Integrated Tactical-Operational Supply Chain Planning with Stochastic Dynamic Considerations

A thesis submitted to
the School of Graduate Studies and Research

in partial fulfillment of the requirements of the degree of

Master of Science

in

Systems Science
by

Hossein Fakharzadeh-Naeini

University of Ottawa

Ottawa, Ontario, Canada, K1N 6N5

Dedicated to

my wife, Faezeh Sadeghi,

and

my Parents,

and

my sister and brothers,

for their unlimited faithful and spiritual support on my study

Table of Contents

List of Figures

List of Tables

Abstract

Integrated robust planning systems that cover all levels of SC hierarchy have become increasingly important. Strategic, tactical, and operational SC plans should not be generated in isolation to avoid infeasible and conflicting decisions. On the other hand, enterprise planning systems contain over millions of records that are processed in each planning iteration. In such enterprises, the ability to generate robust plans is vital to their success because such plans can save the enterprise resources that may otherwise have to be reserved for likely SC plan changes. A robust SC plan is valid in all circumstances and does not need many corrections in the case of interruption, error, or disturbance. Such a reliable plan is proactive as well as reactive. Proactivity can be achieved by forecasting the future events and taking them into account in planning. Reactivity is a matter of agility, the capability of keeping track of system behaviour and capturing alarming signals from its environment, and the ability to respond quickly to the occurrence of an unforeseen event. Modeling such a system behaviour and providing solutions after such an event is extremely important for a SC.

This study focuses on integrated supply chain planning with stochastic dynamic considerations. An integrated tactical-operational model is developed and then segregated into two sub-models which are solved iteratively. A SC is a stochastic dynamic system whose state changes over time often in an unpredictable manner. As a result, the customer demand is treated as an uncertain parameter and is handled by exploiting scenario-based stochastic programming. The increase in the number of scenarios makes it difficult to obtain quick and good solutions. As such, a Branch and Fix algorithm is developed to segregate the stochastic model into isolated islands so as to make the computationally intractable problem solvable. However not all the practitioners, planners, and managers are risk neutral. Some of them may be concerned about the risky extreme scenarios. In view of this, the robust optimization approach is also adopted in this thesis. Both the solution robustness and model robustness are taken into account in the tactical model. Futhermore, the dynamic behaviour of a SC system is handled with the concept of Model Predictive Control (MPC).

Acknowledgements

First, I would like to thank my supervisor, Dr. Ming Liang, for his constructive insightful suggestions, his careful and patient examination of my thesis, and his financial support. Without him, it was impossible for me to complete this study.

In addition, I would like to thank my beloved wife, Faezeh, for her faithful and spiritual support on my study and for the sacrifice she has made for me.

I also want to express my appreciation to my parents who always give their instrumental support.

Nomenclature

Tactical model

Indices:

c	Component families ($c = 1, 2, \ldots, C$)
H	tactical planning horizon
i	product families ($i = 1, 2, 3, \ldots, I$)
j	manufacturing sites ($j = 1, 2, \ldots, J$)
k	distribution centers ($k = 1, 2, \ldots, K$)
s	Suppliers ($s = 1, 2, \ldots, S$)
t	tactical periods ($t = 1, 2, \ldots, T$)

Sets

C_s	set of component families that can be produced in supplier s
C_i	set of component families used to produce product family i
I_j	set of product families that can be manufactured in site j
I_k	set of product families that can be sold in distribution center k
J_s	set of manufacturing sites that can be supplied by supplier s
K_j	set of distribution centers that can be supplied by site j
S_j	set of suppliers that can supply site j

Parameters

$CMOT_j$ over-time cost in manufacturing site j

CP_{cs} production cost of component family c in supplier s

$CSOT_s$ over-time cost in supplier s

D_{ikt} demand of product family i in distribution center k in time period t

HD_{ik} holding cost of one unit of product family i in distribution center k per standard time unit

HJ_{ij} holding cost of one unit of product family i in site j per standard time unit

HS_{cs} holding cost of one unit of component family c in supplier s per standard time unit

JSA_j storage space available in site j SSA_s

KSA_k storage space available in Distribution center k

LB_{ij} minimum production volume of product family i in site j

MOT_j Over-time available in manufacturing site j per standard time period

MR_{ij} manufacturing requirement of product family i in site j

N_t number of standard time units in tactical period t

PA_j production capacity of site j per standard time unit

PI_{ij} production cost of product family i in manufacturing site j

RS_{cs} manufacturing requirement of component family c in supplier s

SA_s production capacity of supplier s per standard time unit

SOT_s Over-time available in supplier s per standard time period. It is assumed that suppliers are capable of satisfying all the demand set by manufacturing sites and consequently this parameter is a big number.

SR_{cs} manufacturing requirement of component family c in supplier s

SSA_s storage space available in supplier s

TSD_{ijk} transportation cost of product family i between manufacturing site j and distribution center k

TSS_{csj} transportation cost of component family c between supplier s and manufacturing site j

UC_{ik} unfulfilled demand cost of one unit of product family i in distribution center k

α_i storage space required to store one unit of product family i in manufacturing site j and distribution center k

β_c storage space required to store one unit of component family c in supplier s

ρ_{ci} amount of component family c used to produce one unit of product family i

Continuous Variables

IPD_{ikt} inventory of product family i in distribution center k at the end of tactical period t

IPS_{ijt} inventory of product family i in manufacturing site j at the end of tactical period t

UD_{ikt} unfulfilled demand of product family i in distribution center k at the end of tactical period t

ICS_{cst} inventory of component family c in supplier s at the end of tactical period t

XCP_{cst} amount of component family c produced in supplier s in time period t

$XMOT_{jt}$ over-time capacity utilized by manufacturing site j in time period t

XP_{ijt} amount of product family i produced in site j in time period t

XSM_{ijkt} amount of product family i shipped from manufacturing site j to distribution center k in time period t

XSS_{csjt} amount of component family c transferred from supplier s to site j in time period t

$XSOT_{st}$ over-time capacity utilized by supplier s in time period t

Integer (binary) Variables

X_{ijt} equal to 1 if product family i is manufactured in site j in time period t, 0 otherwise

Scalars

UB an arbitrarily chosen large number

Operational Model

Indexes

p_i product p of product family i ($p_i = 1, \ldots, P_i$)

l_j manufacturing line l in site j ($l_j = 1, \ldots, L_j$)

t operational period ($t = 1, \ldots, T$)

T operational planning horizon

Sets

I_{l_j} set of product families that can be produced on line l_j of site j

Parameters

D_{ik} demand of product family i in distribution center k at time period 1 taken from tactical model

$D_{p_i ikt}$ demand of product p_i of family i in distribution center k in time period t which is calculated as $w_{p_i ikt} * D_{ik}$

HD_{ikt} holding cost of product family i in distribution center k in short period t, it is assumed that all products from a certain family have the same holding cost

HJ_{ijt} holding cost of product family i in site j in short period t, it is assumed that all products from a certain family have the same holding cost

IPD_{ik1} first stage decision - inventory of product family i in distribution center k at the end of first tactical period, i.e. at the end of operational horizon

IPS_{ij1} first stage decision - inventory of product family i in site j at the end of first tactical period, i.e. at the end of operational horizon

$LB_{il_j j}$ minimum production volume of product family i on assembly line l_j in site j

PI_{il_jj} production cost of product family i in line l_j of site j, it is assumed that all products from a certain family have the production cost

SA_{l_jj} available capacity of production line l_j of site j

ST_{il_jj} set-up time required for product family i produced on assembly line l_j of site j, it is assumed that all products from a certain family have the same set-up time

SUC_{il_jj} set-up cost of product family i on line l_j of site j, it is assumed that all products from a certain family have the same set-up time

SIC_{ij} slack inventory cost of product family i in site j

SID_{ik} slack inventory cost of product family i in distribution center k

w_{p_iikt} percentage of D_{ik1} used to spread the demand over operational horizon, share of product p_i of family i in operational period t from the tactical demand over the first tactical period

X_{ij1} first stage decision – based on tactical planning solution specifies whether product family i can be produced in site j over the operational horizon or not

Variables

IP_{p_iijt} inventory of product p_i of family i in site j at the end of operational period t

IPD_{p_iikt} inventory of product p_i of family i in distribution center k at the end of time period t

SDC_{ik} slack variable introduced for the inventory level of product family i in distribution center k

SIP_{ij} slack variable introduced for inventory level of product family i in site j

$SSM_{p_i ijkt}$ amount of product p_i of family i shipped from site j to distribution center k in time period t

$UD_{p_i ikt}$ unfulfilled demand of product p_i of family i in distribution center k at the end of time period t

$X_{il_j jt}$ equal to 1 if product family i is manufactured on assembly line l_j in site j in time period t, 0 otherwise

$XMOT_{l_j jt}$ amount of over-time assigned to line l_j of manufacturing site j in time period t

$XSP_{p_i il_j jt}$ amount of product p_i of family i produced on line l_j of site j in time period t

$XSU_{il_j jt}$ equal to 1 if a set-up is required for product family i on assembly line l_j of site j in time period t, 0 otherwise

Stochastic tactical model

Indices

z scenarios ($z = 1, 2, \ldots, Z$)

Sets

Z_t set of all scenarios corresponding to the same node on scenario tree in time period t

Continuous Variables

ICS_{cstz} inventory of component family c in supplier s at the end of tactical period t under scenario z

IPD_{iktz} inventory of product family i in distribution center k at the end of tactical period t under scenario z

IPS_{ijtz} inventory of product family i in manufacturing site j at the end of tactical period t under scenario z

UD_{iktz} unfulfilled demand of product family i in distribution center k at the end of tactical period t under scenario z

XCP_{cstz} amount of component family c produced in supplier s in time period t under scenario z

$XMOT_{jtz}$ over-time capacity utilized by manufacturing site j in time period t under scenario z

XP_{ijtz} amount of product family i produced in site j in time period t under scenario z

XSM_{ijktz} amount of product family i shipped from manufacturing site j to distribution center k in time period t under scenario z

$XSOT_{stz}$ over-time capacity utilized by supplier s in time period t under scenario z

XSS_{csjtz} amount of component family c transferred from supplier s to site j in time period t under scenario z

Integer (binary) Variables

X_{ijtz} equal to 1 if product family i is manufactured in site j in time period t under scenario z, 0 otherwise

Parameters

D_{iktz}	demand of product family i in distribution center k in time period t under scenario z

Chapter 1. Introduction

1.1 An Overview of Integrated Robust Supply Chain Planning

Supply chain (SC) is a system, i.e. a combination of elements (e.g., suppliers, manufacturers, distribution centers (DC), wholesalers, retailers and final customers) that have interaction with each other and perform the following functions: procurement of raw material, transformation of raw material into finished products, and distribution of finished products to customers. A forward flow of material and a backward flow of information is the common characteristic of all SC's. To minimize system cost and maximize profit and service level, supply chain planning needs considerable attention.

The fierce competition, instability of customers' behaviour and needs, unforeseen disruptions and events, lack of data, continuous fluctuations in the prices of raw materials and principal components, enormous amount of information to be processed, highly dynamical behaviour of SC system and its environment, and some other complications are driving supply chain planning toward a new era. The development of methodologies to generate stable, reliable, and ever-valid plans that are tractable even for a large number of suppliers, sites, distribution centers, customer zones, products, and components has attracted the attention of practitioners. Planners have to monitor, review, and analyze the internal and external dynamics of the system and update SC plans and sometimes planning approaches for predetermined objectives.

For the above reasons, integrated strategic, tactical, and operational supply chain planning is of a great importance. Companies whose operational plans are generated based on unreal and overestimated/underestimated tactical plans will gradually lose their competitive edge because of wrong decisions. To achieve optimal solutions for SC models, the interdependencies between different planning levels should be taken into account, and

decisions should be taken simultaneously. Integrated methodology can ensure the feasibility of the developed plans at different levels and subsequently the applicability of the strategies.

1.2 Outline of the work

Research in the field of integrated production planning and scheduling has been active for several decades. The subject has been addressed for a variety of manufacturing systems such as refinery, pharmaceutical, home appliances, and consuming goods production systems. Also, uncertainty consideration has been recognized as an important element of supply chain models. Strategic, tactical, and operational planning issues have been traditionally analysed in isolated models and in some cases in an integrated model with no consideration of uncertainty. A comprehensive model that addresses integrated stochastic problems is still a missing link.

As a result, this study is focused on integrated robust SC planning. The advent of fast-growing technologies makes it possible to process an enormous amount of data in a short period of time (a few seconds). Organizations could analyze as many scenarios as required in order to make the most profitable and productive decisions in a reasonable amount of time. Generating SC solutions that are valid for many scenarios is possible now if proper models and solutions can be developed.

Since the independent demand triggers the whole SC planning system, the ability to forecast customer demand is critical. However, as a consequence of the increasingly volatile market and the lack of data, demand cannot be estimated precisely. The uncertain demand makes it difficult to harmonize the activities of all SC elements, which may cause considerable deviation from SC objectives and bull-whip effects in the chain. Therefore, for a SC planning system to generate accurate plans, it is important to review and control demand

variability. In this project, demand is regarded as an uncertain parameter in both stochastic and robust models.

1.3 Organization of thesis

An overview of supply chain planning and its prospects are provided in this chapter. A review of the studies that have been carried out so far by others in the integrated production and supply chain planning and its solution strategies, stochastic programming and its solution methodologies, and robust optimization is presented in Chapter 2. The limitations of previous works and the motivation of this study are also discussed in Chapter 2.

Chapter 3 introduces an integrated tactical operational supply chain planning model that encompasses most important elements of the system including suppliers, manufacturers, and DCs. Two mixed integer linear programming models are presented for the purpose of minimizing cost and maximizing customers' satisfaction.

Chapter 4 presents an efficient algorithm that attempts to solve the integrated problem. As was already discussed, the integrated model is segregated into two sub-models, tactical and operational. Since these models are solved in isolation, the solutions to the two sub-models may be conflicting with each other and the tactical solution may become infeasible or non-optimal at the operational level. Therefore, to converge the solutions of tactical and operational problems the proposed algorithm implements an iterative procedure. The production quantity, inventory level and unfulfilled demand are some criteria used for the proposed method.

In chapter 5, a summary of stochastic programming is presented and the stochastic tactical SC model is introduced. A Branch-and-Fix algorithm is suggested as a solution strategy and its usability is discussed in detail.

Chapter 6 is about robust optimization and why it is necessary. It is shown that a trade-off between the cost, service level, and risk is critical for an enterprise to run its business efficiently.

In chapter 7, two case studies are carried out to illustrate the application of the proposed method and to demonstrate the efficiency of the models and algorithms.

Finally, chapter 8 provides a summary of this study and outlines a few potential future research opportunities and directions.

Chapter 2. Literature Review

The goal of this thesis is to provide an integrated SC plan that stays valid in the presence of foreseen/unforeseen events. As discussed in chapter 1, an iterative strategy is adopted to solve the integrated planning problem and stochastic programming is employed to minimize the cost of here-and-now decisions and the expected cost of wait-and-see decisions. As not all the decision makers are risk neutral, robust optimization is applied to minimize the objective fluctuations originated from uncertainty. As the size of the model grows with the increase in the number of possible scenarios, the tactical model tends to become computationally intractable. As a result, a solution strategy is required to handle this situation. In the following sections, a review of studies conducted in SC planning, integrated models, stochastic SC models, stochastic programming solution strategies, and robust optimization is presented.

2.1 Integrated Supply Chain Planning

A SC is usually modeled as a network composed of vendor nodes, plant nodes, distribution center nodes, market nodes, and transportation arcs that connect the nodes and maintain both the physical and information flow(Pochet and Wolsey 2000). Supply chain management is an area of knowledge dedicated to integrating SC elements so that products are produced and distributed at the right quantity, at the right time, and to the right customer, in order to minimize/maximize cost/profit and satisfy customer needs and service levels.

Supply chain planning plays the role of the coordinator in the process of integrating SC elements and achieving the predefined objectives. It contains a wide range of time scales and different levels of details on each time frame that makes the hierarchical planning strategies a must (Kok and Graves 2003, Maravelias and Sung 2009).

In terms of time scales, supply chain planning is categorized into three levels: strategic, tactical, and operational. Strategic or long-term SC planning is about SC network design and is carried out over a long horizon from several months to several years. It specifies suppliers, manufacturers, DCs, and potential markets as well as the capacities of these nodes. It also selects the transportation methods and routes. Tactical or mid-term SC planning aims to fulfill customer demand over a medium-term horizon of a few months. At this level, all the products, resources, and transportation volume are considered in an aggregated form. Aggregated production and transportation decisions are made and tasks are assigned to the appropriate resources. Operational or short-term SC planning is performed for each individual SC element based on the goals enforced by higher-level plans. This is the most detailed plan that is implemented on the shop floor (Sung and Maravelias 2007).

Interconnections between different levels of supply chain planning cannot be ignored as the operational model is solved to attain the objectives set by the tactical plan. Hence the integration of SC planning levels is necessary in order to obtain a feasible and optimal (or near optimal) planning solution (Harjunkoski and Grossmann 2002, Kanyalkar and Adil 2005, Maravelias 2006, Wu and Ierapetritou 2007, Amaro and Barbosa-Povoa 2008, Maravelias and Sung 2009).

A comprehensive review of integrated production planning and scheduling can be found in Maravelias and Sung (2009). Their focus is on production nodes and the rest of SC elements are not addressed in detail. According to them, integrated modeling approaches fall into four categories: detailed scheduling models, relaxed and aggregated scheduling formulations, off-line surrogate models, and hybrid models for rolling horizon. These models can later be solved using three solution strategies called: hierarchical, iterative and full-space methods. Along this line, Grossmann *et al* (2002) and Shah (2005) presented an overview of the advances and challenges in integrated production planning and scheduling.

The decision on how an integrated model is generated depends on the solution strategy utilized to solve it. Li and Ierapetritou (2011) employed parametric programming to determine the capacity of the production system and embedded the outcome into the tactical model. The goal is to specify the production capacity by the boundary of the production feasibility region. In this way, the possibility of overestimating production plan is minimized because a surrogate scheduling model is present in the tactical model. With the rolling horizon concept, they considered the detailed model for a few recent periods but the rest of the planning horizon is still treated in an aggregated manner.

Another important approach in integrated planning is exploiting the special structure of a mathematical model and split the model into more manageable size using the Lagrangian, Wold-Dantig, and Bender's decomposition methods. These approaches could result in global optimal plans within a reasonable time. Heuristic methods such as genetic algorithm, simulated annealing, tabu search, and particle swarm optimization (PSO) have also been used to solve hard optimization problems (Karimi and McDonald 1997, Yan *et al* 2003, Yan and Zhang 2007, Józefowska and Zimniak 2008, Li and Ierapetritou 2009, Li and Ierapetritou 2010).

One of the disadvantages of full-space strategies is that proficiency in mathematics (mathematical decomposition) and computer science (heuristic and meta-heuristic algorithms) is required. An easier way of solving these integrated models is using iterative methodologies. These methods are not as complicated as full-space methods and the quality of their final solutions is often substantially better than those of the hierarchical strategies. The iterative methods have been problem-specific so far (Erdirik-Dogan and Grossman 2005, Wu and Ierapetritou 2007, Sousa, Shah, and Papageorgiou 2008). In this study iterative approach is employed to ensure the consistency of tactical and operational decisions.

2.2 Dynamic Supply Chain Planning

In the previous section, it was discussed that supply chain is a system composed of a set of elements connected together by information links within system boundaries. SC is also a dynamic system whose behaviour and state changes over time. A dynamic system can be deterministic or stochastic. SC is a stochastic dynamic system, i.e. its behaviour and state changes over time and there are uncertainties about the future states of the system. As a result, a real-time SC optimization strategy should be adopted in order to achieve reliable real-time plans. Hence periodical feedback information is indispensable inSC planning.

For a SC to stay competitive in the global market, it is important to not only develop optimal (or near optimal) and consistent solutions for integrated SC planning models but also to continuously review, monitor, and analyze the SC dynamics. Control-oriented decision framework can be applied to do so. Up to now, many problems in SC modelling with dynamic considerations have been addressed by classical control theory, advanced control theory, dynamic programming and optimal control, model predictive control, and rolling horizon concepts (Puigianer and Lainez 2008, Sarimveis *et al* 2008). A historical review of dynamic SC modelling can be found in Sarimveis *et al* (2008).

The dynamic programming techniques search for a trade-off between present and future costs in deterministic or stochastic dynamic environments where decisions are made in stages. At each stage, decisions are ranked based on the sum of present cost and (expected) future cost for the purpose of minimizing the cumulative cost over all stages.
As stated by Bertsekas (2000) a basic discrete-time system has the following form

$$X_{k+1} = f_k(X_k, U_k, W_k) \quad k = 0, 1, \dots, N-1, \qquad (2-1)$$

where
k indexes discrete time,
x_k is the state of the system and summarizes past information that is relevant for future optimization,

u_k is the control or decision variable to be selected at time k,

w_k is a random parameter (also called disturbance or noise depending on the context),

N is the horizon or number of times the control is applied, and

f_k is a function that describes the mechanism by which the state is updated.

Denoting the cost incurred at time k, denoted by $g_k(x_k, u_k, w_k)$, accumulates over time, the total cost is

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k), \tag{2-2}$$

The optimization is over the controls $u_0, u_1, \dots, u_{N-1}$. We are looking for an optimal control policy that minimize/maximize the objective function and satisfies the constraints while keeping the output as close as possible to the reference trajectories. The overall framework of dynamic programming is illustrated in Fig. 2.1. At each stage the controller applies an optimal control action u_k that depends on the state.

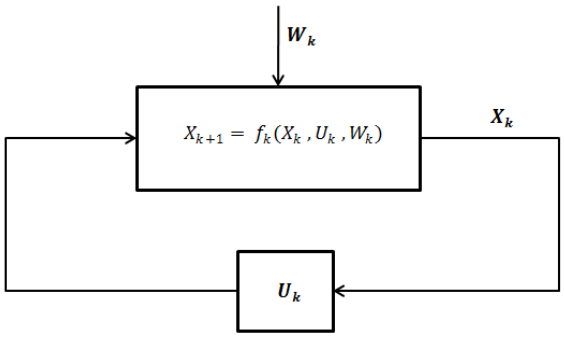

Figure 2.1 Overall framework of dynamic programming

Due to dimensionality, many models that are based on dynamic programming and optimal control cannot be solved analytically. Model predictive control (MPC) is a viable approach to deal with intractability in the optimal closed-loop feedback control design (Bertsekas 2000, Sarimveis *et al* 2008). The basic structure of MPC is shown in Fig. 2.2.

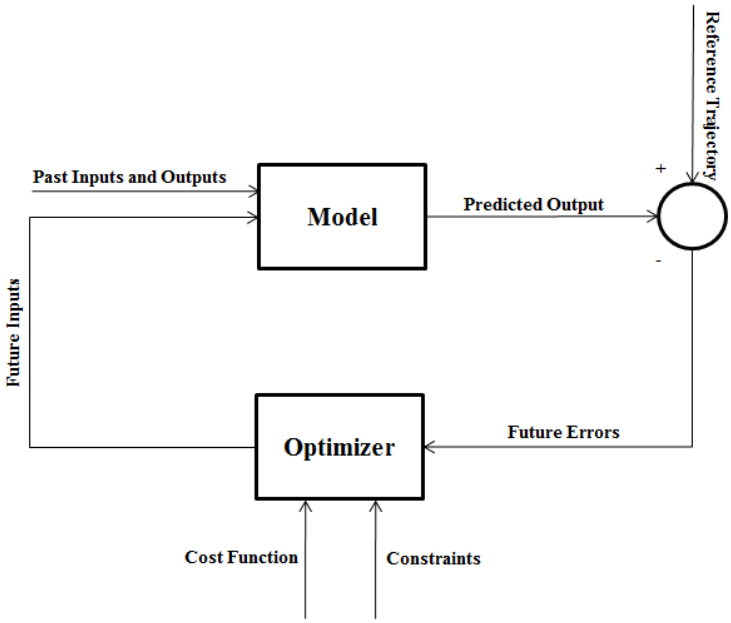

Figure 2.2 Basic framework of Model Predictive Control (MPC)

The methodology of MPC is presented below:

The future outputs for a planning horizon N, are predicted at each instant t using the process model. These predicted outputs $y(t+k|t)\ for\ k = 1 \dots N$ depend on the past inputs and outputs and on the future control signals $u(t+k|t), k = 0 \dots N-1$, which are those to be calculated and to be sent to the system as input.

The set of future control signals is calculated by optimizing a determined objective function in order to keep the process as close as possible to the reference trajectory which can be the inventory level, service level, or maximum number of late orders. This function usually takes the form of errors between the predicted output and the predicted reference trajectory.

The control signal $u(t|t)$ is sent to the process while the next manipulated variables calculated are rejected, because at the next sampling instant $y(t+1)$ is already known and process is repeated with this new value and all the sequences are brought up to date. Thus $u(t+1|t+1)$ is calculated (which in principle will be different from $u(t+1|t)$ because the new information is available) using the receding/rolling horizon concept.

A model is used to predict the future plant outputs, based on the past and current values and on the proposed optimal future control actions. These actions are calculated by the optimizer

taking into account the cost function as well as the constraints (Camacho and Bordons 1995).

Li and Marlin (2009) introduced a novel robust MPC method for SC optimization with uncertainties. They developed the deterministic MPC formulation to be used for comparison with the closed-loop robust stochastic MPC. In their formula, the system-state model is built based on the mass balance of inventory in each node in the SC system. The objective was to minimize cost subject to non-negativity constraint and upper bounds imposed on manufacturing and transportation capacities. In the proposed robust formulation, demand, manufacturing rate and transportation time are considered as uncertain elements.
The work by Bose and Pekny (2000) presented MPC as a solution methodology. In order to address the stochastic dynamic nature of SC under investigation, a forecasting-optimization-simulation framework was suggested. These three modules work together to ensure the desired customer service level and to minimize the average inventory using a scheduling model.

The MPC as a modern control concept was utilized by Wang *et al* (2007) in a two-level hierarchical strategic-tactical decision architecture for supply chain management in a semiconductor manufacturing system. Similar to Li and Marlin's work (2009), the mass inventory balance equations were implemented as the control equation. A weighted objective function composed of three sub-objectives was proposed to keep the predicted output as close as possible to set-points and strategic planning targets.

The concept of MPC was used by Perea-Lopez *et al* (2003) to capture the dynamics of supply chain. They compared the centralized and decentralized SC management approaches to highlight the better results of the former. In centralized SC systems, there is a global manager that coordinates the decision of the entire SC whereas in the decentralized approach each element of SC system makes its own decisions independently. They applied the MPC approach to an operational (detailed) mixed integer linear programming model.

According to the accessible literature, the dynamic programming approaches so far have not been employed for integrated tactical-operational SC planning with robustness consideration.

2.3 Stochastic Programming and Supply Chain Planning

In the above, integrated planning and dynamical behaviour of SC have been reviewed. As was mentioned before, SC is a stochastic dynamic system with continuous change and fluctuations in system state. Customer demand, material price, manufacturing lead time, shipping lead time, time between failures and many other parameters of the system are random variables. Introducing these uncertainties into the model makes the optimization problem extremely complex. Stochastic programming models are well suited for helping modellers in capturing variability.

Stochastic programming was introduced in 1950s (Danzig 1955) and since then it has proven to be an efficient and robust method in capturing stochastic aspects of decision-making problems. SCM can also benefit from this technique.

The essential character of general stochastic SC planning models is that planning activities are divided into two or more stages which subsequently categorize decision variables into two general classes: here-and-now (first-stage) variables and wait-and-see (recourse, second-stage) variables. The decisions related to the first-stage variables are the only ones to be implemented. In most cases, recourse decisions are delayed until the uncertainty unveils. In model predictive control, the here-and-now decisions are made by solving for the control variables in the optimization process for the immediate next period and the rest of the decisions are ignored and decided upon in the next iterations (Birge and Louveau 1994, Kall and Wallace 1994, Kall and Mayer 2005).

In stochastic programming, system parameters are regarded as random variables and their uncertainty can be represented by a set of scenarios with a probability associated with each of them. This way, the uncertainty is represented by a finite number of discrete realizations of stochastic values which makes the modelling and solution process manageable. The only required tool is a methodology for generating the aforementioned scenarios (Puigjaner and Lainez 2008).

In the stochastic models of Puigjander and Lainez (2008), it is assumed that the distribution functions of uncertain parameters are known or can be estimated. In their scenario-based stochastic model a forecasting technique is utilized in order to predict the uncertain parameters of the model. Forecasting module represents each uncertain element as a combination of a forecasted mean value plus an error estimate while latter is a random variable with an estimated distribution function such as the normal distribution function. Monte Carlo sampling can be employed afterwards to create the scenarios required in the stochastic mathematical model. The same approach is exploited by Schutz and Tomasgard (2011) and Guan and Philpott (2011) to forecast customer demand and to generate scenarios applied in the SC mathematical model.

The uncertainty has been considered in the literature in two-stage and multi-stage stochastic programming models to generate more efficient and reliable plans in today's volatile markets. Strategic, tactical, and operational production, inventory, and transportation planning models are no exception in this regard (Gupta and Maranas 2003, Ryu, Dua, and Pistikopoulos 2004, Choi, Realff, and Lee 2005, Li and Ierapetritou 2007, Roghanian, Sadjadi, and Aryanezhad 2007, Bonfill, Espuna, and Puigjaner 2008, Sodhi and Tang 2009).

The notable disadvantage of uncertainty consideration in the form of scenario tree is reflected by the large dimensionality of the resulted model. For this reason, a solution methodology capable of solving these problems accurately in a reasonable amount of time is

imperative. Taking advantage of structure of the model is the focus of the most algorithms developed in the area of stochastic programs. The L-Shaped method, Dantzig-Wolfe decomposition, Basic Factorization method, Nested Decomposition, Bender decomposition, Brand-and-Fix algorithm, Branch-and Price Algorithm, Progressive Hedging Algorithm (PHA), and Dual decomposition are a few to mention. In most of the solution strategies, stochastic model is converted to an equivalent deterministic program in order to maintain the tractability of the problem (Van Slyke and West 1969, Wollmer 1980, Birge 1982, Louveaux 1986, Laporte and Louveaux 1993, Birge and Louveaux 1997, Caroe and Tind 1998, Caroe amd Schultz 1999, Loveaux and Schultz 2003, Alonso-Ayuso *et al* 2003, Sen 2005, Sherali and Zhu 2006, Escudero *et al* 2007, Escudero *et al* 2009, Escudero et al 2010, Gupta and Grossmann 2010).

In this study, we modify the Branch-and-Fix (BF) algorithm developed by Escudero *et al* (2009) to solve our tactical SC planning model. The reason is that the BF algorithm can generate global optimal solutions that are desirable in mid-term and long-term decision making problems as opposed to methodologies that seek for local optimal solutions.

2.4 Robust Supply Chain Planning Model

The stochastic dynamic behaviour of a SC requires planning approaches that are not only reactive but also proactive in an efficient manner. The focus of stochastic programs is on optimizing the objective function composed of cost/profit of here-and-now decisions plus expected value of wait-and-see decisions. This is a reasonable approach for risk neutral practitioners. However this is not always the case. Some decision makers may wish to avert risks. For such decision makers, stochastic programming may not be a desirable tool. Although uncertainty consideration is important to be proactive to the predictable events, the possible extreme scenarios are not handled properly. It is true that we could predict the future and embed that forecast in planning model but the question still remains to be answered when some of the extreme scenarios do occur unexpectedly. To handle this

situation, robust optimization has evolved over the decades. Planners often demand the planning model to generate the best solution in any situation while the model itself is still valid.

As Mulvey *et al* (1994) stated, solution and model robustness have to be considered in order for a model to be valid in all circumstances. As was mentioned before, disruptions, stochastic fluctuations, erroneous data, and interruptions always exist. It is the responsibility of planners to predict and consider their impact on the SC. The solution of such a plan should be reliable and valid in case any of scenarios does occur. The main objective is to produce solutions with the smallest possible expected cost/profit deviation from the scenario-specific optimal solution. Robust program modifies stochastic model by adding the variance to the objective function. Depending on the value of the weight put on this variability, the optimization process may be in favour of the solutions with higher total expected performance and smaller second-stage performance variance to the solutions with lower total expected performance and possibly larger second-stage performance variance (Aghezzaf *et al* 2010).

Al-Qahtani and Ekamel (2010) studied the model and solution robustness in a mixed integer nonlinear programming model formulated to determine the optimal integration strategy and planning in a refinery system. Oil and product prices as well as customer demand are considered as uncertain parameters and handled by a two-stage scenario-based stochastic formula. The variations in the raw material and product prices as well as the deviation from the forecasted demand are added into the problem to increase the robustness of the model and its solution.

A robust tactical SC planning model was formulated by Al-e-hashem and Aryanezhad (2011) as a multi-objective mixed integer linear programming model which was solved using the LP-metrics method.

The design and structure of a supply chain network should be decided upon at strategic level. This involves decisions on the number, locations, and capacities of the potential suppliers, distribution centers, customer zones and assemblers as well as the mode of flow between each pair of facilities. The effect of these decisions will last for several years. Due to the stochastic dynamic behaviour of a SC and its environment, uncertainty needs to be incorporated into the long-term model. Hence the design robustness is an important aspect to consider. There is rich literature on SC network design, e.g., the studies by Klibi *et al* (2010), Pan and Nagi (2010) and Pishvaee *et al* (2011).

In this study, we propose a robust tactical SC planning model to work in tandem with the operational planning model so as to improve the stability and reliability of the proposed approach.

2.5 Motivation and Objectives

As reviewed in this chapter, most studies on integrated SC planning focus only on production node rather than most of the importamt SC components such as suppliers, manufacturers, DCs, transportation systems, and material planning. Also, stochastic dynamic SC programs are concentrated on a single level of planning hierarchy, i.e. strategic, tactical, or operational. Therefore, it is highly desirable to develop a model and a solution method for multi-level planning and to addresses stochastic dynamics of the system as well. This thesis will therefore be directed towards the development of such a model and solution approach.

The objectivres are to:

1. Develop an integrated SC planning model,
2. Expand the integrated model to address the stochastic dynamic behaviour of the system and its environment,
3. Design an efficient solution algorithm to solve the integrated model, and

4. Propose a solution algorithm for two-stage stochastic tactical planning model.

Chapter 3. Framework of the Supply Chain Planning Model

Supply chain planning problems cover a wide range of activities from procurement of raw materials and component parts to distribution and a wide range of time scales from long term to short term. The overall scope of the derived model can be shown in a matrix form (see Fig. 3.1). We are interested in integrated tactical operational planning systems that contain most of the SC elements including procurement, production, distribution and sales, and comprised of production, demand, and transportation planning modules across the SC spectrum.

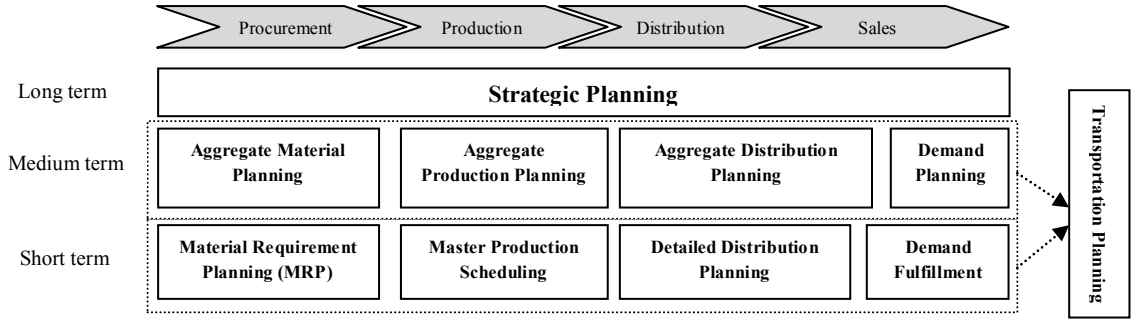

Figure 3.1 Supply chain planning matrix

The hierarchical structure of a SC planning system and interdependencies between the levels of SC planning are illustrated in Fig. 3.2. In this thesis, we will focus on the tactical and operational levels.

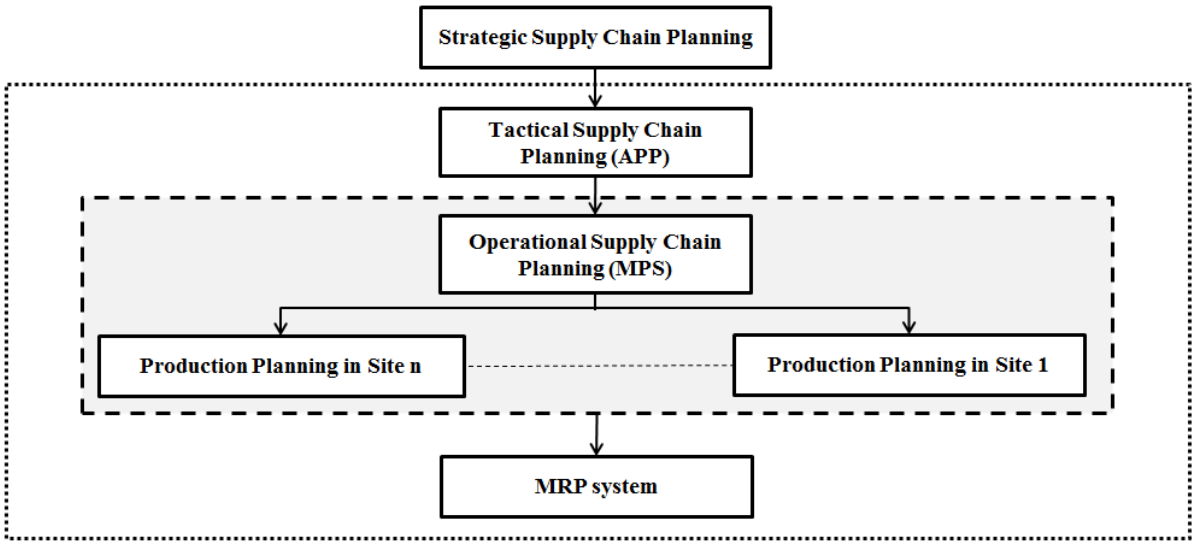

Figure 3.2. Structure of a SC planning systems and interdependencies between SC planning levels. The dash lines specify the boudary of operational SC planning system. In this level, the MPS developed for the elements of SC network is subdivided into more detailed

In this chapter, the overall framework of the SC under investigation will be described and the integrated tactical-operational SC planning model will be formulated.

3.1 Problem Description and Formulation

The main objective in supply chain planning is to fulfill customer demand at minimum total cost. In this study, the total cost is the sum of costs incurred in inventory, production, transportation, overtime activities, backlog and setups. Generally, parameters such as customer demand (deterministic or stochastic), production and transportation capacities, set-up time, and cost rates of production, transportation, holding, shipping, and set-up are given. The consumption rates are also known.

Since the iterative methodology is chosen to solve the integrated model, the integrated SC planning formulation is segregated into tactical (first-stage) and operational (second-stage) sub-problems. The tactical planning horizon can be from several months to a year. It is then divided into a series of equal or unequal time periods. Usually the initial periods are shorter

due to the fact that targets for subsequent periods can be changed as the uncertainty unveils and more information is available. As is illustrated in Fig. 3.3, the first tactical period is the horizon of operational model which is further divided into a set of short-term periods.

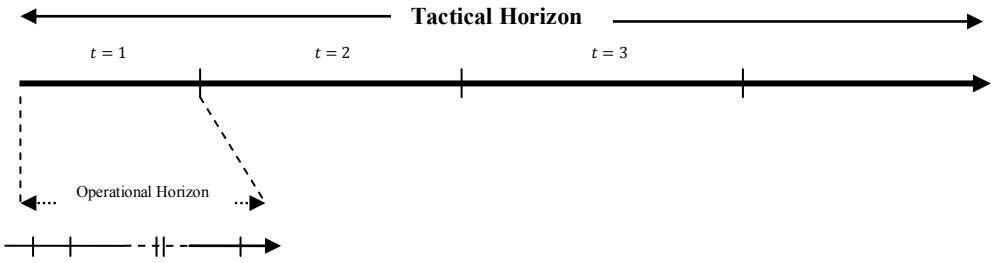

Figure 3.3. Tactical-operational planning time buckets. Communication is achieved via constraints linking tactical targets with final inventory/unfulfilled demand between the two levels at the end of each tactical period

The linking of the two problems is achieved via constraints that enforce the operational model to achieve the objectives set by tactical formulation.

The mathematical model will be developed with reference to the planning structure illustrated in Fig. 3.4 which is comprised of multiple suppliers, manufacturing plants, warehouses and distribution centers. The discrete manufacturing systems are employed to produce discrete products such as cars and computers. Each manufacturing site contains product assembly lines which can be treated as machines referring to the bottleneck workstations on each line. The SC elements include nodes and arcs (possible transportation routes between pairs of nodes). In addition, the model is developed based on the following assumptions:

1- Manufacturing and shipping lead-times are not considered (It is assumed that shipped and processed goods can be received and produced respectively in the same time period. In other words, both the production and shipping lead times are assumed to be zero).
2- There is only one transportation mode, which is incapacitated.

3- Over-time is an option in case the regular work hours are insufficient to meet the deadline.
4- The transportation and customer service capacities of distribution centers are unlimited and the only restriction is the available warehouse space.
5- If the product produced on an assembly line at the end of previous period is the first product to be produced at the beginning of the current period, no set-up time is required. This is reasonable because the product is already in the same place and hence there is no need to repeat the set-up.
6- Set-up time is shorter than the the associated operation interval.
7- Discrete time representation is used for both tactical and operational models.
8- If the demand cannot be satisfied in every period, the unsatisfied demand is backlogged and a backlog cost is paid until the demand is satisfied.
9- The operational time periods have the same length.

Assumptions 1 and 2 are made to simplify the problem. Transportation capacity is not restricted because we assume transportation can be contracted. These two assumptions can be relaxed by minor modification of the model.

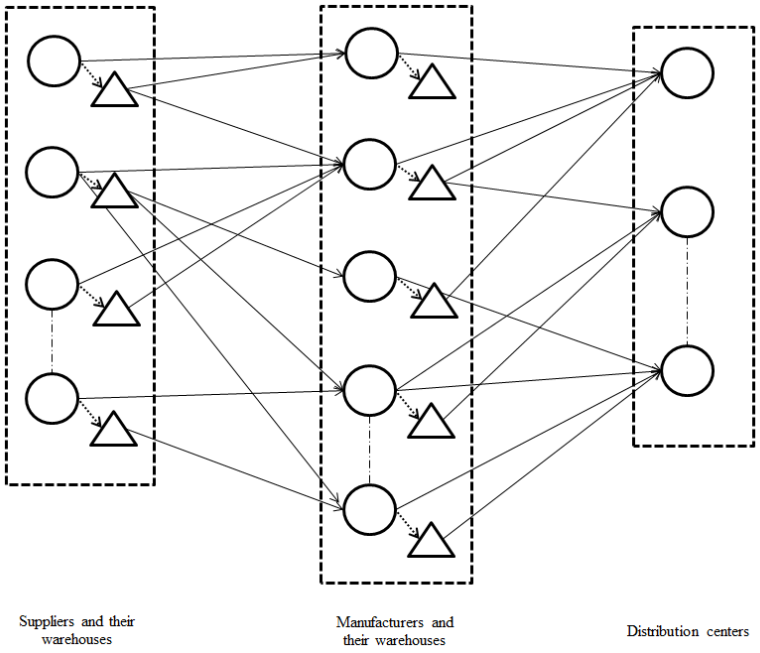

Figure 3.4. Supply chain network structure

3.1.1 First stage supply chain planning model

The first-stage SC planning model is equivalent to our tactical planning problem which is related to decisions on the product mix and quantity on each manufacturing site, the amount of each product family to be shipped from manufacturing sites to distribution centers, the quantities of different components to be supplied from different supplier sites, over-time required and the allocation of suppliers to manufacturing sites.The objective of tactical model is to minimize the total cost comprised of production costs, holding costs, backlog costs, over-time cost, and transportation costs.

3.1.2 Second stage supply chain planning model

In this stage a detailed operational model is built for the first tactical period which is then divided into equal operational periods to assess the feasibility of the proposed tactical plan. The reason for us to consider only the first tactical period here is that it is the only

deterministic period, i.e. we have all the parameters of the system known. Another reason is that by this we reduce the complexity of the model. As we know, customer demand and all the system parameters are uncertain and hence it makes sense to generate a detailed plan for the immediate periods rather than the far later periods which are definitely subjected to significant changes. The tactical decisions made in the first stage serve as input parameters to the second stage. The end-of-period inventory levels in manufacturing sites and distribution centers are sent as input parameters to operational model while the cumulative demand over the operational horizon for products of each product family is equal to the equivalent demand over the first tactical period. The goal of operational planning model is to satisfy the objectives set by the tactical model, to allocate resources to products, and to sequence tasks on assembly lines.

The objective of operational model is then to minimize the same set of costs as considered at the tactical level plus set-up cost, and satisfy the inventory levels set by the tactical program.

3.2 Mathematical Formulation

3.2.1 First stage model

In this section, the tactical planning model is presented. Products, components, and capacities are considered in an aggregated form and there is no notion of a single machine, a single product, or an operator.

Production Limit Constraints

Constraint (3-1) limits the amounts of product families produced in manufacturing sites. UB is an arbitrary large number and LB_{ij} is the minimum production level that has to be

processed if site j is chosen to manufacture product family i. If the product family i is not allocated to site j, X_{ijt} is 0 and the production does not take place.

$$LB_{ij}\, X_{ijt} \leq\ XP_{ijt}\ \leq UB\ \ X_{ijt}\quad \forall i \in\ I_j, j, t \tag{3-1}$$

Material Balance Constraints

Constraints (3-2)-(3-4) express the material balances at the manufacturing sites, distribution centers, and supplier sites. Inventory constraints link all the time periods in the whole tactical horizon. Equation (3-2) states that the amount of product family i at the end of period t is equal to that at the end of period $t-1$ adjusted by any amounts produced and the amounts delivered to distribution centers in period t. Constraint (3-4) establishes similar relationship in supplier sites for purchased component families. Equation (3-3) represents the material balance in distribution centers in a certain time period and contains the backlog variables. If the amount of inventory carried over from previous period plus the amount of products shipped from manufacturing sites cannot satisfy the customer demand at a certain distribution center, a backlog is generated.

$$IPS_{ijt} =\ IPS_{ijt-1} +\ XP_{ijt} -\ \textstyle\sum_{k\in\, k_j} XSM_{ijkt}\quad \forall i\ \in\ I_j, j, k \in\ k_j, t \tag{3-2}$$

$$IPD_{ikt} -\ U_{ikt} =\ IPD_{ikt-1} -\ U_{ikt-1} +\ \textstyle\sum_{j\in j_k} XSM_{ijkt} -\ D_{ikt}\ \forall i\ \in\ I_j, j, k \in\ k_j, t \tag{3-3}$$

$$ICS_{cst} =\ ICS_{cs(t-1)} + XCP_{cst} - \textstyle\sum_{\mathrm{j}\,\epsilon\,\mathrm{J_s}} \mathrm{XSS_{csjt}}\quad \forall c \in\ C_s, s, j \in\ J_s, t \tag{3-4}$$

Constraint (3-3) also ensures that if inventory level is positive, the unfulfilled demand will be zero and vice versa. The unfulfilled demand is backlogged and a cost will be imposed until the backorder is satisfied.

Capacity Constraints

Constraints (3-5)-(3-6) state that the resource utilization cannot exceed the available capacitiesavailable at the suppliers and manufacturing sites. The notion of standard medium time scales is used which means a day, week, or month depending on the supply chain system for which planning model is developed and companies' policies.The lengths of tactical periods are multiples of standard medium time scales. Also, the overall capacity of supplier and manufacturing sites is the sum of the available regular and over-time resources.

We assume that suppliers are capable of providing as many component families as required and the over-time capacity available is unlimited. The idea is that, consumption is charged by a regular price up to a certain level of resource (specified by $XSOT_{st}$) and over that limit overtime cost rate will be imposed.

Constraints (3-7)-(3-8) state that over-time capacity consumed in suppliers and manufacturing sites cannot exceed the available resources.

$$\sum_{i\in I_j} MR_{ij}\ XP_{ijt} \ \leq\ PA_j\, N_t + XMOT_{jt} \quad \forall i \in I_j, j, t \tag{3-5}$$

$$\sum_{c\in C_s} SR_{cs}\ XCP_{cst} \ \leq SA_s\, N_t + XSOT_{st} \quad \forall c \in C_s, S, t \tag{3-6}$$

$$XMOT_{jt} \ \leq\ MOT_j\, N_t \quad \forall j, t \tag{3-7}$$

$$XSOT_{st} \leq\ SOT_s\, N_t \quad \forall s, t \tag{3-8}$$

Storage Constraints

Constraints (3-9)-(3-11) specify the upper limits on the total product families and component families that can be stored in supplier and manufacturing sites as well as distribution centers.

$$\sum_{i \in I_j} \alpha_i \; IPS_{ijt} \; \leq \; JSA_j \quad \forall i \in I_j, j, t \tag{3-9}$$

$$\sum_{i \in I_k} \alpha_i \; IPD_{ikt} \; \leq \; KSA_k \quad \forall k, t \tag{3-10}$$

$$\sum_{c \in C_s} \beta_c \; ICS_{cst} \; \leq \; SSA_s \quad \forall s, t \tag{3-11}$$

Other constraints

The amount of component families required to be shipped from suppliers to a certain manufacturing site is determined via the following constraint, depending on the amounts of correlated product families to be produced in that site.

$$\sum_{s \in S_j} XSS_{csjt} \; = \; \sum_{i \in I_j} \rho_{ci} XP_{ijt} \quad \forall \; c \in \; C_i, j, t \tag{3-12}$$

Objective Function

As mentioned before, the objective of the tactical model is the minimization of the total cost of the whole supply chain network including production costs (*PC*), inventory costs (*IC*), backorder costs (*BC*), transportation costs (*TC*), and over-time costs (*OC*), i.e.,

Min $PC+IC+BC+TC+OC$ (3-13)

where

$$PC = \sum_t \sum_j \sum_{i \in I_j} PI_{ij}\ XP_{ijt} + \sum_t \sum_s \sum_{c \in C_s} CPS_{cs}\ XCP_{cst} \quad (3\text{-}13a)$$

$$IC = \sum_t \sum_j \sum_{i \in I_j} HJ_{ij}\ N_t\ IPS_{ijt} + \sum_t \sum_k \sum_{i \in I_k} HD_{ik}\ N_t\ IPD_{ikt} + \sum_t \sum_s \sum_{c \in C_s} HS_{cs}\ N_t\ ICS_{cst} \quad (3\text{-}13b)$$

$$BC = \sum_t \sum_k \sum_{i \in I_k} UC_{ik}\ UD_{ikt} \quad (3\text{-}13c)$$

$$TC = \sum_t \sum_s \sum_{j \in J_s} \sum_{c \in C_s} TSS_{csj}\ XSS_{csjt} + \sum_t \sum_j \sum_{k \in K_j} \sum_{i \in I_j} TSD_{ijk}\ XSM_{ijkt} \quad (3\text{-}13d)$$

$$OC = \sum_t \sum_s CSOT_c\ XSOT_{st} + \sum_t \sum_j CMOT_j\ XMOT_{jt} \quad (3\text{-}13e)$$

3.2.2 Second-stage model

In this stage we will examine the feasibility of the tactical plan and its proximity to an optimal solution. For this reason, the output of tactical plan will be communicated to the operational level model through a set of parameters. For simplicity, suppliers are excluded from the operational level planning which should not affect the generality of the model because they can be added into the formulation.

The first stage decision variables IPS_{ij1}, XX_{ij1}, and IPD_{ik1} are now fixed at the values determined in the tactical model for the first tactical period. The iterative approach that attempts to achieve the converge of tactical and operational solutions and update the first-stage decision variables will be discussed in the next chapter.

Production Limit Constraints

Constraint (3-14) limits the amounts of products manufactured on the assembly lines in the manufacturing sites. U is an arbitrary large number and L_{il_jj} is the minimum production level that has to be processed if assembly line l_j in site j is chosen to manufacture product family i. If the product family i is not assigned to site j, X_{il_jjt} is 0 and production does not occur.

$$LB_{il_jj}\, X_{il_jjt} \leq \sum_{p_i} XSP_{p_i il_j jt} \leq U\, X_{il_jjt} \;\; \forall \; i \in I_{l_j}, j, t \tag{3-14}$$

Production Constraints

Constraint (3-15) reproduces the first-stage production assignment decisions in the second-stage model. If product family i is not chosen to be produced in manufacturing site j in the first tactical period, we cannot assign any product in family i to any assembly line in site j over the operational horizon.

$$\sum_t \sum_{l_j} X_{il_jjt} \leq M\, XX_{ij1} \;\; \forall \; i \in I_{l_j}, j \tag{3-15}$$

Material Balance Constraints

Constraints (3-16) and (3-17) express the mass balances at the manufacturing sites and distribution centers. Equation (3-16) represents the material balance in distribution centers in a certain time period and contains the backlog variables. A backlog is generated in case the

amount of inventory carried over from previous period and the amount of products shipped from manufacturing sites cannot meet the customer demand at a specific distribution center. The difference between this constraint and equation (3-3) is that products here are not in aggregated form. Equation (3-17) states the amount of product p_i of product family i at the end of short-term period t is equal to that at the end of period $t-1$ adjusted by any amounts produced and the amounts delivered to distribution centers in period t.

$$IPD_{p_i ikt} - UD_{p_i ikt} = IPD_{p_i ik(t-1)} - UD_{p_i ik(t-1)} + \sum_{j|i\in I_{l_j}, j\in J_k} SSM_{p_i ijk(t)}\ D_{p_i ikt}$$

$$\forall\ p_i, i, k, t \qquad (3\text{-}16)$$

$$IP_{p_i ijt} = IP_{p_i ij(t-1)} + \sum_{l_j} XSP_{p_i il_j j} - \sum_{k\in k_j} SSM_{p_i ijkt} \quad \forall i \in I_{l_j}, l_j, j, t \qquad (3\text{-}17)$$

Also, in the same manner as in the tactical model, Constraint (3-16) ensures that if inventory level is positive, the unfulfilled demand will be zero and vice versa. The unfulfilled demand is backlogged and a cost will be imposed until the backorder is satisfied.

Capacity Constraints

Constraint (3-18) states that the resource utilization cannot exceed the available capacity. The capacity balance assumes the same form as constraint (3-5) in the tactical model except for the set-up time term, which stands for the reduction in the resource availability. A change-over will be required if the production system switches from product family i to i' which will consume the available capacity. The operational constraints and set-up cost will ensure the number of set-ups in the operational planning horizon will be minimized.

Constraints (3-19) states that the maximum amount of over-time capacity is utilized in a manufacturing site cannot exceed the available resources.

$$\sum_i \sum_{p_i} MR_{ij} \; XSP_{p_i i l_j j t} \; \leq \; SA_{l_j j} \left(1 - \left(\sum_i ST_{i l_j j} \; XSU_{i l_j j t} \; \right)\right) + XMOT_{l_j j t}$$

$$\forall i \in I_{l_j}, l_j, j, t \tag{3-18}$$

$$XMOT_{l_j j t} \; \leq \; MOT_{l_j j t} \quad \forall l_j, j, t \tag{3-19}$$

Storage Constraints

Constraints (3-20) and (3-21) establish upper limits on the total products that can be stored in manufacturing sites and distribution centers.

$$\sum_{i \in I_j} \sum_{p_i} \propto_i \; IP_{p_i i j t} \; \leq \; JSA_j \quad \forall j, t \tag{3-20}$$

$$\sum_i \sum_{p_i} \propto_i \; IPD_{p_i i k t} \; \leq \; KSA_{kt} \quad \forall \; k, t \tag{3-21}$$

Linking Constraints

One of the purposes of the operation level planning is to obtain the feasible short-term schedule that satisfies the tactical planning results. As such, the requirements imposed by the tactical model should be considered in formulating the operation lelve model. These requirements are represented in the form of required inventory level at the end of operational horizon which is the end of first tactical period. Any deficiency from this stock level will result in a dummy cost which is applied to enforce the outcome of operational level to be as

close as possible to tactical program. Slack variables SIP_{ij} and SID_{ik} are introduced to represent the amount of discrepencies and are penalized in the objective function.

$$\sum_{p_i} IP_{p_i ij(t=T)} \geq IPS_{ij1} - SIP_{ij} \quad \forall i \in I_j, j, t \tag{3-22}$$

$$\sum_{p_i} IPD_{p_i ik(t=T)} \geq IPD_{ik1} - SID_{ik} \quad \forall i, k \tag{3-23}$$

Operational Planning Constraints

Constraints (3-24) to (3-37) express the major difference between our tactical and operational models. The operational planning resembles scheduling type problems where the resource utilization is planned in detail. Between the production periods of two products from different product families, a set-up time is required. If the last product produced in period t is the first product to be manufactured in period $t+1$, no set-up is required otherwise a changeover will be imposed. Different product families can be assigned to an operational period. There has to be at least one product family assigned to the beginning of a period and one to the end of the period (constraints (3-24) and (3-25)). The set-up time of a product family, if required, has to fall inside the same interval as the product is produced. Constraints (3-36) and (3-37) specify the time when a set up is required for a specific product family.

If any production take place on line l_j of site j in time period t (constraint (3-30), constraints (3-26) and (3-27) are ignored. Constraints (3-28) and (3-29) specify the values of L_{il_jjt}, U_{il_jjt} and X_{il_jjt} . Otherwise, the last product family produced on line l_j of site j in time period $t-1$ specifies the values of L_{il_jjt}, U_{il_jjt} and X_{il_jjt} based on constraints (3-26) and (3-27).

Based on the values of L_{il_jjt}, U_{il_jjt} and X_{il_jjt} determined by constraints (3-24) to (3-30), the need for a set up time is evaluated by constraints (3-31) to (3-33).

$$\sum_i L_{il_jjt} = 1 \tag{3-24}$$

$$\sum_i U_{il_jjt} = 1 \tag{3-25}$$

$$L_{il_jjt} + U_{il_jjt} \leq 2\, U_{il_jj(t-1)} + MY_{l_jjt} \tag{3-26}$$

$$L_{il_jjt} + U_{il_jjt} \geq 2\, U_{il_jj(t-1)} - MY_{l_jjt} \tag{3-27}$$

$$L_{il_jjt} + U_{il_jjt} \leq 2\, X_{il_jjt} + M\left(1 - Y_{l_jjt}\right) \tag{3-28}$$

$$L_{il_jjt} + U_{il_jjt} \leq 2 - \left(\frac{\sum_i X_{il_jjt} - 1}{M}\right) + M\left(1 - Y_{l_jjt}\right) \tag{3-29}$$

$$\frac{\sum_i X_{il_jjt} - 0.5}{M} \leq Y_{l_jjt} \leq \sum_i X_{il_jjt} \tag{3-30}$$

$$XSU_{il_jjt} \leq X_{il_jjt} \tag{3-31}$$

$$XSU_{il_jjt} \geq \frac{3 - (X_{il_jjt} + L_{il_jjt} + U_{il_jj(t-1)})}{3} - M\left(1 - X_{il_jjt}\right) \tag{3-32}$$

$$XSU_{il_jjt} \leq 3 - \left(X_{il_jjt} + L_{il_jjt} + U_{il_jj(t-1)}\right) + M\left(1 - X_{il_jjt}\right) \tag{3-33}$$

The only remaining matter to be addressed is the first operational period. Since scheduling constraints deal with current and previous periods, this can raise an issue for the first short-term time period. The issue is that, as constraints (3-26), (3-27), (3-32), and (3-33) contain some variables from period $t-1$, in period one, we have to refer to period zero that however does not exist. This can be handled by adding a dummy product family and a dummy period called product zero and period zero. The constraints (3-34) to (3-36) show how the issue is resolved.

$$L_{0l_jj0} = 1 \quad (3\text{-}34)$$

$$U_{0l_jj0} = 1 \quad (3\text{-}35)$$

$$\sum_t X_{0l_jjt} = 0 \quad (3\text{-}36)$$

Constraint (3-40) states that product family zero cannot be produced in any period.

Objective Function

In the operational model, we ateempt to minimize the total cost of in the SC manufacturing sites, distribution centers and the transportation arcs linking these nodes. The cost function consists of production, set-up, holding, transportation, unfulfilled demand, and over-time cost components (PC_2, SC_2, HC_2, TC_2, UC_2, and OC_2 respectively), as well as the dummy costs (DC) representing the penalty to the deviations from the tactical-level targets. The objective function is written as

$$\text{Min } PC_2 + SC_2 + IC_2 + TC_2 + UC_2 + DC_2 + OC_2 \quad (3\text{-}37)$$

Where

$$PC_2 = \sum_t \sum_j \sum_{l_j} \sum_i \sum_{p_i} PI_{p_i il_j j} \; XSP_{p_i il_j jt} \quad (3\text{-}37a)$$

$$SC_2 = \sum_k \sum_j \sum_{l_j} \sum_i SUC_{il_j j} \; XSU_{il_j jk} \quad (3\text{-}37b)$$

$$IC_2 = \sum_t \sum_j \sum_i \sum_{p_i} IP_{p_i ijt} \; HJ_{ijt} + \sum_t \sum_k \sum_i \sum_{p_i} IPD_{p_i ikt} \; HD_{ikt} \quad (3\text{-}37c)$$

$$TC_2 = \sum_t \sum_k \sum_j \sum_i \sum_{p_i} TSD_{ijk} \; SSM_{p_i ijkt} \quad (3\text{-}37d)$$

$$UC_2 = \sum_t \sum_k \sum_i \sum_{p_i} UD_{p_i ikt} \; UDC_{ik} \tag{3-37e}$$

$$DC_2 = \sum_k \sum_i SID_{ik} \; SDC_{ik} + \sum_j \sum_i SIP_{ij} \; SIC_{ij} \tag{3-37f}$$

$$OC_2 = \sum_t \sum_j \sum_{l_j} COT_{l_j j} \, XMOT_{l_j jt} \tag{3-37g}$$

The above tactic and operation level models are the mathematical expressions of the integrated deterministic dynamic SC planning problem. The constraints in the models set limitations according to the operation requirements and capabilities of the SC elements. Solutions to the SC planning problems can be obtained by solving these models.

Chapter 4. Iterative Solution Algorithm

In the previous chapter, the integrated supply chain planning problem was decomposed into two mixed integer linear programming models which are called tactical and operational programs. The solution of tactical program is the input to the operational model. The details considered in the operational level may cause the infeasibility of targets set by the tactical plans.

To handle this issue, an iterative method is employed and the notion of corrective cuts is used. Depending on the result of each iteration, some cuts are generated in order to make solutions of the two planning levels to converge.

The computational efficiency of iterative method mostly depends on two main factors:

1- How fast each of the two models can be solved, and
2- How many iterations are required to obtain the optimal solution.

Therefore, generating strong cuts is of great importance. The solution procedure should ensure that the tactical solution found infeasible in the operational program does not happen again. Not only those solutions but also any subset of those solutions should not occur. As it is discussed in the following, the value of X_{ij1} found in tactical solution is sent to operational model as parameter. If $SET_1^r = \{(i,j)|X_{ij1} = 1\}$ leads to an infeasible or non-optimal solution in operational problem, it means that any subset of SET_1^r will lead to the same situation.

As was discussed before, model predictive control that is similar to the rolling horizon concept is utilized to enhance the reactivity and pro-activity of the proposed planning

models. It computes a trajectory of future manipulated variables to optimize the future behaviour of supply chain system. In this procedure, the first period is quite important and the solution of the remaining time periods can be updated while the uncertainty unveils over time. The planning process is repeated every period and the tactical formulation is solved for the tactical horizon although only the decisions for the current period are taken into account.

The proposed solution algorithm is developed for the above mentioned mathematical models. However, with some modifications, it can be used for many other hierarchical SC planning problems. The tactical model is solved as a Mixed Integer Linear Programming (MILP) problem which could result in the following cases:

(1) there are some unfulfilled demands for the first period which means that the tactical plan for the current period could not satisfy the aggregated customer demand; and
(2) the tactical solution meets or exceeds customers' demand.

The above two cases further lead to four possible scenarios:

Scenario 1. there are some unfulfilled demands in both the tactical and operational models;

Scenario 2. there is no unfulfilled demand in the tactical model but there are unfulfilled demand sin the operational model;

Scenario 3. there are no unfulfilled demands in both the tactical and operational models; and

Scenario 4. there are some unfulfilled demands in the tactical model but no unfulfilled demand in the operational model.

As Scenario 4 is impossible, only the first three scenairos will be elaborated as follows.

The first scenario happens when there is no backorder in the first tactical period but operational program does not satisfy some customer orders. This could happen due to capacity restriction or overestimation. If all the available capacity is exhausted by the tactical plan, the operational solution is optimum. The tactical model is modified by restricting the production capacity and applying the operational solution to the first tactical period. Resources are limited by adding the following inequalities to the set of tactical constraints:

$$\sum_{i\in I_j} MR_{ij}\, XP_{ijt} \leq \left[PA_j\, N_t + \, XMOT_{jt}\right] * \, CrrCo_{jr}\, \forall j, t \, \neq 1 \qquad (4\text{-}1)$$

where:

$$CrrCo_{jr} = \frac{\sum_{p_{i'},i,l_{j'},j,t} MR_{ijt}\, XSP_{p_i i l_j jt}}{\sum_{p_{i'},i,l_{j'},j,t} MR_{ijt}\, XSP_{p_i i l_j jt} + \sum_{p_{i'},i,l_{j'},j,t} ST_{ijt}\, SA_{l_{j'}j}\, XSU_{i l_j jt}} \qquad (4\text{-}2)$$

If there is any left-over manufacturing capacity that can be employed, the set of constraints (4-3) to (4-5) and inequality (4-1) will be added to the constraints of tactical formulation with a minor modification in calculating $CrrCo_j$.

$$\sum_{(i,j)\in SET_0^r} X_{ij1} + \, X_{i'j'1} \, \geq 1 \quad \forall (i', j') \in SET_1^r \qquad (4\text{-}3)$$

$$\sum_{i\in I_j} MR_{ij}\, XP_{ij} + \, SETUP_j^r \, \leq \, PA_j N_t + XMOT_{jt} + \, M(q_r^+) \quad \forall j, r, t = 1 \qquad (4\text{-}4)$$

$$\sum_{(i,j)\in SET_0^r} X_{ij1} = \, q_r^+ \quad \forall r \qquad (4\text{-}5)$$

where:

$$SET_1^r = \left\{(i,j) | X_{ij1} = 1\right\} \qquad (4\text{-}6)$$

$$SET_0^r = \{(i,j)|X_{ij1} = 0\} \quad (4\text{-}7)$$

$$SETUP_j^r = \sum_t \sum_{lj} \sum_i 0.2 * SA_{l_j j} * ST_{il_j j} * XSU_{il_j jt} \qquad \forall j, r \quad (4\text{-}8)$$

Constraint (4-3) excludes any subset of assignments made at iteration $r-1$ from the solution space of current iteration. During the solution at the lower level at iteration $r-1$, the optimal solution obtained at tactical level and all its possible subsets will be considered due to the inequality (3-19). In scenario one, not only the tactical solution but also all its subsets are not able to satisfy customers demand. Hence, we do not need to further consider these subsets as potential solutions in later iterations and they can be excluded from further investigation at the tactical program. The worst case would be choosing the same assignment as in iteration $r-1$ in later iterations and concluding that customer demand cannot be fully met.

$CrrCo_{jr}$ in this step is computed as follows:

$If\ (q_r^+ = 0\ )\{$

$$CrrCo_{jr} = CrrCo_{j(r-1)} * \frac{\sum_{p_i,i,l_j,j,t} MR_{ijt}\ XSP_{p_i il_j jt}}{\sum_{p_i,i,l_j,j,t} MR_{ijt}\ XSP_{p_i il_j jt} + \sum_{p_i,i,l_j,j,t} ST_{ijt}\ SA_{l_j j}\ XSU_{il_j jt}}$$

$\}\ else\{$

$$CrrCo_{jr} = \frac{\sum_{p_i,i,l_j,j,t} MR_{ijt}\ XSP_{p_i il_j jt}}{\sum_{p_i,i,l_j,j,t} MR_{ijt}\ XSP_{p_i il_j jt} + \sum_{p_i,i,l_j,j,t} ST_{ijt}\ SA_{l_j j}\ XSU_{il_j jt}}$$

$\}$ (4-9)

In the second scenario, there is no backorder in the first tactical period nor in the last operational period. However, further investigation is required to find out if all the targets set

by tactical solution are reached. The only remaining objective that we are interested in is inventory level at the manufacturing sites and distribution centers. Therefore the following check point is essential:

$$\exists\, j, i \in I_j \,|SIP_{ij} > 0 \; or \; \exists\, k, i \in I_k \,|SID_{ik} > 0 \tag{4-10}$$

if the result of the previous logical statement is false, optimal solution has reached; otherwise, the same steps as in scenario one need to be taken. Fig.4.1 to 4.4 illustrates the steps of algorithm in more detail.

If there are backorders in both the first tactical period and at the end of operational horizon, the third scenario is triggered. In this case, optimal solution of operational model is further analyzed and the aggregated production of tactical and operational level solutions are compared. If the absolute value of the difference between the tactical and aggregated operational production levels is higher than a predetermined constant, this can be referred to as an overestimated capacity in tactical model. As a result, capacity is updated and the tactical model is reformulated using the new parameters. If the absolute value of differences is lower than a certain constant, the operational solution is the best schedule for the current period, backorders are allowed, and shortage is due to capacity limitation. The following expression states the above fact:

$$\left|\sum_j XP_{ij1} - \sum_{p_i,l_j,j,t} XSP_{p_i l_j jt}\right| \leq \varepsilon \quad \forall i \tag{4-11}$$

If the result of left-hand-side expression is greater than ε, integer and capacity cuts will be generated to ensure that the later iterations will enhance the quality of integrated solution.

In this approach, a situation where tactical model is not able to satisfy customer demands but operational model is able to do so is impossible. The feasible solution region of operational model is a subset of feasible area of the tactical problem. This can be shown by looking at the structure of two sub-models and how the iterative algorithm is launched. For the first iteration of the solution procedure, the tactical model is solved. However, in the operational model, we are restricting the feasible solution area by introducing set-up times as well as targets set by the tactical problem. In any step of the algorithm, we force the tactical model to disregard the previous solution for the current time period and if this is not successful, we impose the feasible region of operational model on tactical formulation. Under any circumstances, the capacity of the tactical model in the first time period will not be lower than that of the operational model. Therefore it is guaranteed that, if the tactical model cannot make a demand on-time, the operational model will not make it neither. Simply speaking, the tactical model represents a relaxed operational problem at higher level.

The efficiency of iterative approached is demonstrated in chapter 7 through three case studies.

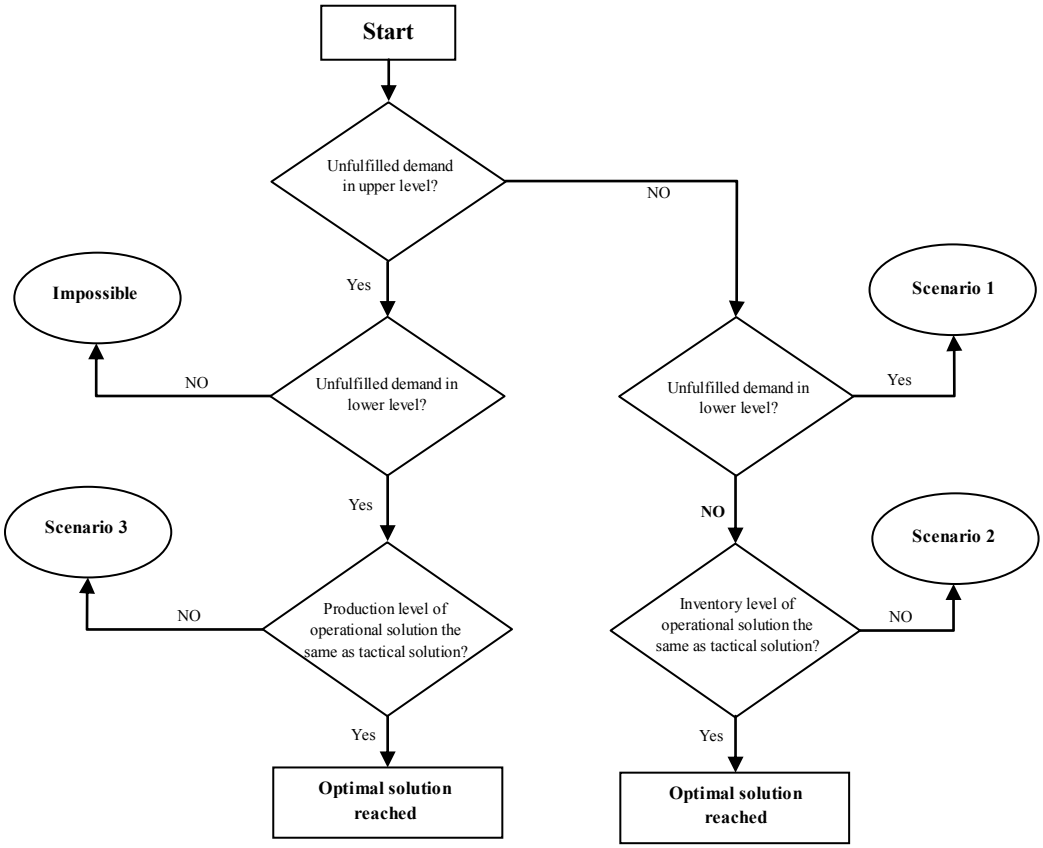

Figure 4.1. Overall framework of the iterative algorithm

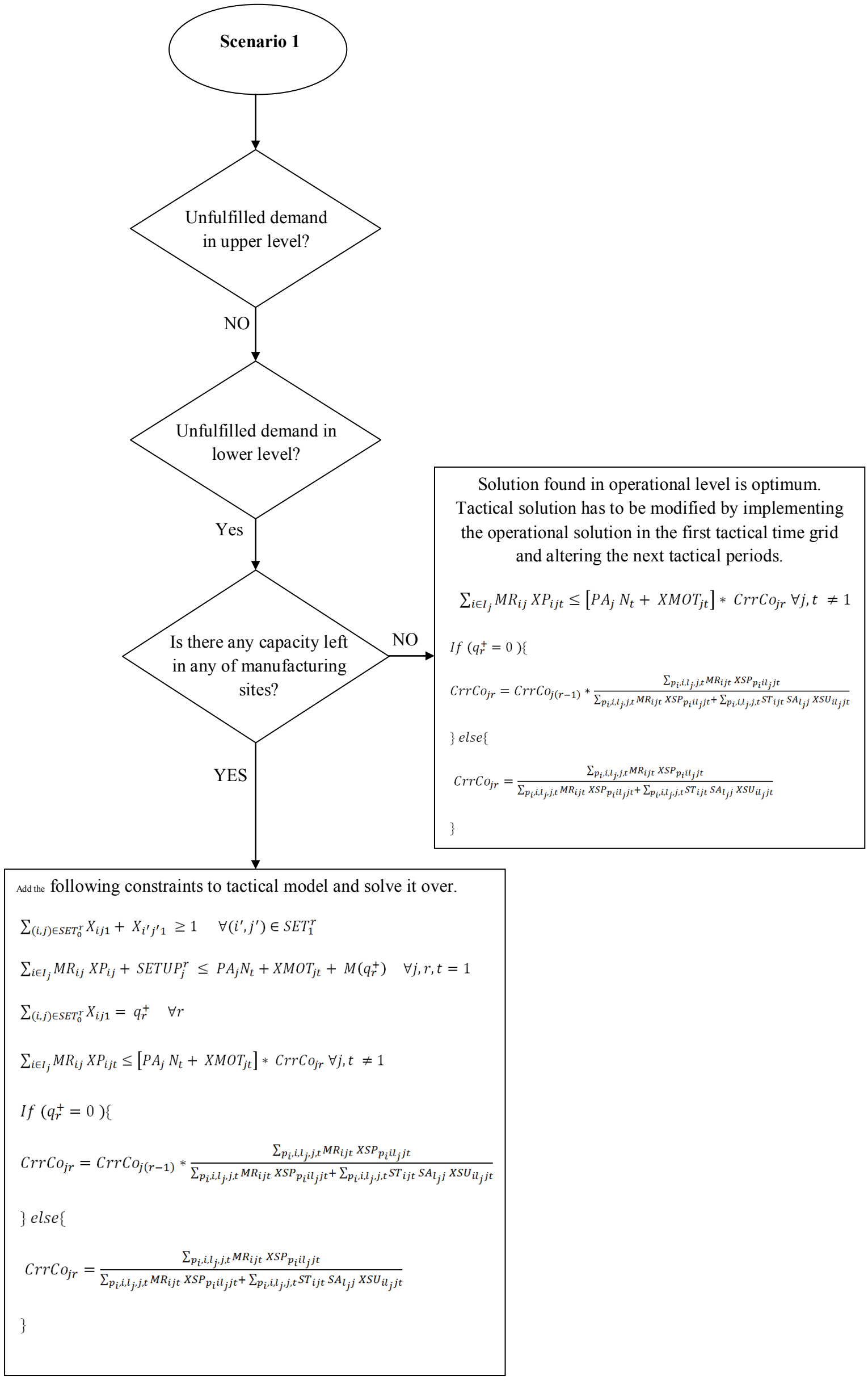

Figure 4.2. Scenario 1 of the iterative algorithm

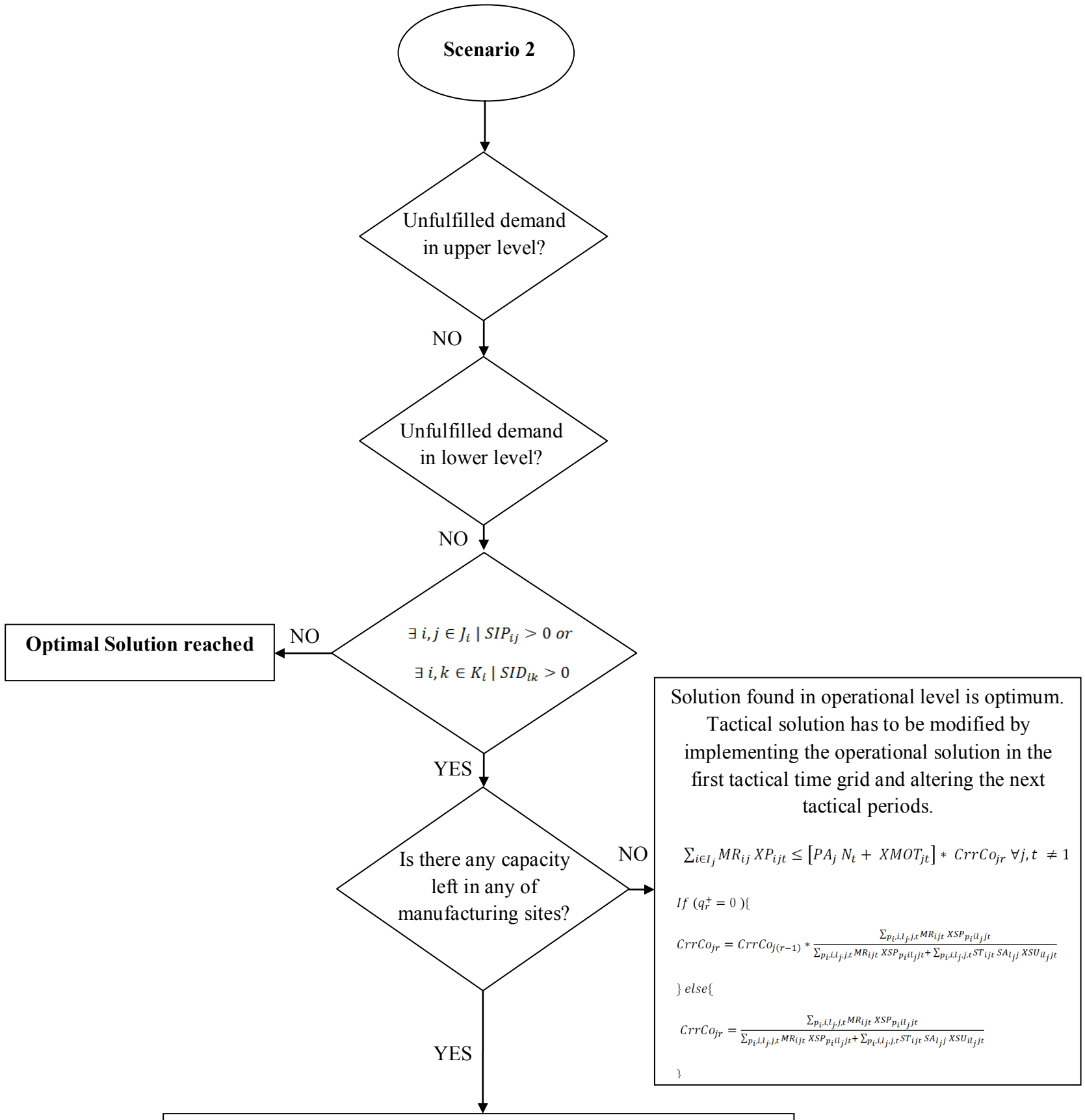

Figure 4.3. Scenario 2 of the iterative algorithm

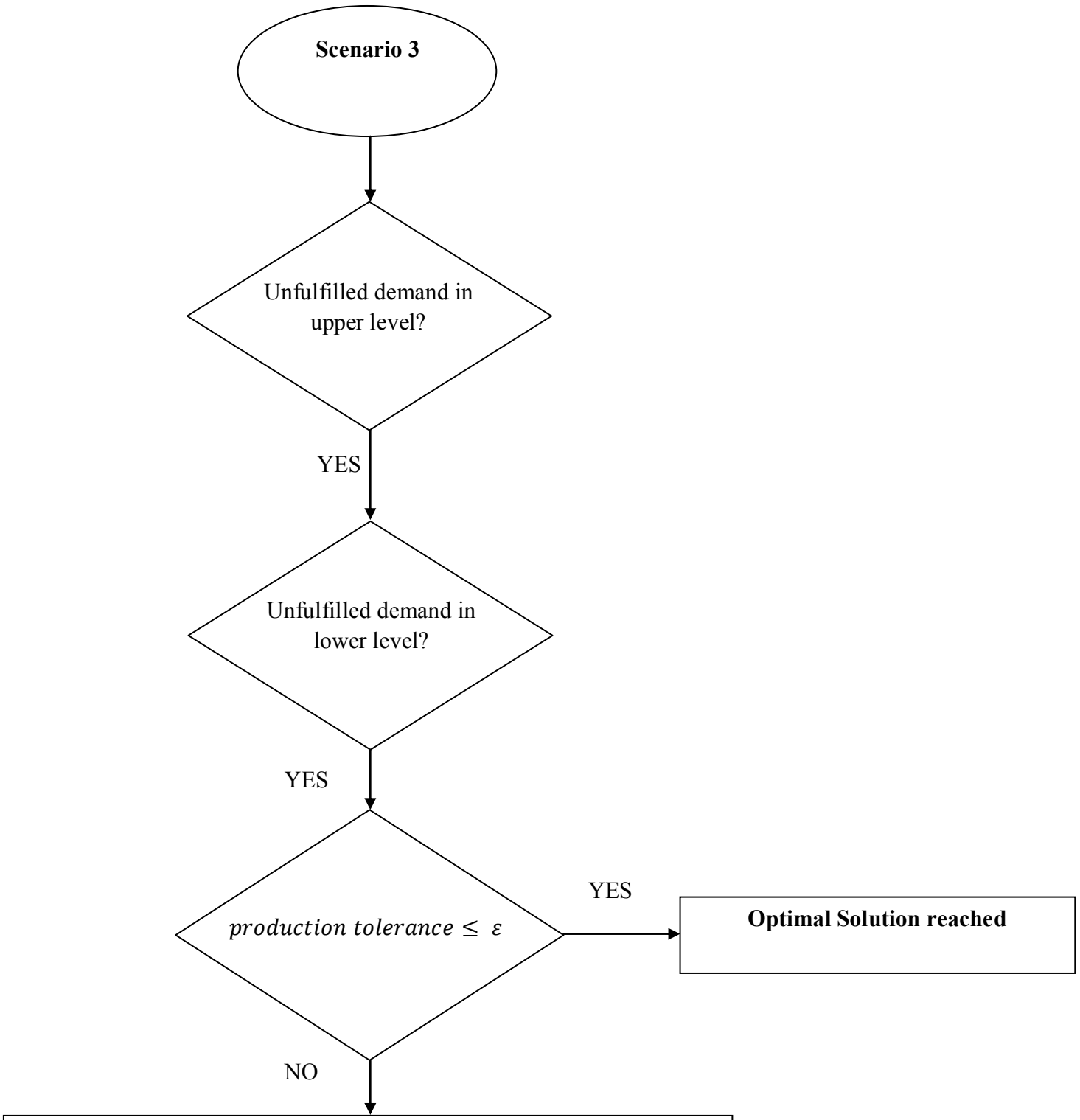

Figure 4.4. Scenario 3 of the iterative algorithm

Chapter 5. Integrated Multi-Stage Scenario-Based Stochastic Supply Chain Planning Model

We have discussed that a supply chain is a stochastic dynamic system whose behaviour changes over time and the future state of the system cannot be predicted with certainty. As a result, it is essential to capture and monitor system dynamics and any disturbances affecting the overall performance of the system elements. Model Predictive control will be employed in this study to address the dynamical behaviour of the system and to keep the overall performance at a desired level in terms of reactivity and proactivity. The overall framework of Model Predictive Control was discussed in Chapter 2. In this chapter, we discuss the control module and propose a multi-stage scenario-based stochastic (MSSBS) SC planning model to be used in the MPC module.

The proposed framework is depicted in Figs. 5.1 and 5.2. The strategy is to incorporate a multi-stage stochastic mixed integer linear programming model within the control framework to account for proactivity provided by stochastic program and reactivity rendered by MPC. The strategy introduced in this study can be described as follows:

(1) The supply chain process is disturbed by all those external and internal changes such as fluctuation in prices, demand, material availability and cost, processing times, equipment breakdowns, and data accuracy. The information regarding these interruptive events as well as current state of supply chain system are captured and sent as input to the control module. The "current state" of the SC system is specified by the inventory level, backorders, available capacity, costs in SC nodes, and any other parameters of the model.

(2) A supervisory filter scans the input data and decides upon its severity and impact level on the current SC state and plan. This is a managerial decision to specify how critical a disturbance is. Practitioners and managers play the role of supervisiory filter in our proposed planning system.

(3) The control module contains a forecasting sub-system and a control model. The forecasting sub-system is responsible for predicting the reference trajectory and uncertain parameters of the control model using historical data and collected information (disturbances). For the uncertain parameters in the control model, scenarios are generated forecasting module.

(4) Once scenarios and reference trajectories are determined by the forecasting module, the control model is updated, the integrated tactical operational model is resolved and the set of future control variables (manipulated variables) is calculated. The control model is composed of a stochastic tactical supply chain planning model and a deterministic operational problem. The iterative algorithm presented in chapter 4 is employed to calculate the manipulated variables.

(5) It will be further discussed later in this chapter that the output of control model can be classified into two types of decisions: first stage and second stage. The first stage or here-and-now decisions are determined prior to the moment when uncertainty is detected. On the other hand, the recourse or wait-and-see decisions are determined when the uncertainty unfolds in the period when its associated decision is implemented. The control or manipulated variables only include the first stage decisions that are associated to the following period.

(6) The strategy is repeated continuously every period or any time a captured interruption passes the supervisory filter. As new information is available, it is captured and analyzed by supervisory filter. If it is recognized as a critical issue, the forecasting module is updated and a new set of manipulated variables are generated by the control algorithm and the horizon advances by one period-step.

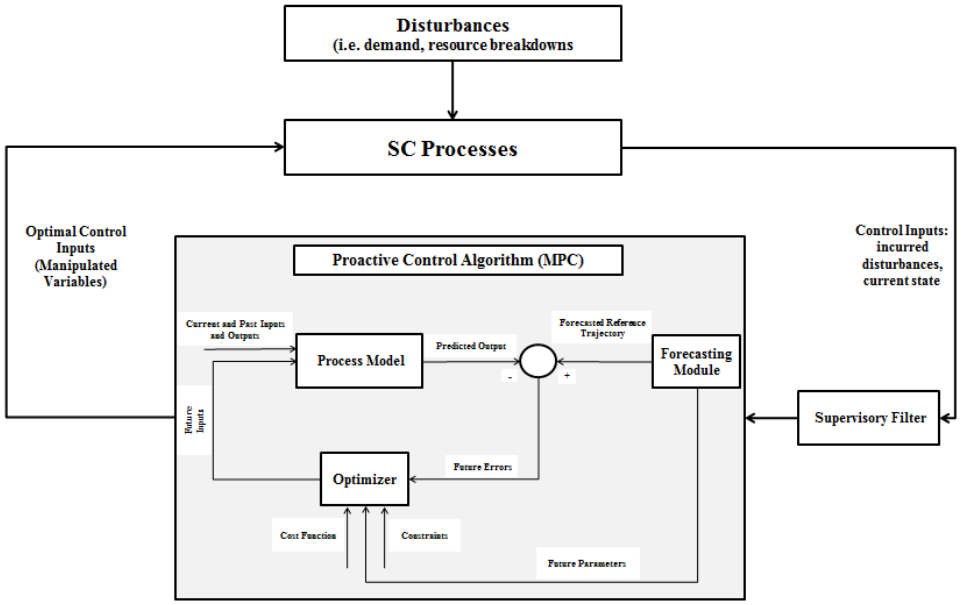

Figure 5.1. Overall framework of proposed reactive-proactive supply chain planning model

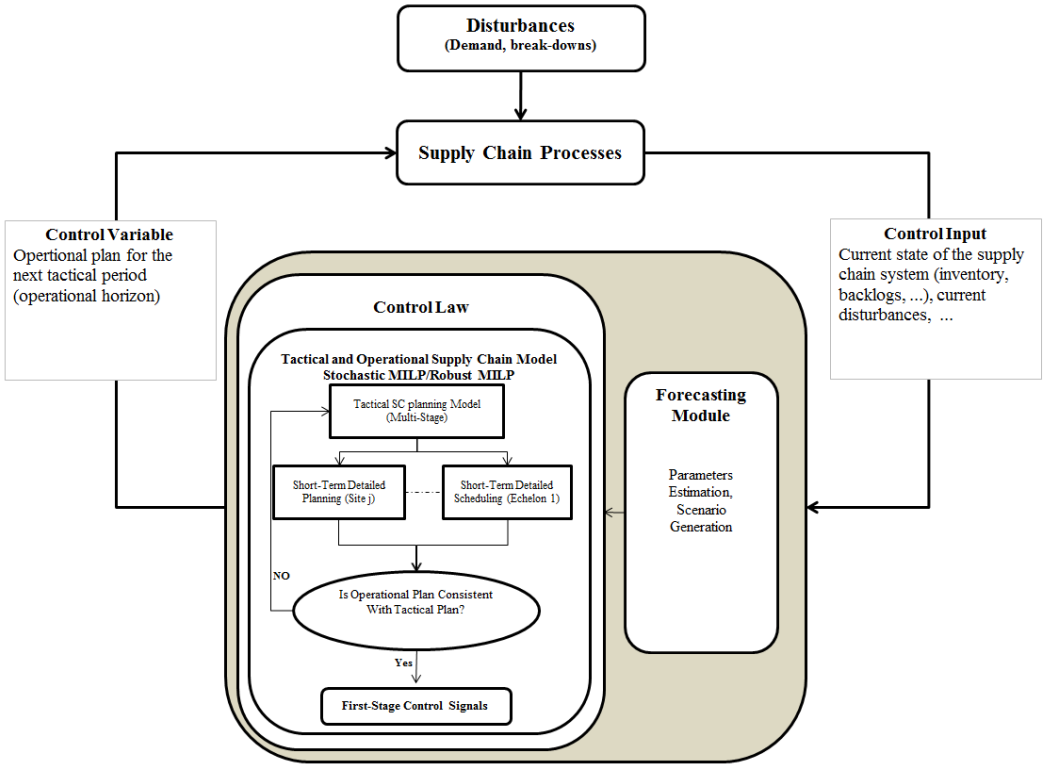

Figure 5.2. Multi-stage stochastic tactical planning model integrated with the operation-level planning problem to play the role of controller in proposed framework

As was already discussed, the control model is a multi-stage stochastic program. We will start with a brief review of stochastic programming and mathematically represent the general models of two-stage and multi-stage stochastic linear programs. We first form the basic two-stage stochastic linear program with fixed recourse as follows.

$$\min z = c^T x + E_\xi [\min q(\omega)^T y(\omega)]$$

$$s.t. Ax = b,$$

$$T(\omega)x + Wy(\omega) = h(\omega),$$

$$x \geq 0, y(\omega) \geq 0 \qquad (5\text{-}1)$$

where c is a known vector in R^{n_1}, b a known vector in R^{m_1}, A and W are known matrices of size $m_1 * n_1$ and $m_2 * n_2$, respectively, and W is called the *recourse matrix,* which we assume as fixed. This allows us to characterize the feasibility region in a convenient manner for computation. If W is not fixed, it will be difficult to do so. x is the vector of first-stage decisions and y is the vector of second-stage decisions. $Ax = b$ composes the deterministic set of constraints while $T(\omega)x + Wy(\omega) = h(\omega)$ constructs the stochastic set of constraints.

For each ω, $T(\omega)$ is $m_2 * n_1, q(\omega) \in R^{n_2} and\ h(\omega) \in R^{m_2}$. Piecing together the stochastic components of the problem, we obtain a vector

$$\xi^T(\omega) = (q(\omega)^T, h(\omega)^T, T_{1.}(\omega), \dots, T_{m_2.}(\omega))$$

with $N = n_2 + m_2 + (m_2 * n_1)$ components, where $T_{i.}(\omega)$ is the ith row of the *technology matrix* $T(\omega)$. E_ξ represents the mathematical expectation with respect to ξ. Let also $\Xi \subseteq R^N$ be the support of ξ, i.e., the smallest closed subset in R^N such that $P\{\xi \in \Xi\} = 1$. The constraints are assumed to almost certainly hold.

Problem (5.1) is equivalent to the deterministic equivalent program:

$$\min z = C^T + Q(x)$$

$$Ax = b,$$

$$x \geq 0$$

$$Q(x) = E_{\xi} Q\big(x, \xi(\omega)\big)$$

$$Q\big(x, \xi(\omega)\big) = \min_{y} \{\, q(\omega)^T\, y(\omega) |\, Wy(\omega) = h(\omega) - T(\omega)x, y \geq 0\}$$

(5-2)

This representation clearly illustrates the sequence of events in the recourse problem. First-stage decisions x are made in the presence of uncertainty about future realizations of ξ. In the second stage, the actual value of ξ becomes known and some corrective actions or recourse decisions y can be made. First-stage decisions are, however, chosen by taking their future effects into account. These future effects are measured by the recourse function, $Q(x)$, which computes the expected value of making decision x.

The difficulty inherent in stochastic programming clearly lies in the computational burden of computing $Q(x)$, for all x in (5-2).

The previous model concerned stochastic programs with two stages. Many practical decision problems, however, involve a sequence of decisions that react to outcomes that evolve over time. The multistage stochastic linear program with fixed recourse then takes the following form.

$$\min z = c^1 x^1 + E_{\xi^2}\big[\min c^2(\omega) x^2(\omega^2) + \ \ldots + \ E_{\xi^H}[\min c^H(\omega)\, x^H(\omega^H)] \ldots\big]$$

$$s.t.$$

$$W^1 x^1 = h^1,$$

$$T^1(\omega)x^1 + W^2 x^2(\omega^2) = h^2(\omega),$$

....

$$T^{H-1}(\omega)x^{H-1}(\omega^{H-1}) + W^H x^H(\omega^H) = h^H(\omega),$$

$$x^1 \geq 0;\ x^t(\omega^t) \geq 0, t = 2, \dots, H;$$

(5-3)

where c^1 is a known vector in R^{n_1}, h^1 is a known vector in R^{m_1}, $R^{m_1}, \xi^t(\omega)^T = (c^t(\omega)^T, h^t(\omega)^T, T_{1.}^{t-1}(\omega), \dots, T_{m_t.}^{t-1}(\omega))$ is a random N_t-vector defined on (Ω, Σ^t, P) (where $\Sigma^t \subset \Sigma^{t+1}$) for all $t = 2, \dots, H$, and each W^t is a known $m_t * n_t$ matrix. The decisions reflected by x values depend on the history up to time t, which is indicated by ω^t. We also assume that Ξ^t is the support of ξ^t.

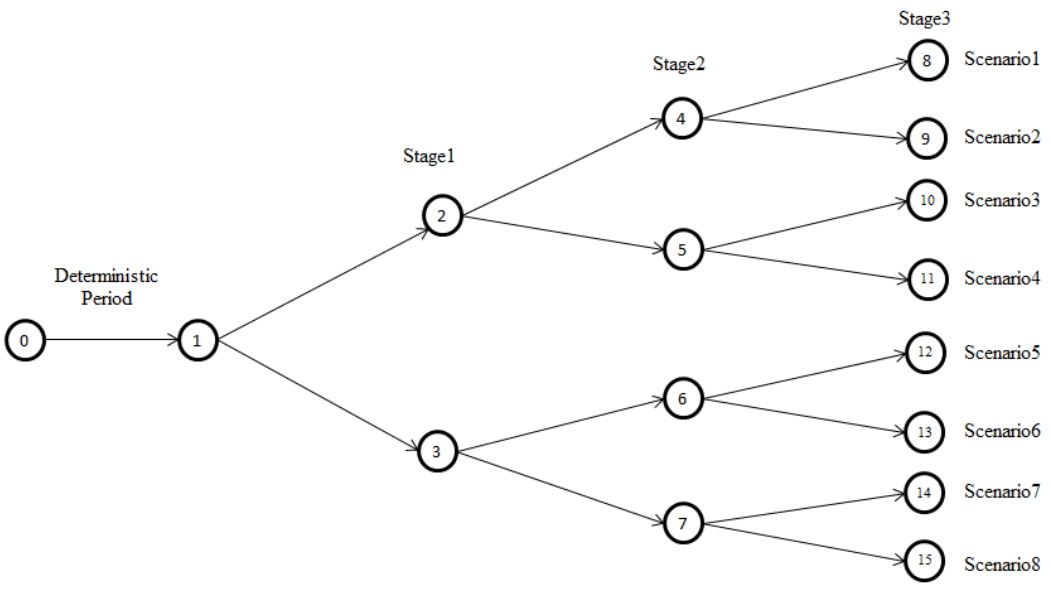

Figure 5.3. Multi-stage scenario tree

In this work, customer demand is assumed an uncertain parameter and its uncertainty is represented by a set of scenarios with associated possibilities. The concept of scenario tree is exploited. We assume that, over the decision stages, n possible scenarios may occur. Scenarios are indicated by an index $z = 1, \dots, Z$, which represents a collection of outcomes that have common characteristics in a specific model. It is required to assign probabilities for each scenario z. The scenario tree contains branches corresponding to different realizations of the random demand. Decision variables are defined with reference to the branches on the scenario tree. Here, the full horizon scenario modelling approach is applied to the multistage problem without considering the history of the process. The difficulty is that, when we have split up the scenarios, we may have lost nonanticipativity of the decisions because they include knowledge of the outcomes up to the end of horizon. A set of nonanticipativity constraints is added to ease this difficulty. This way, the stochastic model can be reformulated as a set of nonanticipativity constraints and a set of isolated problems for each scenario z. This simple additional set of equlities makes it relatively easy to move from a deterministic model to a stochastic model of the same problem. This ease of conversion is quite useful in modelling languages and simplifies development of the iterative and branch-and-fix algorithms in Cplex-Java which are to be utilized to program all the models and algorithms developed in this thesis.

In the following section, the multi-stage scenario-based stochastic tactical supply chain planning model is presented. The branch-and-fix algorithm is discussed in the next chapter as a methodology to solve the large-scale tactical models which are difficult to solve by existing commercial software.

5.1 Stochastic model

Many planning problems involve sequences of decisions over time. The decisions can respond to realizations of outcomes that are not known a priori. The resulting model for optimal decision making is then a multistage stochastic program. The deterministic tactical

model presented in Chapter 3 is converted to a multi-stage stochastic formulation employing the concept of scenario tree. A scenario z is a path from the root to any of the leaves. The cardinality of the set Z of all possible scenarios z is equal to $|N|$ and the probability weight of the scenario z corresponding to the terminal node $n \in N$ is given by P_z. Denoting the imaginary decision at stage t under scenario z by π_{ts}, the deterministic equivalent of stochastic model can be formulated as follows. The scenario tree utilized in this study is represented in Fig. 13. It shows that the first tactical period is considered deterministic.

The descriptions of constraints and objective function, (5-4) to (5-20), are the same as those described in chapter 3. Scenarios are accounted for and nonanticipativity equalities (5-21) are added to the set of constraints. The first tactical period is considered deterministic which means that the customer demand for the first time grid is known at the time of planning.

$$Min_{ij}\, X_{ijtz} \leq XP_{ijtz} \leq Max\ X_{ijtz} \quad \forall i \in I_j, j, t, z \tag{5-4}$$

$$IPS_{ijtz} = IPS_{ijt-1z} + XP_{ijtz} - \sum_{k \in k_j} XSM_{ijktz} \quad \forall i \in I_j, j, k \in k_j, t \tag{5-5}$$

$$IPD_{iktz} - U_{ikt} = IPD_{ikt-1z} - U_{ikt-1z} + \sum_{j \in j_k} XSM_{ijktz} - D_{iktz}\ \forall i \in I_j, j, k \in k_j, t, z \tag{5-6}$$

$$ICS_{cstz} = ICS_{cs(t-1)z} + XCP_{cstz} - \sum_{\mathrm{j} \in \mathrm{J_s}} \mathrm{XSS_{csjtz}} \quad \forall c \in C_s, s, j \in J_s, t, z \tag{5-7}$$

$$\sum_{\mathrm{i} \in \mathrm{I_j}} \mathrm{MR_{ij}}\ \mathrm{XP_{ijtz}} \leq \mathrm{PA_j}\, N_t + XMOT_{jtz} \quad \forall i \in I_j, j, t, z \tag{5-8}$$

$$\sum_{c \in C_s} SR_{cs}\ XCP_{cstz} \leq SA_s\, N_t + XSOT_{stz} \quad \forall c \in C_s, S, t, z \tag{5-9}$$

$$XMOT_{jtz} \leq MOT_j\, N_t \quad \forall j, t, z \tag{5-10}$$

$$XSOT_{stz} \leq SOT_s\, N_t \quad \forall s, t, z \tag{5-11}$$

$$\sum_{i\in I_j} \alpha_i \; IPS_{ijtz} \leq JSA_j \quad \forall i \in I_j, j, t, z \tag{5-12}$$

$$\sum_{i\in I_k} \alpha_i \; IPD_{iktz} \leq KSA_k \quad \forall k, t, z \tag{5-13}$$

$$\sum_{c\in C_s} \beta_c \; ICS_{cstz} \leq SSA_s \quad \forall s, t, z \tag{5-14}$$

$$\sum_{s\in S_j} XSS_{csjtz} = \sum_{i\in I_j} \rho_{ci} XP_{ijtz} \quad \forall c \in C_i, j, t, z \tag{5-15}$$

Objective Function

$$\text{Min } PC_3 + IC_3 + BC_3 + TC_3 + OC_3 \tag{5-16}$$

Where $PC_3, IC_3, BC_3, TC_3,$ and OC_3 are respectively the production, inventory, backorder, transportation, and overtime costs. They are given as follows:

$$PC_3 = \sum_z \sum_t \sum_j \sum_{i\in I_j} P_z \, PI_{ij} \, XP_{ijtz} + \sum_t \sum_s \sum_{c\in C_s} P_z \, CPS_{cs} \, XCP_{cstz} \tag{5-16a}$$

$$IC_3 = \sum_t \sum_j \sum_{i\in I_j} P_z \, HJ_{ij} \, N_t \, IPS_{ijtz} + \sum_t \sum_k \sum_{i\in I_k} P_z \, HD_{ik} \, N_t \; IPD_{iktz} + \sum_t \sum_s \sum_{c\in C_s} P_z \, HS_{cs} \, N_t \, ICS_{cstz} \tag{5-16b}$$

$$BC_3 = \sum_t \sum_k \sum_{i\in I_k} P_z \, UC_{ik} \, UD_{iktz} \tag{5-16c}$$

$$TC_3 = \sum_t \sum_s \sum_{j\in J_s} \sum_{c\in C_s} P_z \, TSS_{csj} \; XSS_{csjtz} + \sum_t \sum_j \sum_{k\in K_j} \sum_{i\in I_j} P_z \, TSD_{ijk} \; XSM_{ijktz} \tag{5-16d}$$

$$OC_3 = \sum_t \sum_s P_z \, CSOT_c \;\; XSOT_{stz} \;\; + \;\; \sum_t \sum_j P_z \, CMOT_j \;\; XMOT_{jtz} \qquad (5\text{-}16e)$$

If two scenarios z and z' are indistinguishable up to a given stage $l, l = 1, \dots, L$, i.e. z and z' follow the same path up to that stage, then the related decisions, up to stage l must be the same. Considering information embodied in the scenario tree structure, we denote τ_{n_l} be the set of scenarios passing node n in stage l and assume that the set of all tactical variables in period t under scenario z form a vector called V_{tz} (5-17). In our stochastic model, every stage is equivalent to one tactical period.

$$V_{tz} = (X_{ijtz}, XP_{ijtz}, XCP_{cstz}, IPS_{ijtz}, IPD_{iktz}, ICS_{cstz}, UD_{iktz}, XSS_{csjtz}, XSM_{ijktz}, XSOT_{stz}, XMOT_{jtz}) \qquad (5\text{-}17)$$

The following nonanticipativity constraints can be formalized:

$$V_{tz} = V_{tz'} \quad \forall \, z, z' \in \tau_{n_{l=t}}, t \qquad (5\text{-}18)$$

The nonanticipativity constraints will increase the dimension of tactical planning problem. However, they make it easier to transform from deterministic to stochastic model. Also, due to the large dimension of stochastic program, this method of generating stochastic programs could result in a model that is computationally tractable. In the next chapter, a branch-and-fix algorithm will be developed to solve large-scale stochastic programs with binary variables within the set of their first stage decision variables.

5.2 Robust tactical supply chain planning model

To deal with real-world planning issues, it is rational to consider uncertainty and treat it with stochastic programming. However, this alone is not sufficient. Although the uncertainty consideration is essential to be proactive to the predictable events but the possible occurrence of extreme scenarios has not been handled properly. It is true that we predict the future and embed that forecast into the tactical program but how to deal with the situation should any of the extreme scenarios occurrs remains to be addressed. As such, robust optimization is employed.

Mulvey (1995) states that a solution to an optimization model is solution-robust if it remains close to optimal for all scenarios, and is model-robust if it remains almost feasible for all scenarios. This is a promising approach to handle the trade-off between the expected cost and its variability in stochastic programming. The structure of stochastic tactical planning model developed in previous section guarantees the model robustness. For solution robustness, the variance of the cost is incorporated into the tactical objective function. This ensures that the solution stays close to optimal if any of the possible scenarios reveal. Allocating higher weight to variance will result in higher expected cost. It means that we reduce the risk and increase the robustness at the expense of increased expected cost.

A robust optimization model is formulated as follows:

$$\text{Min } \theta(x,\ y_1, y_2, \dots, y_\xi) + \ \omega\varphi(\delta_1, \delta_2, \dots, \delta_\xi)$$

$s.t.$:

$$Ax = b,$$

$$B_\xi x + C_\xi y_\xi + \delta_\xi = e_\xi \qquad \forall \xi \in \Omega,$$

$$x \geq 0, y_\xi \geq 0, \delta_\xi \geq 0 \quad \forall \xi \in \Omega, \tag{5-19}$$

The first term in the objective function represents the solution-robustness and the second term expresses the model-robustness. x denotes the vector of first stage decision variables and y_ξ is second-stage variable under scenario ξ. A finite set of scenarios Ω is assumed with which the subset (B_ξ, C_ξ, e_ξ) is associated and the probability of scenario ξ is represented as p_ξ ($\sum_\xi p_\xi = 1$). δ_ξ denotes the infeasibility of the model under scenario ξ. As was mentioned previously, model-robustness is not an issue here because of the specific structure of the constraint set in the tactical model. It is used as a penalty to the violation of control constraints.

Mulvey et al. (1995) used the following expression to represent solution-robustness:

$$\theta(o) = \sum_{\xi \in \Omega} p_\xi \, f(x, y_\xi) + \omega \sum_{\xi \in \Omega} p_\xi \left(f\left(x, y_\xi\right) - \sum_{\xi' \in \Omega} p_{\xi'} f\left(x, y_{\xi'}\right)\right)^2 , \tag{5-20}$$

where ω is the weight assigned to the variance of second-stage decisions. As ω increases, the tactical plan will be less sensitive to changes in customer demand. As can be seen, there is a quadratic term in Eq. (5-24).

To include solution-robustness in the tactical program, the objective function is modified as follows:

Robust objective function:

Min Z = $\sum_z p_z \, (PC^z + IC^z + BC^z + TC^z + OT^z) + \omega \, \sum_z p_z \, [(PC^z + IC^z + BC^z + TC^z + OT^z) - \sum_{z'} p_{z'} \left(PC^{z'} + IC^{z'} + BC^{z'} + TC^{z'} + OT^{z'}\right)]^2$

(5-21)

Where

Production Cost (PC^z) = $\sum_t \sum_j \sum_i PI_{ij}\ XP_{ijtz}$ +$\sum_t \sum_s \sum_c CPS_{cs}\ XCP_{cstz}$

Inventory Cost (IC^z) = $\sum_t \sum_j \sum_i\ HJ_{ij}\ N_t\ IPS_{ijtz}$ + $\sum_t \sum_k \sum_i HD_{ik}\ N_t\ \ IPD_{iktz}$+ $\sum_t \sum_s \sum_c HS_{cs}\ N_t\ ICS_{cstz}$

Backorder Cost (BC^z) = $\sum_t \sum_k \sum_i UC_{ik}\ UD_{iktz}$

Transportation Cost (TC^z) = $\sum_t \sum_s \sum_j \sum_c\ TSS_{csj}\ \ XSS_{csjtz}$ + $\sum_t \sum_j \sum_k \sum_i TSD_{ijk}\ \ XSM_{ijktz}$

Over-time cost in suppliers and manufacturing sites (OT^z) =

$$\sum_t \sum_s CSOT_s XSOT_{stz} + \sum_t \sum_j CMOT_j XMOT_{jtz}$$

The non-linear term in the objective function of robust tactical program can be linearized by the following adjustments:

Min Z = $\sum_z p_z\ (PC^z + IC^z + BC^z + TC^z + OT^z) + \omega\ \sum_z p_z\ (\theta^{z+} + \theta^{z-})$

(5-22)

$(PC^z + IC^z + BC^z + TC^z + OT^z) - \sum_{z'} p_{z'} \left(PC^{z'} + IC^{z'} + BC^{z'} + TC^{z'} + OT^{z'}\right) = \theta^{z+} - \theta^{z-}$

(5-23)

represents the weight placed on the solution variance where the solution is less sensitive to changes in data under all scenarios as increases. In chapter 6, we will show that it is promising to use the proposed model to generate an efficient supply chain plan.

As was discussed, model robustness is not of a great importance in the tactical model since the material balance constraints and all the other restrictions on any of the sites, suppliers, and distribution centers are satisfied under all likely scenarios.

Chapter 6. Branch-and-Fix Algorithm

The multi-stage stochastic planning model has been introduced in chapter 4. In this chapter, a general algorithm proposed by Escudeto et. al. (2009) will be adopted as a solution approach for the two stage stochastic problems with mixed 0-1 first-stage variables. Since first-stage decision variables in the tactical problem contain both 0-1 variables and continuous variables while the second-stage variables are all continuous, this approach is suitable for solving the large-scale two-stage tactical program in question.

The difference between deterministic models and stochastic models is that in stochastic programs some problem data are uncertain which means some of the data can be treated as random with known distribution functions. The particular values that the random variables will take are only known after the uncertainty unfolds. As a result, the set of decisions in stochastic programming with recourse is divided into two groups:

(1) First-stage decisions: the decisions that have to be taken before the uncertainty unveils.
(2) Second-stage: the decisions that can be taken after the uncertainty unveils.

It can be concluded that the definitions of first and second stage decisions (Note: here we are talking about the first and second decisions which should not be confused with the first and second models as mentioned in the previous chapters and hereafter) are only related to before and after the random experiment and may contain many time periods.

The same logic applies to the stochastic tactical supply chain planning model. Assume that decisions upon variables X_{ijt} and XP_{ijt} have to be made at the beginning of the tactical horizon while the other decisions can be postponed until their associated uncertainty reveals. Hence, the stochastic linear program will be formulated as a two-stage stochastic model and the proposed algorithm in this chapter can be employed.

The solution algorithms for mixed integer stochastic linear programming models have been proposed in the literature based on some of popular algorithms such as Benders decomposition, Lagrangian relaxation, Branch-and-Cut procedure, Branch-and-Bound approach, and Wolf-Dantzig algorithm. A variety of mixed integer stochastic models are addressed, which contain programs with

- first-stage 0-1 variables and second stage continuous variables,
- general first stage variable and integer recourse variables,
- 0-1 mixed integer recourse variables and either continuous or first stage 0-1 variables,
- first stage 0-1 variables and 0-1 mixed integer recourse variables, and
- mixed integer variables in both stages.

More details on the related methods are reported in Slyke and Wets (1969), Louveaux (1986), Laporte and Louveaux (1993), Carøe and Schultz (1999), Escudero et. al. (2009), and Escudero et. al. (2010).

Before the algorithm procedure is explained, it is helpful to present a list of mathematical models to understand the steps.

The general two-stage deterministic mixed 0-1 model with both binary and continuous first-stage variables is formulated as:

$\min c_1^T \delta + c_2^T x + q^T y$

$s.t.$

$b_1 \leq A\binom{\delta}{x} \leq b_2,$

$$h_{01} \leq T_0\binom{\delta}{x} \leq h_{02},$$

$$h_1 \leq T\binom{\delta}{x} + Wy \leq h_2,$$

$$x, y \geq 0,$$

$$\delta \in \{0,1\}^n. \qquad (6\text{-}1)$$

where c_1, c_2, and q are the vector of objective function coefficients for the first-stage variables δ and x, and the second stage variable y, respectively. Vector δ has n $0-1$ variables, and the vectors x and y contain only continuous variables.

The stochastic version of problem (6-1) can be represented by:

$$\min c_1^T \delta + c_2^T x + \sum_{\omega \in \Omega} w^\omega q^{\omega T} y^\omega$$

$$s.t.$$

$$b_1 \leq A\binom{\delta}{x} \leq b_2,$$

$$h_{01}^\omega \leq T_0^\omega\binom{\delta}{x} \leq h_{02}^\omega, \omega \in \Omega$$

$$h_1^\omega \leq T\binom{\delta}{x} + W^\omega y^\omega \leq h_2^\omega, \omega \in \Omega$$

$$x, y^\omega \geq 0, \omega \in \Omega$$

$$\delta \in \{0,1\}^n. \qquad (6\text{-}2)$$

where uncertainty is represented by the scenarios ω from the finite set Ω, each with an associated probability of occurrence w^ω, $\omega \in \Omega$.

The above stochastic model can be reformulated to include nonanticipativity constraints discussed in Chapter 4 as follows:

$$\min \sum_{\omega \in \Omega} w^\omega \left(c_1^T \delta^\omega + c_2^T x^\omega + q^{\omega T} y^\omega\right)$$

$s.t.$

$b_1 \leq A\binom{\delta}{x} \leq b_2,$

$h_{01}^{\omega} \leq T_0^{\omega}\binom{\delta}{x} \leq h_{02}^{\omega}, \omega \in \Omega$

$h_1^{\omega} \leq T\binom{\delta}{x} + W^{\omega}y^{\omega} \leq h_2^{\omega}, \omega \in \Omega$

$x, y^{\omega} \geq 0, \omega \in \Omega,$

$\delta^{\omega} \in \{0,1\}^n, \omega \in \Omega,$

$x^{\omega} - x^{\omega'} = 0 \quad \forall \omega, \omega' \in \Omega \mid \omega \neq \omega',$

$\delta^{\omega} - \delta^{\omega'} = 0 \quad \forall \omega, \omega' \in \Omega \mid \omega \neq \omega'.$ (6-3)

To solve the above problem, we present a branch-and-fix algorithm as follows.

Step 0: Initialize $\bar{Z} = +\infty$. $\bar{Z}$ is the upper-level objective function value.

Step 1: Solve the LP relaxation of the original problem (6-3) where $\delta^{\omega} \in [0,1]^n$and compute $\underline{Z}$. If there is any 0-1 variable that takes a fractional value then go to Step 2. Otherwise, the optimal solution to the original problem has been found, i.e., $\bar{Z} = \underline{Z}$ and stop.

Step 2: Initialize $i = 1$ and go to Step 4.

Step 3: Reset $i = i + 1$. If $i = |\gamma| + 1$ (γ is the number of 0-1 variables), go to Step 8.

Step 4: Set $\langle i \rangle = argmax\{\rho_j, j \in \gamma\}$, such that j has not been previously branched on or has not been fixed at in the current branching path. Branch $\delta_{\langle i \rangle}^{\omega} = \sigma_{\langle i \rangle} \forall \omega \in \Omega$. Determine ρ_j as follows in order to select the branching variable: is calculated as follows:

$\rho_j = \min\{\sum_{\omega \in \Omega} \bar{\delta}^{\omega}, TNS - \sum_{\omega \in \Omega} \bar{\delta}^{\omega}\}, j \in \gamma$ (6-4)

where $\bar{\delta}^p$ gives the current values of variables δ_i^ω. The branching parameter $\sigma_{\langle i \rangle}$ is calculated below:

$$\sigma_{\langle i \rangle} = \begin{cases} 0 \; if \; \sum_{\omega\in\Omega} \bar{\delta}^\omega \leq TNS - \sum_{\omega\in\Omega} \bar{\delta}^\omega \;, i \in \gamma \\ 1 \qquad\qquad\qquad\qquad\qquad\qquad othe\ wise. \end{cases} \tag{6-5}$$

TNS is the Total Number of Scenarios.

Step 5: Solve the linear relaxations (6-6) and compute $\underline{Z}$ as follows:

$\text{LP}_{\langle \text{i} \rangle}^{\omega}$: min $w^\omega \left(c_1^T \delta^\omega + c_2^T x^\omega + q^{\omega T} y^\omega\right)$

$s.t.$

$b_1 \leq A\binom{\delta}{x} \leq b_2,$

$h_{01}^\omega \leq T_0^\omega \binom{\delta}{x} \leq h_{02}^\omega, \omega \in \Omega$

$h_1^\omega \leq T\binom{\delta}{x} + W^\omega y^\omega \leq h_2^\omega, \omega \in \Omega$

$x, y^\omega \geq 0, \omega \in \Omega,$

$\delta_j - \bar{\delta}_j = 0 \quad \forall j \in \{1, \dots, i\},$

$$\delta^\omega \in [0,1] \quad \forall j \in \{i+1, \dots, n\}. \tag{6-6}$$

where n is the total number of 0-1 variables, the lower bound is given by $\underline{Z} = \sum_{\omega\in\Omega} z_i^\omega$ where z_i^ω denotes the solution of the $\text{LP}_{\langle \text{i} \rangle}^{\omega}$ model in which the first i 0-1 variables have already been fixed to 0 or 1, $\bar{\delta}_j$, $j = 1, \dots i$.

If $\bar{Z} \leq \underline{Z}$ then go to Step 7.

If there is any 0-1 variable that either takes fractional values or takes different 0 or 1 values for some of the $\hat{p}$ scenarios, go to Step3.

If all the first-stage continuous variables take the same value for all scenarios $\omega \in \Omega$, update $\bar{Z} = \underline{Z}$ and go to step 7.

Step 6: Solve the sub-model (6-7) to satisfy the nonanticipativity constraints (9) for the first-stage continuous variables in the given TNF integer set. Notice that the solution value is denoted by Z^{LP}.

$$c_1^T \bar{\delta} + \min\ c_2^T x^\omega + \sum_{\omega \in \Omega} w^\omega q^{\omega T} y^\omega$$

$$s.t.$$

$$b_1 \le A\binom{\delta}{x} \le b_2,$$

$$h_{01}^\omega \le T_0^\omega \binom{\delta}{x} \le h_{02}^\omega, \omega \in \Omega$$

$$h_1^\omega \le T\binom{\delta}{x} + W^\omega y^\omega \le h_2^\omega, \omega \in \Omega$$

$$x, y^\omega \ge 0, \omega \in \Omega,$$

$$\delta_j - \bar{\delta}_j = 0 \quad \forall j \in \{1, \dots, i\},$$

$$\delta_j - \bar{\delta}_j = 0 \quad \forall j \in \{i+1, \dots, n\}\ (from\ the\ solution\ of\ (6-6)). \qquad (6\text{-}7)$$

Update $\bar{Z} = min\{Z^{LP}, \bar{Z}\}$. If $i = |\gamma|$, go to Step 7.

Solve the sub-model (6-8), where the fractional 0-1 variables are the not yet branched on. Notice that the solution value is denoted by Z^f. If all the fractional 0-1 variables take 0-1 values in the solution to (6-8), update $\bar{Z} = min\{Z^f, \bar{Z}\}$ and go to Step 7.

$$\min c_1^T \delta + c_2^T x + \sum_{\omega \in \Omega} w^\omega q^{\omega T} y^\omega$$

$$s.t.$$

$$b_1 \leq A\binom{\delta}{x} \leq b_2,$$

$$h_{01}^{\omega} \leq T_0^{\omega}\binom{\delta}{x} \leq h_{02}^{\omega}, \omega \in \Omega$$

$$h_1^{\omega} \leq T\binom{\delta}{x} + W^{\omega}y^{\omega} \leq h_2^{\omega}, \omega \in \Omega$$

$$x, y^{\omega} \geq 0, \omega \in \Omega$$

$$\delta_j - \bar{\delta}_j = 0 \quad \forall j \in \{1, \dots, i\},$$

$$\delta_j \in [0,1] \quad \forall j \in \{i+1, \dots, n\}. \tag{6-8}$$

If $Z^{LP} = Z^f$, or $Z^f \geq \bar{Z}$, go to Step 7; otherwise go to Step 3.

Step 7: Prune the branch.

If $\delta_{\langle i \rangle}^{\omega} = \sigma_{\langle i \rangle}, \forall \omega \in \Omega$, go to Step 10.

Step 8: Reset $i \rightarrow i - 1$.

If $i = 0$, stop because the optimal solution $\bar{Z}$ has been found.

Step 9: if $\delta_{\langle i \rangle}^{\omega} = 1 - \sigma_{\langle i \rangle}, \forall \omega \in \Omega$, go to Step 8.

Step 10: Branch $\delta_{\langle i \rangle}^{\omega} = 1 - \sigma_{\langle i \rangle}, \forall \omega \in \Omega$. Go to Step 5.

The branch-and-fix algorithm is implemented in Java using Cplex 12.2 libraries.

Chapter 7. Case Study

The performance of the iterative algorithm is examined in this chapter. Comparisons of deterministic, stochastic, and robust integrated supply chain planning models are also illustrated by solving two case-study problems of a multi-product supply chain system composed of multiple suppliers, manufacturing sites, and distribution centers in different geographical locations. The solution algorithms and integrated planning models were implemented in Java solved with Cplex 12.2 on an Intel 2.27 GHz machine with 4GB memory.

7.1 Case study 1

In the case of iterative algorithm, three scenarios are simulated to examine the efficiency of the proposed procedure. This is illustrated by changing the amount of demands in different mid-term periods. A tactical planning problem with the horizon of 12 weeks is considered which is further divided into four time periods of 1, 2, 4, and 5 weeks, respectively. The integrated model acts as a controller in the planning system and only the manipulated variables elaborated for the first tactical period are implemented. Hence, solution procedure is mostly concentrated on the first tactical period and is only interested to have a general understanding of future tactical periods. The first tactical period is the operational horizon composed of five equal short-term periods (in work days) (Fig. 7.1).

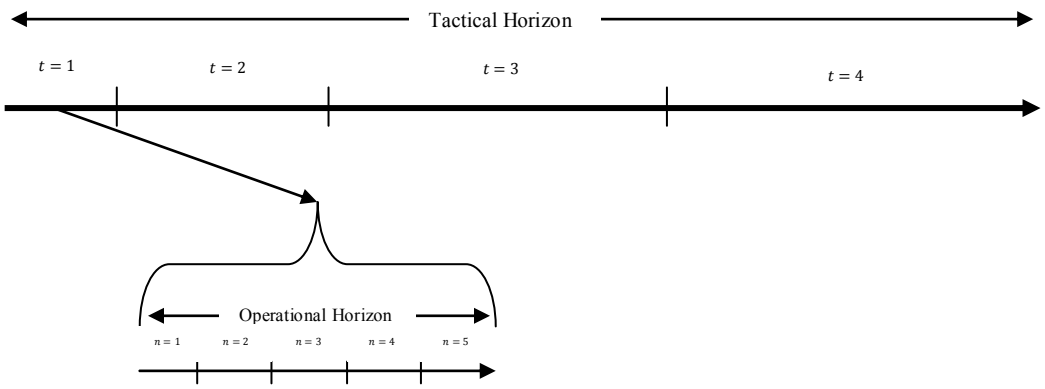

Figure 7.1. Tactical-operational time grids

The supply chain system is composed of three manufacturing sites, 15 suppliers, five distribution centers, four product families, and 20 component families. Each of product families 1, 2, and 4 contains two products while product family 3 has three products. Manufacturing sites 1, 2, and 3 comprise of 3, 2, and 2 assembly lines, respectively. The parameters of both tactical and operational models are represented in tables 7.1 to 7.9.

Overtime cost, production capacity, overtime capacity, warehouse capacity

	Manufacturing site *j*		
	1	2	3
Overtime cost (/unit)	5.6	6.4	7.8
Production capacity (/standard time unit)	2400	3200	1600
Overtime capacity (/standard time unit)	240	320	160
Warehouse capacity	480	640	320

Table 7-1 Overtime costs, production capacities, overtime capacities, and warehouse capacities of manufacturing sites

Manufacturing requirements, holding costs.

	Manufacturing requirements			Holding costs			Production cost		
	Manufacturing site *j*			Manufacturing site *j*			Manufacturing site *j*		
Product family *i*	1	2	3	1	2	3	1	2	3
1	1	1.2	1.1	10	10	13	24	24	27
2	2.6	2.2	2	18	17	14	34	32	28
3	3	3.3	3.2	24	23	19	61	60	44
4	2.4	2.2	2.1	30	38	42	70	77	80

Table 7-2 Manufacturing requirements and holding costs of manufacturing sites

Production Capacity.

	Supplier *s*				
	1	2	3	4	5
Production Capacity (standard time unit)	50000	60000	70000	50000	60000
	6	7	8	9	10
Production Capacity (standard time unit)	50000	60000	70000	50000	60000
	11	12	13	14	15
Production Capacity (standard time unit)	50000	60000	70000	50000	60000

Table 7-3 Production capacities of suppliers

Holding cost

	Distribution center *k*				
Product family *i*	1	2	3	4	5
1	10	11	11	12	13
2	14	14	15	18	18
3	24	24	23	22	19
4	40	42	23	24	38

Table 7-4 Holding costs of distribution centers

Consumption rate.

Product Family i	Component family c									
	1	2	3	4	5	6	7	8	9	10
1	1	3	0	4	2	2	1	1	6	8
2	2	2	1	4	0	1	3	1	6	8
3	0	1	2	4	4	1	3	2	6	8
4	3	2	1	4	1	1	0	2	6	8
	11	12	13	14	15	16	17	18	19	20
1	1	4	2	4	4	8	10	1	1	2
2	0	1	1	3	4	8	10	2	1	2
3	2	4	4	4	4	8	10	0	3	2
4	0	4	0	3	4	8	10	4	3	2

Table 7-5 Consumption rates

Warehouse capacity.

	Distribution center k				
	1	2	3	4	5
Warehouse capacity	100	90	70	80	50

Table 7-6 Warehouse capacities of distribution centers

Transportation cost.

Product family i	Manufacturing site j	Distribution center k				
		1	2	3	4	5
1	1	5	5.1	6	5.6	7
	2	5.1	6	5.6	7	5.2
	3	6	5.6	7	5.2	6.4
2	1	8.9	9	7.1	7.5	8.4
	2	9	7.1	7.5	8.4	8.3
	3	7.1	7.5	8.4	8.3	8
3	1	9.5	9.8	11.1	11.2	11
	2	9.8	11.1	11.2	11	12
	3	11.1	11.2	11	12	9.8
4	1	20.7	20.5	21	19.8	19.9
	2	20.5	21	19.8	19.9	18.2
	3	21	19.8	19.9	18.2	18.9

Table 7-7 Transportation costs between manufacturing sites and distribution centers

Manufacturing requirements.

Component family c	Supplier s														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.2	1	1	1	1	1	1.3	1	1	1	1	1	1.1	1	1
2	1.4	1	1	1	1	1.3	1	1.2	1	1	1	1.1	1	1	1
3	1	2	1	1	1	1	1.8	1	1	1	1	2.2	1	1	1
4	1	1.5	1	1	1	1.7	1	1.6	1	1	1	1	1.5	1	1
5	2.2	1	1	1	1	1	2.3	1	1	1	1	1	2.7	1	1
6	1	1.3	1	1	1	1	1	1.2	1	1	1	1	1	1.3	1
7	1	1	1.8	1	1	1	1.9	1	1	1	1	1	1	1.7	1
8	1	2.3	1	1	1	1	1	2.2	1	1	1	1	1	2	1
9	1	1	1.1	1	1	1	1.2	1	1	1	1	1	1	1	1.1
10	1	1	1.3	1	1	1	1	1	1.2	1	1	1	1.1	1	1.1
11	1	1	3	1	1	1	1	1	3.3	1	1	2.3	1	1	1
12	1	1	1	4	1	1	1	1	1	3	1	4.5	1	1	1
13	1	1	1	2	1	1	1	1	1.8	1	1	1	2.4	1	1
14	1	1	1	1.6	1	1	1	1	1	1.2	1	1	1.1	1	1
15	1	1	1	1.4	1	1	1	1	1.8	1	1	1	1	1.3	1
16	1	1	1	1	1.1	1	1	1	1	1.1	1	1	1	1	1
17	1	1	1	1	1.1	1	1	1	1	1.1	1	1	1	1	1
18	1	1	1	1	1.1	1	1	1	1	1	1.5	1	1	1	1.8
19	1	1	1	1	1	4	1	1	1	1	3.5	1	1	1	4.2
20	1	1	1	1	1	2	1	1	1	1	1.8	1	1	1	1.5

Table 7-8 Manufacturing requirements of component families at suppliers

Production cost.

Component family c	Supplier s														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.5	1	1	1	1	1	1.8	1	1	1	1	1	1.1	1	1
2	1.6	1	1	1	1	1.4	1	1.2	1	1	1	1.1	1	1	1
3	1	1.8	1	1	1	1	1.6	1	1	1	1	1.9	1	1	1
4	1	1.7	1	1	1	1.9	1	1.8	1	1	1	1	1.5	1	1
5	2.2	1	1	1	1	1	2.3	1	1	1	1	1	2.7	1	1
6	1	1.5	1	1	1	1	1	1.3	1	1	1	1	1	1.5	1
7	1	1	2.1	1	1	1	2	1	1	1	1	1	1	2.4	1
8	1	2.5	1	1	1	1	1	2.4	1	1	1	1	1	2.2	1
9	1	1	1.5	1	1	1	1.6	1	1	1	1	1	1	1	1.5
10	1	1	2	1	1	1	1	1	1.9	1	1	1	1.1	1	1.8
11	1	1	3.2	1	1	1	1	1	3.5	1	1	2.5	1	1	1
12	1	1	1	4.5	1	1	1	1	1	3.5	1	5	1	1	1
13	1	1	1	2	1	1	1	1	1.8	1	1	1	2.4	1	1
14	1	1	1	1.9	1	1	1	1	1	1.5	1	1	1.3	1	1
15	1	1	1	2	1	1	1	1	1.8	1	1	1	1	1.8	1
16	1	1	1	1	1.5	1	1	1	1	1.6	1	1	1	1.4	1
17	1	1	1	1	2.1	1	1	1	1	2.1	1	1	1	1.9	1
18	1	1	1	1	2.1	1	1	1	1	1	2.5	1	1	1	2.8
19	1	1	1	1	1	4.5	1	1	1	1	3.9	1	1	1	5.2
20	1	1	1	1	1	2.8	1	1	1	1	1.5	1	1	1	1.9

Table 7-9 Production costs of component families at suppliers

The following sub-cases are simulated and the results are given in table 7.10:

(1) There is no unfulfilled demand in the tactical nor in the operational solutions.
(2) There are unfulfilled demands in both the tactical and operational levels. However, there is a discrepancy in production volume.
(3) There is no unfulfilled demand in both the tactical and operational solutions, but the tactical solution imposes some end-of-horizon inventory on the operational model.

Sub-case 1 represents a situation in which there is no unfulfilled demand in tactical model nor in operational model and the operational model satisfies the targets set by the tactical problem. Table 7.10 shows the computation times for the proposed decomposition algorithm and the full space methods. By full-space model (Fig. 7.2), we mean that the variables and constraints of first tactical period are substituted by the operational planning model.

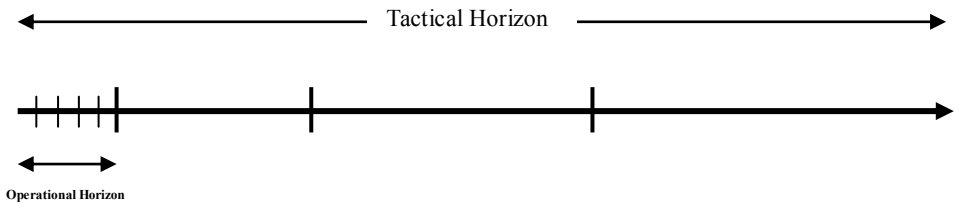

Figure 7.2. Full-space time grids

Table 7.10 shows that iterative algorithm solves the problem in 15.87 seconds of computing time which is 80% lower than the time required to solve the full-space problem with 0.07% deviation form the optimal solution. The solution obtained using the proposed algorithm leads to a cost of 3250876.76 and is reached in one iteration.

Table 7.10 shows that iterative algorithm solves the problem of sub-case 2 in 165.32 seconds of computing time which is 34% lower than the time required to solve the full-space

problem for 7% optimality tolerance. The solution obtained using the proposed algorithm leads to a cost of 34688188.13 and is reached in one iteration.

In sub-case 3, customer demand has changed and there are unfulfilled demands in both SC planning levels. The production tolerance is set to 0.02 and 0.01 which are respectively called sub-case 3a and sub-case 3b. Table 7.10 shows the computing times for iterative algorithm and the full space method. In sub-case 3a, the iterative algorithm yields a solution with a cost of 11251211.47 with a 3% optimality tolerance and is solved in 6.8 seconds, whereas the full-space method terminates in 110.2 seconds. For sub-case 3b, the proposed approach results in a cost of 11586394.24 with a 0.07% optimality tolerance and is solved in 812.3 seconds, whereas the full-space method stops in 110.2 seconds. By reducing the tolerance limit, the operational model is solved 42 times by iterative algorithm and logic and integer cuts are accumulated in the tactical level in each iteration. Hence, the size of planning model increases and this is the only case where the computing time is longer than the full-space method. (table 7.10)

As was illustrated, the iterative algorithm solves the integrated problem in only one iteration in most of the cases. This is because the targets imposed by tactical model tightened the feasible solution region of the operational model. Therefore it takes shorter time to find the optimum solution. The concept of MPC is employed which means the operational planning solution is implemented on the shop floor and the next tactical periods are ignored. It is clear that if the whole tactical horizon is solved as a detailed planning model, the problem will be more complex and difficult to solve.

Results of case studies.

	Full-space Model		Iterative algorithm	
	Total cost	Time to resolve	Total cost	Time to resolve
Sub-case 1 Number of Iterations =1 0.07% gap of cost	3248558.81	76.58	3250876.76	15.87
Sub-case 2 Number of iterations = 1 7% gap of cost	4354029.63	249.62	4688188.13	165.32
Sub-case 3a (ε>0.02) Number of iterations = 1 3% gap of cost	11595438.05	110.2	11251211.47	6.8
Sub-case 3b (ε=0.01) Number of iterations = 42 0.07% gap of cost	11595438.05	110.2	11586394.24	812.3

Table 7-10 Resluts of case studies

Case study 2

In this case study, demand is considered as an uncertain parameter with a finite discrete distribution function that can be represented by a scenario tree. The first period is considered deterministic and the first-stage variable set comprises of all decisions made for the first deterministic time grid and production decisions for the second stage. A scenario tree composed of two stages is considered. The first stage is a deterministic tactical period and the second stage consists of five scenarios of possible customer demand which are shown in Appendix A. This supply chain includes 6 manufacturing sites, 15 suppliers, 15 distribution centers, 8 product families, and 20 component families. Parameters including production costs, transportation costs, inventory costs, capacities, and manufacturing requirements are listed in Appendix A.

Next, the impact of demand uncertainty on the various supply chain planning decisions is investigated. We will also analyze how the planning decisions obtained by neglecting variability in demand perform when exposed to uncertainty. For this purpose, the deterministic model for each scenario is solved. The resulting optimal first-stage decisions are then fixed in stochastic and robust optimization models and they are subsequently solved. Then, the costs and standard deviations are compared in Fig. 7.3. The results suggest that significant cost savings can be achieved by considering demand uncertainty in the planning process. In terms of the standard deviation, robust optimization model outperforms deterministic and stochastic models.

One of the indexes usually utilized to measure the maximum amount a decision maker would be willing to pay in return for complete information about the future is called the *expected value of perfect information* (EVPI). In our example, the objective value of stochastic model is found as 3643750 and the mean value of deterministic solution is 3212921. Therefore, the expected value of perfect information is then 430829 (=3643750-3212921).

Finding the solution for stochastic programming models is more complicated than for deterministic problems. A natural temptation is to solve a simpler problem which is the one obtained by replacing all random parameters by their expected values. This is called the *mean value* problem. Here, the mean value problem is solved and the first-stage decisions are communicated to the stochastic model. The expected value of objective function is found as 4214228 which will result in a discrepancy of 570478 (=4214228-3643750). This is the cost of ignoring uncertainty in decision making.

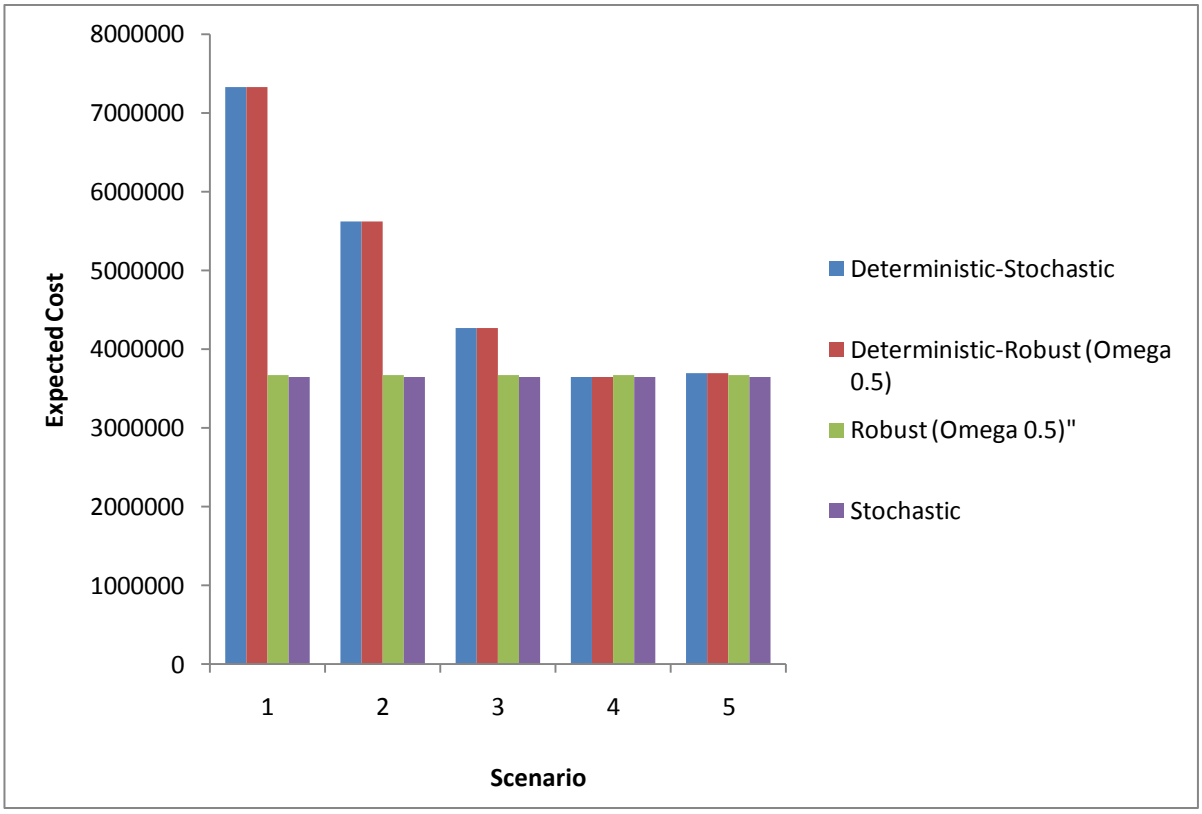

Figure 7.3. Comparison of expected values of recourse decisions

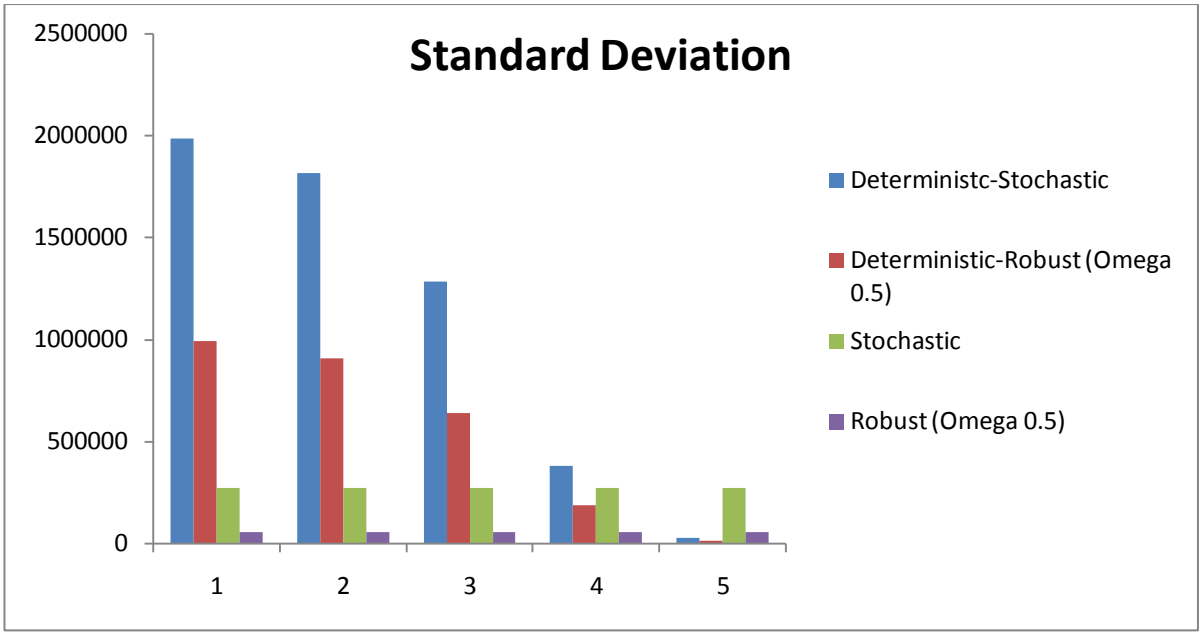

Figure 7.4. Comparison of standard deviations of objective function

Numerical results show that the solution obtained by the robust model is better than the optimal deterministic solution in terms of standard deviation. It is also understandable that the expected value from the robust model is higher than the one obtained by solving the stochastic problem. In robust optimization, the added expected cost offsets supply chain risks. As the value of ω in the objective function of robust model increases, the expected cost goes up but the standard deviation decreases. This is up to managers and practitioners to decide upon the final value of w. Figs. 7.5 and 7.6 show the impact of changing risk coefficient on the values of objective function and standard deviation. The SC plan accomplished by the robust model shows an enhancement in terms of averting extreme risky scenarios.

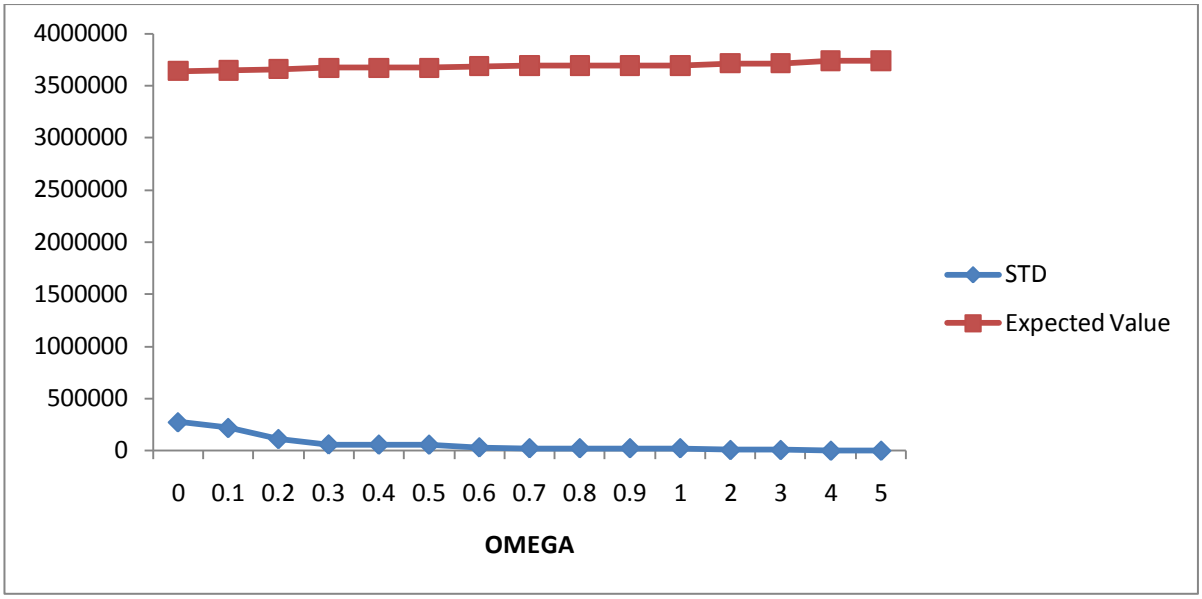

Figure 7.5. Impact of ω on standard deviation and expected value of the robust problem

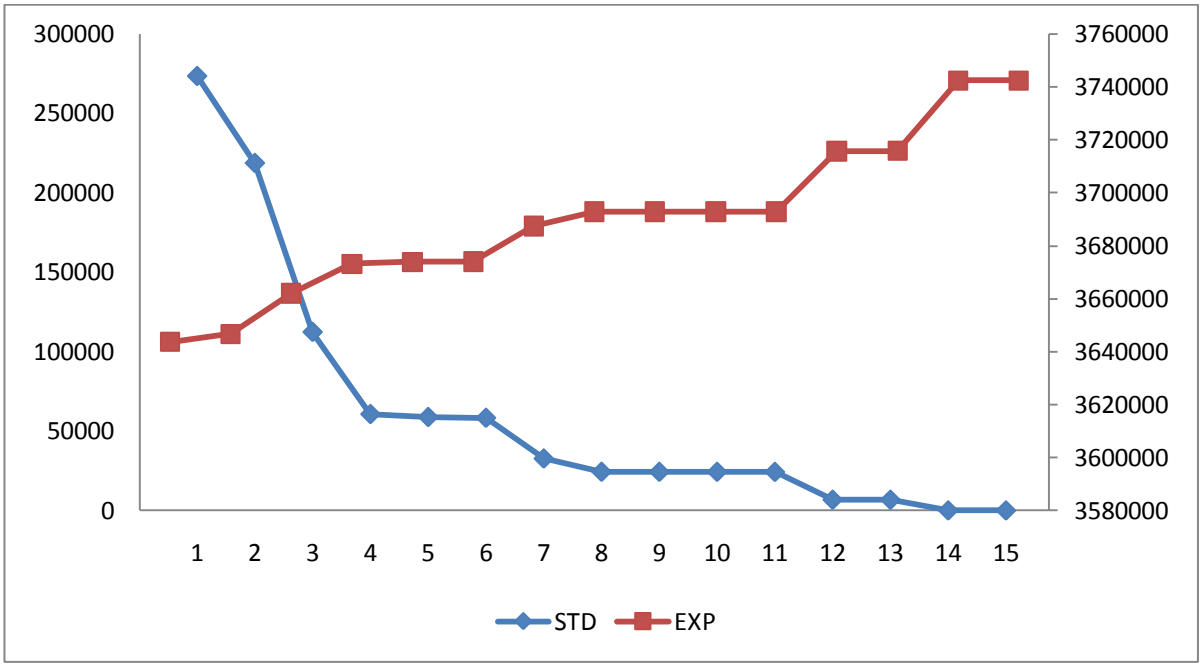

Figure 7.6. Another representation of the impact of ω on expected value and standard deviation of robust problem

Chapter 8. Summary and Directions for Future Studies

The research objectives proposed in Chapter 1 have been achieved and the work carried out in this study is summarised below. Recommendations for future work are also presented in this chapter.

8.1 Summary

This study has provided a methodology to develop multi-level robust supply chain plans and an iterative algorithm to solve the resulted problem. Unlike most related studies, all the elements of supply chain network as well as uncertain nature of planning parameters are brought under a comprehensive structure and taken into account in planning processes. In summary, the following has been accomplished:

1) Three planning problems, i.e., deterministic, stochastic, and robust, have been formulated as multi-level problems. The performances of the associated models have been investigated via case studies. It was concluded that robust and stochastic models provide more reliable plans and robust SC planning model is more advantageous when the enterprise policy is to avert the risky extreme-scenarios.
2) Branch-and-fix algorithm has been proposed to solve large-scale stochastic tactical scenario-based SC planning problems. The proposed procedure breaks the main problem into sub-problems originated from branchs on the scenario tree and solve them independently while keeping track of non-anticipativity constraints.
3) Iterative algorithm has been applied in this study to obtain quick and near optimal solutions to large size integrated SC planning problems. Conditions are checked in each step of the algorithm. Logic and integer cuts are added to converge the solutions of tactical and operational problems.
4) Two case studies have been conducted. The integrated planning model was solved using the iterative algorithm proposed in this study and the result was compared to the solution of full-space method. The comparison clearly illustrates that the iterative

algorithm not only provides close to optimal solutions but also solves the integrated model much faster.

8.2 Future work

To further improve integrated SC planning framework, the above study could be extended in the following directions:

1) *Involvement of procurement planning in the operational model*
   This study is focused on manufacturing and demand planning in the lower level problem and it is assumed that suppliers have enough capacity to procure as many components as required. Incorporation of a procurement module in the SC planning problem would be of particular interest to planners.
2) *Incorporation of lead-time into both tactical and operational planning models*
   Pproduction and transportation lead-times are not considered in this study. It is assumed that a product that is planned in a period will be produced and delivered in the same time period. However, this is not always the case. Therefore, incorporating the lead-time into the integrated problem would be an interesting topic for future research.
3) *Inclusion of a solution algorithm for operational SC planning problem in the proposed SC planning structure*
   This study has concentrated on the tactical SC planning level and its solution algorithms. The proposed structure can be developed to include a procedure for solving the operational model.
4) *Consideration of strategic planning in proposed framework*
   This study is focused on mid-term and short-term SC planning problems. Extending the study to including the strategic planning module would result in more feasible and reliable plans.
5) *Extending stochastic and robust SC planning models by incorporating uncertainty of other system parameters in the proposed framework*

In this study, only the customer demand is treated as an uncertain parameter. Considering the uncertainties of other parameters such as production cost, lead-time, cycle time, procurement cost, inventory cost, and transportation cost is of great importance in stochastic dynamic SC planning.

References

Amaro, A. C. S. and Barbosa-Povoa, A. P. F. D., 2008, "Planning and scheduling of industrial supply chains with reverse flows: A real pharmaceutical case study," *Computers and Chemical Engineering*, **32**, pp. 2606–2625.

Aghezzaf, E., Sitompul, C. and Najid, N. M., 2010, "Models for robust tactical planning in multi-stage production systems with uncertain demands," *Computers & Operations Research*, **37**, pp. 880 – 889.

Al-Qahtani, K. and Elkamel, A., 2010, "Robust planning of multisite refinery networks: Optimization under uncertainty," *Computers and Chemical Engineering*, **34**, pp. 985–995.

Alonso-Ayuso, A., Escudero, L. F. and Ortuno, M. T., 2003, "BFC, branch-and-fix coordination algorithmic framework for solving some types of stochastic pure and mixed 0-1 programs," *European Journal of Operational Research*, **151**, pp. 503-519.

Bertsekas, DP., 2000, "Dynamic programming and optimal control," Belmont, MA: Athena Scientific.

Birge, J. R., 1982, "Decomposition and partitioning methods for multi-stage stochastic linear programs," *Technical Report*, **82**.

Birge, J. R. and Louveaux, F.,1997, "Introduction to Stochastic Programming," USA: Springer.

Bonfill, A., Espuna, A. and Puigjaner, L., 2008, "Proactive approach to address the uncertainty in short-term scheduling," *Computers and Chemical Engineering*, **32**, pp. 1689-1706.

Bose, S. and Pekny, J. F., 2000, "A model Predictive framework for planning and scheduling problems: a case study of consumer goods supply chain," *Computers and Chemical Engineering*, **24**, pp. 329-335.

Camacho, E. F. and Bordons, C., 1995, "Model predictive control in the process industry," London: Springer.

Caroe, C. and Schultz, R., 1999, "Dual decomposition in stochastic integer programming," *Operations Research Letters*, **24**, pp. 37-45.

Caroe, C. and Tind, J., 1998, "L-shaped decomposition of two-stage stochastic programs with integer recourse," *Mathematical Programming*, **83**, pp. 451-464.

Choi, J., Realff, M. J., Lee, J. H., 2005, " Stochastic dynamic programming with localized cost-to-go approximators: Application to large scale supply chain management under demand uncertainty," *Chemical Engineering research and design*, 83, pp. 752-758.

Dantzig, G. B., 1995, "Linear programming under uncertainty," *Management Science*, **3**, pp. 197-206.

Edrik-Dogan, M. and Grossmann, I. E., 2005, "A decomposition method for the simultaneous planning and scheduling of single stage continuous multiproduct plants," *Ind. Eng. Chem. Res.*, ***45*** (1), pp. 299–315.

Escudero, L .F., Garin, M. A., Merino, M. and Perez, G., 2007, "A two-stage stochastic integer programming approaches a mixture of Branch-and-Fix Coordination and Benders Decomposition schemes", *Ann Oper Res*, **152**, pp. 395–420.

Escudero, L .F., Garin, M. A., Merino, M. and Perez, G., 2009, "A general algorithm for solving two-stage stochastic mixed 0-1 first-stage problems," *Computers & Operations Research*, **36**, pp. 2590-2600

Escudero, L .F., Garin, M. A., Merino, M. and Perez, G., 2010, "An exact algorithm for solving large-scale two-stage stochastic mixed-integer problems: Some theoretical and experimental aspects," *European Journal of Operational Research*, **204**, pp. 105–116.

Grossmann, I. E., Van Den Heever, S. A. and Harjunkoski, I., 2002, "Discrete optimization methods and their role in the integration of planning and scheduling," *AICHE SYMPSIUM SERIES*, **326**, pp. 150-168.

Guan, Z. and Philpott, A. B., 2010, "A multistage stochastic programming model for the New Zealand dairy industry," *International Journal of Production Economics,* In Press.

Gupta, A. and Grossmann, I. E., 2010, "Solution strategies for multistage stochastic programming with endogenous uncertainties," *Computers and Chemical Engineering*, In Press.

Gupta, A. and Maranas, C. D., 2003, "Managing demand uncertainty in supply chain planning," *Computers and Chemical Engineering*, **27**, pp. 1219-1227.

Harjunkoski, I. and Grossmann, E., 2002, "Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods," *Computers and Chemical Engineering*, **26**, pp. 1533–1552.

Józefowska, J. and Zimniak, A., 2008, "Optimization tool for short-term production planning and scheduling," *International Journal of Production Economics*, **112**, pp. 109-120.

Kall, P. and Mayer, J., 2005, "Stochastic linear programming: Models, theory, and computation," Springer, USA.

Kall, P. and Wallace, S. W., 1994, "Stochastic programming", John Wiley & Sons, Chichester.

Kanyalkar, A. P. and Adil, G. K., 2005, "An integrated aggregate and detailed planning in a multi-site production environment using linear programming," *International Journal of Production Research*, **43**(20), pp. 4431-4454.

Karimi, I. A. and McDonald, C. M., 1997, "Planning and scheduling of parallel semicontinuous processes. 2. Short-term scheduling," *Industrial Engineering Chemical Research*, **36**, pp. 2701-2714.

Klibi, W., Martel, A. and Guitouni, A., 2010, "The design of robust value-creating supply chain network: A critical review," *European Journal of Operational Research*, **203**, 283-293.

Kok, A.G. and Graves, S.C., 2003, "Handbooks in OR & MS," Vol. 11_ 2003, Elsevier B.V.

Laporte, G. and Louveax, F. V., 1993, "The integer L-shaped method for stochastic integer programs with complete recourse," *Operations Research Letters*, **13**, pp. 133-142.

Li, Z. and Ierapetritou, M. G., 2007, "Process scheduling under uncertainty using multiparametric programming," *AIChE*, **53**, pp. 3183-3203.

Li, Z. and Ierapetritou, M. G., 2009, "Integrated production planning and scheduling using a decomposition framework," *Chemical Engineering Science*, **64**, pp. 3585-3597.

Li, Z. and Ierapetritou, M. G., 2010, "Production planning and scheduling integration through augmented Lagrangian optimization," *Computers and Chemical Engineering*, **34**, pp. 996–1006.

Li, Z. and Ierapetritou, M. G., 2010, "Rolling horizon based planning and scheduling integration with production capacity consideration," *Chemical Engineering Science*, **65**, pp. 5887-5900.

Li, X. and Marlin, T. E., 2009, "Robust supply chain performance via model predictive control," *Computers and Chemical Engineering*, 33, pp. 2134-2143.

Louveax, F. V., 1986, "Multistage stochastic programs with block-separable recourse," *Mathematical Programming Study*, **28**, pp. 48-62.

Louveax, F. V. and Schultz, R., 2003, "Stochastic Integer Programming," *Handbooks in OR & MS*, **10,** Chapter 4, Elsevier Science B.V.

Maravelias, C. T., 2006, "A decomposition framework for the scheduling of single- and multi-stage processes," *Computers and Chemical Engineering*, **30**, pp. 407–420.

Maravelias, C. and Sung, C., 2009, "Integration of production planning and scheduling: Overview, challenges and opportunities," *Computers and Chemical Engineering*, **33**, pp. 1919-1930.

Mirzapour Al-e-hashem, S . M. J., Malekly, H., Aryanezhad, M. B., 2011, "A multi-objective robust optimization model for multi-product multi-site aggregate production planning in a supply chain under uncertainty," *International Journal of Production Economics*, In Press.

Mulvey, J. M. and Vanderbe, R. J., 1994, "Robust optimization of large-scale systems," *Operations Research*, **43**, pp. 264-281.

Pan, F. and Nagi, R., 2010, "Robust supply chain design under uncertain demand in agile manufacturing," *Computers & Operations Research*, **37**, pp. 668-683.

Perea-Lopez, E., Ydstie, B. E. and Grossmann, I., 2003, "A model predictive control strategy for supply chain management," Computers and Chemical Engineering, **27**, pp. 1201–1218.

Pishvaee, M. S., Rabbani, M. and Torabi, S. A., 2011, "A robust optimization approach to closed-loop supply chain network design under uncertainty," *Applied Mathematical Modelling*, **35**, pp. 637–649.

Pochet, Y. and Wolsey, L. A., 2000, "Production planning by Mixed Integer Programming (MIP)," USA: Springer.

Puigjaner, L. and Lainez, J. M., 2008, "Capturing dynamics in integrated supply chain management," *Computers and Chemical Engineering*, **32**, pp. 2582-2605.

Roghanian, E., Sadjadi, S.J. and Aryanezhad, M.B., 2007, "A probabilistic bilevel linear multi-objective programming problem to supply chain planning," *Applied Mathematics and Computation*, **188**, pp. 786-800.

Ryu, J. H., Dua, V., Pistikopoulos, E. N., 2004, "A bilevel programming framework for enterprise-wide process networks under uncertainty," *Computers and Chemical Engineering*, **28**, pp. 1121-1129.

Sarimveis, H., Patrinos, P., Tarantilis, C. D. and Kiranoudis, C. T., 2008, "Dynamic modeling and control of supply chain systems: A review," *Computers & Operations Research*, **35**, pp. 3530 – 3561

Schutz, P. and Tomasgard, A., 2010, "The impact of flexibility on operational supply chain planning," *International Journal of Production Economics,* In Press.

Sen, S., 2005, "Algorithms for stochastic mixed-integer programming models," Handbooks in OR & MS, Vol. 12, Chapter 9, Elsevier B.V.

Shah, N., 2005, "Process industry supply chains: Advances and challenges," *Computers and Chemical Engineering*, **29**, pp. 1225–1235.

Sherali, H., Zhu, X., 2006, "On sloving discrete two-stage stochastic programs having mixed-integer first- and second-stage variable," *Math. Program.,* **108**, pp. 597–616.

Sodhi, M. S. and Tang, C. S., 2009, "Modeling supply chain planning under demand uncertainty using stochastic programming: a survey motivated by asset-liability management," *International Journal of Production Economics*, **121**, pp. 728-738.

Sousa, R., Shah, N. and Papageorgiou, L. G., 2008, "Supply chain design and multilevel planning-An industrial case," *Computers and Chemical Engineering*, **32**, pp. 2643–2663.

Sung, C. and Maravelias, C. T., 2007, "An attainable region approach for production planning of multiproduct processes," *AIChE Journal*, **53**, pp. 1298-1315.

Van Slyke, R. M. and Wets, R., 1969, "L-shaped linear programs with applications to optimal control and stochastic programming," *SIAM Journal on Applied Mathematics*, **17**, pp. 638-663.

Wang, W., Rivera, D. E. and Kempf, K. G., 2007, "Model predictive control strategies for supply chain management in semiconductor manufacturing," *International Journal of Production Economics*, **107**, pp. 56-77.

Wollmer, R. D., 1980, "Two stage linear programming under uncertainty with 0-1 integer first stage variables," *Mathematical Programming*, **19**, pp. 279-288.

Wu, D. and Ierapetritou, M., 2007, "Hierarchical approach for production planning and scheduling under uncertainty," *Chemical Engineering and Processing*, **46**, pp. 1129–1140.

Yan, H. S., Xia, Q. F., Zhu, M. R., Liu, X. L. and Guo, Z. M., 2003, "Integrated production planning and scheduling on automobile assembly lines," *IEEE Transactions*, **35**, pp. 711-725.

Yan, H. S. and Zhang, Z. D., 2007, "A case study on integrated production planning and scheduling in a three-stage manufacturing system," *IEEE Transaction on Automation Science and Engineering*, **4(1)**, pp. 86-92.

Appendix A – Java-Cplex code for tactical SC planning model

```
import ilog.concert.*;
import java.lang.*;
import ilog.cplex.IloCplex;
import java.util.*;
public class RobustTacticalModel {
//INDEX
static final int NOPFi =
RobustTacticalData.NumberOfProductFam;
static final int NOMSj =
RobustTacticalData.NumberOfManuSite;
static final int NOSs =
RobustTacticalData.NumberOfSupplier;
static final int NODk =
RobustTacticalData.NumberOfDC;
static final int NOTPt =
RobustTacticalData.NumberOfTimePeriod;
static final int NOCFc =
RobustTacticalData.NumberOfComponentFam;
int NOSz;
double[] Probability =
new double[NOSz];
double[][][][][] LHSBinary;
double[] RHSBinary;
double[][][][] XXijtzStep6;
boolean Step6 = false;
```

```
int TITI;
double[][][][] DETXPijtz=
new double[RobustTacticalData.NumberOfProductFam]
[RobustTacticalData.NumberOfManuSite][RobustTacticalData.NumberOfTimePeriod][1];
double[][][][] DETIPSijtz=
new double[RobustTacticalData.NumberOfProductFam]
 [RobustTacticalData.NumberOfManuSite][RobustTacticalData.NumberOfTimePeriod][1];
double[][][][] DETXXijtz=
new double[RobustTacticalData.NumberOfProductFam]
[RobustTacticalData.NumberOfManuSite][RobustTacticalData.NumberOfTimePeriod][1];
double[][][][][] DETXSMijktz=
new double[RobustTacticalData.NumberOfProductFam][RobustTacticalData.NumberOfManuSite]
[RobustTacticalData.NumberOfDC][RobustTacticalData.NumberOfTimePeriod][1];
double[][][][] DETIPDiktz=
new double[RobustTacticalData.NumberOfProductFam]
[RobustTacticalData.NumberOfDC][RobustTacticalData.NumberOfTimePeriod][1];
double[][][][] DETUDiktz=
new double[RobustTacticalData.NumberOfProductFam]
[RobustTacticalData.NumberOfDC][RobustTacticalData.NumberOfTimePeriod][1];
double[][][][][] DETXSScsjtz=
new
double[RobustTacticalData.NumberOfComponentFam][RobustTacticalData.NumberOfSupplier]
[RobustTacticalData.NumberOfManuSite][RobustTacticalData.NumberOfTimePeriod][1];
double[][][][] DETICScstz=
new double[RobustTacticalData.NumberOfComponentFam]
[RobustTacticalData.NumberOfSupplier][RobustTacticalData.NumberOfTimePeriod][1];
double[][][][] DETXCPcstz =
```

```
new double[RobustTacticalData.NumberOfComponentFam]
[RobustTacticalData.NumberOfSupplier][RobustTacticalData.NumberOfTimePeriod][1];
double[][][] DETXMOTjtz =
new double[RobustTacticalData.NumberOfManuSite]
[RobustTacticalData.NumberOfTimePeriod][1];
double[][][] DETXSOTstz =
new double[RobustTacticalData.NumberOfSupplier][RobustTacticalData.NumberOfTimePeriod][1];
public RobustTacticalModel(int Scenario, double[][][][] D, double[] Prob, int titi){
NOSz = Scenario;
this.Diktz = D;
this.Probability = Prob;
this.Step6 = false;
this.TITI = titi;
}
public RobustTacticalModel(int Scenario, double[][][][] D, double[] Prob){
NOSz = Scenario;
this.Diktz = D;
this.Probability = Prob;
this.Step6 = false;
}
public RobustTacticalModel(int Scenario, double[][][][] D, double[] Prob,
double[][][][][] A, double[] B, double[][][][] X, boolean StepSix){
NOSz = Scenario;
this.Diktz = D;
this.Probability = Prob;
this.LHSBinary = A;
```

```
this.RHSBinary = B;
this.XXijtzStep6 = X;
this.Step6 = StepSix;
}
public RobustTacticalModel(int scenario, double[][][][] D, double[] Prob, double[][][][] Xijt,
double[][][][] XPijt,double[][][][][] XSMijkt,double[][][][][] XSScsjt,double[][][][] IPSijt,
double[][][][] IPDikt,double[][][][] UDikt,double[][][][] ICScst,double[][][][] XCPcst,
double[][][] XMOTjt,double[][][] XSOTst, int titi){
NOSz = scenario;
this.Diktz = D;
this.Probability = Prob;
this.Step6 = false;
DETXPijtz= XPijt;
DETIPSijtz=IPSijt;
DETXXijtz=Xijt;
DETXSMijktz=XSMijkt;
DETIPDiktz=IPDikt;
DETUDiktz=UDikt;
DETXSScsjtz=XSScsjt;
DETICScstz=ICScst;
DETXCPcstz =XCPcst;
DETXMOTjtz =XMOTjt;
DETXSOTstz =XSOTst;
this.TITI = titi;
}
```

```
//PARAMETERS
static double[] Nt = RobustTacticalData.DataNt;
static final double[] PAj = RobustTacticalData.DataPAj;
static final double[][] JSAjt = RobustTacticalData.DataJSAjt;
static final double[][] KSAkt = RobustTacticalData.DataKSAkt;
static final double[][] MINij = RobustTacticalData.DataMINij;
static final double[][] MAXij = RobustTacticalData.DataMAXij;
static final double[][] ROci = RobustTacticalData.DataROci;
static final double[][] MRij = RobustTacticalData.DataMRij;
double[][][][] Diktz =
new double[NOPFi][NODk][NOTPt][NOSz];
static final double[] ALFAi = RobustTacticalData.DataALFAi;
static final double[] MOTj = RobustTacticalData.DataMOTj;
static final double[][][] TSDijk = RobustTacticalData.DataTSDijk;
static final double[][][] TSScsj = RobustTacticalData.DataTSScsj;
static final double[][] PIij = RobustTacticalData.DataPIij;
static final double[][] HJij = RobustTacticalData.DataHJij;
static final double[][] HDik = RobustTacticalData.DataHDik;
static final double[][] UCik = RobustTacticalData.DataUCik;
static final double[] OPj = RobustTacticalData.DataOPj;
static final double[][] INITIALINVENTORYij = RobustTacticalData.DataINITIALINVENTORYij;
static final double[] SAs = RobustTacticalData.DataSAs;
static final double[] SOTs = RobustTacticalData.DataSOTs;
static final double[][] SRcs = RobustTacticalData.DataSRcs;
static final double[][] INITIALINVENTORYcs =
RobustTacticalData.DataINITIALINVENTORYcs;
static final double[] BETAc = RobustTacticalData.DataBETAc;
```

```
static final double[][] SSAst = RobustTacticalData.DataSSAst;
static final double[][] CPScs = RobustTacticalData.DataCPScs;
static final double[][] HScs = RobustTacticalData.DataHScs;
static final double[] CSOTs = RobustTacticalData.DataCSOTs;
static final double[] CMOTj = RobustTacticalData.DataCMOTj;
double[][][][] Binaryijtz =
new double[NOPFi][NOMSj][NOTPt][NOSz];
double ObjectiveFunctionValue;
double[][][][] XPijtzValue =
new double[NOPFi][NOMSj][NOTPt][NOSz];
double[][][][][] XSMijktzValue =
new double[NOPFi][NOMSj][NODk][NOTPt][NOSz];
double[][][][][] XSScsjtzValue =
new double[NOCFc][NOSs][NOMSj][NOTPt][NOSz];
double[][][][] IPSijtzValue =
new double[NOPFi][NOMSj][NOTPt][NOSz];
double[][][][] IPDiktzValue =
new double[NOPFi][NODk][NOTPt][NOSz];
double[][][][] UDiktzValue =
new double[NOPFi][NODk][NOTPt][NOSz];
double[][][][] ICScstzValue =
new double[NOCFc][NOSs][NOTPt][NOSz];
double[][][][] XCPcstzValue =
new double[NOCFc][NOSs][NOTPt][NOSz];
double[][][] XMOTjtzValue =
new double[NOMSj][NOTPt][NOSz];
```

```
double[][][] XSOTstzValue =
new double[NOSs][NOTPt][NOSz];
//Degree of importance used for Variance
double OMEGA = 0;

public void MixedIntegermodel(){
try{
IloCplex cplex = new IloCplex();
IloNumVar[][][][] XPijtz =
new IloNumVar[NOPFi][NOMSj][NOTPt][NOSz];
for (int i = 0; i < NOPFi; i++ ){
for (int j = 0; j < NOMSj; j++){
for (int k = 0; k < NOTPt; k++){
XPijtz[i][j][k] = cplex.numVarArray(NOSz, 0.0, Double.MAX_VALUE);
}}}
IloNumVar[][][][][] XSMijktz =
new IloNumVar [NOPFi][NOMSj][NODk][NOTPt][NOSz];
for( int i = 0; i < NOPFi; i++){
for(int j = 0; j < NOMSj; j++){
for(int k = 0; k < NODk; k++){
for(int l = 0; l < NOTPt; l++){
XSMijktz[i][j][k][l] = cplex.numVarArray(NOSz, 0.0, Double.MAX_VALUE);
}}}}
IloNumVar[][][][][] XSScsjtz =
new IloNumVar[NOCFc][NOSs][NOMSj][NOTPt][NOSz];
for( int i = 0; i < NOCFc; i++){
```

```
for(int j = 0; j < NOSs; j++){
for(int k = 0; k < NOMSj; k++){
for(int l = 0; l < NOTPt; l++){
XSScsjtz[i][j][k][l] = cplex.numVarArray(NOSz, 0.0, Double.MAX_VALUE);
}}}}
IloNumVar[][][][] IPSijtz =
new IloNumVar[NOPFi][NOMSj][NOTPt][NOSz];
for (int i = 0; i < NOPFi; i++ ){
for (int j = 0; j < NOMSj; j++){
for (int k = 0; k < NOTPt; k++){
IPSijtz[i][j][k] = cplex.numVarArray(NOSz, 0.0, Double.MAX_VALUE);
}}}
IloNumVar[][][][] IPDiktz =
new IloNumVar[NOPFi][NODk][NOTPt][NOSz];
for (int i = 0; i < NOPFi; i++ ){
for (int j = 0; j < NODk; j++){
for (int k = 0; k < NOTPt; k++){
IPDiktz[i][j][k] = cplex.numVarArray(NOSz, 0.0, Double.MAX_VALUE);
}}}
IloNumVar[][][][] UDiktz =
new IloNumVar[NOPFi][NODk][NOTPt][NOSz];
for (int i = 0; i < NOPFi; i++ ){
for (int j = 0; j < NODk; j++){
for (int k = 0; k < NOTPt; k++){
UDiktz[i][j][k] = cplex.numVarArray(NOSz, 0.0, Double.MAX_VALUE);
}}}
```

```
IloNumVar[][][] XMOTjtz =
new IloNumVar[NOMSj][NOTPt][NOSz];
for (int j = 0; j < NOMSj; j++ ){
for (int t = 0; t < NOTPt; t++){
XMOTjtz[j][t] = cplex.numVarArray(NOSz, 0.0, Double.MAX_VALUE);
}}
IloNumVar[][][][] ICScstz =
new IloNumVar[NOCFc][NOSs][NOTPt][NOSz];
for(int c = 0; c < NOCFc; c++){
for(int s = 0; s < NOSs; s++){
for(int t = 0; t < NOTPt; t++){
ICScstz[c][s][t] = cplex.numVarArray(NOSz, 0.0, Double.MAX_VALUE);
}}}
IloNumVar[][][][] XCPcstz =
new IloNumVar[NOCFc][NOSs][NOTPt][NOSz];
for(int c = 0; c < NOCFc; c++){
for(int s = 0; s < NOSs; s++){
for(int t = 0; t < NOTPt; t++){
XCPcstz[c][s][t] = cplex.numVarArray(NOSz, 0.0, Double.MAX_VALUE);
}}}
IloNumVar[][][] XSOTstz =
new IloNumVar[NOSs][NOTPt][NOSz];
for(int s = 0; s < NOSs; s++){
for(int t = 0; t < NOTPt; t++){
XSOTstz[s][t] = cplex.numVarArray(NOSz, 0.0, Double.MAX_VALUE);
}}
```

```
IloNumVar[][][][] XXijtz =
new IloNumVar[NOPFi][NOMSj][NOTPt][NOSz];
for (int i = 0; i < NOPFi; i++ ){
for (int j = 0; j < NOMSj; j++){
for (int k = 0; k < NOTPt; k++){
XXijtz[i][j][k] = cplex.boolVarArray(NOSz);
}}}
//Constraints
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
for(int i =0 ; i < NOPFi; i++){
IloConstraint range = cplex.addLe(cplex.prod(MINij[i][j],
XXijtz[i][j][t][z]), XPijtz[i][j][t][z]);
}}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
for(int i =0 ; i < NOPFi; i++){
IloConstraint range = cplex.addLe(XPijtz[i][j][t][z],
cplex.prod(MAXij[i][j], XXijtz[i][j][t][z]));
}}}}
for(int z = 0; z < NOSz; z++){
for(int j = 0; j < NOMSj; j++){
for(int i =0 ; i < NOPFi; i++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
```

```
for (int k = 0; k < NODk; k++){
Expr1.addTerm(1.0, XSMijktz[i][j][k][0][z]);
}
IloConstraint range =
cplex.addEq(cplex.sum(IPSijtz[i][j][0][z],
cplex.prod(-1.0, XPijtz[i][j][0][z]),Expr1),INITIALINVENTORYij[i][j]);
}}}
for(int z = 0; z < NOSz; z++){
for(int t = 1; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
for(int i =0 ; i < NOPFi; i++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for (int k = 0; k < NODk; k++){
Expr1.addTerm(1.0, XSMijktz[i][j][k][t][z]);
}
IloConstraint range =
cplex.addEq(cplex.sum(IPSijtz[i][j][t][z],cplex.prod(-1.0, IPSijtz[i][j][t-1][z]),
cplex.prod(-1.0, XPijtz[i][j][t][z]),Expr1),0.0);
}}}}
for(int z = 0; z < NOSz; z++){
for (int k = 0; k < NODk; k++){
for(int i = 0; i< NOPFi; i++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for(int j = 0; j < NOMSj; j++){
Expr1.addTerm(-1.0, XSMijktz[i][j][k][0][z]);
}
```

```
IloConstraint range =
cplex.addEq(cplex.sum(IPDiktz[i][k][0][z],
cplex.prod(-1.0, UDiktz[i][k][0][z]), Expr1),-Diktz[i][k][0][z]);
}}}
for(int z = 0; z < NOSz; z++){
for(int t = 1; t < NOTPt; t++){
for (int k = 0; k < NODk; k++){
for(int i = 0; i< NOPFi; i++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for(int j = 0; j < NOMSj; j++){
Expr1.addTerm(-1.0, XSMijktz[i][j][k][t][z]);
}
IloConstraint range =
cplex.addEq(cplex.sum(IPDiktz[i][k][t][z],
cplex.prod(-1.0, IPDiktz[i][k][t-1][z]),
cplex.prod(-1.0, UDiktz[i][k][t][z]),
UDiktz[i][k][t-1][z] , Expr1),-Diktz[i][k][t][z]);
}}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for(int i = 0; i < NOPFi; i++){
Expr1.addTerm(MRij[i][j], XPijtz[i][j][t][z]);
}
IloConstraint range =
```

```
cplex.addLe(cplex.sum(Expr1,
cplex.prod(-1.0, XMOTjtz[j][t][z])), PAj[j]*Nt[t]);
}}}
for (int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
cplex.addLe(XMOTjtz[j][t][z], MOTj[j]*Nt[t]);
}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
for(int c = 0; c < NOCFc; c++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
IloLinearNumExpr Expr2 = cplex.linearNumExpr();
for(int i = 0; i < NOPFi; i++){
Expr1.addTerm(ROci[c][i], XPijtz[i][j][t][z]);
}
for (int s = 0; s < NOSs; s++){
Expr2.addTerm(1.0,XSScsjtz[c][s][j][t][z]);
}
IloConstraint range =
cplex.addEq(Expr2 , Expr1);
}}}}
for(int z = 0; z < NOSz; z++){
for (int s = 0; s < NOSs; s++){
for(int c = 0; c < NOCFc; c++){
```

```
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for(int j = 0; j < NOMSj; j++){
Expr1.addTerm(1.0,XSScsjtz[c][s][j][0][z]);
}
IloConstraint range =
cplex.addEq(cplex.sum(ICScstz[c][s][0][z],
cplex.prod(-1, XCPcstz[c][s][0][z]),Expr1), INITIALINVENTORYcs[c][s]);
}}}
for(int z = 0; z < NOSz; z++){
for(int t = 1; t < NOTPt; t++){
for (int s = 0; s < NOSs; s++){
for(int c = 0; c < NOCFc; c++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for(int j = 0; j < NOMSj; j++){
Expr1.addTerm(1.0,XSScsjtz[c][s][j][t][z]);
}
IloConstraint range =
cplex.addEq(cplex.sum(ICScstz[c][s][t][z],
cplex.prod(-1,ICScstz[c][s][t-1][z]),
cplex.prod(-1, XCPcstz[c][s][t][z]),Expr1), 0);
}}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for (int s = 0; s < NOSs; s++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for(int c = 0; c < NOCFc; c++){
```

```
Expr1.addTerm(SRcs[c][s], XCPcstz[c][s][t][z]);
}
IloConstraint range =
cplex.addLe(cplex.sum(Expr1,
cplex.prod(-1.0,XSOTstz[s][t][z])),SAs[s]*Nt[t]);
}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for (int s = 0; s < NOSs; s++){
IloConstraint range =
cplex.addLe(XSOTstz[s][t][z], SOTs[s]*Nt[t]);
}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for(int i = 0; i < NOPFi; i++){
Expr1.addTerm(ALFAi[i], IPSijtz[i][j][t][z]);
}
IloConstraint range =
cplex.addLe(Expr1, JSAjt[j][t]);
}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int k = 0; k < NODk; k++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
```

```
for(int i = 0; i < NOPFi; i++){
Expr1.addTerm(ALFAi[i], IPDiktz[i][k][t][z]);
}
IloConstraint range =
cplex.addLe(Expr1, KSAkt[k][t]);
}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for (int s = 0; s < NOSs; s++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for(int c = 0; c < NOCFc; c++){
Expr1.addTerm(BETAc[c], ICScstz[c][s][t][z]);
}
IloConstraint range =
cplex.addLe(Expr1, SSAst[s][t]);
}}}
if(this.TITI == 1){
for(int i = 0; i < RobustTacticalData.NumberOfProductFam; i++){
for(int j = 0; j < RobustTacticalData.NumberOfManuSite; j++){
IloConstraint range =
cplex.addEq(XPijtz[i][j][0][0], DETXPijtz[i][j][0][0]);
System.out.println("XPijtz"+i+""+j+"00 ="+ DETXPijtz[i][j][0][0]);
IloConstraint range1 =
cplex.addEq(XPijtz[i][j][1][0], DETXPijtz[i][j][1][0]);
IloConstraint range2 =
cplex.addEq(IPSijtz[i][j][0][0], DETIPSijtz[i][j][0][0]);
```

```
IloConstraint range3 =
cplex.addEq(XXijtz[i][j][0][0], DETXXijtz[i][j][0][0]);
IloConstraint range4 =
cplex.addEq(XXijtz[i][j][1][0], DETXXijtz[i][j][1][0]);
}}
for(int i = 0; i < NOPFi; i++){
for(int j = 0; j < NOMSj; j++){
for (int k = 0; k < NODk; k++){
IloConstraint range =
cplex.addEq(XSMijktz[i][j][k][0][0], DETXSMijktz[i][j][k][0][0]);
}}}
for(int i = 0; i < NOPFi; i++){
for(int k = 0; k < NODk; k++){
IloConstraint range =
cplex.addEq(IPDiktz[i][k][0][0], DETIPDiktz[i][k][0][0]);
IloConstraint range1 =
cplex.addEq(UDiktz[i][k][0][0], DETUDiktz[i][k][0][0]);
}}
for(int c = 0; c < NOCFc; c++){
for (int s = 0; s < NOSs; s++){
for(int j = 0; j < NOMSj; j++){
IloConstraint range =
cplex.addEq(XSScsjtz[c][s][j][0][0], DETXSScsjtz[c][s][j][0][0]);
}}}
for(int c = 0; c < NOCFc; c++){
for (int s = 0; s < NOSs; s++){
```

```
IloConstraint range =
cplex.addEq(ICScstz[c][s][0][0], DETICScstz[c][s][0][0]);
IloConstraint range1 =
cplex.addEq(XCPcstz[c][s][0][0], DETXCPcstz[c][s][0][0]);
IloConstraint range2 =
cplex.addEq(XCPcstz[c][s][1][0], DETXCPcstz[c][s][1][0]);
}}
for (int s = 0; s < NOSs; s++){
IloConstraint range =
cplex.addEq(XSOTstz[s][0][0], DETXSOTstz[s][0][0]);
IloConstraint range1 =
cplex.addEq(XSOTstz[s][1][0], DETXSOTstz[s][1][0]);
}
for(int j = 0; j < NOMSj; j++){
IloConstraint range =
cplex.addEq(XMOTjtz[j][0][0], DETXMOTjtz[j][0][0]);
IloConstraint range1 =
cplex.addEq(XMOTjtz[j][1][0], DETXMOTjtz[j][1][0]);
}
}
if(NOSz > 1){
for(int i = 0; i < NOPFi; i++){
for(int j = 0; j < NOMSj; j++){
for(int z = 0; z < NOSz-1; z++){
IloConstraint range =
cplex.addEq(XPijtz[i][j][0][z], XPijtz[i][j][0][z+1]);
```

```
IloConstraint range1 =
cplex.addEq(XPijtz[i][j][1][z], XPijtz[i][j][1][z+1]);
IloConstraint range2 =
cplex.addEq(IPSijtz[i][j][0][z], IPSijtz[i][j][0][z+1]);
IloConstraint range3 =
cplex.addEq(XXijtz[i][j][0][z], XXijtz[i][j][0][z+1]);
IloConstraint range4 =
cplex.addEq(XXijtz[i][j][1][z], XXijtz[i][j][1][z+1]);
}}}
for(int i = 0; i < NOPFi; i++){
for(int j = 0; j < NOMSj; j++){
for (int k = 0; k < NODk; k++){
for(int z = 0; z < NOSz-1; z++){
IloConstraint range =
cplex.addEq(XSMijktz[i][j][k][0][z], XSMijktz[i][j][k][0][z+1]);
}}}}
for(int i = 0; i < NOPFi; i++){
for(int k = 0; k < NODk; k++){
for(int z = 0; z < NOSz-1; z++){
IloConstraint range =
cplex.addEq(IPDiktz[i][k][0][z], IPDiktz[i][k][0][z+1]);
IloConstraint range1 =
cplex.addEq(UDiktz[i][k][0][z], UDiktz[i][k][0][z+1]);
}}}
for(int c = 0; c < NOCFc; c++){
for (int s = 0; s < NOSs; s++){
```

```
for(int j = 0; j < NOMSj; j++){
for(int z = 0; z < NOSz-1; z++){
IloConstraint range =
cplex.addEq(XSScsjtz[c][s][j][0][z], XSScsjtz[c][s][j][0][z+1]);
}}}}
for(int c = 0; c < NOCFc; c++){
for (int s = 0; s < NOSs; s++){
for(int z = 0; z < NOSz-1; z++){
IloConstraint range =
cplex.addEq(ICScstz[c][s][0][z], ICScstz[c][s][0][z+1]);
IloConstraint range1 =
cplex.addEq(XCPcstz[c][s][0][z], XCPcstz[c][s][0][z+1]);
IloConstraint range2 =
cplex.addEq(XCPcstz[c][s][1][z], XCPcstz[c][s][1][z+1]);
}}}
for (int s = 0; s < NOSs; s++){
for(int z = 0; z < NOSz-1; z++){
IloConstraint range =
cplex.addEq(XSOTstz[s][0][z], XSOTstz[s][0][z+1]);
IloConstraint range1 =
cplex.addEq(XSOTstz[s][1][z], XSOTstz[s][1][z+1]);
}}
for(int j = 0; j < NOMSj; j++){
for(int z = 0; z < NOSz-1; z++){
IloConstraint range =
cplex.addEq(XMOTjtz[j][0][z], XMOTjtz[j][0][z+1]);
```

```
IloConstraint range1 =
cplex.addEq(XMOTjtz[j][1][z], XMOTjtz[j][1][z+1]);
}}
}
IloNumVar[] OBSz =
new IloNumVar[NOSz];
OBSz = cplex.numVarArray(NOSz, 0.0, Double.MAX_VALUE);
IloNumVar[] TetaPositivez =
new IloNumVar[NOSz];
TetaPositivez =
cplex.numVarArray(NOSz, 0.0, Double.MAX_VALUE);
IloNumVar[] TetaNegativez =
new IloNumVar[NOSz];
TetaNegativez =
cplex.numVarArray(NOSz, 0.0, Double.MAX_VALUE);
IloNumVar[] STD =
new IloNumVar[1];
STD = cplex.numVarArray(1, 0.0, Double.MAX_VALUE);
//Building the objective function of Each Scenario
//in order to develop the main Objective Function
for(int z = 0; z < NOSz; z++){
IloLinearNumExpr obj = cplex.linearNumExpr();
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
for(int i = 0; i < NOPFi; i++){
double coff = Nt[t]*HJij[i][j];
```

```
obj.addTerm(coff, IPSijtz[i][j][t][z]);
}}}
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
for(int i = 0; i < NOPFi; i++){
double coff = PIij[i][j];
obj.addTerm(coff, XPijtz[i][j][t][z]);
}}}
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
double coff = CMOTj[j];
obj.addTerm(coff, XMOTjtz[j][t][z]);
}}
for(int t = 0; t < NOTPt; t++){
for (int s = 0; s < NOSs; s++){
for(int c = 0; c < NOCFc; c++){
double coff = CPScs[c][s];
obj.addTerm(coff, XCPcstz[c][s][t][z]);
}}}
for(int t = 0; t < NOTPt; t++){
for (int s = 0; s < NOSs; s++){
for(int c = 0; c < NOCFc; c++){
double coff = Nt[t]*HScs[c][s];
obj.addTerm(coff, ICScstz[c][s][t][z]);
}}}
for(int t = 0; t < NOTPt; t++){
```

```
for (int s = 0; s < NOSs; s++){
double coff = CSOTs[s];
obj.addTerm(coff, XSOTstz[s][t][z]);
}}
for(int t = 0; t < NOTPt; t++){
for (int k = 0; k < NODk; k++){
for(int j = 0; j < NOMSj; j++){
for(int i = 0; i < NOPFi; i++){
double coff = TSDijk[i][j][k];
obj.addTerm(coff, XSMijktz[i][j][k][t][z]);
}}}}
for(int t = 0; t < NOTPt; t++){
for (int k = 0; k < NODk; k++){
for(int i = 0; i< NOPFi; i++){
double coff = UCik[i][k];
obj.addTerm(coff, UDiktz[i][k][t][z]);
}}}
for(int t = 0; t < NOTPt; t++){
for(int k = 0; k < NODk; k++){
for(int i = 0; i< NOPFi; i++){
double coff = Nt[t]*HDik[i][k];
obj.addTerm(coff, IPDiktz[i][k][t][z]);
}}}
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
for (int s = 0; s < NOSs; s++){
```

```
for(int c = 0; c < NOCFc; c++){
double coff = TSScsj[c][s][j];
obj.addTerm(coff, XSScsjtz[c][s][j][t][z]);
}}}}
IloConstraint range =
cplex.addEq(OBSz[z], obj);
}
// Building up the constraints in order to handle
//the nonlinear objective function
IloLinearNumExpr ExpectedValue = cplex.linearNumExpr();
for(int z = 0; z < NOSz; z++){
ExpectedValue.addTerm(Probability[z],OBSz[z]);
}
for(int z = 0; z < NOSz; z++){
IloConstraint range =
cplex.addEq(cplex.sum(OBSz[z],cplex.prod(-1, ExpectedValue),
cplex.prod(-1, TetaPositivez[z]), TetaNegativez[z]), 0);
}
IloLinearNumExpr FinalObjectiveFunction = cplex.linearNumExpr();
IloLinearNumExpr TotalSTD = cplex.linearNumExpr();
for(int z = 0; z < NOSz; z++){
//Adding Expected Value to the Objective Function
FinalObjectiveFunction.addTerm(Probability[z],OBSz[z]);
//Adding Variance to the Objective Function
//with a degree of importance Called OMEGA
double Coefficient = Probability[z]*OMEGA;
```

```
FinalObjectiveFunction.addTerm(Coefficient, TetaPositivez[z]);
FinalObjectiveFunction.addTerm(Coefficient, TetaNegativez[z]);
TotalSTD.addTerm(Probability[z], TetaPositivez[z]);
TotalSTD.addTerm(Probability[z], TetaNegativez[z]);
}
cplex.addEq(cplex.sum(STD[0],cplex.prod(-1, TotalSTD)), 0);
cplex.addMinimize(FinalObjectiveFunction);
// Solve model
cplex.solve();
System.out.println("Solution status = " + cplex.getStatus());
if ( cplex.solve()) {
double[][][][] Xijtz =
new double[NOPFi][NOMSj][NOTPt][NOSz];
double[][][][] XXPijtz =
new double[NOPFi][NOMSj][NOTPt][NOSz];
double[][][][] IIPSijtz =
new double[NOPFi][NOMSj][NOTPt][NOSz];
double[][][] XXMOTjtz =
new double[NOMSj][NOTPt][NOSz];
double[][][][][] XXSMijktz =
new double[NOPFi][NOMSj][NODk][NOTPt][NOSz];
double[][][][][] XXSScsjtz =
new double[NOCFc][NOSs][NOMSj][NOTPt][NOSz];
double[][][][] IIPDiktz =
new double[NOPFi][NODk][NOTPt][NOSz];
double[][][][] UUDiktz =
```

```
new double[NOPFi][NODk][NOTPt][NOSz];
double[][][][] IICScstz =
new double[NOCFc][NOSs][NOTPt][NOSz];
double[][][][] XXCPcstz =
new double[NOCFc][NOSs][NOTPt][NOSz];
double[][][] XXSOTstz =
new double[NOSs][NOTPt][NOSz];
System.out.println();
System.out.println("Solution status = " + cplex.getStatus());
System.out.println();
double ObjFunValue = cplex.getObjValue();
System.out.println(" cost = " + cplex.getObjValue());
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
for(int i = 0; i < NOPFi; i++){
Xijtz[i][j][t][z] = cplex.getValue(XXijtz[i][j][t][z]);
System.out.println("XX" + i + ""+ j + ""+ t +""+z+ " = "
+cplex.getValue(XXijtz[i][j][t][z]));
}}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int k = 0; k < NODk; k++){
for(int i = 0; i < NOPFi; i++){
UUDiktz[i][k][t][z] = cplex.getValue(UDiktz[i][k][t][z]);
System.out.println("UD" + i + ""+ k + ""+ t +""+z+ " = " +
```

```
cplex.getValue(UDiktz[i][k][t][z]));
}}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
for(int i = 0; i < NOPFi; i++){
XXPijtz[i][j][t][z] = cplex.getValue(XPijtz[i][j][t][z]);
System.out.println("XP" + i + ""+ j + ""+ t +""+z+ " = " +
cplex.getValue(XPijtz[i][j][t][z]));
}}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
for(int i = 0; i < NOPFi; i++){
IIPSijtz[i][j][t][z] = cplex.getValue(IPSijtz[i][j][t][z]);
System.out.println("IPS" + i + ""+ j + ""+ t +""+z+ " = " +
cplex.getValue(IPSijtz[i][j][t][z]));
}}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
XXMOTjtz[j][t][z] = cplex.getValue(XMOTjtz[j][t][z]);
System.out.println("XMOT"+ j + ""+ t +""+z+ " = " +
cplex.getValue(XMOTjtz[j][t][z]));
}}}
for(int z = 0; z < NOSz; z++){
```

```
for(int t = 0; t < NOTPt; t++){
for(int k = 0; k < NODk; k++){
for(int j = 0; j < NOMSj; j++){
for(int i = 0; i < NOPFi; i++){
XXSMijktz[i][j][k][t][z] = cplex.getValue(XSMijktz[i][j][k][t][z]);
System.out.println("XSM" + i + ""+ j + ""+k+""+t +""+z+ " = " +
cplex.getValue(XSMijktz[i][j][k][t][z]));
}}}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int j = 0; j < NOMSj; j++){
for (int s = 0; s < NOSs; s++){
for(int c = 0; c < NOCFc; c++){
XXSScsjtz[c][s][j][t][z] = cplex.getValue(XSScsjtz[c][s][j][t][z]);
System.out.println("XSS" + c + ""+ s + ""+j+""+t +""+z+ " = " +
cplex.getValue(XSScsjtz[c][s][j][t][z]));
}}}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for(int k = 0; k < NODk; k++){
for(int i = 0; i < NOPFi; i++){
IIPDiktz[i][k][t][z] = cplex.getValue(IPDiktz[i][k][t][z]);
System.out.println("IPD" + i + ""+ k + ""+ t +""+z+ " = " +
cplex.getValue(IPDiktz[i][k][t][z]));
}}}}
for(int z = 0; z < NOSz; z++){
```

```
for(int t = 0; t < NOTPt; t++){
for (int s = 0; s < NOSs; s++){
for(int c = 0; c < NOCFc; c++){
IICScstz[c][s][t][z] = cplex.getValue(ICScstz[c][s][t][z]);
System.out.println("ICS" + c + ""+ s + ""+t +""+z+ " = " +
cplex.getValue(ICScstz[c][s][t][z]));
}}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for (int s = 0; s < NOSs; s++){
for(int c = 0; c < NOCFc; c++){
XXCPcstz[c][s][t][z] = cplex.getValue(XCPcstz[c][s][t][z]);
System.out.println("XCP" + c + ""+ s + ""+t +""+z+ " = " +
cplex.getValue(XCPcstz[c][s][t][z]));
}}}}
for(int z = 0; z < NOSz; z++){
for(int t = 0; t < NOTPt; t++){
for (int s = 0; s < NOSs; s++){
XXSOTstz[s][t][z] = cplex.getValue(XSOTstz[s][t][z]);
System.out.println("XSOT"+ s + ""+t +""+z+ " = " +
cplex.getValue(XSOTstz[s][t][z]));
}}}
for(int z = 0; z < NOSz; z++){
System.out.println("OBSz"+ z +" = " + cplex.getValue(OBSz[z]));
}
System.out.println("STD = " + cplex.getValue(STD[0]));
```

```
this.Binaryijtz = Xijtz;
this.ObjectiveFunctionValue = ObjFunValue;
this.XPijtzValue = XXPijtz;
this.IPSijtzValue = IIPSijtz;
this.XMOTjtzValue = XXMOTjtz;
this.XSMijktzValue = XXSMijktz;
this.XSScsjtzValue = XXSScsjtz;
this.IPDiktzValue = IIPDiktz;
this.UDiktzValue = UUDiktz;
this.ICScstzValue = IICScstz;
this.XCPcstzValue = XXCPcstz;
this.XSOTstzValue = XXSOTstz;
}
cplex.end();
}
catch (IloException ex) {
System.out.println("Concert Error: " + ex);
}
}
public double[][][][] returnBinary(){
return this.Binaryijtz;
}
public double returnObjectiveFunctionValue(){
return this.ObjectiveFunctionValue;
}
public double[][][][] returnXP(){
```

```
return this.XPijtzValue;
}
public double[][][][][] returnXSM(){
return this.XSMijktzValue;
}
public double[][][][][] returnXSS(){
return this.XSScsjtzValue;
}
public double[][][][] returnIPS(){
return this.IPSijtzValue;
}
public double[][][][] retunIPD(){
return this.IPDiktzValue;
}
public double[][][][] retunUD(){
return this.UDiktzValue;
}
public double[][][][] retunICS(){
return this.ICScstzValue;
}
public double[][][][] returnXCP(){
return this.XCPcstzValue;
}
public double[][][] returnXMOT(){
return this.XMOTjtzValue;
}
```

```
public double[][][] returnXSOT(){
return this.XSOTstzValue;
}
}
```

Appendix B – Java-Cplex code for operational SC planning model

```
import ilog.concert.*;
import java.lang.*;
import ilog.cplex.IloCplex;
import java.util.*;
public class OperationalModel {
double[][] XXij1= new double[OperationalData.NumofProductFamily]
[OperationalData.NumofManuSite];
double[][] IPDik1 = new double[OperationalData.NumofProductFamily]
[OperationalData.NumofDC];
double[][] IPSij1 = new double[OperationalData.NumofProductFamily]
[OperationalData.NumofManuSite];
public OperationalModel(double[][] Xij1, double[][] IPDDik1, double[][] IPSSij1){
this.XXij1 = Xij1;
this.IPDik1 = IPDDik1;
this.IPSij1 = IPSSij1;
}
double[][][][] Xiljjt =
new
double[OperationalData.NumofProductFamily][OperationalData.MaximumNumberofManuLines]
[OperationalData.NumofManuSite][OperationalData.NumofOperationalPeriod];
double[][][][][] XSPpiiljjt =
new
double[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily]
[OperationalData.MaximumNumberofManuLines][OperationalData.NumofManuSite][OperationalD
ata.NumofOperationalPeriod];
double[][][][] XSUiljjt =
new
double[OperationalData.NumofProductFamily][OperationalData.MaximumNumberofManuLines]
[OperationalData.NumofManuSite][OperationalData.NumofOperationalPeriod];
double[][][][][] SSMpiijkt =
```

```
new
double[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily]
[OperationalData.NumofManuSite][OperationalData.NumofDC][OperationalData.NumofOperationa
lPeriod];
double[][][][] IPpiijt =
new
double[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily]
[OperationalData.NumofManuSite][OperationalData.NumofOperationalPeriod];
double[][] SIPij =
new double[OperationalData.NumofProductFamily][OperationalData.NumofManuSite];
double[][] SDCik =
new double[OperationalData.NumofProductFamily][OperationalData.NumofDC];
double[][][][] IPDpiikt =
new
double[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily]
[OperationalData.NumofDC][OperationalData.NumofOperationalPeriod];
double[][][][] UDpiikt =
new
double[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily]
[OperationalData.NumofDC][OperationalData.NumofOperationalPeriod];
double[][][] XMOTljjt =
new double[OperationalData.MaximumNumberofManuLines][OperationalData.NumofManuSite]
[OperationalData.NumofOperationalPeriod];
public void OperationalModelMath(){
try{
final double[][][][] Dpiikt =
new
double[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily]
[OperationalData.NumofDC][OperationalData.NumofOperationalPeriod];
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
Dpiikt[pi][i][k][t] = OperationalData.Wpiikt[pi][i][k][t]*OperationalData.Dik1[i][k];
```

```
}}}}
IloCplex cplex = new IloCplex();
IloNumVar[][][][] Xiljjt =
new
IloNumVar[OperationalData.NumofProductFamily][OperationalData.MaximumNumberofManuLine
s]
[OperationalData.NumofManuSite][OperationalData.NumofOperationalPeriod];
for(int i = 0; i < Xiljjt.length; i++){
for(int j = 0; j < Xiljjt[i].length; j++){
for( int lj = 0; lj < Xiljjt[i][j].length; lj++){
Xiljjt[i][j][lj] = cplex.boolVarArray(OperationalData.NumofOperationalPeriod);
}}}
IloNumVar[][][][][] XSPpiiljjt =
new
IloNumVar[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily
]
[OperationalData.MaximumNumberofManuLines][OperationalData.NumofManuSite][OperationalD
ata.NumofOperationalPeriod];
for(int pi = 0; pi < XSPpiiljjt.length; pi++){
for(int i = 0; i < XSPpiiljjt[pi].length; i++){
for(int lj = 0; lj < XSPpiiljjt[pi][i].length; lj++){
for(int j = 0; j < XSPpiiljjt[pi][i][lj].length; j++){
XSPpiiljjt[pi][i][lj][j] = cplex.numVarArray(OperationalData.NumofOperationalPeriod, 0,
Double.MAX_VALUE);
}}}}
IloNumVar[][][][] XSUiljjt =
new
IloNumVar[OperationalData.NumofProductFamily][OperationalData.MaximumNumberofManuLine
s]
[OperationalData.NumofManuSite][OperationalData.NumofOperationalPeriod];
for(int i = 0; i < XSUiljjt.length; i++){
for(int lj = 0; lj < XSUiljjt[i].length; lj++){
for(int j = 0; j < XSUiljjt[i][lj].length; j++){
XSUiljjt[i][lj][j] = cplex.boolVarArray(OperationalData.NumofOperationalPeriod);
```

```
}}}
IloNumVar[][][][][] SSMpiijkt =
new
IloNumVar[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily
]
[OperationalData.NumofManuSite][OperationalData.NumofDC][OperationalData.NumofOperationa
lPeriod];
for(int pi = 0; pi < SSMpiijkt.length; pi++){
for(int i = 0; i < SSMpiijkt[pi].length; i++){
for(int j = 0; j < SSMpiijkt[pi][i].length; j++){
for(int k = 0; k < SSMpiijkt[pi][i][j].length; k++){
SSMpiijkt[pi][i][j][k] = cplex.numVarArray(OperationalData.NumofOperationalPeriod, 0,
Double.MAX_VALUE);
}}}}
IloNumVar[][][][] IPpiijt =
new
IloNumVar[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily
]
[OperationalData.NumofManuSite][OperationalData.NumofOperationalPeriod];
for(int pi = 0; pi < IPpiijt.length; pi++){
for(int i = 0; i < IPpiijt[pi].length; i++){
for(int j = 0; j < IPpiijt[pi][i].length; j++){
IPpiijt[pi][i][j] = cplex.numVarArray(OperationalData.NumofOperationalPeriod, 0,
Double.MAX_VALUE);
}}}
IloNumVar[][] SIPij =
new IloNumVar[OperationalData.NumofProductFamily][OperationalData.NumofManuSite];
for ( int i = 0; i < SIPij.length; i++){
SIPij[i] = cplex.numVarArray(OperationalData.NumofManuSite, 0, Double.MAX_VALUE);
}
IloNumVar[][] SDCik =
new IloNumVar[OperationalData.NumofProductFamily][OperationalData.NumofDC];
for ( int i = 0; i < SDCik.length; i++){
SDCik[i] = cplex.numVarArray(OperationalData.NumofDC, 0, Double.MAX_VALUE);
```

```
}
IloNumVar[][][][] IPDpiikt =
new
IloNumVar[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily
]
[OperationalData.NumofDC][OperationalData.NumofOperationalPeriod];
for(int pi = 0; pi < IPDpiikt.length; pi++){
for(int i = 0; i < IPDpiikt[pi].length; i++){
for ( int k = 0; k < IPDpiikt[pi][i].length; k++){
IPDpiikt[pi][i][k] = cplex.numVarArray(OperationalData.NumofOperationalPeriod, 0,
Double.MAX_VALUE);
}}}
IloNumVar[][][][] UDpiikt =
new
IloNumVar[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily
]
[OperationalData.NumofDC][OperationalData.NumofOperationalPeriod];
for(int pi = 0; pi < UDpiikt.length; pi++){
for(int i = 0; i < UDpiikt[pi].length; i++){
for ( int k = 0; k < UDpiikt[pi][i].length; k++){
UDpiikt[pi][i][k] = cplex.numVarArray(OperationalData.NumofOperationalPeriod, 0,
Double.MAX_VALUE);
}}}
IloNumVar[][][] XMOTljjt =
new
IloNumVar[OperationalData.MaximumNumberofManuLines][OperationalData.NumofManuSite]
[OperationalData.NumofOperationalPeriod];
for(int lj = 0; lj < XMOTljjt.length; lj++){
for ( int j = 0; j < XMOTljjt[lj].length; j++){
XMOTljjt[lj][j] = cplex.numVarArray(OperationalData.NumofOperationalPeriod, 0,
Double.MAX_VALUE);
}}
IloNumVar[][][][] Liljjt =
```

```
new
IloNumVar[OperationalData.NumofProductFamily][OperationalData.MaximumNumberofManuLine
s]
[OperationalData.NumofManuSite][OperationalData.NumofOperationalPeriod];
for( int i = 0; i < Liljjt.length; i++){
for ( int lj = 0; lj < Liljjt[i].length; lj++){
for ( int j = 0; j < Liljjt[i][lj].length; j++){
Liljjt[i][lj][j] = cplex.boolVarArray(OperationalData.NumofOperationalPeriod);
}}}
IloNumVar[][][][] Uiljjt =
new
IloNumVar[OperationalData.NumofProductFamily][OperationalData.MaximumNumberofManuLine
s]
[OperationalData.NumofManuSite][OperationalData.NumofOperationalPeriod];
for( int i = 0; i < Uiljjt.length; i++){
for ( int lj = 0; lj < Uiljjt[i].length; lj++){
for ( int j = 0; j < Uiljjt[i][lj].length; j++){
Uiljjt[i][lj][j] = cplex.boolVarArray(OperationalData.NumofOperationalPeriod);
}}}
IloNumVar[][][] Yljjt =
new
IloNumVar[OperationalData.MaximumNumberofManuLines][OperationalData.NumofManuSite]
[OperationalData.NumofOperationalPeriod];
for ( int lj = 0; lj < Yljjt.length; lj++){
for ( int j = 0; j < Yljjt[lj].length; j++){
Yljjt[lj][j] = cplex.boolVarArray(OperationalData.NumofOperationalPeriod);
}}
IloNumVar[][] XPOperationaLij1 =
new IloNumVar[OperationalData.NumofProductFamily][OperationalData.NumofManuSite];
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
XPOperationaLij1[i] = cplex.numVarArray(OperationalData.NumofManuSite, 0,
Double.MAX_VALUE);
}
//Building the Constraints
```

```
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for (int j = 0; j < OperationalData.NumofManuSite; j++){
for (int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for( int i = 0; i < OperationalData.NumofProductFamily; i++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for( int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
Expr1.addTerm(-1.0, XSPpiiljjt[pi][i][lj][j][t]);
}
IloConstraint range =
cplex.addLe(cplex.sum(cplex.prod(OperationalData.MinProcustioniljj[i][lj][j],
Xiljjt[i][lj][j][t]),Expr1), 0);
}}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for (int j = 0; j < OperationalData.NumofManuSite; j++){
for (int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for( int i = 0; i < OperationalData.NumofProductFamily; i++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for( int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
Expr1.addTerm(1.0, XSPpiiljjt[pi][i][lj][j][t]);
}
IloConstraint range =
cplex.addLe(cplex.sum(cplex.prod(-1000000, Xiljjt[i][lj][j][t]),Expr1), 0);
}}}}
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for (int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for (int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
Expr1.addTerm(1, Xiljjt[i][lj][j][t]);
}}
IloConstraint range =
cplex.addLe(Expr1,10000*XXij1[i][j]);
}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
```

```
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
Expr1.addTerm(1, Liljjt[i][lj][j][t]);
}
cplex.addEq(Expr1, 1);
}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
Expr1.addTerm(1, Uiljjt[i][lj][j][t]);
}
cplex.addEq(Expr1, 1);
}}}
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
cplex.addLe(cplex.sum(Liljjt[i][lj][j][t],Uiljjt[i][lj][j][t],
cplex.prod(-2,Uiljjt[i][lj][j][t-1]),cplex.prod(-100,Yljjt[lj][j][t])),0);
cplex.addGe(cplex.sum(Liljjt[i][lj][j][t],Uiljjt[i][lj][j][t],
cplex.prod(-2,Uiljjt[i][lj][j][t-1]),cplex.prod(100,Yljjt[lj][j][t])),0);
cplex.addLe(cplex.sum(Liljjt[i][lj][j][t],Uiljjt[i][lj][j][t],
cplex.prod(-2,Xiljjt[i][lj][j][t]),cplex.prod(100,Yljjt[lj][j][t])),100);
}}}}
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
IloLinearNumExpr TotalXiljjt = cplex.linearNumExpr();
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
TotalXiljjt.addTerm(1, Xiljjt[i][lj][j][t]);
```

```
}
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
cplex.addLe(cplex.sum(Liljjt[i][lj][j][t],Uiljjt[i][lj][j][t],
cplex.prod(0.000001,TotalXiljjt),cplex.prod(100000,Yljjt[lj][j][t])),100002.000001);
}}}}
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
IloLinearNumExpr TotalXiljjt = cplex.linearNumExpr();
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
TotalXiljjt.addTerm(1, Xiljjt[i][lj][j][t]);
}
cplex.addGe(cplex.sum(Yljjt[lj][j][t], cplex.prod(-0.00001,TotalXiljjt)), -0.000005);
cplex.addLe(cplex.sum(Yljjt[lj][j][t], cplex.prod(-1, TotalXiljjt)), 0);
}}}
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
double coef = (double)1/3;
cplex.addLe(cplex.sum(XSUiljjt[i][lj][j][t],cplex.prod(-1, Xiljjt[i][lj][j][t])),0);
cplex.addGe(cplex.sum(XSUiljjt[i][lj][j][t],cplex.prod(-100000, Xiljjt[i][lj][j][t]),
cplex.prod((double)1/3, cplex.sum(Xiljjt[i][lj][j][t],Liljjt[i][lj][j][t],Uiljjt[i][lj][j][t-1]))),-99999);
cplex.addLe(cplex.sum(XSUiljjt[i][lj][j][t], Xiljjt[i][lj][j][t],Liljjt[i][lj][j][t],
Uiljjt[i][lj][j][t-1],cplex.prod(100000, Xiljjt[i][lj][j][t])), 100003);
}}}}
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
IloLinearNumExpr Expr = cplex.linearNumExpr();
for(int j = 0; j < OperationalData.NumofManuSite; j++){
Expr.addTerm(-1, SSMpiijkt[pi][i][j][k][1]);
}
```

```
cplex.addEq(cplex.sum(IPDpiikt[pi][i][k][1], cplex.prod(-1, UDpiikt[pi][i][k][1]),Expr), -
Dpiikt[pi][i][k][1]);
}}}
for(int t = 2; t < OperationalData.NumofOperationalPeriod; t++){
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
IloLinearNumExpr Expr = cplex.linearNumExpr();
for(int j = 0; j < OperationalData.NumofManuSite; j++){
Expr.addTerm(-1, SSMpiijkt[pi][i][j][k][t]);
}
cplex.addEq(cplex.sum(IPDpiikt[pi][i][k][t], cplex.prod(-1,UDpiikt[pi][i][k][t]),
cplex.prod(-1, IPDpiikt[pi][i][k][t-1]), UDpiikt[pi][i][k][t-1], Expr), -Dpiikt[pi][i][k][t]);
}}}}
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
IloLinearNumExpr Expr = cplex.linearNumExpr();
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
Expr.addTerm(1, IPDpiikt[pi][i][k][OperationalData.NumofOperationalPeriod-1]);
}
cplex.addGe(cplex.sum(Expr, SDCik[i][k]), IPDik1[i][k]);
}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int k = 0; k < OperationalData.NumofDC; k++){
IloLinearNumExpr Expr = cplex.linearNumExpr();
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
Expr.addTerm(OperationalData.ALFAi[i], IPDpiikt[pi][i][k][t]);
}}
cplex.addLe(Expr, OperationalData.KSAk[k]);
}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
```

```
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
IloLinearNumExpr Expr2 = cplex.linearNumExpr();
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
Expr1.addTerm(OperationalData.MRij[i][j], XSPpiiljjt[pi][i][lj][j][t]);
Expr2.addTerm(OperationalData.STiljj[i][lj][j], XSUiljjt[i][lj][j][t]);
}}
cplex.addLe(cplex.sum(Expr1, cplex.prod(OperationalData.SAljj[lj][j]*0.2/* it is multiplied by
two because SA is for 5 periods */, Expr2), cplex.prod(-1, XMOTljjt[lj][j][t])),
0.2*OperationalData.SAljj[lj][j]);
}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
cplex.addLe(XMOTljjt[lj][j][t], OperationalData.MOTljjt[lj][j][t]);
}}}
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
IloLinearNumExpr Expr2 = cplex.linearNumExpr();
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
Expr1.addTerm(-1, XSPpiiljjt[pi][i][lj][j][1]);
}
for(int k = 0; k < OperationalData.NumofDC; k++){
Expr2.addTerm(1, SSMpiijkt[pi][i][j][k][1]);
}
cplex.addEq(cplex.sum(IPpiijt[pi][i][j][1],Expr1, Expr2), 0);
}}}
for(int t = 2; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
```

```
IloLinearNumExpr Expr2 = cplex.linearNumExpr();
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
Expr1.addTerm(-1, XSPpiiljjt[pi][i][lj][j][t]);
}
for(int k = 0; k < OperationalData.NumofDC; k++){
Expr2.addTerm(1, SSMpiijkt[pi][i][j][k][t]);
}
cplex.addEq(cplex.sum(IPpiijt[pi][i][j][t], cplex.prod(-1, IPpiijt[pi][i][j][t-1]), Expr1, Expr2), 0);
}}}}
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
Expr1.addTerm(1, IPpiijt[pi][i][j][OperationalData.NumofOperationalPeriod-1]);
}
cplex.addGe(cplex.sum(Expr1, SIPij[i][j]), IPSij1[i][j]);
}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
Expr1.addTerm(OperationalData.ALFAi[i], IPpiijt[pi][i][j][t]);
}
cplex.addLe(Expr1, OperationalData.JSAj[j]);
}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
cplex.addEq(Xiljjt[0][lj][j][t], 0);
                                                    }}}
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
cplex.addEq(Liljjt[0][lj][j][0], 1);
```

```
cplex.addEq(Uiljjt[0][lj][j][0], 1);
}}
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int i = 1; i < OperationalData.NumofProductFamily; i++){
cplex.addEq(Xiljjt[i][lj][j][0], 0);
}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
cplex.addEq(Xiljjt[i][3][0][t], 0);
cplex.addEq(Xiljjt[i][2][1][t], 0);
cplex.addEq(Xiljjt[i][3][1][t], 0);
cplex.addEq(Xiljjt[i][2][2][t], 0);
cplex.addEq(Xiljjt[i][3][2][t], 0);
}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
cplex.addEq(XSPpiiljjt[2][1][lj][j][t], 0);
cplex.addEq(XSPpiiljjt[2][2][lj][j][t], 0);
cplex.addEq(XSPpiiljjt[2][4][lj][j][t], 0);
}}}
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
IloLinearNumExpr Expr1 = cplex.linearNumExpr();
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
Expr1.addTerm(1, XSPpiiljjt[pi][i][lj][j][t]);
}
}
}
cplex.addEq(XPOperationaLij1[i][j], Expr1);
}}
```

```
//Objective Function
IloLinearNumExpr obj = cplex.linearNumExpr();
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for( int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++)
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
obj.addTerm(OperationalData.PIpiiljj[pi][i][lj][j], XSPpiiljjt[pi][i][lj][j][t]);
}}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for( int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++)
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
obj.addTerm(OperationalData.SUCiljj[i][lj][j], XSUiljjt[i][lj][j][t]);
}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
obj.addTerm(OperationalData.HJpiij[pi][i][j], IPpiijt[pi][i][j][t]);
}}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
obj.addTerm(OperationalData.HDpiik[pi][i][k], IPDpiikt[pi][i][k][t]);
}}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
obj.addTerm(OperationalData.TSDijk[i][j][k], SSMpiijkt[pi][i][j][k][t]);
}}}}}
```

```
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
obj.addTerm(OperationalData.UDCpiik[pi][i][k], UDpiikt[pi][i][k][t]);
}}}}
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
obj.addTerm(OperationalData.SIDik[i][k], SDCik[i][k]);
}}
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
obj.addTerm(OperationalData.SICij[i][j], SIPij[i][j]);
}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for( int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
obj.addTerm(OperationalData.COTljj[lj][j], XMOTljjt[lj][j][t]);
}}}
cplex.addMinimize(obj);
//Solve Model
cplex.solve();
System.out.println("Solution status = " + cplex.getStatus());
if ( cplex.solve()) {
System.out.println();
System.out.println("Solution status = " + cplex.getStatus());
System.out.println();
double ObjFunValue = cplex.getObjValue();
System.out.println(" cost = " + cplex.getObjValue());
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for( int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
this.Xiljjt[i][lj][j][t] = cplex.getValue(Xiljjt[i][lj][j][t]);
```

```
System.out.println("X" + i + ""+ lj + ""+ j +""+t+ " = " + cplex.getValue(Xiljjt[i][lj][j][t]));
}}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for( int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
this.XSPpiiljjt[pi][i][lj][j][t] = cplex.getValue(XSPpiiljjt[pi][i][lj][j][t]);
System.out.println("XSP" + pi +""+ i + ""+ lj + ""+ j +""+t+ " = " +
cplex.getValue(XSPpiiljjt[pi][i][lj][j][t]));
}}}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for( int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
this.XSUiljjt[i][lj][j][t] = cplex.getValue(XSUiljjt[i][lj][j][t]);
System.out.println("XSU" + i + ""+ lj + ""+ j +""+t+ " = " + cplex.getValue(XSUiljjt[i][lj][j][t]));

}}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
this.IPpiijt[pi][i][j][t] = cplex.getValue(IPpiijt[pi][i][j][t]);
System.out.println("IP" + pi +""+ i +""+ j +""+t+ " = " + cplex.getValue(IPpiijt[pi][i][j][t]));
}}}}
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
this.SIPij[i][j] = cplex.getValue(SIPij[i][j]);
System.out.println("SIP" + i +""+ j +" = " + cplex.getValue(SIPij[i][j]));
}}
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
this.SDCik[i][k] = cplex.getValue(SDCik[i][k]);
```

```
System.out.println("SDC" + i +""+ k +" = " + cplex.getValue(SDCik[i][k]));
}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
this.IPDpiikt[pi][i][k][t] = cplex.getValue(IPDpiikt[pi][i][k][t]);
System.out.println("IPD" + pi +""+ i +""+ k +""+ t +""+" = " +
cplex.getValue(IPDpiikt[pi][i][k][t]));
}}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
this.UDpiikt[pi][i][k][t] = cplex.getValue(UDpiikt[pi][i][k][t]);
System.out.println("UD" + pi +""+ i +""+ k +""+ t +""+" = " + cplex.getValue(UDpiikt[pi][i][k][t]));

}}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for( int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
this.XMOTljjt[lj][j][t] = cplex.getValue(XMOTljjt[lj][j][t]);
System.out.println("XMOTljjt" + lj +""+ j +""+ t + " = " + cplex.getValue(XMOTljjt[lj][j][t]));
}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for( int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
System.out.println("L" + i + ""+ lj + ""+ j +""+t+ " = " + cplex.getValue(Liljjt[i][lj][j][t]));
}}}}
for(int t = 0; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for( int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
```

```
System.out.println("U" + i + ""+ lj + ""+ j +""+t+ " = " + cplex.getValue(Uiljjt[i][lj][j][t]));
}}}}
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for( int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
System.out.println("Y"+ lj + ""+ j +""+t+ " = " + cplex.getValue(Yljjt[lj][j][t]));
}}}
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
System.out.println("XPOperationaLij1"+i+""+j+" = "+ cplex.getValue(XPOperationaLij1[i][j]));
}}
}
cplex.end();
}
catch (IloException ex) {
System.out.println("Concert Error: " + ex);
}
}
public double[][][][] returnXiljjt(){
return this.Xiljjt;
}
public double[][][][][] returnXSPpiiljjt(){
return this.XSPpiiljjt;
}
public double[][][][] returnXSUiljjt(){
return this.XSUiljjt;
}
public double[][][][][] returnSSMpiijkt(){
return this.SSMpiijkt;
}
public double[][][][] returnIPpiijt(){
return this.IPpiijt;
}
public double[][] returnSIPij(){
```

```
return this.SIPij;
}
public double[][] returnSDCik(){
return this.SDCik;
}
public double[][][][] returnIPDpiikt(){
return this.IPDpiikt;
}
public double[][][][] returnUDpiikt(){
return this.UDpiikt;
}
public double[][][] returnXMOTljjt(){
return this.XMOTljjt;
}
}
```

Appendix C – Java-Cplex code for iterative algorithm

```
public class IterativeTacticalOperationalAlgorithm {
public static double[][][][] DDiktz =
public static int TotalNumberOfBinaryVariables =
TacticalData.NumberOfProductFam*TacticalData.NumberOfManuSite*
TacticalData.NumberOfTimePeriod;
public static double[][][][][] LHS =
new double[TotalNumberOfBinaryVariables][TacticalData.NumberOfProductFam]
[TacticalData.NumberOfManuSite][TacticalData.NumberOfTimePeriod]
[TacticalData.TotalNumberOfScenarios];
public static double[] RHS = new double[TotalNumberOfBinaryVariables];
public static double[][][][] BinaryVariable;
static final int TotalIterations = 1024;
static double[][][][] Xijt =
new double[TacticalData.NumberOfProductFam][TacticalData.NumberOfManuSite]
[TacticalData.NumberOfTimePeriod][0];
static double[][][][] XPijt =
new double[TacticalData.NumberOfProductFam][TacticalData.NumberOfManuSite]
[TacticalData.NumberOfTimePeriod][0];
static double[][][][][] XSMijkt =
new double[TacticalData.NumberOfProductFam][TacticalData.NumberOfManuSite]
[TacticalData.NumberOfDC][TacticalData.NumberOfTimePeriod][0];
static double[][][][][] XSScsjt =
new double[TacticalData.NumberOfComponentFam][TacticalData.NumberOfSupplier]
[TacticalData.NumberOfManuSite][TacticalData.NumberOfTimePeriod][0];
static double[][][][] IPSijt =
new double[TacticalData.NumberOfProductFam][TacticalData.NumberOfManuSite]
[TacticalData.NumberOfTimePeriod][0];
static double[][][][] IPDikt =
new double[TacticalData.NumberOfProductFam][TacticalData.NumberOfDC]
[TacticalData.NumberOfTimePeriod][0];
```

```
static double[][][][] UDikt =
new double[TacticalData.NumberOfProductFam][TacticalData.NumberOfDC]
[TacticalData.NumberOfTimePeriod][0];
static double[][][][] ICScst =
new double[TacticalData.NumberOfComponentFam][TacticalData.NumberOfSupplier]
[TacticalData.NumberOfTimePeriod][0];
static double[][][][] XCPcst =
new double[TacticalData.NumberOfComponentFam][TacticalData.NumberOfSupplier]
[TacticalData.NumberOfTimePeriod][0];
static double[][][] XMOTjt =
new double[TacticalData.NumberOfManuSite][TacticalData.NumberOfTimePeriod][0];
static double[][][] XSOTst =
new double[TacticalData.NumberOfSupplier][TacticalData.NumberOfTimePeriod][0];
static double[] RemainedCapacityj =
new double[TacticalData.NumberOfManuSite];
static double[] QPit =
new double[TotalIterations];
static double[] Qit =
new double[TotalIterations];
static double[][][][] Xiljjt =
new
double[OperationalData.NumofProductFamily][OperationalData.MaximumNumberofManuLines]
[OperationalData.NumofManuSite][OperationalData.NumofOperationalPeriod];
static double[][][][][] XSPpiiljjt =
new
double[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily]
[OperationalData.MaximumNumberofManuLines][OperationalData.NumofManuSite]
[OperationalData.NumofOperationalPeriod];
static double[][][][] XSUiljjt =
new
double[OperationalData.NumofProductFamily][OperationalData.MaximumNumberofManuLines]
[OperationalData.NumofManuSite][OperationalData.NumofOperationalPeriod];
static double[][][][][] SSMpiijkt =
```

```
new
double[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily]
[OperationalData.NumofManuSite][OperationalData.NumofDC]
[OperationalData.NumofOperationalPeriod];
static double[][][][] IPpiijt =
new
double[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily]
[OperationalData.NumofManuSite][OperationalData.NumofOperationalPeriod];
static double[][] SIPij =
new double[OperationalData.NumofProductFamily][OperationalData.NumofManuSite];
static double[][] SDCik =
new double[OperationalData.NumofProductFamily][OperationalData.NumofDC];
static double[][][][] IPDpiikt =
new
double[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily]
[OperationalData.NumofDC][OperationalData.NumofOperationalPeriod];
static double[][][][] UDpiikt =
new
double[OperationalData.MaxNumofProductsInFamilies][OperationalData.NumofProductFamily]
[OperationalData.NumofDC][OperationalData.NumofOperationalPeriod];
static double[][][] XMOTljjt =
new double[OperationalData.MaximumNumberofManuLines][OperationalData.NumofManuSite]
[OperationalData.NumofOperationalPeriod];
static double[][][] SETrij1 =
new
double[TotalIterations][TacticalData.NumberOfProductFam][TacticalData.NumberOfManuSite];
static int CurrentIteration;
static double[][] setUprj = new double[TotalIterations][TacticalData.NumberOfManuSite];
static double[][][] SetXPrij = new double[TotalIterations][TacticalData.NumberOfProductFam]
[TacticalData.NumberOfManuSite];
static double[][] crrfactorrj =  new double[TotalIterations][TacticalData.NumberOfManuSite];
static double Epsilon = 0.01;
public static void main(String[] args){
CurrentIteration = 0;
```

```
for(int it = 0; it < TotalIterations; it++){
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
setUprj[it][j] = 0;
crrfactorrj[it][j] = 1;
}
}
for(int it = 0; it < TotalIterations; it++){
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
for(int i = 0; i < TacticalData.NumberOfProductFam; i++){
SetXPrij[it][i][j] = 0;
}}}
step0(crrfactorrj,0);
}
public static void step0(double[][] CrrFactorj, int StopOrNot){
double[][] crrfactorj = CrrFactorj;
int stopornot = StopOrNot;
for(int n = 0; n < TotalNumberOfBinaryVariables; n++){
RHS[n] = 0;
for(int i = 0; i < TacticalData.NumberOfProductFam; i++){
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
for(int t = 0; t < TacticalData.NumberOfTimePeriod; t++){
for(int z = 0; z < TacticalData.TotalNumberOfScenarios; z++){
LHS[n][i][j][t][z] = 0;
}}}}}
TacticalModel tacticalModel = new TacticalModel(1, DDiktz, 1,LHS, RHS, crrfactorj,
setUprj, SETrij1, CurrentIteration, stopornot,SetXPrij);
BreakA:{
tacticalModel.MixedIntegermodel();
if(stopornot == 111){
break BreakA;
}

BinaryVariable = tacticalModel.returnBinary();
Xijt = tacticalModel.returnBinary();
```

```
XPijt = tacticalModel.returnXP();
XSMijkt = tacticalModel.returnXSM();
XSScsjt = tacticalModel.returnXSS();
IPSijt = tacticalModel.returnIPS();
IPDikt = tacticalModel.retunIPD();
UDikt = tacticalModel.retunUD();
ICScst = tacticalModel.retunICS();
XCPcst = tacticalModel.returnXCP();
XMOTjt = tacticalModel.returnXMOT();
XSOTst = tacticalModel.returnXSOT();
RemainedCapacityj = tacticalModel.returnRemainedCapacity();
QPit = tacticalModel.returnQPrij();
Qit = tacticalModel.returnQrij();
double[][] Xij1 = new
double[TacticalData.NumberOfProductFam+1][TacticalData.NumberOfManuSite];
double[][] IPSSij1 = new
double[TacticalData.NumberOfProductFam+1][TacticalData.NumberOfManuSite];
double[][] IPDDik1 = new
double[TacticalData.NumberOfProductFam+1][TacticalData.NumberOfDC];
for(int i = 0; i < TacticalData.NumberOfProductFam; i++){
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
Xij1[i+1][j] = Xijt[i][j][0][0];
IPSSij1[i+1][j] = IPSijt[i][j][0][0];
}}
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
Xij1[0][j] = 0;
IPSSij1[0][j] = 0;
}
for(int i = 0; i < TacticalData.NumberOfProductFam; i++){
for(int k = 0; k < TacticalData.NumberOfDC; k++){
IPDDik1[i+1][k] = IPDikt[i][k][0][0];
}}
for(int k = 0; k < TacticalData.NumberOfDC; k++){
IPDDik1[0][k] = 0;
```

```
}
OperationalModel operationalModel = new OperationalModel(Xij1, IPDDik1,  IPSSij1);
operationalModel.OperationalModelMath();
Xiljjt = operationalModel.returnXiljjt();
XSPpiiljjt = operationalModel.returnXSPpiiljjt();
XSUiljjt = operationalModel.returnXSUiljjt();
SSMpiijkt = operationalModel.returnSSMpiijkt();
IPpiijt = operationalModel.returnIPpiijt();
SIPij = operationalModel.returnSIPij();
SDCik = operationalModel.returnSDCik();
IPDpiikt = operationalModel.returnIPDpiikt();
UDpiikt = operationalModel.returnUDpiikt();
XMOTljjt = operationalModel.returnXMOTljjt();
BreakB:{
for(int t = 0; t < 1; t++){
for(int k = 0; k < TacticalData.NumberOfDC; k++){
for(int i = 0; i < TacticalData.NumberOfProductFam; i++){
if(UDikt[i][k][t][0] > 0){
Step2(UDpiikt, Xijt, SIPij, SDCik, XSPpiiljjt, XSUiljjt, XPijt, RemainedCapacityj);
break BreakB;
}}}}
Step1(UDpiikt, Xijt, SIPij, SDCik, XSPpiiljjt, XSUiljjt, RemainedCapacityj,XPijt);
}}}
public static void Step2(double[][][][] UDpiikt, double[][][][] Xijt, double[][] SIPij,
double[][] SDCik,double[][][][][] XSPpiiljjt,double[][][][] XSUiljjt, double[][][][] Xpijt,
double[] RemainedCapacityj){
double[][][][] udpiikt = UDpiikt;
double[][][][] xijt = Xijt;
double[][] sipij = SIPij;
double[][] sdcik = SDCik;
double[][][][][] xsppiiljjt = XSPpiiljjt;
double[][][][] xsuiljjt =  XSUiljjt;
double[][][][] xpijt = XPijt;
double[] remainedcapacity = RemainedCapacityj;
```

```
BreakB:{
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int i = 1; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
if(udpiikt[pi][i][k][OperationalData.NumofOperationalPeriod-1] > 0){
Step2B2(xijt, xsppiiljjt, xsuiljjt, xpijt, remainedcapacity);
break BreakB;
}}}}}}
public static void Step2B2(double[][][][] Xijt, double[][][][][] XSPpiiljjt,
double[][][][] XSUiljjt, double[][][][] XPijt, double[] RemainedCapacityj){
double[][][][] xijt = Xijt;
double[][][][][] xsppiiljjt = XSPpiiljjt;
double[][][][] xsuiljjt =  XSUiljjt;
double[][][][] xpijt = XPijt;
double[] remainedcapacity = RemainedCapacityj;
BreakB:{
double TotalProductionInTactical = 0;
double totalProductionInOperational = 0;
double difference = 0;
for(int i = 0; i < TacticalData.NumberOfProductFam; i++){
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
totalProductionInOperational += xsppiiljjt[pi][i+1][lj][j][t];
}}}}}
for(int i = 0; i < TacticalData.NumberOfProductFam; i++){
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
TotalProductionInTactical += xpijt[i][j][0][0];
}}
difference = Math.abs((totalProductionInOperational-TotalProductionInTactical));
double Ratio = difference / TotalProductionInTactical;
if(Ratio > Epsilon){
Step2B2B2(xijt, xsppiiljjt, XSUiljjt, remainedcapacity);
```

```
break BreakB;
}}}
public static void Step2B2B2(double[][][][] Xijt,double[][][][][] XSPpiiljjt,
double[][][][] XSUiljjt, double[] RemainedCapacityj){
double[][][][] xijt = Xijt;
double[][][][][] xsppiiljjt = XSPpiiljjt;
double[][][][] xsuiljjt = XSUiljjt;
double[] remainedcapacity = RemainedCapacityj;
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
for(int i = 0; i < TacticalData.NumberOfProductFam; i++){
SETrij1[CurrentIteration][i][j] = xijt[i][j][0][0];
}}
double[] productionj = new double[TacticalData.NumberOfManuSite];
double[] setUpj = new double[TacticalData.NumberOfManuSite];
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
productionj[j] = 0;
setUpj[j] = 0;
}
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int i = 1; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
productionj[j] = productionj[j]+ (OperationalData.MRij[i][j]* xsppiiljjt[pi][i][lj][j][t]);
setUpj[j] = OperationalData.SAljj[lj][j]*0.2*OperationalData.STiljj[i][lj][j]*
xsuiljjt[i][lj][j][t];
setUprj[CurrentIteration][j] += setUpj[j];
}}}}}
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
double total = productionj[j]+setUprj[CurrentIteration][j];
if(CurrentIteration == 0){
crrfactorrj[CurrentIteration+1][j] = (productionj[j]/total);
}else if(CurrentIteration > 0){
if(QPit[CurrentIteration] == 0 && Qit[CurrentIteration] == 0){
```

```
crrfactorrj[CurrentIteration+1][j] = crrfactorrj[CurrentIteration][j]*(productionj[j]/total);
}else{
crrfactorrj[CurrentIteration+1][j] = (productionj[j]/total);
}}}
CurrentIteration++;
int StopOrNot = 222;
step0(crrfactorrj, StopOrNot);
}
public static void Step1(double[][][][] UDpiikt, double[][][][] Xijt,
double[][] SIPij, double[][] SDCik,double[][][][][] XSPpiiljjt,
double[][][][] XSUiljjt, double[] RemainedCapacityj, double[][][][] Xpijt){
double[][][][] udpiikt = UDpiikt;
double[][][][] xijt = Xijt;
double[][] sipij = SIPij;
double[][] sdcik = SDCik;
double[][][][][] xsppiiljjt = XSPpiiljjt;
double[][][][] xsuiljjt =  XSUiljjt;
double[][][][] xpijt = XPijt;
double[] remainedcapacity = RemainedCapacityj;
BreakB:{
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int i = 1; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
        if(udpiikt[pi][i][k][OperationalData.NumofOperationalPeriod-1] > 0){
Step1B1(xijt, xsppiiljjt, xsuiljjt,remainedcapacity, xpijt);
break BreakB;
}}}}
Step1B2(xijt, xsppiiljjt, xsuiljjt, sipij, sdcik, remainedcapacity, xpijt);
}}
public static void Step1B2(double[][][][] Xijt,double[][][][][] XSPpiiljjt,
double[][][][] XSUiljjt, double[][] SIPij, double[][] SDCik,
double[] RemainedCapacityj, double[][][][] Xpijt){
double[][][][] xijt = Xijt;
double[][][][][] xsppiiljjt = XSPpiiljjt;
```

```
double[][][][] xsuiljjt =  XSUiljjt;
double[][] sipij = SIPij;
double[][] sdcik = SDCik;
double[][][][] xpijt = XPijt;
double[] remainedcapacity = RemainedCapacityj;
BreakA:{
for(int k = 0; k < OperationalData.NumofDC; k++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
if(sdcik[i][k] > 0){
Step1B2B2(xijt,xsppiiljjt,xsuiljjt,remainedcapacity, xpijt );
break BreakA;
}}}
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int i = 0; i < OperationalData.NumofProductFamily; i++){
if(sipij[i][j] > 0){
Step1B2B2(xijt,xsppiiljjt,xsuiljjt, remainedcapacity, xpijt);
break BreakA;
}}}
for(int i = 0; i < TacticalData.NumberOfProductFam; i++){
double TotalProductionInTactical = 0;
double totalProductionInOperational = 0;
double difference = 0;
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
totalProductionInOperational += xsppiiljjt[pi][i+1][lj][j][t];
}}}}
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
TotalProductionInTactical += xpijt[i][j][0][0];
}
difference = Math.abs((totalProductionInOperational-TotalProductionInTactical));
double Ratio = difference / TotalProductionInTactical;
}}}
```

```
public static void Step1B2B2(double[][][][] Xijt,double[][][][][] XSPpiiljjt,
double[][][][] XSUiljjt, double[] RemainedCapacityj, double[][][][] Xpijt){
double[][][][] xijt = Xijt;
double[][][][][] xsppiiljjt = XSPpiiljjt;
double[][][][] xsuiljjt =  XSUiljjt;
double[][][][] xpijt = XPijt;
double[] remainedcapacity = RemainedCapacityj;
BreakC:{
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
if(remainedcapacity[j] != 0){
Step1B2B2B1(xijt, xsppiiljjt, xsuiljjt, remainedcapacity, xpijt);
break BreakC;
}}
Step1B2B2B2(xijt, xsppiiljjt, xsuiljjt, remainedcapacity, xpijt);
}}
public static void Step1B2B2B2(double[][][][] Xijt,double[][][][][] XSPpiiljjt,
double[][][][] XSUiljjt, double[]RemainedCapacityj, double[][][][] Xpijt){
double[][][][] xijt = Xijt;
double[][][][][] xsppiiljjt = XSPpiiljjt;
double[][][][] xsuiljjt = XSUiljjt;
double[] remainedcapacity = RemainedCapacityj;
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
for(int i = 0; i < TacticalData.NumberOfProductFam; i++){
SETrij1[CurrentIteration][i][j] = xijt[i][j][0][0];
}}
double[] productionj = new double[TacticalData.NumberOfManuSite];
double[] setUpj = new double[TacticalData.NumberOfManuSite];
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
productionj[j] = 0;
setUpj[j] = 0;
}
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
```

```
for(int i = 1; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
productionj[j] = productionj[j]+ (OperationalData.MRij[i][j]* xsppiiljjt[pi][i][lj][j][t]);
setUpj[j] = OperationalData.SAljj[lj][j]*0.2*OperationalData.STiljj[i][lj][j]*xsuiljjt[i][lj][j][t];
setUprj[CurrentIteration][j] += setUpj[j];
}}}}}
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
if(CurrentIteration == 0){
crrfactorrj[CurrentIteration+1][j] = (productionj[j]/(productionj[j]+
setUprj[CurrentIteration][j]));
}else if(CurrentIteration > 0){
if(QPit[CurrentIteration] == 0 && Qit[CurrentIteration] == 0){
crrfactorrj[CurrentIteration+1][j] = crrfactorrj[CurrentIteration][j]*
(productionj[j]/(productionj[j]+setUprj[CurrentIteration][j]));
}else{
crrfactorrj[CurrentIteration+1][j] = (productionj[j]/(productionj[j]+
setUprj[CurrentIteration][j]));
}}}
CurrentIteration++;
int StopOrNot = 1222;
step0(crrfactorrj, StopOrNot);
}
public static void Step1B2B2B1(double[][][][] Xijt,double[][][][][] XSPpiiljjt,
double[][][][] XSUiljjt, double[] RemainedCapacityj, double[][][][] Xpijt){
double[][][][][] xsppiiljjt = XSPpiiljjt;
double[][][][] xsuiljjt = XSUiljjt;
double[][][][] xpijt = XPijt;
double[] productionj = new double[TacticalData.NumberOfManuSite];
double[] setUpj = new double[TacticalData.NumberOfManuSite];
double[] remainedcapacity = RemainedCapacityj;
for(int i = 0; i < TacticalData.NumberOfProductFam; i++){
double TotalProductionInTactical = 0;
double totalProductionInOperational = 0;
double difference = 0;
```

```
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
totalProductionInOperational += xsppiiljjt[pi][i+1][lj][j][t];
}}}}
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
TotalProductionInTactical += xpijt[i][j][0][0];
}
difference = Math.abs((totalProductionInOperational-TotalProductionInTactical));
double Ratio = difference / TotalProductionInTactical;
}
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
productionj[j] = 0;
setUpj[j] = 0;
}
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int i = 1; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
productionj[j] = productionj[j]+ (OperationalData.MRij[i][j]* xsppiiljjt[pi][i][lj][j][t]);
setUpj[j] = OperationalData.SAljj[lj][j]*0.2*OperationalData.STiljj[i][lj][j]*xsuiljjt[i][lj][j][t];
setUprj[CurrentIteration][j] += setUpj[j];
}}}}}
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
if(CurrentIteration == 0){
crrfactorrj[CurrentIteration+1][j] = (productionj[j]/(productionj[j]+setUprj[CurrentIteration][j]));
}else if(CurrentIteration > 0){
if(QPit[CurrentIteration] == 0 && Qit[CurrentIteration] == 0){
crrfactorrj[CurrentIteration+1][j] = crrfactorrj[CurrentIteration][j]*
(productionj[j]/(productionj[j]+setUprj[CurrentIteration][j]));
}else{
crrfactorrj[CurrentIteration+1][j] = (productionj[j]/(productionj[j]+
```

```
setUprj[CurrentIteration][j]));
}}}
for(int j = 0 ; j < OperationalData.NumofManuSite; j++){
for(int i = 1; i < OperationalData.NumofProductFamily; i++){
double TotalProduction= 0;
for( int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
TotalProduction +=  xsppiiljjt[pi][i][lj][j][t];
}}}
SetXPrij[CurrentIteration][i][j] = TotalProduction;
}}
CurrentIteration++;
step0(crrfactorrj, 1221);
}
public static void Step1B1(double[][][][] Xijt,double[][][][][] XSPpiiljjt,
double[][][][] XSUiljjt, double[] RemainedCapacityj, double[][][][] Xpijt){
double[][][][] xijt = Xijt;
double[][][][][] xsppiiljjt = XSPpiiljjt;
double[][][][] xsuiljjt =  XSUiljjt;
double[] remainedcapacity = RemainedCapacityj;
BreakC:{
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
if(remainedcapacity[j] != 0){
Step1B1B2(xijt, xsppiiljjt, xsuiljjt, remainedcapacity);
break BreakC;
}}}
Step1B1B1(xijt, xsppiiljjt, xsuiljjt, remainedcapacity );
}
public static void Step1B1B1(double[][][][] Xijt,double[][][][][] XSPpiiljjt,
double[][][][] XSUiljjt, double[]RemainedCapacityj){
double[][][][][] xsppiiljjt = XSPpiiljjt;
double[][][][] xsuiljjt = XSUiljjt;
double[] productionj = new double[TacticalData.NumberOfManuSite];
```

```
double[] setUpj = new double[TacticalData.NumberOfManuSite];
double[] remainedcapacity = RemainedCapacityj;
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
productionj[j] = 0;
setUpj[j] = 0;
}
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int i = 1; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
productionj[j] = productionj[j]+ (OperationalData.MRij[i][j]* xsppiiljjt[pi][i][lj][j][t]);
setUpj[j] = OperationalData.SAljj[lj][j]*0.2*OperationalData.STiljj[i][lj][j]*
xsuiljjt[i][lj][j][t];
setUprj[CurrentIteration][j] += setUpj[j];
}}}}}
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
if(CurrentIteration == 0){
crrfactorrj[CurrentIteration+1][j] = (productionj[j]/(productionj[j]+
setUprj[CurrentIteration][j]));
}else if(CurrentIteration > 0){
if(QPit[CurrentIteration] == 0 && Qit[CurrentIteration] == 0){
crrfactorrj[CurrentIteration+1][j] = crrfactorrj[CurrentIteration][j]*
(productionj[j]/(productionj[j]+setUprj[CurrentIteration][j]));
}else{
crrfactorrj[CurrentIteration+1][j] = (productionj[j]/
(productionj[j]+setUprj[CurrentIteration][j]));
}}}
for(int j = 0 ; j < OperationalData.NumofManuSite; j++){
for(int i = 1; i < OperationalData.NumofProductFamily; i++){
double TotalProduction= 0;
for( int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
```

```
TotalProduction +=  xsppiiljjt[pi][i][lj][j][t];
}}}
SetXPrij[CurrentIteration][i][j] = TotalProduction;
}}
CurrentIteration++;
step0(crrfactorrj, 111);
}
public static void Step1B1B2(double[][][][] Xijt,double[][][][][] XSPpiiljjt,
double[][][][] XSUiljjt, double[] RemainedCapacityj){
double[][][][] xijt = Xijt;
double[][][][][] xsppiiljjt = XSPpiiljjt;
double[][][][] xsuiljjt = XSUiljjt;
double[] remainedcapacity = RemainedCapacityj;
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
for(int i = 0; i < TacticalData.NumberOfProductFam; i++){
SETrij1[CurrentIteration][i][j] = xijt[i][j][0][0];
}}
double[] productionj = new double[TacticalData.NumberOfManuSite];
double[] setUpj = new double[TacticalData.NumberOfManuSite];
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
productionj[j] = 0;
setUpj[j] = 0;
}
for(int j = 0; j < OperationalData.NumofManuSite; j++){
for(int t = 1; t < OperationalData.NumofOperationalPeriod; t++){
for(int lj = 0; lj < OperationalData.MaximumNumberofManuLines; lj++){
for(int i = 1; i < OperationalData.NumofProductFamily; i++){
for(int pi = 0; pi < OperationalData.MaxNumofProductsInFamilies; pi++){
productionj[j] = productionj[j]+ (OperationalData.MRij[i][j]* xsppiiljjt[pi][i][lj][j][t]);
setUpj[j] = OperationalData.SAljj[lj][j]*0.2*OperationalData.STiljj[i][lj][j]*
xsuiljjt[i][lj][j][t];
setUprj[CurrentIteration][j] += setUpj[j];
}}}}}
for(int j = 0; j < TacticalData.NumberOfManuSite; j++){
```

```
double total = productionj[j]+setUprj[CurrentIteration][j];
if(CurrentIteration == 0){
crrfactorrj[CurrentIteration+1][j] = (productionj[j]/(productionj[j]+
setUprj[CurrentIteration][j]));
}else if(CurrentIteration > 0){
if(QPit[CurrentIteration] == 0 && Qit[CurrentIteration] == 0){
crrfactorrj[CurrentIteration+1][j] = crrfactorrj[CurrentIteration][j]*(productionj[j]/
(productionj[j]+setUprj[CurrentIteration][j]));
}else{
}}}
CurrentIteration++;
int StopOrNot = 112;
step0(crrfactorrj, StopOrNot);
}
}
```