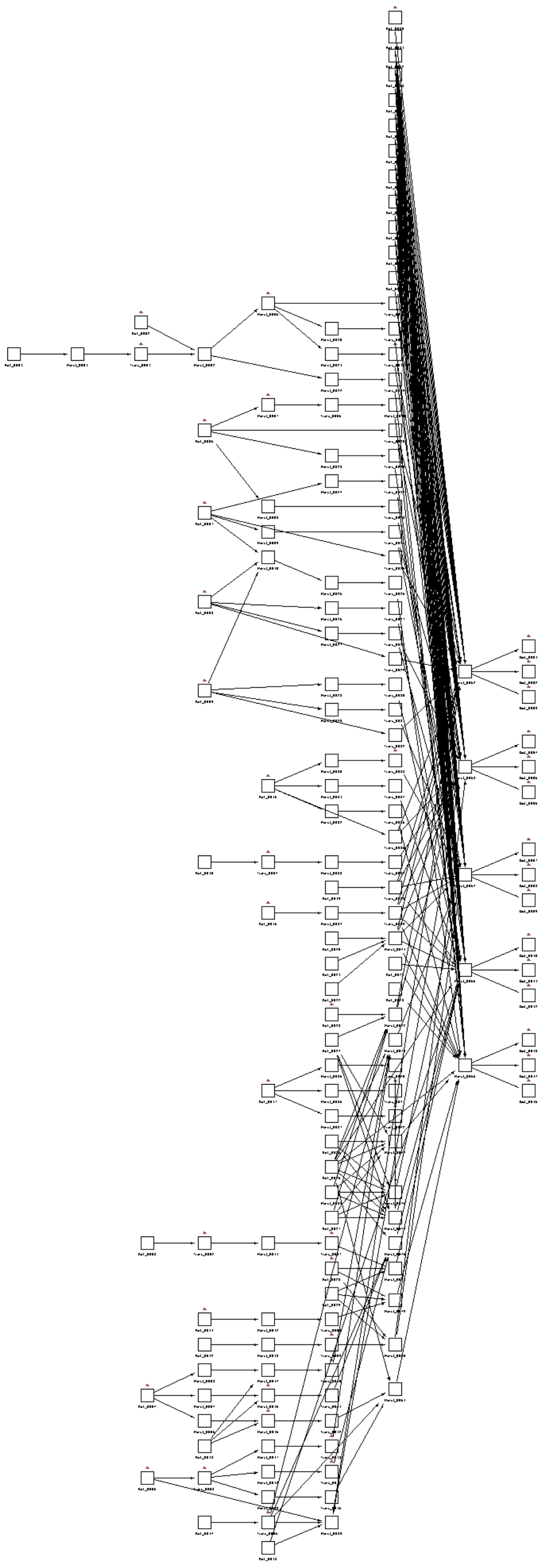

FIGURE 6.5 *Test Case 19*

Table 6.1
Origins of the selected supply chains

Chain	Description
1	Industrial organic chemicals
3	Computer peripheral equipment
5	Food preparations
7	Construction machinery and equipment
8	Electromedical and electrotherapeutic apparatus
19	Computer peripheral equipment

6.2.2 Number of Stages

If the number of stages required to accurately model the lags in the supply chain is too large, we can increase the number of time periods per stage. The trade-off is that decisions inside a stage are made for more than one time period, and that with respect to time periods (not stages), the resulting policy is not non-anticipative anymore. The solution is too optimistic. In most cases we would expect this to still be a good approximation however, as we are relaxing only the non-anticipativity requirement for subsets of production decisions that are close in time.

6.2.3 Lower Bound Accuracy

When we use a sampled cut, small inaccuracies may grow much larger the farther away we move from the point where the cut was obtained. Assuming normality in the cut errors, the chance of a large deviation for a particular cut may be small, but is bound to happen eventually at some cut, after a large enough number of iterations. The main problem with this is that the lower bound given by the first stage master problem is growing monotonical. Each cut can only help to push up the bound. If a big error cut manages to push up the bound by a large amount and overshoots the actual optimal value, further cuts will not be able to contribute anymore as they erroneously will not

Table 6.2
Test case parameters and numerical results

Ch	Stgs	Deph	RVs	Scens	Prods	Its	Time	Conv	Cuts	SSz
1	12	5	96	4.2×10^{77}	228	240	4	$< 1\%$	240	100
3	20	6	224	1.3×10^{181}	760	3370	253	14%	200	80
5	9	3	192	1.7×10^{155}	558	2640	154	$< 1\%$	500	40
7	12	5	192	1.7×10^{155}	984	4770	95	11%	200	20
8	12	4	64	5.6×10^{51}	1116	2790	525	10%	1000	40
19	12	5	480	$> 1.0 \times 10^{300}$	3924	320	70	$< 1\%$	300	100

Table 6.3
Descriptions of columns of Table 6.2

Column	Description
Ch	Test case number as in [59]
Stgs	Number of stages in the multistage Benders decomposition
Scens	Number of random scenarios resulting from the distribution model
Prods	Number of productions
Its	Number of fast-forward, fast-backward iterations
Time(min)	Running time in minutes
Conv(%)	Gap, if any, between best upper bound and lower bound
Cuts	Number of expected cuts used in each stage
SSz	Number of samples used to calculate expected cuts
Deph	Largest number of stages required to produce an end item
RVs	Number of basic independent random variables used to construct the distribution model

be binding. The most stable approach seems to be to dispose of cuts randomly or in a round-robin fashion, so that eventually a large error cut will disappear. This entails a non-monotonic lower bound. The drawback is a trade-off between choosing a larger number of cuts to allow for a large enough coverage of the cutting plane approximation, and being able to remove cuts contributing large lower bound errors sufficiently often.

As another interesting approach to address the cut sampling error issue, we can remove any cut that was obtained at a distance too far from the current subproblem solution, at progressively shorter distances. In other words, as the faces of the cutting plane approximation become more refined, we don't need any contribution to faces from cuts that originated too far away. For example, after computing subproblem solutions for a new cut for stage $t-1$, we can examine the supporting cuts at stage t and the distances from their origins, and then drop the cuts that exceed a certain threshold. We did not implement this idea, but suggest it may make a significant impact in the quest to obtain sufficient coverage while keeping the impact of cut errors low.

Finally, we may be able to completely avoid the expected cut sampling issue by computing exact cuts: by spreading out the independent variables (or variable vectors) over dummy stages for a given stage t, we reduce the number of outcomes per stage to the discrete number of outcomes of one random variable, instead of the product of these outcomes over all the variables in that stage. Slower convergence because of more stages and sparser coverage is the price to pay for exact lower bounds in this approach. Also it requires that the model make use of only discrete or discretized random variables.

6.3 Redundant Solution Space

One observation we made is that production decisions tend to vary quite a bit from stage to stage, even when the stages have equivalent independent distributions and are far away from the finite horizon. The reason is that the solution space allows for such variations without impacting total profit. For example, if it is optimal to allow for a certain amount of stockouts, it is not relevant to total profit whether they occur all in one period or are spread out evenly over stages. Clearly we would prefer a smooth picture, if only to interpret better what the solution is telling us. One possibility is to smooth the solution with an exponential smoother, for example, or moving average.

If productions were required to vary not too much from the previous stage, another advantage of a smaller variance would be, most likely, faster convergence, via reduction of the solution space. It is not clear however how this could be achieved without distorting the problem too much. One possibility is to require productions to be close to the previous-stage decisions (as opposed to closeness to the previous iteration, as in regularization [49] or bundle methods). We would impose a quadratic penalty term or trust region for all item productions (or a subset thereof, such as obtained by leaving out the inventory productions). This is an area for future testing and research.

6.4 Conclusions

We have shown that it is feasible to use a multistage Benders decomposition approach to calculate an approximately optimal solution of a supply chain problem formulation that considers decisions and future outcomes over many stages into the future, and that has an expressive demand distribution that might well be used for solving practical problems. We solved test models ranging from 9 to 20 stages.

Bibliography

[1] H. Arsham. Initialization of the simplex algorithm: an artificial-free approach. *SIAM Rev.*, 39(4):736–744, 1997.

[2] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.*, 4:238–252, 1962.

[3] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, 1997.

[4] P. Billington, J. McClain, and L. Thomas. Mathematical programming approaches to capacity-constrained MRP systems: Review, formulation and problem reduction. *Management Sci.*, 29(10):1126–1141, 1983.

[5] J. Birge. *Solution Methods for Stochastic Dynamic Linear Programs.* PhD thesis. Technical Report SOL 80-29, Systems Optimization Laboratory, Stanford University, 1980.

[6] L. M. Bregman, Y. Censor, and S. Reich. Dykstra's algorithm as the nonlinear extension of Bregman's optimization method. *J. Convex Anal.*, 6(2):319–333, 1999.

[7] T. C. Brown, P. D. Feigin, and D. L. Pallant. Estimation for a class of positive nonlinear time series models. *Stochastic Process. Appl.*, 63(2):139–152, 1996.

[8] R. Cambini, G. Gallo, and M. G. Scutellà. Flows on hypergraphs. *Math. Programming*, 78(2):195–217, 1997.

[9] R. Correa and C. Lemaréchal. Convergence of some algorithms for convex minimization. *Math. Programming*, 62(2, Ser. B):261–275, 1993.

[10] J. Cox, J. Blackstone, and M. Spencer. *APICS Dictionary.* American Production and Inventory Control Society.

[11] G. B. Dantzig. Optimal solution of a dynamic Leontief model with substitution. *Econometrica*, 23:295–302, 1955.

[12] G. B. Dantzig and P. Wolfe. The decomposition algorithm for linear programs. *Econometrica*, 29:767–778, 1961.

[13] A. Dax. The adventures of a simple algorithm. *Linear Algebra Appl.*, 361:41–61, 2003. Ninth Conference of the International Linear Algebra Society (Haifa, 2001).

[14] R. Durrett. *Probability: Theory and Examples.* Duxbury Press, Belmont, CA, second edition, 1996.

[15] R. L. Dykstra. An algorithm for restricted least squares regression. *J. Amer. Statist. Assoc.*, 78(384):837–842, 1983.

[16] Ş. Erengüç, N. Simpson, and A. Vakharia. Integrated production/distribution planning in supply chains: An invited review. *European J. Oper. Res.*, 115(2):219–236, 1999.

[17] D. Fogarty, J. Blackstone, Jr., and T. Hoffmann. *Production & Inventory Management.* South-Western Publishing, Cincinnati, OH, second edition, 1991.

[18] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *39th Annual Symposium on Foundations of Computer Science (Palo Alto, 1998)*, pages 300–309, Los Alamitos, CA, 1998. IEEE Comput. Soc. Press.

[19] S. C. Graves and S. P. Willems. Supply chain design: Safety stock placement and supply chain configuration. *J. of Operations Management*, 25(2):219–238, Mar 2007.

[20] G. K. Grunwald, R. J. Hyndman, L. Tedesco, and R. L. Tweedie. Non-Gaussian conditional linear AR(1) models. *Aust. N. Z. J. Stat.*, 42(4):479–495, 2000.

[21] J. D. Hamilton. *Time Series Analysis.* Princeton University Press, Princeton, NJ, 1994.

[22] T. Hastie and R. Tibshirani. Varying-coefficient models. *J. of the Royal Statistical Society. Series B. Methodological*, 55(4):757–796, 1993.

[23] M. Hillman and H. Keltz. Managing risk in the supply chain – a quantitative study. White paper, AMR Research, Inc., January 2007.

[24] G. Infanger. *Planning Under Uncertainty.* Boyd & Fraser Pub. Co., 1998.

[25] G. Infanger and D. P. Morton. Cut sharing for multistage stochastic linear programs with interstage dependency. *Math. Programming*, 75(2, Ser. B):241–256, 1996. Approximation and computation in stochastic programming.

[26] J. JáJá. *An Introduction to Parallel Algorithms.* Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, 1992.

[27] K. Jansen. Approximation algorithm for the mixed fractional packing and covering problem. *SIAM J. Optim.*, 17(2):331–352 (electronic), 2006.

[28] J. E. Kelley, Jr. The cutting-plane method for solving convex programs. *J. Soc. Indust. Appl. Math.*, 8:703–712, 1960.

[29] G. J. Koehler, A. B. Whinston, and G. P. Wright. The solution of Leontief substitution systems using matrix iterative techniques. *Management Sci.*, 21(11):1295–1302, 1974/75.

[30] H. Lee, V. Padmanabhan, and S. Whang. Information distortion in a supply chain: The bullwhip effect. *Management Sci.*, 50(13):1875–1886, 2004.

[31] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Math. Programming*, 69(1, Ser. B):111–147, 1995. Nondifferentiable and large-scale optimization (Geneva, 1992).

[32] B. Lus and A. Muriel. Measuring the impact of increased product substitution on pricing and capacity decisions under linear demand models. Accepted by: Production and Operations Management.

[33] V. A. Mabert. The early road to materials requirement planning. *J. of Operations Management*, 25(2):346–356, 2007.

[34] V. A. Mabert, A. Soni, and M. A. Venkataramanan. Enterprise resource planning: Measuring value. *Production and Inventory Management Journal*, 42(3/4):46–51, 2001.

[35] D. Medhi. Parallel bundle-based decomposition for large-scale structured mathematical programming problems. *Ann. Oper. Res.*, 22(1-4):101–127, 1990. Supercomputers and Large-scale Optimization: Algorithms, Software, Applications (Minneapolis, MN, 1988).

[36] H. Min and G. Zhou. Supply chain modeling: past, present and future. *Comput. Ind. Eng.*, 43(1-2):231–249, 2002.

[37] A. S. Nemirovsky and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization.* A Wiley-Interscience Publication. John Wiley & Sons Inc., New York, 1983. Translated from the Russian and with a preface by E. R. Dawson, Wiley-Interscience Series in Discrete Mathematics.

[38] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.

[39] J. Orlicky and G. Plossl. *Orlicky's Material Requirements Planning.* McGraw-Hill, 1994.

[40] S. A. Plotkin, D. B. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.*, 20(2):257–301, 1995.

[41] S. M. Robinson. Bundle-based decomposition: description and preliminary results. In *System Modelling and Optimization (Budapest, 1985)*, volume 84 of *Lecture Notes in Control and Inform. Sci.*, pages 751–756. Springer, Berlin, 1986.

[42] R. T. Rockafellar. *Convex Analysis.* Princeton Mathematical Series, No. 28. Princeton University Press, Princeton, NJ, 1970.

[43] R. T. Rockafellar. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Math. Oper. Res.*, 1(2):97–116, 1976.

[44] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control Optimization*, 14(5):877–898, 1976.

[45] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Math. Oper. Res.*, 16(1):119–147, 1991.

[46] C. H. Rosa and A. Ruszczyński. On augmented Lagrangian decomposition methods for multistage stochastic programs. *Ann. Oper. Res.*, 64:289–309, 1996. Stochastic programming, algorithms and models (Lillehammer, 1994).

[47] R. Ruiz-Benitez. *Optimal Implementation and Benefits of Rolling Inventory.* PhD thesis, University of Massachusetts at Amherst, 2003.

[48] A. Ruszczyński. A regularized decomposition method for minimizing a sum of polyhedral functions. *Math. Programming*, 35(3):309–333, 1986.

[49] A. Ruszczyński. Parallel decomposition of multistage stochastic programming problems. *Math. Programming*, 58(2, Ser. A):201–228, 1993.

[50] A. Ruszczyński. Some advances in decomposition methods for stochastic linear programming. *Ann. Oper. Res.*, 85:153–172, 1999. Stochastic programming. State of the art, 1998 (Vancouver, BC).

[51] G. Sierksma and G. A. Tijssen. Degeneracy degrees of constraint collections. *Math. Methods Oper. Res.*, 57(3):437–448, 2003.

[52] L. G. Sprague. Evolution of the field of operations management. *J. of Operations Management*, 25(2):219–238, Mar 2007.

[53] C. Stefanescu, V. deMiguel, K. Fridgeirsdottir, and S. Zenios. Revenue management with correlated demand forecasting.

[54] L. W. G. Strijbosch and J. J. A. Moors. Modified normal demand distributions in (R, S)-inventory control. *European J. Oper. Res.*, 172(1):201–212, 2006.

[55] S. Tayur. Moving beyond multi-stage inventory optimization. *Supply & Demand Chain Executive*, June/July 2007.

[56] A. F. Veinott, Jr. Extreme points of Leontief substitution systems. *Linear Algebra and Appl.*, 1:181–194, 1968.

[57] A. F. Veinott, Jr. Minimum concave-cost solution of Leontief substitution models of multi-facility inventory systems. *Operations Research*, 17(2):262–291, 1969.

[58] T. Vollmann. *Manufacturing Planning and Control Systems for Supply Chain Management*. McGraw-Hill, 2005.

[59] S. P. Willems. Real-world multi-echelon supply chains used for inventory optimization. *Manufacturing & Service Operations Management*, 10(1):19–23, 2008.

[60] R. Wittrock. *Advances in a Nested Decomposition Algorithm for Solving Staircase Linear Programs*. PhD thesis. Technical Report SOL 83-2, Systems Optimization Laboratory, Stanford University, 1980.

[61] N. E. Young. Sequential and parallel algorithms for mixed packing and covering. In *42nd IEEE Symposium on Foundations of Computer Science (Las Vegas, NV, 2001)*, pages 538–546. IEEE Computer Soc., Los Alamitos, CA, 2001.