NOT TO BE TAKEN FROM THIS ROOM

HIERARCHICAL COORDINATION FRAMEWORK IN SUPPLY CHAIN NEGOTIATIONS

by

Murat Özdemir

BS. in I.E., İstanbul Teknik Üniversitesi, 1996

Submitted to the Institute for Graduate Studies in Science and Engineering in partial fulfillment of the requirements for the degree of

Master of Science

in

Industrial Engineering

Bogazici University Library
39001100370819

Boğaziçi University

1999

Dedicated to

Ahmet

Sabahat

and Mustafa N. Özdemir

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my thesis supervisor, Assoc. Prof. Gülay Barbarasoğlu for her leadership, guidance, advices and encouragement not only during this thesis but also during my assistantship to her.

I would like to thank to Assist. Prof. Taner Bilgiç and Assoc. Prof. Tülin Yazgaç for their valuable comments and suggestions on this thesis.

I would like to thank to Orhan Feyzioğlu, not only for helping me with my very first C++ code but for everything we shared during several years. I am very thankful to Deniz Aksen for his support for GAMS and I wish him good luck with his GAMS book. I would like to express my gratitude to Alper E. Murat, Saltuk B. Akbulut and Gün Evren Gören and all IE research assistants for their various helps and continous moral support.

I am deeply grateful to all citizens of Turkish Republic who supported my education with their taxes for 19 years, up to now.

ABSTRACT

This study examines a hierarchical coordination framework between a supplier of a strategic item and his buyers in an environment where a mutual information exchange between the supplier and buyers is possible. A organizational hierarchical structure is defined via a generic supply chain contract where the supplier is in the leadership position. A three-level decision process is designed to help the supplier in making decisions: aggregate capacity planning, disaggregation of this aggregate capacity among the buyers and then price determination. Since the pricing model is a mixed integer nonlinear problem that is not easy to solve by exact methods, as a pioneering effort in supply chain management, a global metaheuristic algorithm SUPRIME (SUpply PRIce MEtaheuristic) is developed by integrating the principles of simulated annealing (SA) and tabu search (TS) procedures. Also the hierarchical structures in this study are mapped to Hierarchical Organization Planning (HOP) framework.

ÖZET

Bu çalışma paralel bilgi alışverişinin mümkün olduğu bir ortamda stratejik bir mamül sağlayan tedarikçi ile onun müşterileri arasındaki hiyerarşik koordinasyon yapısını inceler. Tedarikçinin lider pozisyonunda olduğu genel bir tedarik zinciri sözleşmesi ile organizasyonel hiyerarşik yapı tanımlanmıştır. Üç seviyeli karar süreci (toplam kapasite seviyesinin belirlenmesi, toplam kapasitenin müşterilere bölüştürülmesi ve fiyat belirleme) ile tedarikçi kararlarını verir. Fiyatlandırma modeli çözümü kolay olmayan karışık tam sayılı, doğrusal olmayan bir model olduğundan bu modelin çözümü için, tabu tarama ve ve tavlama benzetimi yöntemlerinden esinlenilerek bir melez meta-sezgisel yöntem SUPRIME (SUpply PRIce MEtaheuristic) geliştirilmiştir. Ayrıca bu çalısmadaki hiyerarşik yapılar, Hiyerarşik Organizasyonel Planlama (HOP) çercevesi içerisinde ifade edilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Today the dynamics of global economy and the competitive market environment force all manufactures to be on time, flexible to changes and serve with low costs. Productivity and efficiency studies in the history of industrial engineering have performed evident results to decrease costs. However competitiveness force companies to make improvements not only in bound but also in the framework of the supply chain management.

It is essential to automate and probably more important to integrate the business functions at all levels of production and logistics systems by adapting an information technology over common or different remote databases.

High development speed in the information techniques and continuously decreasing information systems costs helped the manufacturing management system known in 60's as Material Requirements Planning (MRP) spread world wide in the early 80's. With the practicality of MRP, new concepts are added rapidly such as Financial modules (costing, accounting, financial reporting), Capacity Requirements Planning (CRP), Distribution Resources Planning (DRP), Human Resources (HR) introducing new concepts like Manufacturing Resources Planning (MRP II) and Enterprise Resources Planning (ERP). The integration and information sharing between business partners and the modern management approach has given rise up to Supply Chain Management; a concept of late 90s and promising to be popular in the last milenium.

In the progress of manufacturing management systems, MRP, MRP II and DRP provided quick material requirement calculations and information flow across different sites whereas MRP II, DRP, human and financial resources planning functionalities are covered in the early 90's most popular solution ERP. However, many standard commercial ERP solutions are lacking intelligent supply chain management issues as of 1999. Thus it poses an academic and practical challenge to develop supply chain tools.

Today manufacturers are making partnership agreements with their suppliers and customers in order to improve the total supply chain efficiency. Hence the survival of a company depends on the ability of the whole supply chain to meet the requirements of the end customers with consistent quality, minimum cost and quick response capability. Ignoring the supply chain partnership between the customers and suppliers would be the lose of opportunities on both sides such as quality, lower prices, shorter lead times, shorter cycle times, larger volumes, long term agreements, training, stable forecasts, advanced planning information, etc. In this respect, coordination within a supply chain system implies that all members of the chain mutually share relevant information with the advancements in information technology and implement their decision and information exchange processes on a formal basis. Since uncoordinated planning may lead to severe inconsistencies and infeasibilities due to conflicts resulting from local optima, a coordinated approach to the supply chain may generate significant savings by trading off the costs associated with all the members rather than minimizing production, sales and purchase costs of each member separately. Thus it becomes an important issue both in theory and practice to determine the coordination strategies and policies, and the coordination of planning practices along the supply chain poses a crucial design problem that still remains to be an unexplored research area.

This study involves a hierarchical coordination framework in an information sharing supply chain between a supplier and his buyers in making production and symmetrical purchase/sales decisions. A generic supply chain contract is designed between the supplier and his buyers, and at the contract renewal time the supplier needs to decide upon the supply commitment and the price to be declared to each buyer for the upcoming contract term. A three-level decision process is designed to help the supplier in making these decisions. In this approach, the overall-planning problem is first divided into a set of subproblems, each of which possesses its own design, scope and focus of decision modelling. It consists of aggregate capacity planning, disaggregation of this aggregate capacity among the buyers and then price determination. Most emphasis is placed upon revealing the negotiation aspects of the pricing model with respect to varying degrees of coordination. Since pricing strategies in the supply chain play important roles in the competitiveness of end product manufacturers, their production models are specifically incorporated into the pricing model of the supplier, which aims to formulate the supply

partnership expectations of all parties involved. Pricing strategies have been studied by many researchers and there exist novel studies in literature; however, they basically assume uniform demand and obtain lot sizing results through differentiation-based algorithms. Since consumer demand is assumed to be a constant process in all these studies, the traditional EOQ-based results are derived under different cooperation schemes. This study exhibits a pioneering attempt to study pricing and sales decisions simultaneously in a dynamic demand environment and develops a mixed integer mathematical programming model with non-linear relations. Although the ideas covered in these studies have inspired this research to a great extent, here demand is assumed to change dynamically over time and mathematical programming is used as the design tool to describe the related problems; consequently a completely different solution is needed especially to cope with non-linearity in the joint pricing-sales model. As a pioneering effort in supply chain management, a global metaheuristic algorithm SUPRIME (SUpply PRIce MEtaheuristic) is developed by integrating the principles of simulated annealing (SA) and tabu search (TS) procedures. The performance of the algorithm is assessed on a large set of test problems and shown to generate good results in considerably short computational times. Another aim of the study is to adapt the hierarchical coordination studied in this thesis to Hierarchical Organization Planning (HOP) framework.

This study is organized as follows: Section 2 includes the literature survey about the related topics. Section 3 states the supply chain management problem this study is based on. Section 4 provides the hierarchical planning structure proposed as a decision support model that can be used by a supplier in making supply and price determination decisions. Section 5 provides the mathematical formulation of the aggregate model. Analytical Hierarchical Planning method is proposed as a tool for solving capacity allocation problem in Section 6. In Section 7, the supplier's determination of the price of the supply item is modeled. The metaheuristic developed as a global search algorithm for the price and purchase decisions of a supplier simultaneously in a coordinated supply chain environment is presented in Section 8. Section 9 explores the hierarchical planning structure in HOP framework. Finally, in Section 10 results and possible future studies are stated.

2. LITERATURE SURVEY

The areas of research that inspired this study can be classified as hierarchical planning, integrated manufacturing-distribution systems and supply chain management, analytical hierarchical planning, pricing and metaheuristics for combinatorial optimization problems.

2.1. Hierarchical Planning

Hierarchical production planning provides an encouraging approach to overcome the impracticality and the complexity in applying a monolithic approach to a large-scale problem. The basic idea of this approach is to reduce the problem complexity by decomposing the planning process into isolated subproblems that can be solved separately.

Hierarchical approach defines interdependencies in a great variety of ways. One approach is to define a hierarchical product structure as proposed by Hax and Candea [1] in terms of items, families and types and formulate the resulting decision problem accordingly.

Another issue in defining subproblems is to consider the existing organizational structures adequately. In this respect, Schneeweiss [2] gives a general framework for hierarchical interdependencies within an organization. He presents a uniform conceptual framework to discuss the structure of hierarchical negotiations, principal-agent relationships, various kinds of hierarchical planning procedures, and hierarchical algorithms.

Dempster et al. [3] provide a first survey of fields application of hierarchical planning, and show that this type of problems may also be modeled as a multistage stochastic program.

Simchi-Levi [4] proposes a hierarchical approach to the design and control of probabilistic distribution systems where only a subset of all potential customers needs service on any given working day. Three main subproblems corresponding to the three levels of decision – strategic, tactical and operational – are integrated into a hierarchical design, and the total system cost is proven to be asymptotically optimal.

In principal, hierarchical planning is based on a top-down control. However infeasibilities on the lower levels, or inconsistencies due to disaggregation may require some modifications. To solve these problems, Günther [5] suggests that three types of decision making on the superior level have to be considered: fixed, frozen and tentative plans by using the rolling horizon approach with anticipation.

Saad [6] extends the earlier hierarchical production planning approach initiated by Hax and Meal [1] into a hierarchical model for multi-plant, multi-product firms which incorporates both hierarchical and functional interface interacting simultaneously as in practice. He uses goal programming to solve the problem of feasibility which arises in disaggregating the aggregate production plan into detailed schedules at the product-family disaggregation level, and uses the branch and bound at the item-disaggregation level.

Kistner and Steven [7] review applications of operations research to solve planning problems that arise in subsystems of production control. They also consider some theoretical aspects of hierarchical production planning. They state that heuristic decomposition may be applied for tuning decisions on the tactical and operative levels of the planning process.

Erschler et al. [8] consider the consistency of aggregate model in a rolling horizon environment. They emphasize that the knapsack algorithm in connection with the look-ahead rule guarantees

A study by Tsubone et al. [9] verifies the validity of a hierarchical production planning system for a two-stage process. They clarify, through a simulation model, the relationship between the production planning rules and the buffer inventory role in terms

of manufacturing performance in order to support the choice of an optimum production system.

2.2. Integrated Manufacturing-Distribution and Supply Chain Management

Although there exists a rich library of state of the art research about supply chain management and software solutions supporting this concept, among which a few will be cited below.

Glover et al. [10] develop a network flow model of the production scheduling inventory and distribution decisions. They embed this model in a decision support system that is used to analyze both short-run planned decisions and long range strategic decisions.

Bloemhof-Ruward et al. [11] propose and analyse alternative model formulations for the problem of coordinating product and by-product flows in a two level distribution network. Their problem is an extension of the classical facility location problem in which they consider a variant of the two level location problem.

Iyogun [12] describes an algorithm for a distribution problem involving several retailers and several warehouses and multiple items where external demands for items are constant, continuous and deterministic and each facility has a fixed setup cost and incurs echelon holding cost for any unused inventory.

Slats et al. [13] provide a literature review on the logistic chain modeling. Recent development in the market demands, technology and structuring and redesign of logistic processes are described. They conclude that Operations Research models and techniques are well suited to analyze the local performance of logistic sub-chains and processes but in order to fully support the analysis of the performance of an integrated logistic chain the traditional OR approach is not sufficient.

Nam et al. [14] focus on the outsourcing bidding process pertinent to the selection of one contractor by a user firm. The paper explores bidding situations where the vendors

have different levels of expertise and cost structures. The model allows the user firm to determine the optimal expected contract cost. The conclusions emerge from the analysis of representative examples used in the paper.

Williams [15] describes how a supplier partnership can be set up to avoid the typical purchasing relationship-price being inversely proportional to quantity and having the purchaser take all the risk of product obsolescence. It also describes how rate based replenishment replaced time based delivery, and how all these advantages were achieved at reduced administrative costs.

Vidal and Goetschalckx [16] provide a review of strategic production - distribution models with emphasis on global supply chain models. They focused on mixed integer programming models. In addition, they concentrate on the identification of the relevant factors included in the formulations, and the specific characteristics of solution methods and computational experiences with a special emphasis on global logistics.

Thomas and Griffin [17] review the literature addressing coordinated planning between two or more stages of the supply chain (the fundamental stages of the supply chain are procurement, production, and distribution), placing particular emphasis on models that would lend themselves to a total supply chain model.

Maloni and Benton [18] seek to provide a review of supply chain research from both the qualitative conceptual and analytical operations research perspectives. They state that the expanding importance of supply chain integration presents a challenge to operations researchers to focus more attention on supply chain modeling, and there are numerous opportunities for operations researchers to provide support for the current conceptual based supply chain research.

Petrovic et al. [19] deal with the fuzzy modeling and simulation of a supply chain in an uncertain environment. The objective is to determine stock levels and order quantities for each inventory in a supply chain during a finite time horizon to obtain an acceptable delivery performance at a reasonable total cost for a whole supply chain. Two sources of uncertainty inherent in the external environment in which the supply chain operates were

identified and modeled: customer demand and external supply of raw material. They were interpreted and represented by fuzzy sets.

Desirey and Özatalay [20] introduce the rough-cut value management analysis to show that traditional customer ranking by using revenue may not generate the best value for the business. They propose a weighting model to obtain the demand value for each customer to achieve the maximum business value with supply capabilities. Moreover, they carry out the case study about this research in Dupont plastic business.

Bassok et al. [21] study a type of supply contract that is frequently used in the electronic industry. A common feature of these supply contracts is that at the beginning of the contract, the buyer makes purchasing commitments to the supplier for each period. The buyer may have some flexibility to purchase quantities that actually deviate from the original commitments. Moreover as time passes and more information about the actual demand is collected, the buyer may update the previous commitments. Finally, they develop a heuristic that is easy to implement and that determines nearly optimal commitments and purchasing quantities.

Ng et al [22] study the utilization of ERP application considered to be the most effective computer application in the modern manufacturing industry which meets requirements like fast response, accuracy, integrated planning of all sources. However only a few manufacturers can design and implement the system successfully. Therefore the authors propose a systematic implementation design and methodology for ERP called hierarchical design pyramid (HDP).

Contemporary manufacturing management packages can be found proprietary and can require expensive efforts to fit the changing needs of a specific enterprise. Maione and Piscitelli [23] give guidelines to design operational control software for a flexible manufacturing system, with generic, flexible and object oriented features.

2.3. Analytical Hierarchical Planning

After AHP's first introduction by Saaty [24], it has been widely used in many applications. Weber et al. [25] provide an extensive literature survey on vendor selection criteria and quantitative approaches to vendor selection.

Barbarosoğlu and Yazgaç [26] developed an AHP tree structure where the goal is to develop a general model to solve the supplier selection problem in current global and competitive world-class manufacturing. The primary objectives affecting the supplier selection are grouped under three main categories: performance assessment, business structure / manufacturing capability assessment, and quality system assessment. The whole set of subcriteria constitutes a five-level incomplete hierarchy.

Diakoukaki et al. [27] propose a method for determination of objective weights which is based on the fundamental notions of Multiple Criteria Decision Method (MCDM): the contrast intensity and the conflicting character of the evaluation criteria. The extraction of subjective preferences is either difficult or undesirable. In this study, the latter notion is of greater importance in interfirm comparisons because the financial indices used are often highly correlated. The method developed is applied to a sample of industrial firms. The results are compared to those obtained by other sets of objective weights and show this method ensures a better compromise of the criteria examined.

2.4. Metaheuristics for Combinatorial Optimization Problems

Genetic algorithms, simulated annealing, tabu search and neural networks are the most popular global search algorithms for solving complex optimization problems. Genetic algorithms and neural networks are inspired from biological sciences whereas simulated annealing was derived from the analogy of heat treatment, annealing, of physical science thermodynamics. Tabu search is a intelligent problem solving method for avoiding cycling. Since it is not possible to solve complex problem with exact algorithms, search algorithms are employed to find a near optimum solution in a desirable time period.

Tabu search metaheuristic was introduced independently by Glover [28] and Hansen [29] in 1986. This metaheuristic is based on putting intelligence or memory in the problem solving method. In order avoid getting trapped in local optima, the idea is to keep the history of accepted movements in a finite memory. Each time a solution is accepted the solution is kept in memory while an ex-solution is removed from the memory or history.

Osman [30] defines heuristics, combinatorial optimization and reviews, classifies, summarizes the recently emerging heuristic approaches to combinatorial problem solving. He focuses on the recent approaches derived from artificial intelligence and natural science by analogies and explains how metaheuristics can move out of local optima where local search methods would be trapped. He gives sample algorithms for metaheuristic approaches.

Glover et al [31] describe the development and use of tabu search approach to solve binary quadric programs. Computational experience is reported, showing that the approach optimally solves the most difficult problems reported in the litrerature. For challenging problems of limited size that are capable of being approached by exact produres, they can show that they find optimal solutions in shorter run times.

2.5. Pricing and Discounting

Chiang et al. [32] analyze the traditional quantity discount problem from the perspective of game theory, including both the noncooperative and cooperative models. For the noncooperative case, the Stackelberg equilibrium is derived. For the cooperative case, the Pareto Optimality criteria are used to find a group of optimal strategies. Both scenarios are illustrated thorough an example which quantifies the benefits resulting from cooperation between the buyer and the seller for the game-theoretic solutions using geometric programming.

Leavy [33] deals with the buyer-supplier relationship. He examined this relationship from both sides in an effort to understand the strategic implications for each

party implied in the traditional (buyer and supplier as competitors) and JIT (buyer and supplier as partners) perspectives.

Abad [34] presents a new model of vendor-buyer co-ordination for the case where the buyer's requirements are fixed. The model allows for multiple lot strategy by the supplier. The problem of supplier pricing is viewed as a two person fixed threat bargaining game and Pareto efficient solutions are characterized. A pricing scheme is designed for a supplier who is supplying to a population of buyers.

Li and Huang [35] explore cooperative relationships between two members in a simple buyer-seller system where the buyer is in a monopolistic position. They begin with a situation where the seller, as the leader, has the power to enforce his strategies on the buyer, but vice versa is not true. Then the analysis is extended to a situation where the buyer can also influence the seller's decisions and address the issue of system cooperation. The mutual incentives for cooperation and individual disincentives for co-operation are illustrated. A quantity discount scheme is developed to implement a profit sharing mechanism.

Shinn et al. [36] deal with the problem of determining the retailer's optimal price and lot size simultaneously under conditions of permissible delay in payments. It is assumed that the ordering cost consists of a fixed set-up cost and a freight cost, where the freight cost has a quantity discount offered due to the economies of scale. The constant price elasticity demand function is adapted, which is a decreasing function of retail price. Investigation of the properties of an optimal solution allows us to develop an algorithm whose validity is illustrated through an example problem.

Huang et al. [37] deal with a situation in which the buyer is in a monopolistic position with respect to the seller, and examines the issues and advantages of co-operation in a seller-buyer inventory control system. The combination of an equal profit sharing role implemented via quantity discounting is demonstrated as the best mechanism for achieving system co-operation.

Weng [38] presents models for determining optimal all-unit and incremental quantity discount policies and investigates the effect of quantity discounts on increasing demand and ensuring pareto–efficient transactions under general price-sensitive demand functions. He develops optimal quantity discount policies, investigates their interrelationships and their benefits to both the supplier and the buyer, and gains managerial insights for the scenarios of maximizing the supplier's profit and the joint profit. He develops simple and efficient solution approaches for determining the all-unit and the incremental optimal decision policies for general price-sensitive demand functions.

Weng and Wong [39] investigate some managerial insights related to the all-unit quantity discount policies under various conditions. They develop models that are general treatments for dealing with four major issues: one buyer or multiple buyers; constant or price-elastic demand; the relationship between the supplier's production schedule or ordering policy and the buyer's ordering size; and the supplier either manufacturing or purchasing the item. They also propose algorithms to find optimal decision policies. At the end, they provide the supplier with both the optimal all-unit quantity discount policy and the optimal production (or ordering) strategy.

Weng [40] studies the effect of coordination in a manufacturing and distribution system consisting of one manufacturer and one distributor. The coordinated pricing and production/ordering policies that maximize the expected profits of the manufacturer and the distributor, as well as the distributor's optimal pricing and ordering policies without coordination, are developed. The focus of his study is placed on factors that make coordination an effective strategy for the manufacturer and distributor, coordination strategies and the coordinated policies that maximize both parties' expected profits and the joint expected profit.

Sueyoshi [41] presents a new application of Data Envelopment Analysis (DEA) to empirically determine prices of multiple products. This paper directs toward a new DEA application to the determination of a marginal cost pricing system or a profit based pricing system. Nonlinear properties of the new DEA approach are explored.

3. PROBLEM DESCRIPTION

The purpose of the study is to examine interrelations between the supply chain members, develop organizational and mathematical models for the structure of supply chain, supply procurement and commitment, and pricing strategies in a way to optimize the overall benefits of an integrated communication network.

In many situations, a supplier has a number of buyers whose ordering policies and supply chain performances are likely to differ from one another, and the supplier needs to allocate his capacity among them so as to provide the right quantity at an acceptable price.

The supplier under consideration is assumed to manufacture a strategic product for the supply chain in a high value-added capacity environment, and the buyers are assumed to manufacture different types of products which all require the same supply item. The supplier is not in a monopolistic situation, but once an agreement is reached, a strategic partnership will naturally evolve between the supplier and the buyers, enhancing the supplier's bargaining authority. The buyers are assumed to be behaving independently without knowing each others reactions and decisions while each aims to make their own production and purchase decisions concurrently. However a communication network such as remote connection, electronic data interchange (EDI) or internet-based e-commerce is known to exist to ensure that information flows freely between the supplier and each of the buyers. Thus the supplier possesses precise information about the production planning models of the buyers, so that in making pricing decisions, he is capable to analyze the individual reactions of the buyers. Symmetrically the supplier shares information on his production operations and cost drivers with the buyers to justify a fair price.

Furthermore there is a formal partnership plan in which the schedules, time fences and commitments are respected. This plan can be best expressed as a generic supply replenishment contract similar to Williams [15] between the supplier and his buyers to ensure long-term commitment, supply visibility and ease of operation.

A generic supply chain partnership contract exists between the supplier and the buyers. A two-year rolling forecast with monthly time buckets is provided, which is updated monthly by the buyers, by dropping off the past month and adding a new month. These forecasts are sent to the supplier monthly by the buyer. This conveys the advantage of giving the supplier long-term visibility. Given these forecasts, at the start of year T, the supplier and the buyers should renew the contract for year T+1. This implies that the supply replenishment and the price for the very next year T has already been determined a year before.

From the supplier's point of view, the contract renewal for year T+1 covers the supply quantity commitment and the price for year T+1. This one-year lead time which may be considered as the capacity-planning fence provides sufficient time to the supplier to carry out necessary capacity expansion if the buyers' requirements exceed the current capacity and the supplier finds it feasible to make that investment. Symmetrically it provides some time to the buyer to find some other supply source if the supplier is not able to provide all the requirements for year T+1. In the contract it is allowed that the supply quantity allocated to the buyer not be sufficient to meet the buyer's total requirements in T+1 and the final supply commitment be agreed upon at the end of negotiations. Once an agreement is reached, this is a 2-year commitment from the supplier's point of view. A simple scenario is depicted in TABLE 3.1.

TABLE 3.1 Supplier's commitment plan stated by the contract: An example

	t=1	t=2	.....	t=12	t=13	t=14	.....	t=24
Foreçast at t=0	30	40	.....	35	60	75		70
Supply Commitment before Contract Ren.	30	60	.....	60				
Supply Commitment after Contract Ren.	30	60	.....	60	55	85	.....	70

As far as shared risk and mutual commitment is concerned, each buyer also commits to purchasing a quantity equal to the manufacturing lead-time of the supply item which is taken to be 3 months in this study. Like the forecast, this commitment is a rolling mechanism with the last month's quantity dropped off and a new month's quantity added. According to the contract terms, the buyer is committed to the first three months

requisitions already stated, and he can not change the future requisitions in the contract by more than a given percentage till the end of the contract term. Once the contract is renewed, the supplier's commitment should be considered as an upper bound for monthly supply delivery. TABLE 3.2 shows such a scenario on the buyers's fluctuating forecasts and commitments. If the maximum percentage by which the forecasts can be decreased is specified as 50% in the contract, as the forecasts are updated at the end of the first month, the forecast for month t=6 for example could not be stated less than 30 units. The overall roll of the contract is depicted in FIGURE 3.1.

TABLE 3.2 Buyer's commitment plan stated by the contract: An example

Months for T=1	t=1	t=2	t=3	t=4	t=5	t=6	.....	t=12
Supply Commitment for T=1	30	60	25	40	70	80	.....	60
Forecast at t=0	30	40	20	35	60	75		35
Buyer Commitment	30	40	20					
Forecast at t=1		35	20	30	55	75		35
Buyer Commitment		40	20	35				
Forecast at t=2			25	25	50	70		35
Buyer Commitment			20	35	55			
Forecast at t=3				30	45	65	.....	30
Buyer Commitment				35	55	70		

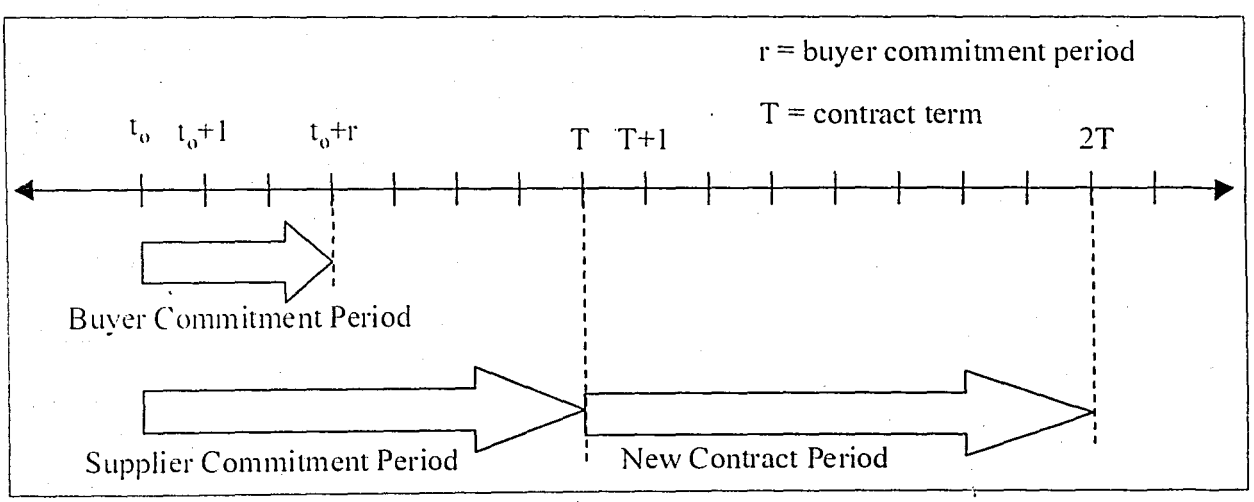


FIGURE 3.1 Outline of the contract between the buyers and the contractor

Since this research is basically concerned with developing the coordinated monthly production plans, the issue of designing the weekly delivery process is not studied in detail.

However a contract is complete only with delivery specifications. Any contract will naturally advocate a JIT delivery schedule with short and frequent replenishments. Thus a rate-based replenishment can be used in contracts such that the weekly delivery of the supply item is based on the actual consumption of the buyers and not on some forecasted replenishment schedule. This is accomplished by setting a maximum and a minimum stock level and communicating the weekly inventory on hand to the supplier electronically.

Since the design of the mathematical models to follow requires further clarification of the general contract features, some structural assumptions are made:

(a) The price of the supply item is presumed to possess a strong effect upon the competitiveness of the buyers to the extent that it may change their production and purchase plans.

(b) Demand is treated in a deterministic manner by simply expressing the fluctuations in the form of statistical expectations.

(c) Supply commitment determination should depend on both qualitative and quantitative measures value based on the past supply chain coordination, not mainly on the price.

(d) Information and visibility contained within supplier scheduling provides the supplier a far better opportunity for cost reduction which will naturally arise out of economies-of-scale in combining and sequencing runs to minimize change-over time and establishing long-term commitments with their own suppliers. This cost reduction is quantified as a decrease in the unit variable production cost of the supply item. Thus pricing is considered to be no longer an issue of dispute, but rather a technical parameter since fair share is to be exercised and all cost drivers and elements are to be acknowledged by all parties.

(e) Capacity disaggregation can be done either on a buyer basis where some aggregate quota is declared to the buyer without explicitly identifying the product or on a buyer/product basis where each such combination is treated separately and different quotas are declared. The choice is clearly dependent on the amount of information the supplier possesses and the variability among the products.

(f) The fixed costs involved in sale and purchase activities are not included in the production models of the supplier and the buyers since long-term supply chain coordination tends to reduce fixed procurement cost on behalf of both sides. However if

this is not the situation, it can be included into the objective functions without affecting the rest of the analysis.

(g) Since this research presumes that a single price will be declared to all the buyers for the sake of consistency, all the customers are integrated within the same price model. However if it is rational to declare different prices to different buyers, then the supplier needs to solve the pricing problem independently for each buyer to get a different price, the latter situation being computationally easier to solve.

None of these assumptions are restrictive enough to discredit the general applicability of the proposed approach. They can be relaxed and necessary modifications can be done easily wherever needed in the models to follow. Some suggestions to cope with these are also provided in next sections.

4. HIERARCHICAL PLANNING

Given a general contract and assumptions in the previous section, it is utmost important to design a supplier planning process which is compromised and accepted by all the buyers. This study proposes a hierarchical approach that aims to help the supplier decide on the supply capacity and the price to be declared to the buyers at the contract renewal time. In fact there exist two hierarchies: one is mainly for the supplier and the other is the overall supply chain hierarchy.

4.1. Supplier Hierarchical Planning

The supplier hierarchy that is summarized in FIGURE 4.1 consists of three main levels:

The first step is mainly concerned with determining the aggregate monthly capacity levels of the supplier for year T+1 aggregating the demand estimates of all buyers and possible variations as stated in the contracts. Although many alternative models can be used in order to calculate monthly capacity level, in this study a simplified linear program is proposed to determine the capacity utilization level for each period t=1,...,24 within a production smoothing formulation. In the second step, it is recommended to employ an assessment model for evaluating and ranking the buyers according to the value-added of their orders and their current supply chain performance. In other words, the supplier should decide on how to use the available supply capacity for the greatest business value and needs a proactive way of selecting the customers and the product segments. Thus, the results of the assessment procedure can be summarized in the form of some priority weights that are in turn used in a simple knapsack disaggregation model. The later concept may insinuate the family disaggregation model of Bitran and Hax [42]. Thus the aggregate capacity is disaggregated according to the results of this assessment.

Once the supply commitment to each buyer is determined, the supplier will determine the price for the supply item for year T+1. In fact the price is to be determined

by the total quantity contained in the forecasts for year T+1 and considering the buyer's production decisions and profit sharing perspective. Of course this approach does not override the negotiation process; instead it aims to develop reasonable figures to initialize the negotiation process between the supplier and the buyers. Supply commitments and the prices to be included in the new contract can only be finalized at the end of negotiations.

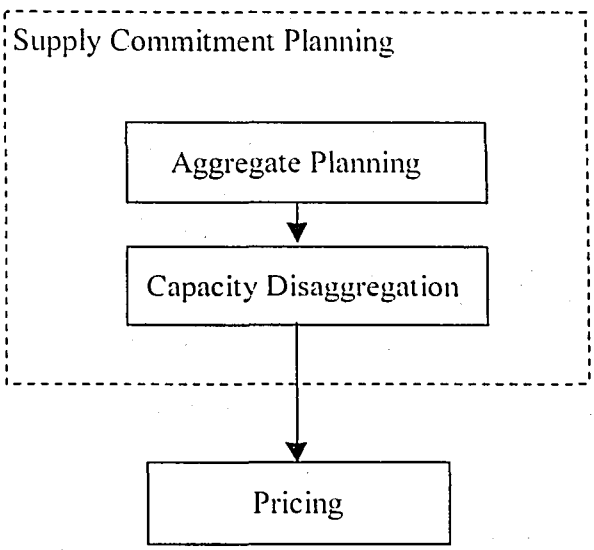


FIGURE 4.1 Decision hierarchy of the supplier

One modification that can be easily done in the sequence of decisions is to first decide on the price of the supply item and then to allocate the aggregate capacity among the buyers given the price. This situation is more applicable if the supply commitment is heavily dependent upon the pricing policy and the contract price should be included in disaggregating the aggregate capacity.

4.2. Supply Chain Hierarchy

Since global competition forces the manufacturers and their suppliers to coordinate their activities and decisions within the framework of supply chain management, all members should mutually share relevant information with the advancements in information technology and implement their decision process and information exchange on a formal basis by integrating their individual decision processes.

The supply chain hierarchy deals basically with the information exchange and communication framework between the supplier and his buyers. The complete coordination process is best described by a hierarchical planning system. First, the supplier sets the supply quotas and the price to each of the buyers and provides the estimated supply quantities using their estimated requsitions given in FIGURE 4.2. Then the buyers decide on the purchase and production quantities simultenęously, declare them to the supplier and estimates the production targets. Then the supplier finalizes the operational production schedule. In doing so, the supplier is committed fulfill the buyers requisitions as long as they satisfy the previously stated contract terms. The organizational coordination between decision making units of the buyers and the supplier is given in FIGURE 4.3 resulting from this hierarchy.The mathematical models developed to describe each of the decisions will be covered in detail in the following sections.

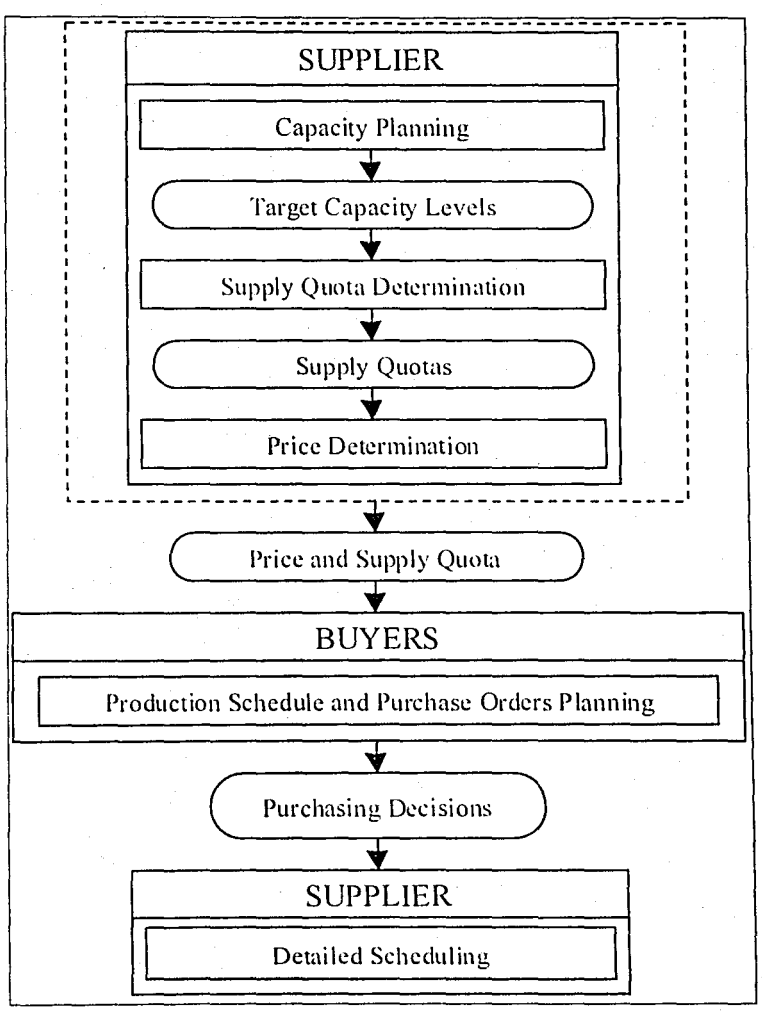


FIGURE 4.2 Supply chain hierarchy

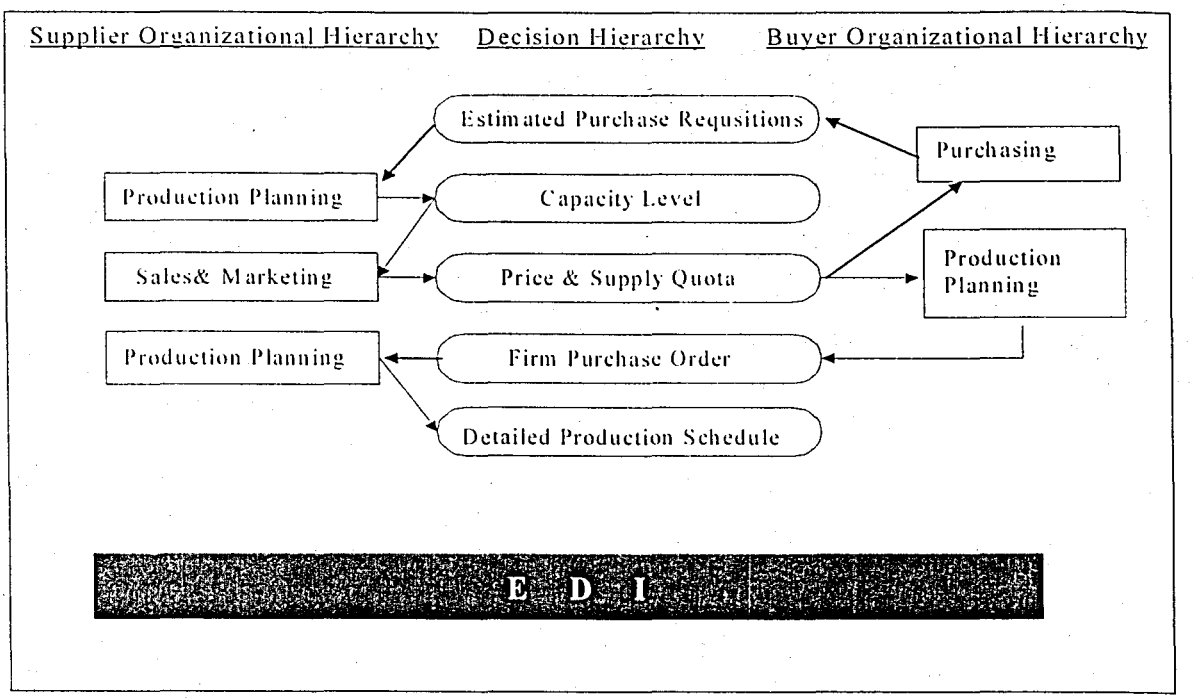


FIGURE 4.3 Organizational coordination in the supply chain hierarchy

4.3. Buyer's Production-Purchase Model

Since this research assumes that the buyers and suppliers are all connected over a global information technology infrastructure, the supplier possesses precise knowledge about the buyer's cost and capacity structure and demand estimates over the short term horizon so as to anticipate the buyer's reaction. In fact the supplier uses the following formulation as the integrated production-purchase model of buyer j:

PROBLEM (MOB(j)), j=1,....M:

$$Z_u^j = \min \sum_{t=13}^{24} \sum_{i=1}^{N} \left(s_{ijt} Y_{ijt} + c_{ijt} . X_{ijt} + h_{ijt} I_{ijt}^{+} + \pi_{it} I_{ijt}^{-} \right) + \sum_{t=13}^{24} (pu_{jt} US_{jt} + hm_{jt} IS_{jt}) \qquad (4.1)$$

subject to:

$$I^{+}_{ij,t-1} - I^{-}_{ij,t-1} + X_{ijt} - I^{+}_{ijt} + I^{-}_{ijt} = d_{ijt} \quad i=1....N,\ t=13,....,24 \tag{4.2}$$

$$IS_{j,t-1} + US_{jt} - \sum_{i=1}^{N} a_{ij} X_{ijt} - IS_{jt} = 0 \quad t=13,....24 \tag{4.3}$$

$$X_{ijt} \le CAP_{ijt} Y_{ijt} \quad i=1,....,N,\ t=13,....24 \tag{4.4}$$

$$US_{jt} \le Q^{j}_{t} \quad t=13,....24 \tag{4.5}$$

$$X_{ijt},\ I^{+}_{ijt},\ I^{-}_{ijt},\ US_{jt},\ IS_{jt} \ge 0;\ Y_{ijt} = \{0,1\}\ i=1,....,N,\ t=13,....24,\ j=1,..,M \tag{4.6}$$

Index Set:

i = {1...N}= set of products

j = {1...M}= set of buyers

t = {13,....24}= set of periods covered in the new contract period

Decision Variables for Buyer j:

X_{ijt} = production amount of product i for buyer j in period t

I^{+}_{ijt} = inventory of product i for buyer j in period t

I^{-}_{ijt} = shortage of product i for buyer j in period t

US_{jt} = non-coordination purchase amount of the supply for buyer j in period t

IS_{jt} = inventory level of the supply item in the stock of buyer j in period t

$$Y_{ijt} = \begin{cases} 1 & \text{if } X_{ijt} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Parameters for Buyer j:

s_{ijt} = fixed cost of producing product i for buyer j in period t

c_{ijt} = unit variable cost of producing product i for buyer j in period t

h_{ijt} = unit holding cost of product i for buyer j in period t

π_{ijt} = shortage cost of product i for buyer j in period t

pu_{jt} = non-coordination purchase price of the supply item for buyer j in period t

hm_{jt} = unit holding cost of the supply item for buyer j in period t

d_{ijt} = demand requirement of product i for buyer j in period t

a_{ij} = amount of the supply item needed in one unit of product i for buyer j

CAP_{ijt} = total production capacity of product i for buyer j in period t (in units)

Q^{j}_{t} = supply quota to be allocated to buyer j in period t

In the mixed integer mathematical model given above, the buyer's problem can be stated as the problem of determining the production schedule for N end products within a single-level capacitated lot sizing approach and the purchase plan for the supply item given the price pu and the supply capacity over t=13,...,24. Here backlogging can be interpreted as that portion of supply purchases not planned from this particular supplier and the backlogging cost as the cost of providing the supply item from some other supplier. In MOB(j). an aggregate supply capacity denoted by Q^{j}_{t} is included for each buyer. If the capacity allocation is to be made on a buyer/product basis, it can be easily modified by redefining Q^{ij}_{t} as supply quota allocated to buyer j to produce product i in period t.

The objective function (4.1) aims to minimize the production- and inventory-related costs of the products they manufacture themselves as well as the purchase and inventory holding costs of the supply item. The assumption (h) precludes the existence of

binary variables for the fixed purchase cost. The material balance for the end products are expressed by (4.2) while the material balance for the supply item is given in (4.3). Constraints (4.4) simply guarantee that setup be done whenever there is a production in any period, and (4.5) express that the purchase orders of the buyers can not exceed the supply quota declared by the supplier.

5. AGGREGATE CAPACITY PLANNING

The aim of the aggregate capacity planning for the supplier is to determine the capacity utilization level for each period over the whole planning horizon t=1,...24 within a production smoothing formulation for the supplier.

As the top level of the hierarchy, the supplier deploys a mathematical model to plan capacity in the medium term. At the contract renewal time, the supplier should formulate a model which adequately represents his constraints and costs related with capacity. Although a number of models could be used to serve this purpose, in this study a simplified mixed integer program is proposed to determine the capacity utilization level for each period t=1,...24 within a production smoothing formulation.

The model includes economies/diseconomies of scales in capacity costs (fixed and variable costs defined either in all units intervals or incremental intervals with multiple cost breaks), penalties for increasing and decreasing capacity levels from one period to the next, inventory holding costs for years T and T+1, and the cost of not satisfying total supply chain requirements in year T+1.

Problem (CAS):

$$Z = \min \sum_{t=1}^{24} \left\{ \sum_{j=1}^{K} (ss_{jt}\alpha_{jt} + sc_{jt}CAP_{jt}) + a_t\Delta_t^+ + b_t\Delta_t^- + hs_tSI_t^+ + \pi_tSI_t^- \right\} \qquad (5.1)$$

subject to:

$$SI_{t-1}^+ + SX_t - SI_t^+ = FS_t \qquad t=1,2,3 \qquad (5.2)$$

BOĞAZİÇİ ÜNİVERSİTESİ KÜTÜ

$$SI^{+}_{t-1} + SX_t - SI^{+}_t = ES_t \qquad t=4,\ldots,12 \tag{5.3}$$

$$SI^{+}_{t-1} - SI^{-}_{t-1} + SX_t - SI^{+}_t + SI^{-}_t = RS_t \qquad t=13,\ldots,24 \tag{5.4}$$

$$\sum_{j=1}^{K} CAP_{jt} = \sum_{j=1}^{K} CAP_{j,t-1} + \Delta^{+}_t - \Delta^{-}_t \quad t=1,\ldots 24 \tag{5.5}$$

$$SX_t \le \sum_{j=1}^{K} CAP_{jt} \quad t=1,\ldots 24 \tag{5.6}$$

$$K_{j-1}\alpha_{jt} \le CAP_{jt} \le K_j\alpha_{jt} \quad j=1\ldots K,\ t=1,\ldots 24 \tag{5.7}$$

$$\sum_{j=1}^{K} \alpha_{jt} \le 1 \quad t=1,\ldots 24 \tag{5.8}$$

$$\Delta^{+}_t \le KAP \quad t=1,\ldots 24 \tag{5.9}$$

$$SX_t,\ CAP_{jt},\ SI_t^+,\ SI_t^-,\ \Delta_t^+,\ \Delta_t^- \geq 0\ ;\ \alpha_{jt}\{0,1\} \qquad (5.10)$$

Index Set:

t ={1.....24}= set of periods in the planning horizon

j ={1...K}= set of capacity intervals

Decision Variables:

SX_t = production amount in period t

CAP_{jt} = capacity utilization level in interval j in period t

SI_t^+ = units of ending inventory in period t

SI_t^- = units of supply item backordered at the end of period t

Δ_t^+ = increase in production rate from period t-1 to t

Δ_t^- = decrease in production rate from period t-1 to t

$$\alpha_{jt} = \begin{cases} 1 & \text{if } CAP_{jt} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Parameters:

$(K_{j-1}..K_j)$ = limits of the capacity interval j for the supply item such that $K_{j-1} \leq K_j$, j=1....K. $K_0=0$ and K_K is taken as unlimited in period t

FS_t = aggregate committed replenishment of all buyers in period t, t=1,2,3.

ES_t = aggregate forecasted replenishment of all buyers in period t, t=4,....,12

RS_t = aggregate demand forecast of all the buyers for the new contract year in period t. t=13.....24

sc_{jt} = unit variable cost which applies in capacity interval j in period t

ss_{jt} = fixed cost which applies in capacity interval j in period t

a_t = cost to increase the capacity level by one unit from period t-1 to period t

b_t = cost to decrease the capacity level by one unit from period t-1 to period t

hs_t = unit holding cost of the supply item in the supplier's stock in period t

π_t = unit backorder cost of the supply item in period t

KAP = maximum possible capacity level increase between consecutive intervals

The objective function (5.1) minimizes the sum of capacity costs, inventory-related costs and production rate change costs in years T and T+1 over t=1,....24. It can be expanded by including the employment level and overtime decisions if necessary.

Constraint (5.2) to (5.4) express the material balance in accordance with the contract terms. Constraints (5.2) and (5.3) are written separately to emphasize the difference between FS_t and ES_t. Here FS_t denotes the replenishments committed by the buyers and already planned by the supplier while ES_t shows the expected replenishments which are to be confirmed later by the buyers and to be planned yet by the supplier following the committed lead-time. Since the supplier has already committed to delivering the replenishment for the very first year (t = 1,...,12) the material balance with stocks and backlogs.

By assumption (e), backlogging is included in (5.4). Since some order flexibility is included in the contract by permitting the buyers to increase their initial requirements by a given percentage beyond the committed lead-time, RS_t should be determined by inflating the sum of the buyers' forecasts in month t by that percentage to obtain the highest possible requisitions.

Capacity fluctuations are controlled by constraints (5.5). Constraint (5.6) show the relationship between capacity and production levels. Constaints (5.7) together with constraints (5.8) sets the fixed cost that will apply to the capacity utilization level variable.

Constraints (5.9), on the other hand, imply that capacity increase in each period is upper-bounded. Constraints (5.10) simply guarantee non-negativity bounds of appropriate variables.

Constraints (5.5) and (5.6) are included to show the all-unit capacity-cost model. If incremental modeling is required, (5.5) and (5.6) can be replaced respectively by (5.11) and (5.12).

$$(K_j - K_{j-1})\alpha_{j+1} \leq CAP_{jt} \quad j=1,....K,\ t=t_0,....T \tag{5.11}$$

$$(K_j - K_{j-1})\alpha_j \geq CAP_{jt} \quad j=1,....K,\ t=t_0,....T \tag{5.12}$$

In both cases since no relative magnitude restriction is imposed on ss_{jt} and sc_{jt} over different intervals, both economies and diseconomies of scale in capacity can be treated in the above formulation. A diseconomies of scale situation may especially arise if production over a certain level is motivated by an attempt to preserve the market share and not to lose buyers.

Since (CAS) is designed to be an aggregate model, the problem dimensionality does not pose a complexity problem and the existence of binary variables does not inhibit the possibility of solving it by using a commercial software. However if this is not the situation, the inclusion of fixed cost can be avoided intentionally to facilitate the use of linear programming codes for solving the aggregate problem by eliminating economies and diseconomies of scale from the model formulation.

6. DETERMINATION OF SUPPLY QUOTAS TO CUSTOMERS

After the aggregate capacity plan of the supplier is determined, the disaggregation model has to realize the allocation of this capacity among the buyers. The value of the demand should be reconciled against the business strategy and demand that does not provide adequate strategic value should be identified. That is, the supplier should determine the importance of the buyers and then allocate its capacity. In determining the value of a buyer, this study proposes Analytical Hierarchical Process (AHP) method to be employed to disaggregate capacity among buyers. Although there are linear programming-based models available to assist decision makers in allocation of order quantity among buyers, there are certain qualitative issues that are not easily quantifiable, and thus AHP developed by Thomas L. Saaty [24] may be very useful tool for splitting large-volume purchases among several buyers.

6.1. Determination of Customer Priority Indices and AHP

The AHP provides a framework to cope with the multiple criteria situations involving intuitive, rational, qualitative and quantitative aspects. It first structures the problem in the form of a hierarchy to capture the basic elements of a problem and then derives ratio scales to integrate the perceptions and purposes into a synthesis. In the hierarchical structure, all elements in a level are pairwisely compared with respect to the elements in the above level, and paired comparisons are used to elicit judgments. Then the synthesis of judgments is obtained as a result hierarchical re-composition in order to find the best decision.

6.1.1. Construction of the AHP Tree to Evaluate The Buyers

In order to determine the prioritization of the buyers, supply chain performances of the buyers should be evaluated. This is a multi-objective decision problem encompassing many tangible and intangible factors in a hierarchical manner. In fact in literature and practice there exist a large number of supply chain measures among which a comprehensive set is selected to form a company independent AHP tree as listed below, and the AHP tree proposed to TEE initially is given in Appendix A.

(1) Evaluation of Strategic Cooperation

1.1 Supplier improvement / certification programs

1.2 Defining common mission and vision

1.2.1 Defining common performance criteria

1.2.2 Participation in benchmarking studies

1.3 Commercial support

1.3.1 Turnover

1.3.2 Man hours spent in marketing department

1.3.3 Trade potential and efficiency of the procurement organization of the buyer firm

1.3.4 Directive and cooperative leadership in case of customer complaints

1.3.5 Creating quick answers and solutions to questions and problems

1.3.6 Number of suppliers (at the product base)

1.3.7 Supplier contract application approaches

1.3.8 Participation in cost reduction studies

1.3.9 The opportunity of direct shipment to the production line

1.3.10 Shipment and transportation conditions

1.4 Technical support

1.4.1 Effectiveness of technical consulting and support services

1.4.2 Simultaneous engineering practices

1.4.3 Support to develop new product competency

(2) Evaluation of Infrastructure and Managerial Functions

2.1 Payment performance

2.1.1 Deviations from payment term

2.1.2 Deviations from payment amount

2.1.3 Late payments

2.1.4 Guarantee value

2.1.4 Amount of checks protested

2.1.5 Error in maturity dates

2.1.6 Customer risk accounts (weekly)

2.1.7 Ratio of checks less than 50 million TL.

2.1.8 Ease of relation with the customer at account adjustments

2.2 Manufacturing infra-structure

2.2.1 Flexibility (adaptation to changes in demand)

2.2.4 Manufacturing technologies (CAD/CAM, CAPP, robotics, automation etc.)

2.2.5 Management technologies (ERP, JIT, TPM, TQM etc.)

2.2.6 New product design and development competency

2.3 Competitive advantage

2.3.1 Market share

2.2.2 Expected improvement in the market in long and middle term

2.2.3 Competitive priorities

2.2.4 Action plans

2.2.5 Competitive strategies

2.4 Quality management

2.4.1 Quality certificates

2.4.2 Total Quality Management

2.4.3 Statistical Process Control

2.4.4 Quality control systems and number of quality control personnel

(3) Evaluation of Supply Chain Management System

3.1 Simultaneous planning system

3.1.1 Percentage of orders changed

3.1.2 Percentage of changes in order amounts

3.1.3 Re-planning period / frequency

3.1.4 Accuracy of forecasts

3.1.5 Length of planning period

3.2 Sales management system

3.2.1 Structure of formal sales procedures

3.2.2 Man hours spent in sales department

3.2.3 Number of monthly sales documents

3.2.4 Ratio of urgent orders to total number of orders

3.2.5 Ratio of actual orders to quoted orders

3.3 Costing / pricing system

3.3.1 Average cost of an order

3.3.2 Costs of supply chain financing and planning

3.3.3 Level (cost) of inventories

3.3.4 Actual cost / standard cost

3.3.5 Actual shipment cost

3.4 Information management system

3.4.1 E-commerce

3.4.2 Information sharing

3.4.3 Documentation system

3.4.4 Reporting quality

3.4.5 Data accuracy

6.1.2. AHP Implementation in TEE

The applicability of the AHP approach in supply quota determination is tested in TEE, a leading electric motor manufacturer in Turkey. TEE was established in 1965 with the partnership of General Electric, Arçelik A.Ş. and Türkiye İş Bankası in Topkapı-Istanbul. Today the company has 3 production (TOPEM, SUTEM, ESKOM) sites and a coordination center and 99% of its shares belong to Arçelik. There are total 2127 employees of which 351 are white collar in TEE. The company has 140 domestic and 35 export buyers and supply critical items (motors) for the buyers' production.

Once TEE managers were given full knowledge and understanding of factors and the supply chain tree was explained to them, they require some modifications in the

original tree since they stated that supply chain relationship shows variations between domestic and export buyers; furthermore it is indicated that each of the three plants has a different supply chain management approach for each buyer group. Since TOPEM has two main buyers, one local and one export, they used a single AHP tree. Thus five different AHP trees are constructed for SUTEM Local Customers, SUTEM Export Customers, ESKOM Local Customers, ESKOM Export Customers and TOPEM.

The prioritization procedure is done separately for each main factor by the related departments. The factors are presented in the matrix form and department managers give weights to the sequence of factors by using the classical AHP rating scale in comparisons.

After forming preference matrices, the process moves to deriving relative weights for the factors. Here, the calculations involved in AHP are done by using Expert Choice that easily handles analytical hierarchical planning problems. Consistency of all matrices is controlled. The consistency ratio (CR) is a degree to which the pairwise comparisons are consistent. This check allows to detect misjudgments in comparisons. Not only this reduces careless errors, but also it reveals to the managers their own unsuspected bias or exaggeration concerning one or more of the comparisons. The matrices with CR larger than 0.10 are revalued and corrected.

The prioritization is given separately for SUTEM Domestic Customers, SUTEM Export Customers, ESKOM Local Customers, ESKOM Export Customers and TOPEM. The prioritization levels for all plants are given in Appendix B.

6.2. Capacity Disaggregation

Once the target capacities for each month are determined by the supply chain tool in this study, the next issue is to allocate the aggregate capacity among the buyers. This decision is particularly more important for the supplier who provides strategic items for the supply chain in a high value added capacity environment and whose capacity utilization may limit the supply chain throughput. In fact, it is necessary to measure the buyers' value

in terms other than just revenue and to be able to express the value in terms of supply chain performances.

Capacity disaggregation is done on a buyer basis. Therefore, the disaggregation of the capacity among the buyers are realized as determining priority indices for buyers. These indices are obtained by using the AHP methodology.

Then these priority indices are deployed in three disaggregation models:

(a) Weighted Allocation Model

(b) Ordered Allocation Model

(c) Regular Knapsack Model

6.2.1. Weighted Allocation Model and Its Implementation:

In this model, the aim is to allocate the capacity among the buyers according to their weights obtain from AHP. Thus the supply chain efficiency of each buyer from the supplier's point of view is defined as a direct buyer priority. According to this priority, supply quotas are allocated to buyers with the following calculation:

$$Q_t^j = CAP_t \, wt_j \qquad (6.1)$$

CAP_t : Aggregate capacity of supplier in month t

Q_t^j : Allocation amount for buyer j in month t

wt_j : Weight of buyer j obtained from AHP

Since the main emphasis is given to buyers' weights, the demand values are not considered in this model. Therefore, in case of a buyer having small priority and high

demand may not be satisfied if the supplier capacity is not enough. The supplier may allocate more items than the demands of the buyers with highest priority.

6.2.2. Ordered Allocation Model and Its Implementation

Here, the priorities for each buyer are used to obtain a rank order among the buyers. Allocation is realized through this order. In this model, the first priority belongs to the highest weighed buyer. After all demand of this buyer is met, the buyer with the second highest weight is considered. The procedure is going on in this way. The supplier continues the allocation as much as the aggregate capacity allows.

CAP_t : Aggregate capacity of supplier in month t.

$w_1 > w_2 > w_3 > w_4 \ldots\ldots > w_n$: Buyers are ranked in a decreasing priority order.

Available capacity is allocated to the buyers by their weight order, by satisfying whole demand of that order buyer as long as there is available remaining capacity and passing to next level.

6.2.3. Regular Knapsack Model and Its Implementation

This is a mathematical model that takes into account the penalty of not satisfying and/or exceeding specific buyer demands. Since buyers in any supply chain expect JIT replenishment form their suppliers, allocating more than they need is as undesirable as not meeting their demand. The format of the proposed model is similar to the Regular Knapsack Model (RKM) of Bitran and Hax [42,43]. The main constraint which dictates that the sum of the disaggregated supply capacity for all buyers should equal the capacity already determined in the top level aims to establish consistency between these two levels. Since customers in any supply chain expect JIT replenishment from their suppliers, allocating more than they need is as undesirable as not meeting their demand. Supply chain

coordination implies the replenishment of the right quantity at the right time and disaggregation process should take this into account. In doing this, some customer differentiation is implemented by using their AHP weights such that the more supply chain efficient they are, the higher their weights are. To serve this purpose, a continuous knapsack problem can be defined to minimize the sum of the weighted squared deviations from the stated customer orders.

PROBLEM (SAP(t)),t=13,....24:

$$\text{Min } W_t = \sum_{j=1}^{M} w_j (Q_t^j - RS_{jt})^2 \qquad (6.2)$$

subject to:

$$\sum_{j=1}^{M} Q_t^j = CAP_t \qquad (6.3)$$

$$Q_t^j \geq 0 \quad j=1,....M \qquad (6.4)$$

CAP_t : Aggregate capacity of supplier in month t

Q_t^j : The supply capacity to be allocated to buyer j in period t

w_j : The weight assigned to buyer j coming from AHP.

RS_{jt} : Demand forecast of buyer j for period t

Here the objective function (6.2) minimizes the weighted deviations from buyer forecasts where w_j is used as a relative measure of deviating from the given forecasts

(obtained at the end of some assessment approach cited above). Constraint (6.3) assures that the total supply capacity committed to all buyers in t does not exceed CAP_t determined by (CAS). This problem is solved for each time period independently, and the optimal supply quantities can be found by solving the convex objective function subject to the knapsack constraint (6.3) and given by

$$Q_t^j = RS_{jt} - \alpha_j\left(\sum_{j=1}^{M} RS_{jt} - CAP_t\right) \quad \text{where} \quad \alpha_j = \prod_{\substack{k=1 \\ k\neq j}}^{M} w_k \Big/ \sum_{r=1}^{M}\prod_{\substack{k=1 \\ k\neq r}}^{M} w_k \tag{6.5}$$

This implies that capacity deficit (excess) is allocated to buyers in proportion to α_j which is a function of w_j. Let's consider the situation in which there are five buyers with demand forecasts in t given by RS_{jt} = (70,50,30,80,130) and the normalized weights w_j = (0.15, 0.25, 0.30, 0.10, 0.20). Suppose $CAP_t = 300 < \sum_{j=1}^{M} RS_{jt} = 360$. Then this capacity deficit is allocated in proportion with α_j = (0.230, 0.138, 0.115, 0.345, 0.172). Thus the supply commitments by (6.5) are found to be

$$Q_t^1 = 56.20 \quad Q_t^2 = 41.72 \quad Q_t^3 = 23.10 \quad Q_t^4 = 59.30 \quad Q_t^5 = 119.68$$

As it can be seen, the disaggregation variables are not bounded in (SAP) since the supplier is prompted to behave exactly according to the allocation scheme derived out of the assessment procedure without imposing any more constraints. However if this is not the case, one can modify the iterative RKM algorithm of Bitran and Hax [42, 43] by using the objective function of (6.2) of (SAP) and the bounding constraints. Implementations of models are shown for TEE-Sutem Plant in Appendix C.

7. PRICING MODEL

The important issue that remains to be solved by the supplier is the determination of the price of the supply item while anticipating the buyer's reaction to that price. Traditionally the buyers try to reduce the price as low as possible while the supplier is obliged to accept the lowest possible margin out of the fear of losing business and tries to figure out how to regain the profit. In supply chain management, these tactics are eliminated in favour of a "fair" price in the sense that the supplier can make a profit, and the buyer can still remain competitive in the market place. In addition to lower prices, supply chain partnership also results in constant pricing over a longer period of time. To arrive at such a price compromise, both parties should reveal their costing elements and should share information to enhance mutual cost reduction efforts. Since supply chain implies long-term commitment, the price is determined on the entire business volume, not for a given item for a specific purchase order. Thus the price breaks in the traditional sense where they exist mainly to enable the supplier to promote a larger quantity at one time are not as significant as before. In supply chain management, on the other hand, it is necessary to quantify the expected cost savings of the supplier and to justify a rational share of these savings to the buyers.

In order to give a sight into possible negotiation schemes, let's consider an example similar to the one given in Schorr [45]. Suppose that without any supply chain coordination the unit cost of the supply item is \$0.80 and the selling price is \$1.00. This yields a 20% profit margin and a unit profit of \$0.20 to the supplier. Now let's assume that a supply chain partnership has been arranged and the unit cost is reduced to \$0.60. Then negotiations may commence and lead to different pricing alternatives and it becomes necessary to compare their attractiveness both from the supplier's and buyer's point of views. The supplier deserves a reasonable profit and the buyer deserves a reasonable price reduction in this long-term commitment. The buyer's expectations of this cost reduction play an even more important role if the pricing under consideration clearly affects his own price and the market position. One such an alternative may require that the profit margin be preserved on the supply item and the unit selling price be \$0.75 while the supplier gets a profit of \$0.15 per unit. Although the profit margin remains the same, the supplier may not

be very happy with this since his profit is reduced in actual terms. In a second alternative, the amount of profit per item may be fixed at \$0.20, the selling price is \$0.80, so that the profit margin is increased to 25%. This situation is less desirable from the buyer's point of view and he may refuse to accept it since this price is higher than the price in the first alternative. Of course it is possible to generate many alternatives similar to these, but the issue is to find a compromise both parties can agree upon. The price is naturally to be determined by the profit the supplier is willing to sacrifice and the cost reduction the buyer is planning to attain. The pricing model is based upon the analysis of non-coordination and coordination situations.

In the non-coordination case, first the supplier will solve each buyer's problem MOB(j) independently using the price pu to determine their individual expected purchases without supply chain coordination. Then let $TC_j^* = Z^j$ and US_j^* be the associated purchase orders. Having determined the purchase orders from all the buyers, the supplier can compute his total non-coordination revenue and profit directly by totalling $TR^* = \sum_{j=1}^{M} \sum_{t=13}^{24} pu_{jt} US_{jt}^*$ and $\Pi^* = \sum_{j=1}^{M} \sum_{t=13}^{24} (pu_{jt} - cu) US_{jt}^*$.

In supply chain coordination, rational buyers will expect to lower their costs below TC_j^* due to supply chain commitment they provide to the supplier. Thus if Z_c^j denotes the supply chain cost of buyer j, the buyer's cost expectation can be expressed by $Z_c^j \le \alpha_j . TC_j^*$ for some parameter α_j $(0 < \alpha_j < 1)$. Specifically Z_c^j is defined by

$$Z_c^j = \sum_{t=13}^{24} \sum_{i=1}^{N} \left(s_{ijt} . Y_{ijt} + c_{ijt} . X_{ijt} + h_{ijt} . I_{ijt}^{+} + \pi_{ijt} I_{ijt}^{-} \right) + \sum_{t=13}^{24} (p_{jt} DS_{jt} + hm_{jt} IS_{jt}) \qquad (7.1)$$

where $p_{jt} \in [pl,pu]$ is the supply chain price to be determined by the supplier and DS_{jt} is the supply chain purchase order of the buyer j in period t.

The supplier on the other hand will be able to reduce his production costs to c since he can manage his production operations better in supply chain partnership. Since he is expected to behave rationally himself, he will anticipate an increase in his profit position or at least the profit of the non-coordination case. Thus, the supply chain expectation of the supplier can be expressed by

$$\sum_{t=13}^{24}\sum_{j=1}^{M}(p_{jt}-c)\,DS_{jt} \geq \Pi^{*} \tag{7.2}$$

Alternatively, he may sacrifice some part of his revenue on behalf of long-term business commitment and consent a portion of his uncoordinated revenue. That is,

$$\sum_{t=13}^{24}\sum_{j=1}^{M}p_{jt}\,DS_{jt} \geq \beta.TR^{*} \quad \text{for some } 0<\beta\leq 1 \tag{7.3}$$

The supplier's production cost is not explicitly treated in (7.3) since the profit expectations are indirectly included in the estimation of β. Either (7.2) or (7.3) should be included into any pricing model in order to incorporate supplier's expectations. The parameters α and β should be determined through a value and costing analysis on a win-win basis and firmly confirmed at the end of negotiations. Here α measures the degree of profit sharing of the buyers, and β is a quantified measure of the cost reduction which the supplier can attain through visibility contained in the contract. The supplier will then try to determine the price in accordance with these mutual expectations. Since the pricing model is designed to reveal the buyers' reactions to the changes in the price, it mainly includes the production and purchase constraints of the buyers.

The production and inventory-related costs of the supplier have already been covered in the first level and the consequences of capacity and production decisions have been conveyed to this level through Q_t^j; thus, it is not necessary to include them in the

pricing model once more. Although the fixed cost of running business with different buyers has not been addressed, assumption (f) eliminates this need. By assumption (g), the supplier should consider the reactions of all the buyers simultaneously within the same model to determine the common price p_t.

Since it is assumed that an aggregate capacity planning has already been done, the production and inventory-related costs of the supplier should have been already covered and the consequences of capacity and production decisions conveyed to this level through monthly aggregate capacity parameters; thus it is not necessary to include them in the pricing model once more. It is assumed that all the buyers will be considered simultaneously within the same model to determine the common price p_t. Naturally, the objective function of the supplier will be sales maximization in the pricing model integrating all buyers. The integrated price model (IPM) is as below:

PROBLEM (IPM):

$$R_d = \text{Max} \sum_{t=13}^{24} \sum_{j=1}^{M} p_t DS_{jt} \tag{7.4}$$

subject to:

$$\sum_{t=13}^{24} \sum_{j=1}^{M} (p_{jt} - c) DS_{jt} \geq \Pi \tag{7.5}$$

$$\sum_{t=13}^{24} \sum_{i=1}^{N} \left(s_{ijt} . Y_{ijt} + c_{ijt} . X_{ijt} + h_{ijt} . I_{ijt}^{+} + \pi_{ijt} I_{ijt}^{-} \right) + \sum_{t=13}^{24} \left(p_t DS_{jt} + hm_{jt} IS_{jt} \right) \leq \alpha_j TC_j^{*} \qquad j=1,....M \tag{7.6}$$

$$IS_{j,t-1} + DS_{jt} - \sum_{i=1}^{N} a_{ij} X_{ijt} - IS_{jt} = 0 \quad j=1,....M,\ t=13,....24 \tag{7.7}$$

$$I^{-}_{ij,t-1} - I^{-}_{ij,t-1} + X_{ijt} - I^{+}_{ijt} + I^{-}_{ijt} = d_{ijt} \quad i=1,....N,\ j=1,....M,\ t=13,...,24 \tag{7.8}$$

$$pl \le p_t \le pu \quad i=1,...N,\ j=1....M,\ t=13,....,24 \tag{7.9}$$

$$X_{ijt} \le CAP_{ijt}\, Y_{ijt} \quad i=1,....,N,\ j=1,...,M,\ t=13,...,24 \tag{7.10}$$

$$DS_{jt} \le Q_{jt} \quad j=1,...,M,\ t=13,...,24 \tag{7.11}$$

$$X_{ijt} \ge 0,\ I^{-}_{ijt} \ge 0,\ I^{-}_{ijt} \ge 0,\ Y_{ijt} = 0,1,\ DS_{jt} \ge 0,\ IS_{jt} \ge 0 \quad i=1,...,N,\ j=1,...,M,\ t=13,...,24 \tag{7.12}$$

The objective function (7.4) represents the supplier's revenue maximization. Constraints (7.5) and (7.6) express the profit and total cost expectations of the supplier and the buyers respectively in supply chain coordination as discussed before. The material balances for the supply item and the end products in the stocks of buyers are given by (7.7) and (7.8). The fact that the price can assume values in a given range is ensured in (7.9)

Constraints (7.10) and (7.11) ensure that fixed costs are incurred when production and purchase is scheduled in period t, respectively.

Naturally as the values of α_j's are decreased from 1.00, the price and the profit of the supplier continuously declines until minimum values for α_j's denoted as αmin_j are reached. At $\alpha_j = \alpha min_j$, the constraints (7.5) are satisfied as active constraints and the supplier's profit turns out to be π^* ; no feasible solution can be found for (IPM) for values lower than αmin_j . Thus feasible solutions can only be found for certain values of $\alpha_j \in [1.00, \alpha min_j]$. In case a feasible solution can not be found for (IPM) for a given set of α_j values as dictated by the buyers, new values should be negotiated to be used in the model.

Although (IPM) can be used as a tool for negotiations between the supplier and the buyers, it considers the problem from the supplier's point of view and attempts to establish a compromise with buyer-defined α_j's. Instead a rational choice of α_j's may be obtained by assuming a complete coordination situation as if there is a central agent aiming to optimize the objective functions of all parties simultaneously. This implies that the central agent's objective function should address the maximization of the supplier's profit and minimization of buyers' cost. Thus the complete coordination situation can be described by the following model:

CIPM:

$$CP = \text{Max} \sum_{t=13}^{24} \sum_{j=1}^{M} p_{jt} DS_{jt} - \sum_{j=1}^{M} \Big\{ \sum_{t=13}^{24} \sum_{i=1}^{N} \left(s_{ijt} . Y_{ijt} + c_{ijt} . X_{ijt} + h_{ijt} . I_{ijt}^{+} + \pi_{ijt} I_{ijt}^{-} \right) + \sum_{t=13}^{24} \left(f_{jt} Z_{jt} + p_t DS_{jt} + hm_{jt} IS_{jt} \right) \Big\} \quad (7.13)$$

subject to (7.5)-(7.12) with α_j=1.0

Once (CIPM) is solved, the compromise α_j for buyer j can be found by determining the ratio between the cost of buyer j as found in the solution of (CIPM) and

TC_j^*. That percentage will denote the cost reduction that buyer j can achieve in a supply chain coordination.

Obviously the multiplicative term $p_t.DS_{jt}$ of variables in (IPM) and (CIPM) appearing in (7.4), (7.5), and (7.6) makes these problems nonlinear which are difficult to solve optimally. Thus it is necessary to design a global search procedure to get a good solution for the mixed integer models (IPM) and (CIPM) with nonlinear mathematical relations. and the metaheuristic algorithm **SUPRIME** designed to serve this purpose are described in the following section.

Below. a simple single simple pass enumerative search is given to discover the dynamics of the problem where prices are selected iteratively in order to avoid the nonlinearity of the problems. Both problems (IPM) and (CIPM) will have feasible solutions only over a certain price range $pl \leq p(1) \leq price \leq p(2) \leq pu$. For price<p(1), (7.5) requires large purchase quantities which can not be found satisfying (7.6) and (7.10) while for price>p(2) the complicating constraints become (7.6). Thus it is important to determine the critical prices p(1) and p(2) and solve either of the problems incrementing the price iteratively over [p(1),p(2)]. Although the purchase quantities will tend to decrease as price increases. the objective function due to the multiplicative relationship in (7.4) does not depict a uniform behaviour as a function of price. It might increase or decrease depending on the relative effect of price decrease upon the purchase quantities. Thus it is necessary to search the whole region over [p(1),p(2)] in order to finalize the solution. Search steps are below:

(a) Define a proper step size Δ as an input parameter which can be adjusted to the level of accuracy desired, n*= min { n: (IPM(n)) is feasible}, n**= max { n: (IPM(n)) is feasible}. Start with n=1, n*=0, n**=0 and p(n) = pl.

(b) Construct (IPM) or (CIPM) by using p(n). If the problem is infeasible, go to [d]. Otherwise, if n*=0, let n*=n, and p(1)=p(n*). Determine DS(n) for each buyer and the objective function value $R_d(n)$ or $CR_d(n)$. Go to (c).

(c) Let n=n+1, p(n)=p(n-1)+Δ and go to (b).

(d) If n*=0, let n = n+1 and p(n)=p(n-1)+Δ and go to (b); otherwise let n**=n and p(2)=p(n**) and terminate. The solution with max{ $R_d(n)$ or $CR_d(n)$} is chosen as a local optimum solution.

It is important to note that in any solution to the problem (IPM) and (CIPM) price works against holding cost of the supply item and backordering costs of end items while backordering and inventory holding costs act against production and setup costs. Thus the backlogging cost plays an important role in the production and purchase decisions of the buyers. Low backlogging cost indicates that there exist other supplier alternatives technologically compatible with the supplier under consideration while not permitting backlogging implies that the buyers are obliged to him. In fact it can be treated as a measure of the technological (or market) dependence of the buyers upon the supplier.

Three scenarios related with the backlogging desirability is covered by generating low backlog cost, medium backlog cost and no backlogging situations for the test problems to be described in detail in Section 8. For the sake of illustration, an example Class A problem with M=3 and T=8 will be considered here. As a first step, the non-coordination case is analyzed for the three backlog scenarios and MOB(j) is solved for each buyer independently. That implies that the three customers buy a supply commodity from a supplier at pu=$8.00 without any supply chain cooperation. TC_j^* for each j and Π* for the supplier are determined and given in TABLE 7.1.

TABLE 7.1 Non-coordination results at pu=$8.00 and cu=$6.00.

	TC_1^*	TC_2^*	TC_3^*	Π^*
Low-Backlog Cost	13858	13889	12537	3048
Medium-Backlog Cost	14327	14768	13570	3742
No Backlogging	15857	15914	14686	4684

As the second step, a supply chain partnership is assumed where the unit production cost has been reduced from cu=$6.00 to c=$5.00. Then (IPM) is developed by using TABLE 7.1 results and different values of α_j in (7.6). Although all numerical results are not reported here for the sake of brevity, by assuming a common α for all buyers, the minimum α value at which a feasible solution can be found, the corresponding price and Π^* are reported in TABLE 7.2.

TABLE 7.2 IPM results at cu=$6.00.

	α^*	p^*	Π^*
Low-Backlog Cost	0.965	6.725	3048
Medium-Backlog Cost	0.9300	6.864	3742
No Backlogging	0.9505	6.997	4684

Observed results about the dynamics of the problem can be summarized as follows:

(a) Constraint (7.6) is binding and satisfied as strict equalities at αTC_j^*.

(b) For any given backlog situation, as α decreases, that is, as cost reduction expectations of buyers increase, the price they are willing to pay decreases and accordingly the revenue and the profit of the supplier declines.

(c) (IPM) becomes infeasible at a certain α value and turns out to be infeasible for lower values. This simply implies that it becomes impossible to satisfy the supplier's profit and the buyers' cost reduction expectations simultaneously as the buyers expect to gain more. The critical α value at which (IPM) becomes infeasible increases as the backlogging cost increases.

(d) At a given α value, on the other hand, the price increases as the backlogging cost increases. This is very intuitive since as the desirability of backlogging decreases, the buyers are forced to produce and thus to buy the supply item even at a high price in order to satisfy (7.6).

(e) Different (α-p) and (α-Π) combinations provide a full set of scenarios and it becomes very easy to determine the sensitivity of the buyer cost functions and the supplier profit to the price.

Similar observations are made on all the test problems so that the above conclusions can be generalized for (IPM). Next (CIPM) is constructed by using the results in Table 1. The compromise α_j values, price and π^* for the three backlog cases are shown in Table 3. Results indicate that the price obtained by (CIPM) is lower than that of (IPM) only for the low-backlog cost situation; in the other two cases, not only the compromise prices of (CIPM) turn out to be greater than those of (IPM), but also (CIPM) generates higher profit to the supplier. Furthermore the fact that α_j^* values are relatively high indicates that the buyers' cost reduction fractions are lower in the later two cases.

In this example. (CIPM) favours the buyers only when the back-logging cost is low. However. the complete analysis of test problems indicates that coordination through (CIPM) may be to the benefit of any party depending upon the relative magnitude of all costs in the supplier-buyer relationships. This is indeed dependent upon the technological competence of the supplier that is conveyed to the model through backlogging variables and parameters.

TABLE 7.3 CIPM results at c=$5.00.

	α_1^*	α_2^*	α_3^*	p^*	π^*
Low-Backlog Cost	0.9635	0.9755	0.9800	6.47	3048
Medium-Backlog Cost	0.9890	0.9790	0.9840	7.64	5889
No Backlogging	1.0000	0.9980	0.9990	7.98	7023

8. METAHEURISTIC FOR PRICING MODEL

Algorithms that are lacking a proven convergence to the optimum are called heuristics. Heuristics are mainly used in combinatorial optimization when the exact problem solution procedure is not known or if exists the known exact procedure computationally is very expensive.

This part of the research proposes a pricing methodology that attempts to formulate the supply chain expectations of all parties of the supply chain. A heuristic for solving the pricing problem is proposed in this section. The main motivation for developing a heuristic for the pricing problem is that there are no commercial solvers capable to solve mixed integer non-linear problems for large-scale problems as covered in this study. The proposed metaheuristic includes a hybrid approach to simulated annealing (SA) and tabu search (TS) algorithms that are covered briefly in the following sections. The aim is to combine separate advantages of SA and TS in order to reach a near global optimum solution for a non-unimodel problem like the one covered in this section.

8.1. Heuristics In Combinatorial Optimization

Lawler [46] defines combinatorial optimization is the mathematical study of finding an optimal arrangement, grouping, ordering, or selection of discrete objects. A combinatorial (or discrete) optimization problem (COP) asks to find a solution of minimizing (maximizing) an objective function over a combinatorial (or discrete) set of feasible solutions.

In most of these problems including the problems defined in this study, the optimal solution is computationally hard to obtain. Therefore it is an important issue to find approximate algorithms or so called heuristics, which can provide near optimal solutions. Nicholson [47] defines a heuristic as a procedure "... for solving problems by an approach in which the problem can be interpreted and exploited intelligently to obtain a reasonable solution".

A metaheuristic algorithm is a computer method for solving approximately, difficult, large and/or complex combinatorial optimization problems for which optimal solutions can not be found in a realistic amount of computing time using even the most powerful computer. Such an algorithm organizes and directs the search of subordinate methods such as local search, and employ different operational and organizational strategies in order to enhance their performance.

The local optima obtained by the improvement methods depend heavily on the starting feasible solution, and the search mechanisms. These local optima are often of low quality. Metastrategy innovative approaches are devised so as to avoid being trapped in poor local optima. They allow the local search method to be continued after a local minimum is detected where no further search for a global minimum can be performed. Hence, they can be viewed as enhanced versions of local search techniques. They also increase the chances of obtaining near optimal solutions when applied to new problem areas.

In general local search algorithms start with an initial solution S and a search is made to find a neighbour S′ of S. If the neighbour improves the solution S′ is assigned to S, otherwise S remains as the solution. Search and assignment criterion is repeated until a stopping criterion is satisfied. This greedy approach is not guaranteed to find a near global optimum because the best solution that can be found depends on the initial solution. FIGURE 8.1 shows how local search can be trapped in local optima depending on the initial solution.

On the other hand a metaheuristic like simulated annealing (SA) or a hybrid use of SA is more likely to find near global optimum solutions as SA can avoid local optima since non-improving moves can also be accepted with a probability as shown in FIGURE 8.1.

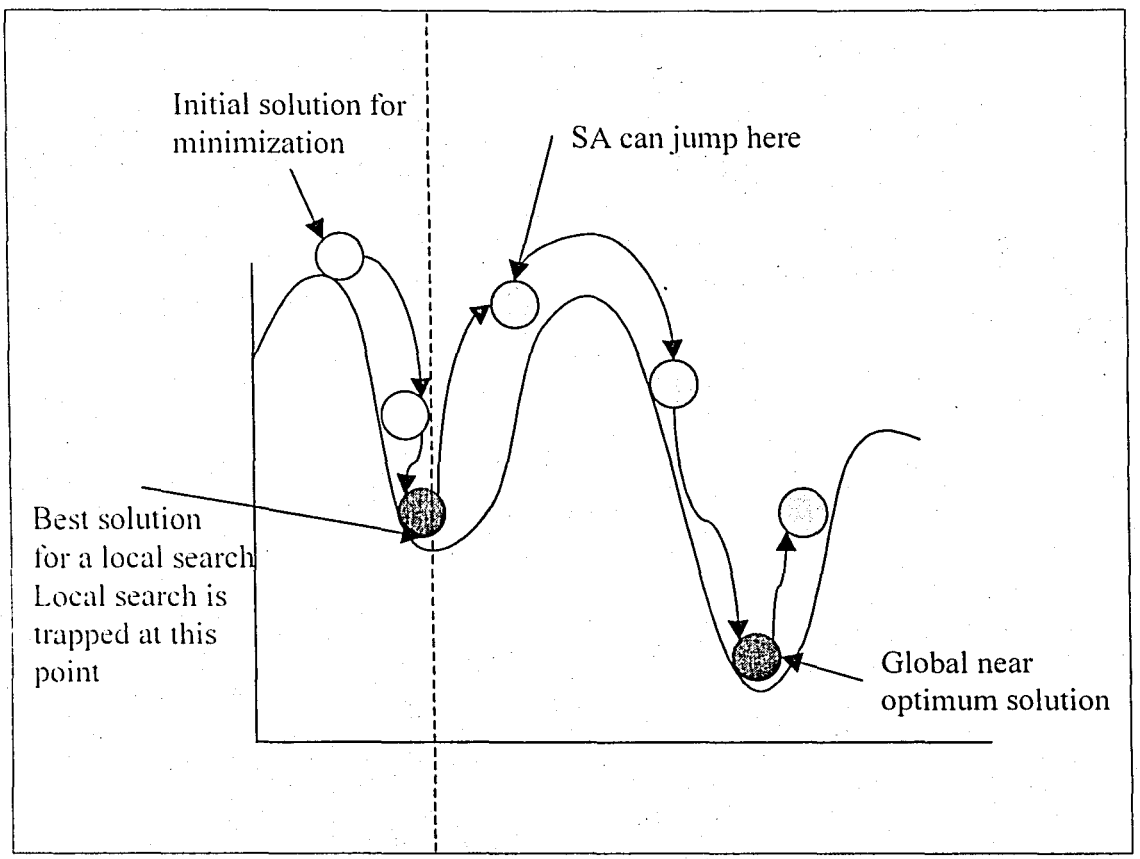


FIGURE 8.1 Local search vs. global search heuristics

8.1.1. Simulated Annealing

The simulated annealing metaheuristic was originated from analogy between the heat process of solids called annealing and the problem of solving combinatorial optimization problems. In the condensed matter physics. annealing is the process in which a solid in a heat path is melted by increasing the temperature of the heat path to a high maximum value at which all molecules of the crystal randomly arrange themselves into a liquid phase. Then the temperature of the melted crystal is decreased until the crystal structure is frozen by reaching a low ground state. The aim is to get a homogenous crystal structure: however, if the cooling is done in a quick manner, widespread irregularities and defects can be locked into the crystal structure. The formation of these irregularities is called rapid quenching which results in meta-stable structures.

In this analogy, the states of the solid corresponds to the feasible solution of a combinatorial optimization problem; the energy of the states correspond to the objective function value of the solutions; the maximum energy or ground state corresponds to an optimal solution; the rapid quenching process can be viewed as analogous to local optimization via steepest descent. When the external temperature is zero, no state transition can lead to a state of higher energy. Thus, as in local optimization, uphill moves are prohibited and the algorithm is trapped in a local minimum. When crystals are grown in practice, the bad local optima are avoided by a process of careful annealing. In this process, the temperature (T) descends slowly through a series of levels, each held long enough for the crystal melt to reach equilibrium at that temperature. As long as the temperature is not zero, uphill moves remain possible. By keeping the temperature from getting too far from the current energy level, we can hope to avoid local optima, until we are relatively close to the ground state.

In 1953 Metropolis et al. [48] propose a method the annealing process based on Monte Carlo simulation. So, the name simulated annealing refers to this simulation technique with the annealing process. The process for the simulating the evolution to thermal equilibrium works as follows: Given the current state, S, a small perturbation is applied by a small displacement of a randomly chosen molecule. If the difference in the energy level, Δ, between the current state and the newly generated state S' is negative the new perturbed state is of a lower energy and S' is accepted and the process continues from the new state. On the other side if Δ is positive, a random number $\theta \in [0, 1]$ from the uniform distribution is drawn, and if $\theta \leq e^{-\Delta/T}$ then S' is accepted; otherwise S is retained as the current state. This acceptance rule is referred to as Metropolis criterion. Thus a move which is not improving can be accepted according to a random process which is a function of the current temperature and the difference in the energy level.

It is clear that the simulated annealing is not a strictly defined algorithm but an approach to solve problems, permitting non-improving moves in order to avoid local optima. Using this generic approach, different designs of algorithms can be generated. An important issue of using this approach is making decisions on the parameters and rules of the cooling schedule such as: initial value of temperature (T), cooling rate and the

temperature update rules, the number of iterations to be performed at each temperature, and the stopping criterion.

8.1.2. Tabu Search

The tabu metaheuristic is based on putting intelligence or memory in the problem solving method. In order avoid getting trapped in local optima, the idea is to keep the history of accepted movements in a finite memory. Each time a solution is accepted the solution is kept in memory while an ex-solution is removed from the memory or history. Traces of the accepted moves are kept in memory in order to avoid accepting one of the old solutions again. Since this memory or history is a list of ex-solutions and accepting the solutions in that list is prohibited, this list is called tabu list and this procedure is called tabu search.

This search method which provides a guiding framework for search works as follows: Given an initial solution S, some neighbours of S is explored and the one with the biggest improvement and which is not in tabu list is accepted as the new solution. The new solution (S′) is assigned to the existing solution. Tabu list is updated including the newly accepted solution while one of the old members of the list is removed from the list. This iterative procedure is repeated until a stopping criterion is satisfied.

This search method is again open to customizations and it is again an approach rather than a strictly defined algorithm. In the design of TS metaheuristics decisions should be made on the some basic strategies. A tabu strategy should be made about which solutions should enter the tabu list, how long or until what event it will stay in the tabu list. Aspiration criteria that are used for accepting a solution although it is in the tabu list can be set. The main idea is not to reject a good enough solution for being in the tabu list. A short term strategy can be set for interplaying all the above mentioned forbidding, accepting and the aspiration criteria. A learning strategy can be put into the algorithm for adding more intelligence in the heuristic. This strategy gathers information during the runs and the

gathered information can be used in the next iterations of the algorithm as intermediate or long term memory. Probably a last decision is made when to finish the algorithm. This is the stopping criterion.

8.1.3. Generic Algorithms

Generic Algorithms (GAs) have been developed by Holland [49]. They are search algorithms based on the mechanics of natural selection and natural genetics in biological systems. The secrets of adoption and survival that are best learned from the careful study of rule in biological systems. These rules include features for self-repair, self-guidance, and reproduction, that are barley exist in the most sophisticated artificial systems. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. While randomized, genetic algorithms are no simple random walk. They efficiently exploit historical information too speculate on new search points with expected improved performance. The GA`s differ from normal optimization and search procedures in four different ways:

(a) GAs work with a coding of the parameter set, not the parameters themselves.
(b) GAs search from a population of points, not a single points.
(c) GAs use payoff information (objective function), not derivatives or other auxiliary knowledge.
(d) GAs use probabilistic transaction rules, not deterministic rules.

Generic algorithms have found their major applications in the optimization of functions over bounded, but otherwise unrestricted, domains.

8.2. A Hybrid Metaheuristic Algorithm SUPRIME

A hybrid metaheuristic algorithm SUPRIME standing for Supply Price Metaheuristic is developed to solve either (IPM) or (CIPM) by using both the principles of tabu search and simulated annealing approaches.

Although the SA approach can be used as a stand-alone procedure, its search capability can be enhanced when integrated with the TS memory structures known to avoid cycling between solutions and redundant transitions to non-improving solutions. The main flow of the search is to generate a price, try to improve the solution at that price until no further improvement can be expected and then try another price. The structure of the algorithm consists of initialization, solution improvement and intensification phases. Accordingly it first carries out an initialization phase by generating a number of initial solutions among which the one with the best objective function value is selected to initiate the improvement phase. The initial solution to the solution improvement phase is selected to be feasible with respect to constraints (7.7)-(7.13), but may be infeasible with respect to (7.5) and (7.6). Although the solution improvement phase aims at attaining feasibility as well as improving the objective function in a given number of moves, it incorporates a neighbourhood transition scheme based upon relaxing constraints (7.5) and (7.6) and enables state transitions that conform only to the restrictions imposed by (7.7)-(7.13) of the defined search space. The moves that lead to violations of (7.5) and (7.6) are penalized and included into the objective function. The acceptance of non-improving moves is coordinated within simulated annealing structure, and each time a different price is generated, the simulated annealing temperature is reset to its initial value. Finally the intensification phase is carried out around the best solution of the improvement phase by reannealing the temperature and not permitting infeasible neighbourhood transition. Information gained during the improvement phase is communicated to the price generating procedure of the intensification phase. Accepting only the feasible moves over a restricted solution space will naturally accelerate the convergence to a better solution.

One important aspect of the algorithm is to include a short term memory structure that helps the algorithm learn the nonimproving regions and a long term memory structure

that measures the likelihood of improving the objective function at different temperatures and that abandons the search around a given price if considerable improvement can not be expected from further exploration. The short term memories are limited to each phase while the long-term memory is expanded to the whole search. The general structure is outlined in FIGURE 8.2. In order to provide a complete characterization of **SUPRIME**, it is necessary to describe the generation of the initial solution, the cooling scheme, memory structures, the tabu listing and the perturbance algorithm necessary to generate the neighbourhood transitions. The pseudo code of the algorith is given in Appendix D and the code itself in C++ is given in Appendix E.

8.2.1. Generation of an Initial Solution

A number (M1) of initial solutions are constructed by generating different prices in the interval given by the supplier and assigning a random purchase order to each buyer that is upper-bounded by Q_{jt} without violating (7.10) and a random lot size to every item produced by each buyer in each period. Here the random lot size in period t is determined as the sum of a random number of consecutive periods' demand requirements following t. The inventory level of the supply item is then calculated by (7.7) using these random quantities. If inventory constraints (7.7) are not satisfied, then first DS_{jt} of the first period in which infeasibility occurs is tried to be increased without violating (7.10) and (7.12). If feasibility is not attained yet, the production quantities X_{ijt} are decreased until $IS_{jt} \geq 0$. Then all required quantities are computed by (7.8), (7.11) and (7.12). Then the initial solution with the best objective function value is chosen to initiate the main improvement search. It is important to note that a solution that is infeasible with respect to (7.5) and (7.6) can be accepted as an initial solution for the solution improvement phase.

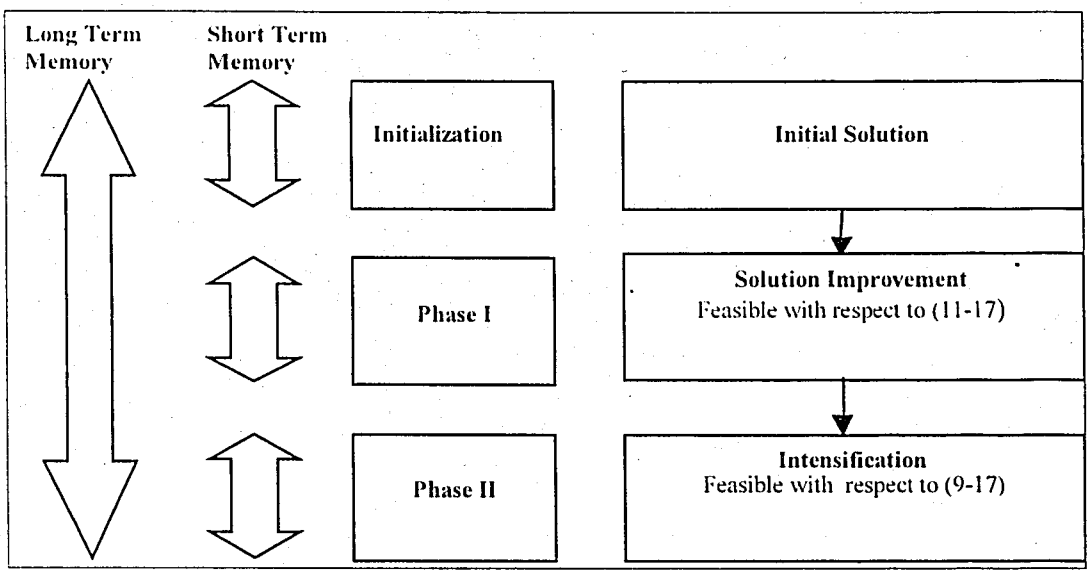


FIGURE 8.2 Outline of **SUPRIME**

8.2.2. Generation of a Neighbour Solution

A move is generated as the result of perturbing the purchase orders and production quantities in a given solution. The perturbation consists of either increasing or decreasing some DS_{jt} depending on the feasibility of the current solution with respect to (7.5) and (7.6). A neighbouring solution is one where exactly one DS_{jt} is modified and the related production quantities X_{ijt} are updated accordingly. The main idea in a move is to determine a buyer/period pair for which a possible purchase quantity can be increased or decreased in order to improve the objective function. The modification of the purchase/sales decisions will naturally affect either some production quantities X_{ijt} or the inventory level of the supply item IS_{jt} or both. The state of the variables in the current solution and the costs will determine how this change will be propagated to X_{ijt} and IS_{jt}. Given any DS_{jt}, one may either consume up all the available supply item to manufacture the final products so that X_{ijt} and inventory levels will increase or the supply item is not processed, but kept in its raw form in stock and IS_{jt} will increase. Here the cost of final product shortage and

inventory holding costs act against the inventory holding cost of the supply item. The best decision will be determined by considering all these cost items. The detailed description of the procedure to generate a neighbour is provided in Figure D.1. Since the move is constructed according to the feasibility state of the current solution, possible cases of feasibility are considered separately:

(A) If the solution is feasible with respect to (7.5), but infeasible with respect to (7.6): First, the buyer/period pair (j*,t*) is selected randomly, so that the purchase quantity of buyer j* will be modified in period t*. Since the modification can be either of "increase" or "decrease" type, the procedure **(neighbour)** is executed as it is.

(B) If the solution is feasible with respect to (7.6), but infeasible with respect to (7.6): Here feasibility and thus objective function improvement can only be achieved by the "increase" option: thus only the "increase" part of the procedure **(neighbour)** is executed.

(C) If the solution is feasible with respect to both (7.5) and (7.6): This case is similar to case (A) where both "increase" and "decrease" options are possible.

(D) If the solution is infeasible with respect to both (7.5) and (7.6): This case is similar to case (A) where both "increase" and "decrease" options are possible.

8.2.3. Neighbourhood Transition Schemes

Any SA algorithm should specify the mechanisms for incorporating feasibility constraints while performing the solution transitions. In fact there are two main techniques that can be used to incorporate the constraint relaxation. In the first approach only feasible solutions are considered, and the search space is restricted to the feasible space and hence only feasible moves are implemented by the algorithm that directly restricts the move definition according to the current state of slack in the constraints. This naturally requires the development of a heuristic to restore the feasibility of only the starting solution and

naturally reduces the computational burden of restoring feasibility between any two moves, which is generally the situation in previous studies employing this approach. In the second approach infeasible moves are permitted with a resulting penalty. All (or some) constraints of the problem can be relaxed and incorporated into the objective function by using a penalty function. Thus the search can proceed through the infeasible region hoping that a feasible solution with a better objective function value can be found. The fact that any point in the search space can be accepted provides great flexibility. Despite all its advantages, this flexibility may cause the search to wander around a large space and to converge even to an infeasible local minimum. In this model since the constraints (7.5) and (7.6) are critical, the relaxation of these are accepted at different points of the algorithm. In the solution improvement phase, solutions that are infeasible with respect to one or both are accepted. Permitting the execution of the moves that are infeasible with respect to (7.5) and (7.6) diversifies the search, improves the solution quality due to amplified search effort, and still converges to a feasible good solution. Consequently as DS_{j^*,t^*} is being modified, the feasibility of (7.5) and (7.8) is not taken into account and no restriction is imposed upon the increase and decrease quantities. This implies that the search is permitted to move to an infeasible neighbouring solution even from a feasible one. Although one may suspect that this might lead to a nonconverging dispersion, the correct choice of the penalty terms for infeasibility and the high number of computational runs guarantees convergence to a feasible and a better solution. The intensification phase, on the other hand, is always initiated from the best feasible solution found in the improvement phase and does not permit transitions to infeasible solutions once a feasible solution is found.

8.2.4. Tabu Listing

A tabu restriction is applied in order to prevent cycling in the search process. Here a move is considered for execution only if it is not found in the tabu list which stores the objective function value, price and purchase quantities associated with a given number of recent moves. The tabu list is updated with each executed move by adding the most recent

move to the end of the list and deleting the oldest one from the list. The size of the tabu list is specified according to the problem dimensionality in the algorithms. If a pre-specified number (N1) of consecutive moves all result in tabu moves, the oldest move in the tabu list is selected to continue the search.

8.2.5. The Cooling Scheme

It is well known that the choice of the cooling schedule is a critical factor for a successful implementation of the SA approach. A cooling scheme includes the starting temperature, the cooling function according to which the temperature will be reduced and the Markovian probabilities in accepting a non-improving move. Once a neighbour is generated, it is important to determine whether it improves the cost or not. Whenever the move leads to an improvement over the current cost, it is immediately implemented. However, the objective function difference Δ between two consecutive moves may be negative and a move may result in a profit lower than the incumbent one. When a pre-specified number (N2) of consecutive moves does not result in improvement, the possibility of transition to a non-improving neighbour is considered. Then the acceptance of non-improving moves is randomized by assigning a probability of acceptance (PA) calculated by

$$PA(\Delta, tSA) = \exp(\Delta / tSA) \qquad (8.1)$$

where Δ=profit of the new move- profit of the incumbent solution. After each non-improving move, the temperature tSA is reduced by $tSA \leftarrow tSA * \beta$ where β is the annealing parameter for which different values are tested in preliminary experimentation and the best results are obtained for β=0.85 in the first phase and β = 0.95 in the second phase. In the solution improvement phase, the initial value for the temperature is determined by $0.05(\sum_{j=1}^{M} TC_j + TP)M$ as a measure of the worst-case objective function value.

In the intensification phase, on the other hand, reannealing is done by initializing the temperature from 10% of the best objective function value obtained in the improvement phase. One way to deal with different changing sensitivities of the temperature parameters in the search is to re-scale the annealing parameter either periodically or at some pre-established events. In this study reannealing is performed by setting the temperature to the initial value of the undergoing phase whenever a new price is generated. Furthermore, if tSA falls below a very small value in any part of the algorithm, then it is reset to this initial value, so that PA does not become infinitesimally small after a large number of moves. A maximum number (N3) of non-improving transitions are permitted with a given price. Then a new price is generated to continue the search. The algorithm is executed with N4 different prices. The general outline of **SUPRIME** is given in Figure D.2.

8.2.6. Short Term Memory Structure

In order to enhance the effectiveness of the search, it is necessary to influence the process to head in new directions and to diversify the search whenever further improvements can not be expected in the selected direction. This is accomplished by affecting the choice of the modification type and the purchase quantity variable to be modified in the move construction. The frequency of selecting variables and modification types are recorded in the form of short term memory listing. If a certain modification type is selected repeatedly for a given number of consecutive iterations without bringing along any improvement, the other modification is directly implemented as a diversification mechanism. Furthermore if a certain DS_{jt} is selected repeatedly for a given number of iterations without considerable improvement, that variable is not considered for modification; instead the variable with the lowest frequency at that iteration is selected for further exploration.

8.2.7. Long Term Memory Structure

Here it is intended to construct a memory mechanism to estimate the distribution of cost improvements at checkpoint temperatures given by {TC(k): TC(k) ≤TC(k+1), k=1....r}. This idea is first introduced in Sadeh et al. When a new price is generated, the annealing is re-scaled from the initial temperature, and at each TC(k) it is determined if continuing to the next temperature checkpoint TC(k+1) is likely to improve the current best solution at the given price considerably. If the result is negative, a cut-off condition is said to hold true, the run with the current price is abandoned and a new price is generated. This condition does not override N3 limitation; that is, if the cut-off condition does not hold true for N3 non-improving transitions, the current price is still abandoned and the search is resumed with a new price. Let p(s) be the price studied in price-run s and {p(r): r=1....s}be the ordered array of prices generated up to the current run. Then

R_r^0 = profit at tSA=TC(0) with price p(r) (8.2)

R_r^k = profit at tSA=TC(k) with price p(r) (8.3)

R_s^k = profit at tSA=TC(k) with price p(s) (8.4)

R* = profit of the best solution found in the revious runs (8.5)

Then the expected profit improvement below TC(k) denoted as EPI(k) is expected to have the average and the standard deviation given by

$$\mu_s^k = \sum_{r=1}^{s-1}(R_r^k - R_r^0)/s-1 \tag{8.6}$$

$$\sigma_s^k = \left[\sum_{r=1}^{s-1}\left\{(R_r^k - R_r^0) - \mu_s^k\right\}^2/s-2\right]^{1/2} \tag{8.7}$$

Thus the cut-off criterion at temperature TC(k) can be stated as:

$$\left[(R_s^k - \mu_s^k) - R^*\right]/\sigma_s^k \geq CC \tag{8.8}$$

where CC is a threshold value . The threshold value used in the experimentation is chosen to be 3.0 in accordance with Sadeh et al.

8.3. Experimental Design

Since the difficulty to find an optimal solution, consequently the performance of the algorithm, clearly depends on the dimensionality of the problem and the inclusion of binary variables, the metaheuristic algorithm **SUPRIME** described in this paper was applied to 384 problems which can be categorized into 4 data sets:

(a) Problem Class A: (M=3; T=4,8,12; Fixed cost excluded)

(b) Problem Class B: (M=3; T=4,8,12; Fixed cost included)

(c) Problem Class C: (M=10; T=4,8,12; Fixed cost excluded)

(d) Problem Class D: (M=10; T=4,8,12; Fixed cost included)

It is important to note that in any solution to (IPM) and (CIPM) price works against holding cost of the supply item and backordering costs of end items while backordering and inventory holding costs act against production and setup costs. Thus, it is crucial to analyze the interrelated effect of these cost parameters by varying their relative magnitudes. Since the tightness of the feasible space affects the performance, different scenarios on related attributes are designed and possible combinations are generated to test the overall integrated effect. The manner in which problem parameters are generated is described below:

(A) Two demand scenarios are developed: The first one assumes a random linear trend while the second one includes seasonality upon the linear base generated normal distribution with a mean of 100 units and a standard deviation of 10 (that is, N(100,10)). In the second scenario, demand is assumed to possess a seasonal trend with the base level multiplied with a seasonality factor. The seasonality factor is assumed to depend upon the cycle time of a sine wave and generated from uniform distribution over 1.0 and 1.5 (that is, U(0.5,1.5)for both base level cases.

(B) The lower and upper limits for the price are randomly generated from U(2,100) and U(5pl,50pl), respectively.

(C) Production costs are generated from two different Uniform distributions: the first distribution is U(2pl,2pu) which represents a high material cost contribution to the overall cost of the end items in a material-intensive system and the second one is U(10pl,10pu) which implies a lower material cost burden.

(D) The holding cost of the end items are defined by $h_{ijt} = 0.25.c_{ijt}$

(E) Two different cases are considered for the shortage cost of the end items: the first situation corresponds to a high shortage penalty where $\pi_{ijt} = 2c_{ijt}$ and encourages the buyers to buy the supply item even when the price is relatively high. The other case is defined by $\pi_{ijt} = 0.50c_{ijt}$ which might lead to shortages if the price of the supply item is high enough.

(F) In classes A and C, fixed costs are not included in order to avoid binary variables whereas in classes B and D fixed costs are defined by $s_{ijt} = \gamma . c_{ijt}$ and $f_{jt} = \gamma . pu$ where γ is generated from N(100, 10).

(G) The holding cost of the supply item is generated from two different distributions: U(1.25pl, 1.25pu) and N((pl+pu)/2, 0.10pl).

(H) Two different capacity levels are assumed where capacity is found by $CAP_{ijt} = \rho(\sum_{j=1}^{N} d_{ijt} / T)$ for each buyer. In a "tight" capacity situation ρ is taken as 1.2 and in a "loose" capacity situation ρ is taken as 1.6.

(I) In defining Q_{jt}, CAP_t is equally apportioned among the buyers in all cases.

Considering the five attributes mentioned above (d_{ijt}, c_{ijt}, π_{ijt}, hm_{ijt},CAP_{ijt}) there arise (2×2×2×2×2=32) factor combinations of manufacturing scenarios to be tested for each problem, and the four classes of test problems mentioned above are generated according to this factorial design. Class A and B problems are replicated 3 times while one replication is done for class C and D problems. The number of initial solutions (M1) considered before the solution improvement phase is 20 for Class A and B problems and 10 for Class C and D. The values of the other parameters used in the experimentation are given in TABLE 8.1.

TABLE 8.1 Parameters used in the experiments

	N1	N2	N3	N4
Solution Improvement Phase	30	30	2000	50
Intensification Phase	10	20	1000	20

In order to be able to compare the performances of the solution procedures against a commercial optimizer, the test problems without binary variables are also solved by using GAMS XA-Solver with a maximum limit of 1,000,000 iterations. The algorithm is coded in C++ and the computation times are observed on a Pentium-150 32MB Ram computer. The numerical results for Class A and C problems are reported in TABLE 8.2 where AVG. STD. and MAX denote the average percentage deviation between the best solutions obtained by **SUPRIME** and GAMS, the percentage standard deviation and the maximum percentage deviations, respectively.

TABLE 8.2 Results for Class A and C problems

	AVG	DEV	MAX	CPU
Class A/ T=4/ Low π	0.000	0.000	0.000	8.879
Class A/ T=4/ High π	0.053	0.041	0.113	11.748
Class A/ T=8/ Low π	0.006	0.007	0.019	23.549
Class A/ T=8/ High π	0.114	0.046	0.269	65.787
Class A/ T=12/ Low π	0.058	0.034	0.096	16.651
Class A/ T=12/ High π	0.132	0.047	0.253	81.019
Class C/ T=4/ Low π	0.000	0.000	0.000	18.801
Class C/ T=4/ High π	0.054	0.051	0.164	20.089
Class C/ T=8/ Low π	0.004	0.002	0.007	60.769
Class C/ T=8/ High π	0.153	0.089	0.257	382.928
Class C/ T=12/ Low π	0.052	0.020	0.085	93.212
Class C/ T=12/ High π	0.173	0.066	0.200	510.180

The results in TABLE 8.2 reveal that the performance of **SUPRIME** is compatible with a well-known commercial optimizer namely GAMS for some problems while the performance declines with problem dimensionality and feasible region tightness. Naturally as the problem gets bigger, the quality of the solutions declines and the computational time increases. For any problem size, as the backlogging cost is increased, the feasible region becomes very tight and it becomes relatively difficult to find a good solution. Another

parameter that impairs the performance is the demand seasonality; when demand shows a seasonal pattern. poor solutions are obtained. The two conflicting cost parameters are basically the inventory holding cost and the backlogging cost of the end items. Their relative magnitudes determine the difficulty of satisfying (7.5) and (7.6). When the inventory holding and the backlogging costs are comparatively low, it is easiest to satisfy both (7.5) and (7.6) by increasing the price and the purchase orders. When the inventory holding cost is low, but the backlogging cost is high, (7.5) and (7.6) are satisfied by high purchase orders at lower prices. When the inventory holding cost is high, but the backlogging cost is low, (7.5) and (7.6) are satisfied by lower purchase orders at high prices. The most difficult situation to satisfy (7.5) and (7.6) is when both costs are high.

The main contribution of the algorithm arises when the problem under consideration includes binary variables. To the knowledge of the authors, there are no commercial software capable to solve mixed integer non-linear problems for large scale problems as covered in this study. Since Class B and D problems are mixed integer nonlinear models, GAMS could not be used at all; thus only the results of **SUPRIME** are displayed in Table 3. In addition to the previously-made observations, the results in TABLE 8.3 show that the computational time does not increase with the inclusion of binary variables. For some problems the intensification phase does not change the solution obtained in the improvement phase while it brings along considerable improvement above the solution of the improvement phase in some other problems. In fact there are certain problems for which the first phase can not even obtain a feasible solution and feasibility can only be obtained by the intensification phase. This further emphasizes the fact that neighbourhood transition strategies and consequently the constraint relaxation schemes play significant roles in the search capabilities of metaheuristics. It is conjectured that the same performance and disparity around the optimum will hold true for these problems as well. Solutions of all problems are given in Appendix F where GAMS solutions for groups A and C are also compared.

TABLE 8.3 Solutions for Class B and D problems

Class D	Problem		Imp(OF)	Int(OF)	CPU	Class B	Problem		Imp(OF)	Int(OF)	CPU
1 101	11	1 n	44512	44512	47.62	1 031	11	1 N	13182	13182	19.21
1 101	11	1 s	41952	41952	41.40	1 031	11	1 S	11040	11040	21.05
1 101	11	2 n	44512	44512	40.39	1 031	11	2 N	13155	13177	18.67
1 101	11	2 s	41952	41952	38.64	1 031	11	2 S	11017	11037	19.36
1 101	12	1 n	44512	44512	49.62	1 031	12	1 N	13184	13184	20.54
1 101	12	1 s	41637	41923	53.02	1 031	12	1 S	10994	11016	22.26
1 101	12	2 n	44512	44512	46.12	1 031	12	2 N	13184	13184	21.13
1 101	12	2 s	41952	41952	45.63	1 031	12	2 S	11040	11040	21.54
1 101	21	1 n	25153	26749	73.51	1 031	21	1 N	7982	8978	3.21
1 101	21	1 s	31313	32393	25.23	1 031	21	1 S	8246	9025	6.31
1 101	21	2 n	29530	30320	28.40	1 031	21	2 N	8255	8310	3.40
1 101	21	2 s	30399	32631	47.14	1 031	21	2 S	8438	8438	5.76
1 101	22	1 n	-3373710	10807	115.59	1 031	22	1 N	10552	10553	2.58
1 101	22	1 s	-441717	14905	125.42	1 031	22	1 S	5589	5977	4.42
1 101	22	2 n	-3778910	13380	86.75	1 031	22	2 N	10205	10441	2.63
1 101	22	2 s	-85513	13522	98.38	1 031	22	2 S	7018	7018	2.12
1 102	11	1 n	91288	91456	57.88	1 032	11	1 N	26176	26176	19.44
1 102	11	1 s	94288	94288	54.55	1 032	11	1 S	27840	27840	14.42
1 102	11	2 n	91392	91392	64.66	1 032	11	2 N	26176	26176	22.81
1 102	11	2 s	94368	94368	59.23	1 032	11	2 S	27840	27840	29.60
1 102	12	1 n	91000	91000	60.22	1 032	12	1 N	25962	25977	23.34
1 102	12	1 s	94152	94152	55.41	1 032	12	1 S	27248	27248	17.02
1 102	12	2 n	91079	91079	59.14	1 032	12	2 N	26032	26032	25.73
1 102	12	2 s	94248	94280	56.45	1 032	12	2 S	27312	27312	18.68
1 102	21	1 n	39829	40331	241.86	1 032	21	1 N	21888	21954	9.84
1 102	21	1 s	45673	46113	248.91	1 032	21	1 S	15186	16286	9.63
1 102	21	2 n	48795	49160	250.86	1 032	21	2 N	23455	23478	8.72
1 102	21	2 s	51432	51432	291.86	1 032	21	2 S	15592	15628	8.59
1 102	22	1 n	88511	88799	248.75	1 032	22	1 N	20852	20994	6.54
1 102	22	1 s	91712	91728	256.84	1 032	22	1 S	20616	20623	5.21
1 102	22	2 n	88932	88948	256.95	1 032	22	2 N	20635	21132	6.10
1 102	22	2 s	91512	91534	254.44	1 032	22	2 S	20534	20804	5.25
1 103	11	1 n	92057	92086	104.83	1 033	11	1 N	29613	29778	18.67
1 103	11	1 s	117023	117046	105.50	1 033	11	1 S	32777	32777	21.04
1 103	11	2 n	84534	86690	106.77	1 033	11	2 N	30678	30678	20.82
1 103	11	2 s	118373	119561	105.26	1 033	11	2 S	33365	33365	19.39
1 103	12	1 n	83072	83357	89.71	1 033	12	1 N	29606	30231	16.58
1 103	12	1 s	116914	118208	96.97	1 033	12	1 S	32584	32584	16.67
1 103	12	2 n	82213	83522	95.70	1 033	12	2 N	29944	30039	13.88
1 103	12	2 s	117976	118274	94.45	1 033	12	2 S	32640	32728	16.30
1 103	21	1 n	36290	36290	35.85	1 033	21	1 N	16077	16077	4.95

TABLE 8.3 Solutions for Class B and D problems, continued

1	103	21	1 s	70292	70542	51.75	1	033	21	1 S	21698	21803	13.63
1	103	21	2 n	40688	41127	36.88	1	033	21	2 N	12748	12748	4.45
1	103	21	2 S	64255	64494	57.97	1	033	21	2 S	22392	22392	13.32
1	103	22	1 N	68749	68749	46.99	1	033	22	1 N	12849	12849	3.82
1	103	22	1 S	76437	76564	58.01	1	033	22	1 S	27879	28745	9.76
1	103	22	2 N	61009	61195	42.55	1	033	22	2 N	10232	10232	3.73
1	103	22	2 S	68502	68734	59.42	1	033	22	2 S	27830	28544	10.64

Legend: the first digit indicates the existence of binary variables. The next two digits indicate the number of buyers. The fourth digit indicates the number of periods (1: T=4; 2: T=8; 3: T=12). The next digit show the production cost (1: c=low; 2: c=high). The sixth digit indicates the shortage cost (1:π =low; 2: π=high). The next digit indicates the holding cost of the supply item (1: hm=high; 2: hm=low). The last digit shows demand (n: d=random; s=seasonal)

9. HIERARCHICAL PLANNING

9.1. The HOP Framework

A common way to handle complex organizations is to decompose it into more manageable subsystems. These subsystems are not often of equal ranking but show a typical hierarchical interdependence. In other words one of the components of the whole organizations has either more power on the other one or one of the subsystem make earlier decisions that have influence on the other one.

Hierarchical planning includes two basic issues: constructing of the hierarchy; that is to impose a hierarchical structure on a non-structured system, and planning within a hierarchy which is the development of planning procedures within an organization that is already hierarchically structured. Schneeweiss [50] identifies the first type planning with a hierarchical structuring and modeling task, whereas the second type is concerned with planning and leadership activities within and between the management levels of an organization.

Structural hierarchies in contrast to organizational hierarchies are of a more permanent nature. Structural hierarchies can be real structure or formal structure hierarchies and organizational structures exit due to decision time or leadership hierarchies. The characteristics of structural hierarchies are the symmetric status of information and only one decision-maker agent that uses the information available at a specific point of time.

A type of structural hierarchy, real structure hierarchy, results from a separation a real world problem into smaller hierarchically dependent parts. All decisions are made by a decision-maker at the same time. The second type of structural hierarchies is formal structure hierarchy. In contrast to the real structure hierarchy it is not aimed at better understanding or handling a system but at improving its formal mathematical tractability.

They do not represent real world objects and are pure intellectual to better manipulate planning tasks.

On the other hand, organizational hierarchies may predominantly be characterized by the asymmetry of their information status. This asymmetry arises from to reasons: the decision-maker has different information at different levels (decision hierarchy) or, secondly, each level has another decision-maker possessing his own distinct information (leadership hierarchy).

Decision time hierarchy of the organizational hierarchies consists of levels that make decisions at different points of time and which consequently are, in general, based on different information.

Finally, leadership hierarchy of the organizational hierarchies describes the relationship between leader and follower. Here information asymmetry is not caused by the different point of time at which a decision has to be made but by the existence of different decision-makers that might possess different information and, in addition, might follow different objectives.

Schneeweiss [50] formalizes all these concepts within the HOP Framework. Another aspect of this study is to identify the HOP structure in the Supply Chain approach proposed in this study.

9.1.1. Interdepences in Hierarchical Systems

In the hierarchical decomposition of complex systems three major interdependence stages between the subsystems may occur that can be observed in FIGURE 9.1.

In a first step, in finding a feasible decision, the top-level takes into account the relevant characteristics of the base level. These characteristics are called "anticipated base level". Often an anticipated base level is just an aggregated base level or some particularly important aspect that has to be considered. Choosing an anticipated base level and taking

into account its impact on top decision is called an anticipation. Generally an anticipation can be regarded as a bottom up influence of the base level on the top level.

Having anticipated the base level, the top level makes a decision that influences the base level that is called instruction (IN). Generally instruction can be regarded as a top down influence of the top level on the base level.

If the base level is in position to react on the top level's instruction this influence is called a reaction (RE). A reaction is representing a feedback influence whereas the anticipation can be considered as feedforward influence.

In the frame work of this approach, when a reaction is available, a communication or even a negotiation process is also applicable. Finally after a series of instruction-reaction cycles, top and base level agree on a decision that results in an implementation. This implementation influences the object system, that is the system top and base level are supposed to control. All decisions prior to the implementation will be called ex-ante and those after this final decision ex post.

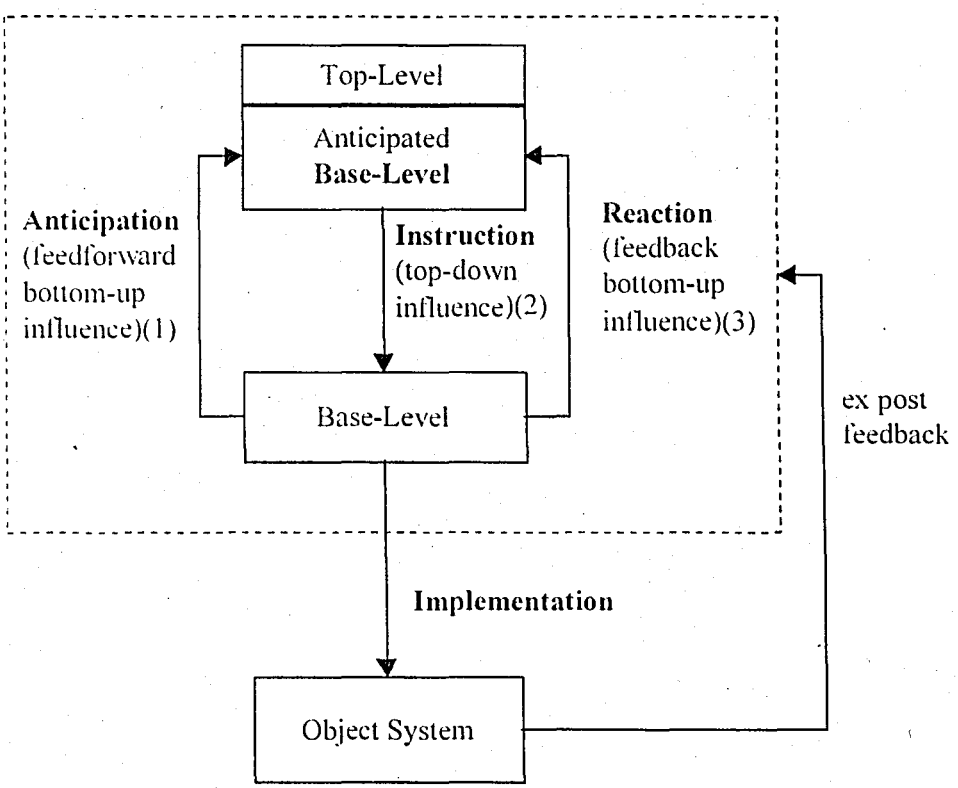


FIGURE 9.1 Interdependence of hierarchical levels

9.1.2. General Characteristics of Hierarchical Planning Structures

FIGURE 9.2 [50] shows a hierarchical system with different kinds of subsystems having particular interrelations and outputs. In order to derive an optimal instruction, the top level anticipates the base level and hypothetically applies different instructions IN as 'stimuli' to the anticipated base level. These instructions give rise to possible reactions and finally result in an optimal instruction IN = IN* which is definitely communicated to the base level. The base level then takes this instruction into account, derives an optimal reaction RE* and passes to the top level. After some negotiation cycles, the hierarchical system finally results in a decision IN**.

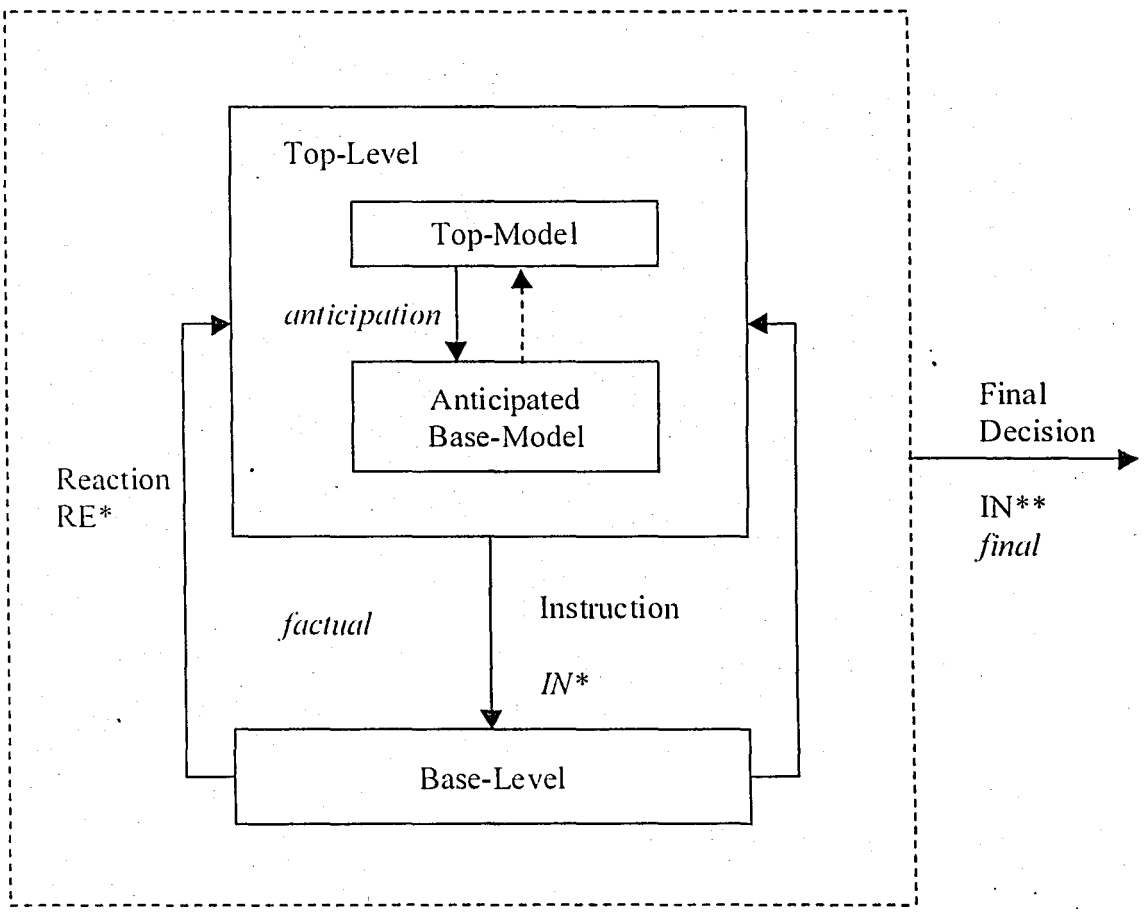


FIGURE 9.2 General hierarchical planning structure

In FIGURE 9.3 a more formal description is given. Each level is described by a decision model M depending on a decision field (or action space) A and a preference structure (criterion) C. Hence one has $M^T = M^T(C^T, A^T)$ for a top-level, and $M^B = M^B(C^B, A^B)$ for the base-level. The anticipated base level is described by the decision model $\hat{M}^B = \hat{M}(\hat{C}^B, \hat{A}^B)$ where the hat will always be taken to denote estimates. The variable $a^T \in A^T$ denotes the decision of the top-level with a^{T*} being an optimal value. The decision a^T implies an instruction $IN = IN(a^T)$.

The anticipation function AF(IN) describes the possible reaction of the base-level with respect to an instruction IN. It is only through AF(IN) that the base-level is taken into account by the top level.

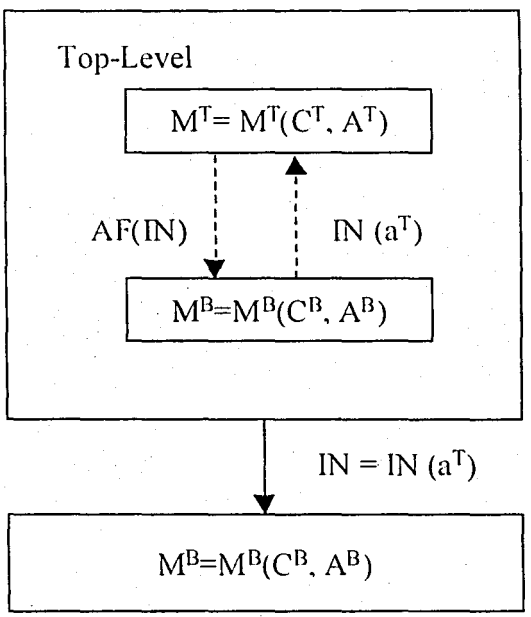


FIGURE 9.3 Coupling equations

9.2. HOP Mapping

The complete coordination process explained in the previous sections via mathematical models can be described by a hierarchical planning system between the supplier and the buyers. Since the supplier is assumed to have a leading role in the design

of coordination presented in this treatise, from the hierarchical planning point of view, the supplier can be placed in the TOP level and the buyers are in the BASE level within an organizational hierarchy (leadership hierarchy) with explicit anticipation.

Furthermore, in the supplier's top-level decision process there exists a formal structure hierarchy with a pure top-down (non-specific anticipation) attribute imbedded within the decision process itself. Although all supply chain decisions are very much interrelated to be described within the framework of a single model, due to mathematical tractability, a formal hierarchical planning approach is proposed where each of these issues is treated in a sequential manner. The mapping results of hierarchical structures in this study are shown in FIGURE 9.4, FIGURE 9.5 and
FIGURE 9.6. and the hierarchical structures can be summarized as given in FIGURE 9.7.

Type of Anticipation: Non-specific anticipation

Top level(CAS):

$$a^{T*} = \arg\min \sum_{t=1}^{24} \left\{ \sum_{j=1}^{K} \left(ss_{jt}\alpha_{jt} + sc_{jt}CAP_{jt} \right) + a_t\Delta_t^+ + b_t\Delta_t^- + hs_tSI_t^+ + \pi_tSI_t^- \right\}$$

$$a^T = \left\{ CAP_{jt}, \alpha_{jt}, \Delta_t^+ \Delta_t^- SI_t^+, SI_t^-, SX_t \right\}$$

$$C^T = C^{TT} = \sum_{1}^{24} \left\{ \sum_{j=1}^{K} \left(ss_{jt}\alpha_{jt} + sc_{jt}CAP_{jt} \right) + a_t\Delta_t^+ + b_t\Delta_t^- + hs_tSI_t^+ + \pi_tSI_t^- \right\}$$

A^T defined by (5.1)-(5.10)

Feed-forward Influence (No anticipation function): $A^T = A^T(\hat{ES}_t)$

Instruction: $IN_t^* = IN_t(a^{T*}) = CAP_t^*$ t =1.....24

Base level (SAP): $a^{B*} = \arg\min \sum_{j=1}^{M} w_j (Q_t^j - RS_{jt})^2$ $a^B = \left\{ Q_t^j \right\}$

$C^B = \sum_{j=1}^{M} w_j (Q_t^j - RS_{jt})^2$ $A_{IN^*}^B = A_{CAP}^B$ defined by (6.3)-(6.4)

Instruction CAP_t^* is entering the base level through (6.3)

FIGURE 9.4 Capacity Hierarchy (CAS-AHP-SAP)

Type of Anticipation: Non-specific anticipation

Top level(SAP):

$$a^{I} = \arg\min \sum_{j=1}^{M} w_j (Q_t^j - RS_{jt})^2$$

$$a^{T} = \{Q_t^j\}$$

$$C^{T} = C^{TT} = \sum_{j=1}^{M} w_j (Q_t^j - RS_{jt})^2$$

A^{T} defined by (6.3)-(6.4)

Feed-forward Influence(No anticipation function): $A^{T} = A^{T}(CAP_t)$

Instruction: $IN_t^* = IN_t(a^{T*}) = Q_t^j \qquad t = 13,.....,24$

Base level(IPM or CIPM):

$$a^{B*} = \arg\max \sum_{t=13}^{24} \sum_{j=1}^{M} p_t DS_{jt}$$

$$a^{B} = \{p_t . DS_{jt}\}$$

$$C^{B} = \sum_{t=13}^{24} \sum_{j=1}^{M} p_t DS_{jt}$$

$A_{IN^*}^{B} = A_{x}^{B}$ defined by (7.5) to (7-12)

Instruction $\hat{x}_{tj} \equiv Q_t^j$ is entering the base level through (7.11)

FIGURE 9.5 Supply quota hierarchy

Type of Anticipation: Explicit anticipation

Top level(DIS):

$$a^{T*} = \arg\max \sum_{13}^{24} \sum_{j=1}^{M} p_t DS_{jt}$$

$$a^{T} = \{p_t, DS_{jt}\}$$

$$C^{T} = C^{TT} = \sum_{t=13}^{24} \sum_{j=1}^{M} p_t DS_{jt}$$

A^{T} defined by (7.5)-(7.12)

Feed-forward Influence(Explicit anticipation function):

$$AF(IN) = a^{B''}(IN)$$

$$a^{B''} = \arg\min \sum_{t=13}^{24} \sum_{i=1}^{N} \left(s_{ijt} \cdot Y_{ijt} + c_{ijt} \cdot X_{ijt} + h_{ijt} \cdot I_{ijt}^{+} + \pi_{ijt} I_{ijt}^{-}\right) + \sum_{t=13}^{24} \left(p_t DS_{jt} + hm_{jt} IS_{jt}\right)$$

$\hat{A}_{IN}^{B}$ defined by (4.2)-(4.6).

Instruction: $IN_t^* = IN_t(a^{T*}) = \{p_t\}$

Base level(MOB):

$$a^{B*} = \arg\min \sum_{=} \sum_{=} \left(s \ .Y \ + c \ .X \ + h \ .I^{+} + \pi \ I^{-}\right) + \sum_{=} \left(p \ DS \ + hm \ IS \ \right)$$

$$a^{B} = \{Y_{it}, X_{it}, I_{it}^{+}, I_{it}^{-}, S_{jt}, IS_t\}$$

$$C^{B} = \sum_{=} \sum_{=} \left(s \ .Y \ + c \ .X \ + h \ .I^{+} + \pi \ I^{-}\right) + \sum_{=} \left(p \ DS \ + hm \ IS \ \right)$$

$A_{IN^*}^{B} = A_{p}^{B}$ defined by (4.2)-(4.6)

FIGURE 9.6 Price discounting hierarchy

The above analysis shows that a one-to-one mapping can be defined between the HOP framework and the hierarchical planning described in this paper.

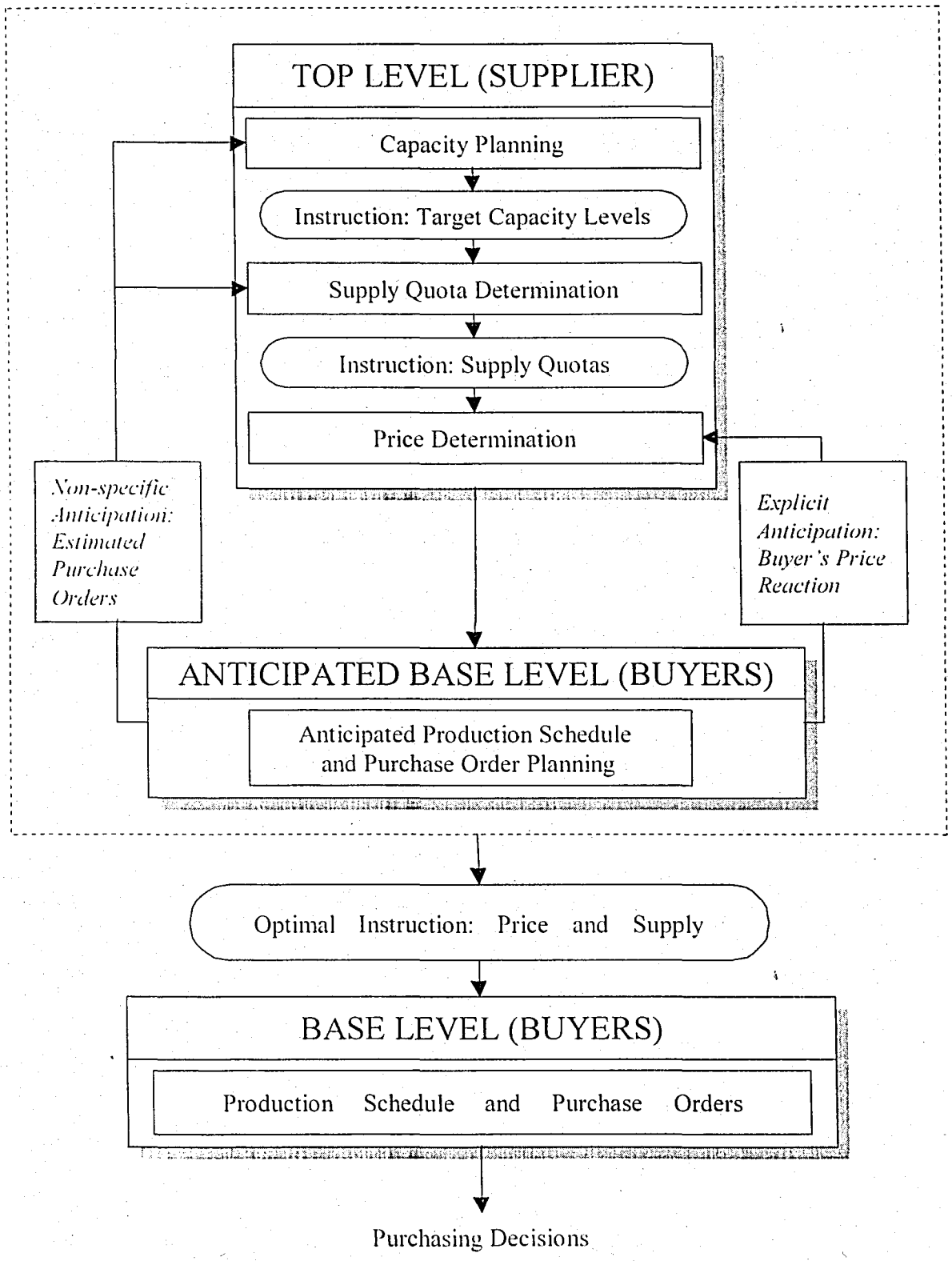


FIGURE 9.7 Hierarchical structure of supply chain

10. CONCLUSION

This research proposes a hierarchical planning structure for a supplier that manufactures a high value, critical item and its buyers. It is assumed that a supply chain partnership with each customer is structured in the form of a supply contract that is confirmed and accepted by all members. Its practice is based upon information-sharing and mutual improvement conditions. Thus it is presumed that supply coordination will eventually reduce the cost and price of the purchase item and both the supplier and the customers will be able to quantify the actual benefits from this.

It is an important issue to design supplier planning process in accordance with the contract and to ensure that it is confirmed and accepted by the buyers as well. This sequential process involves aggregate capacity planning, capacity disaggregation among the customers and price determination. Different mathematical models are introduced to represent these decisions. Since the pricing model turns out to be a nonlinear mixed integer problem, emphasis is placed upon designing a robust solution procedure to cope with the binary variables and nonlinear relations arising in that model, and a global search algorithm is developed by integrating the principles of simulated annealing (SA) and tabu search (TS). Experimentation with test problems reveal that the algorithm is generating high-quality solutions very rapidly and can be used very effectively in real life supply chain negotiations.

This research can be concluded by stating that the best customers are not necessarily the largest customers and the supplier needs a methodology to re-evaluate the customers at contract renewal times and to enhance customer value-added. This can be done by designing and confirming a formal planning structure with all his buyers and deploying operations research techniques capable to treat quantitative and qualitative features in this respect.

In order to determine the supply chain performances of the buyers, Data Envelopment Analysis (DEA) can also be used to replace the AHP approach in a further research. As it is well-known, DEA is a method that can be used to determine the

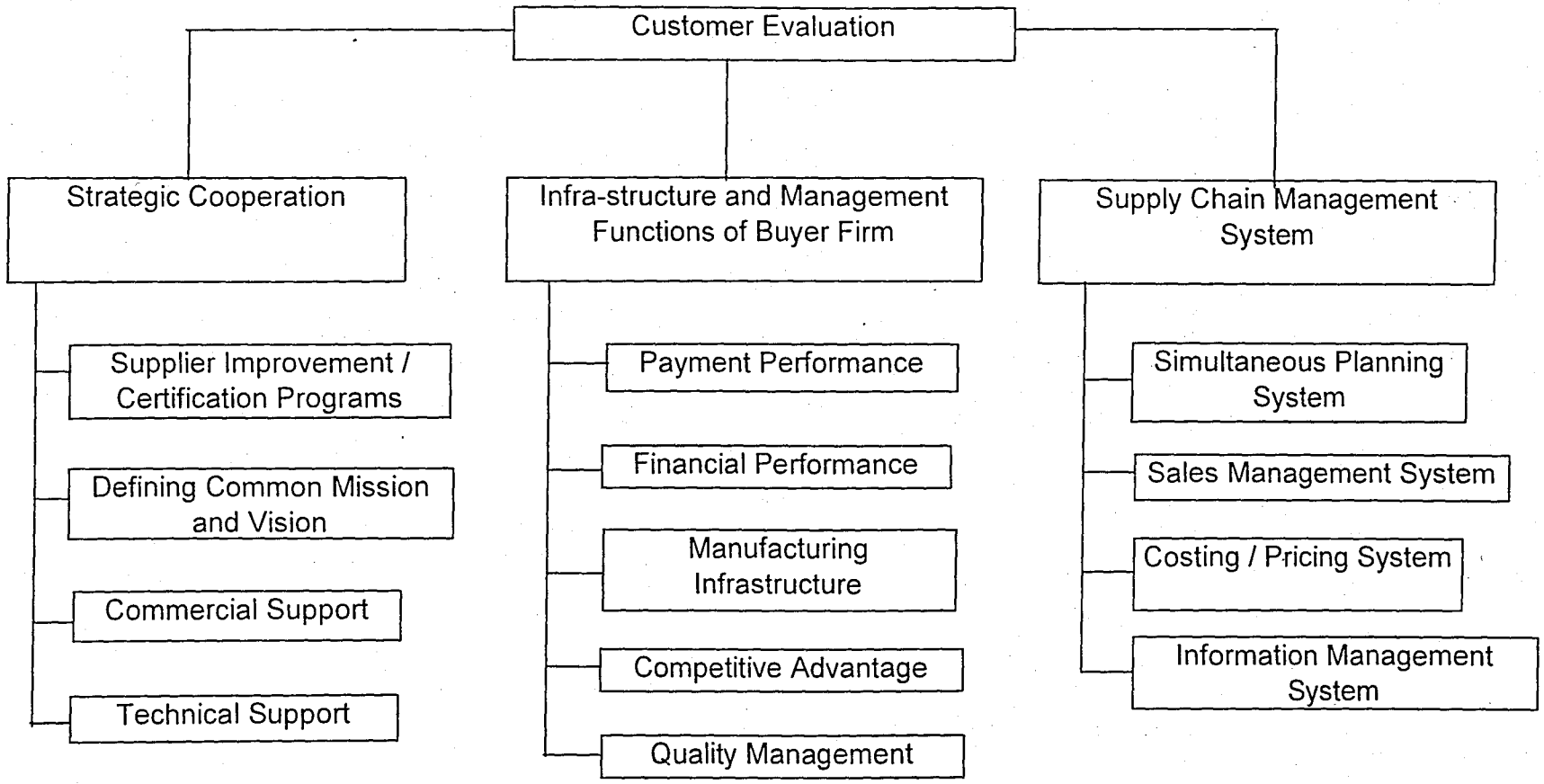
efficiency of decision making units (DMU's) from the point of view of the central unit. In the supply chain context. the central unit can be chosen as a supplier, the DMU's are the buyers. the single input is the supply quota and theoutputs are the criteria that are already used in the AHP. Then the DEA efficiency results could be used in a relevant resource allocation model.

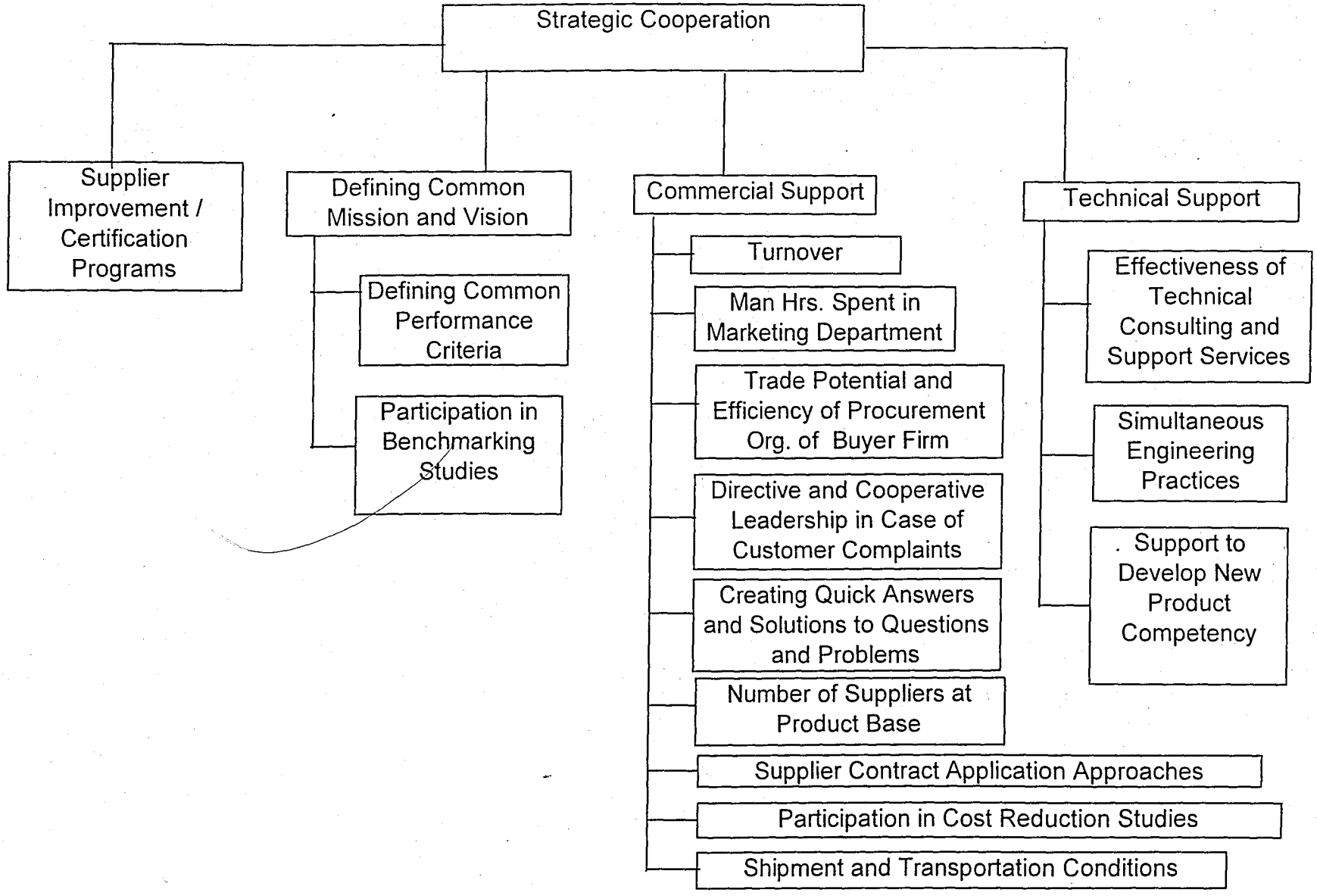
Another natural extension of the approach would be to include a stochastic demand process. One major issue in supply chain planning is to protect the supplier against the unexpected fluctuations in the buyer's requisitions and to provide a stable supplier plan. In order to achieve this. demand randomness should be included into the (CAS) model as a chance constraint.

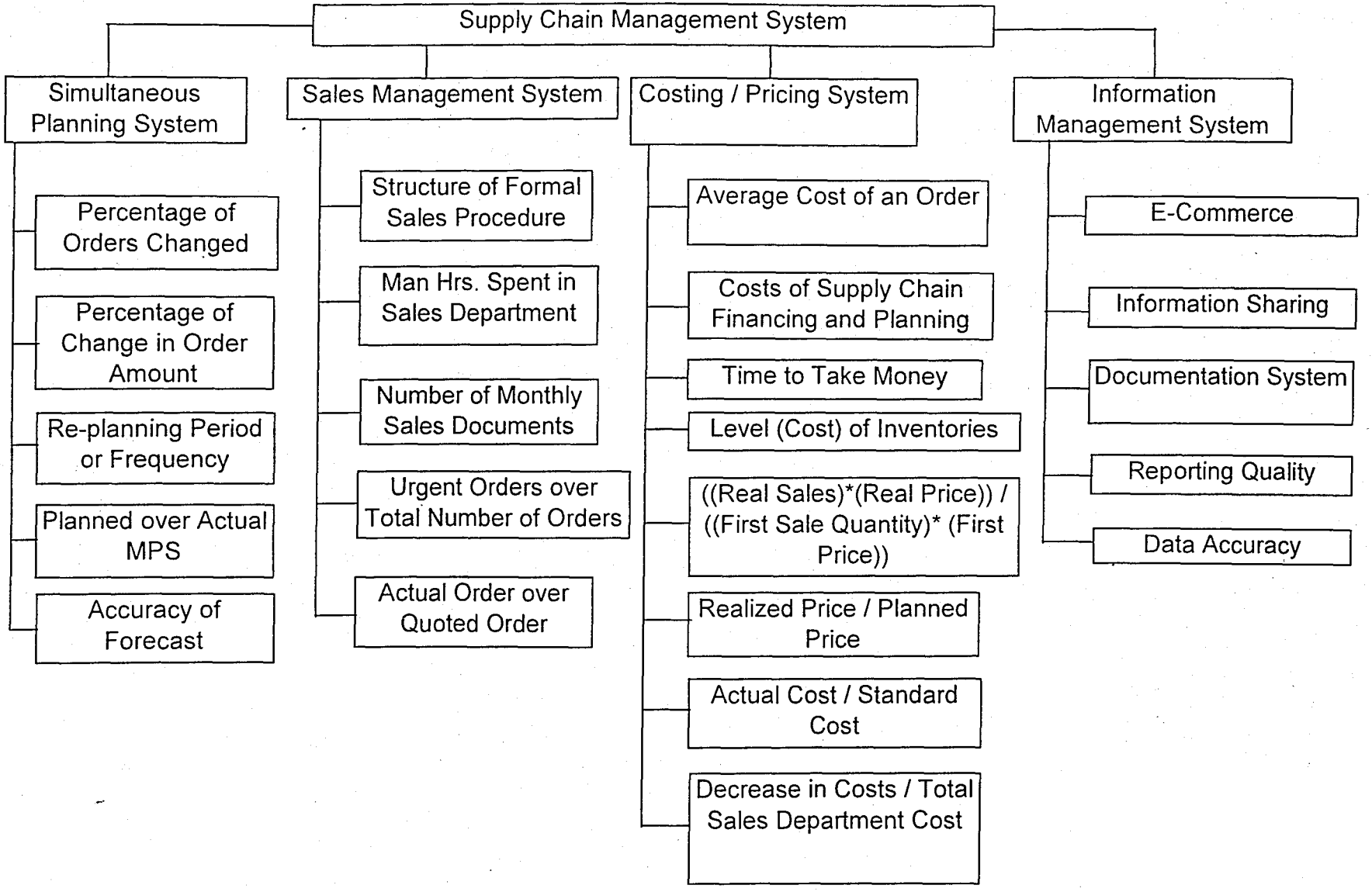
In order to validate the applicability of the proposed approach, a real life application that will complete the whole decision cycle is most recommended as a future study.

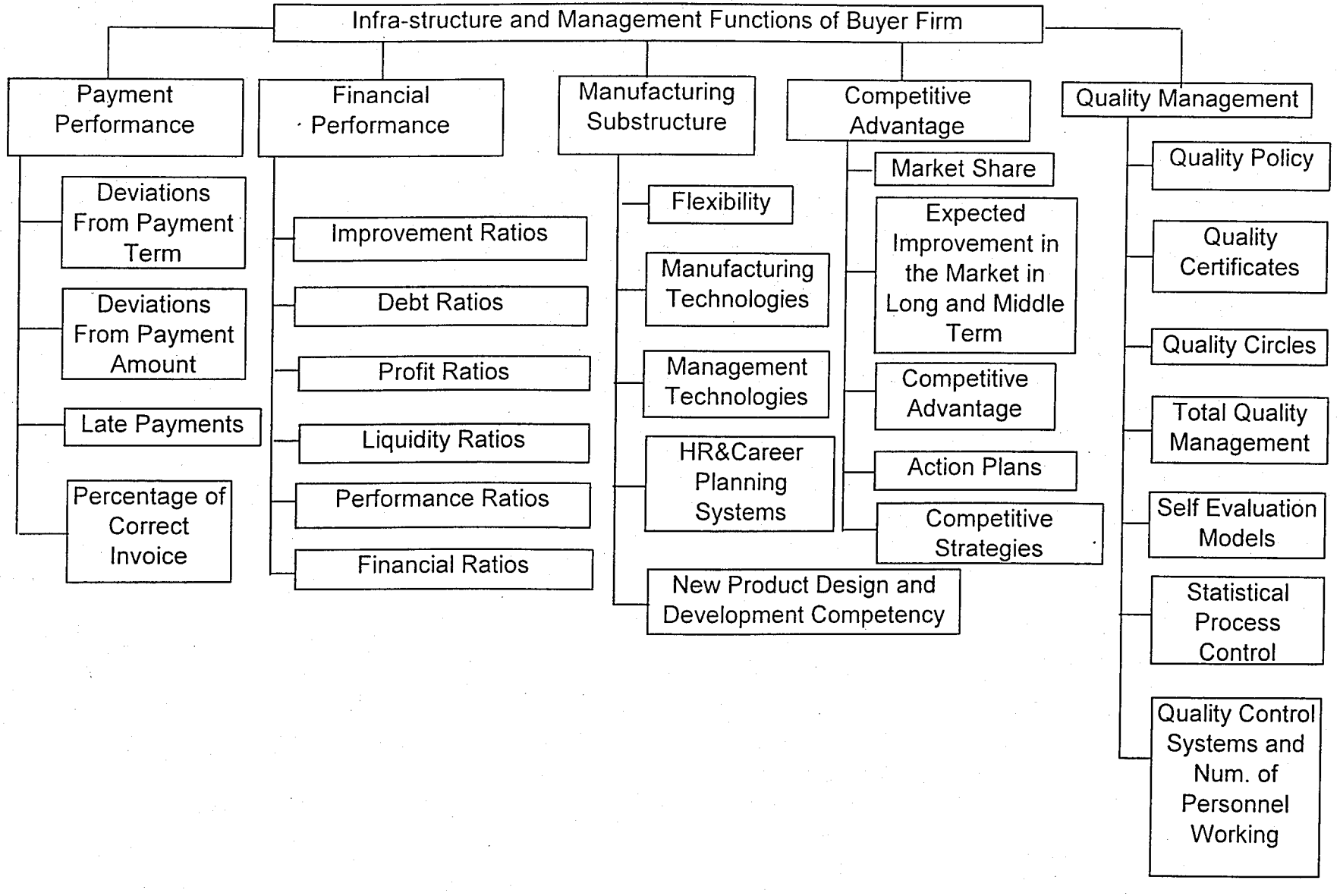
APPENDIX A

AHP TREE


Customer Evaluation
Strategic Cooperation
Infra-structure and Management Functions of Buyer Firm
Supply Chain Management System
Supplier Improvement / Certification Programs
Defining Common Mission and Vision
Commercial Support
Technical Support
Payment Performance
Financial Performance
Manufacturing Infrastructure
Competitive Advantage
Quality Management
Simultaneous Planning System
Sales Management System
Costing / Pricing System
Information Management System


Strategic Cooperation
Supplier Improvement / Certification Programs
Defining Common Mission and Vision
Defining Common Performance Criteria
Participation in Benchmarking Studies
Commercial Support
Turnover
Man Hrs. Spent in Marketing Department
Trade Potential and Efficiency of Procurement Org. of Buyer Firm
Directive and Cooperative Leadership in Case of Customer Complaints
Creating Quick Answers and Solutions to Questions and Problems
Number of Suppliers at Product Base
Supplier Contract Application Approaches
Participation in Cost Reduction Studies
Shipment and Transportation Conditions
Technical Support
Effectiveness of Technical Consulting and Support Services
Simultaneous Engineering Practices
Support to Develop New Product Competency


Supply Chain Management System
Simultaneous Planning System
Sales Management System
Costing / Pricing System
Information Management System
Percentage of Orders Changed
Percentage of Change in Order Amount
Re-planning Period or Frequency
Planned over Actual MPS
Accuracy of Forecast
Structure of Formal Sales Procedure
Man Hrs. Spent in Sales Department
Number of Monthly Sales Documents
Urgent Orders over Total Number of Orders
Actual Order over Quoted Order
Average Cost of an Order
Costs of Supply Chain Financing and Planning
Time to Take Money
Level (Cost) of Inventories
((Real Sales)*(Real Price)) / ((First Sale Quantity)* (First Price))
Realized Price / Planned Price
Actual Cost / Standard Cost
Decrease in Costs / Total Sales Department Cost
E-Commerce
Information Sharing
Documentation System
Reporting Quality
Data Accuracy


Infra-structure and Management Functions of Buyer Firm
Payment Performance
Financial Performance
Manufacturing Substructure
Competitive Advantage
Quality Management
Deviations From Payment Term
Deviations From Payment Amount
Late Payments
Percentage of Correct Invoice
Improvement Ratios
Debt Ratios
Profit Ratios
Liquidity Ratios
Performance Ratios
Financial Ratios
Flexibility
Manufacturing Technologies
Management Technologies
HR&Career Planning Systems
New Product Design and Development Competency
Market Share
Expected Improvement in the Market in Long and Middle Term
Competitive Advantage
Action Plans
Competitive Strategies
Quality Policy
Quality Certificates
Quality Circles
Total Quality Management
Self Evaluation Models
Statistical Process Control
Quality Control Systems and Num. of Personnel Working

APPENDIX B

CUSTOMER WEIGHTS

Table B.1 Customer Evaluations with AHP-Eskom Local Customers

	Customer Evaluation	Strategic Cooperation	Infra-structure and Management Functions of Buyer Firm	Supply Chain Management System
CUSTOMERS		0.183	0.742	0.075
Klimasan	**0.180**	0.261	0.150	0.279
Uğur Soğutma	**0.220**	0.352	0.169	0.408
SFA	**0.311**	0.237	0.342	0.187
Profilo	**0.289**	0.150	0.340	0.126

Table B.2 Customer Evaluations with AHP-Eskom Export Customers

	Customer Evaluation	Strategic Cooperation	Infra-structure and Management Functions of Buyer Firm	Supply Chain Management System
CUSTOMERS		0.540	0.163	0.297
Merloni	**0.371**	0.403	0.363	0.318
Densi	**0.105**	0.086	0.079	0.155
Power Cold	**0.126**	0.110	0.097	0.171
Europair	**0.207**	0.211	0.244	0.180
Firmofrio	**0.190**	0.190	0.217	0.175

Table B.3 Customer Evaluations with AHP-TOPEM

	Customer Evaluation	Strategic Cooperation	Infra-structure and Management Functions of Buyer Firm	Supply Chain Management System
CUSTOMERS		0.600	0.200	0.200
Arçelik	**0.669**	0.675	0.596	0.725
Danfoss	**0.331**	0.325	0.404	0.275

Table B.4 Customer Evaluations with AHP-Sutem Local Çutomers

	Customer Evaluation	Strategic Cooperation	Infra-structure and Management Functions of Buyer Firm	Supply Chain Management System
CUSTOMERS		0.279	0.649	0.072
Standart Pompa	0.170	0.092	0.200	0.203
Toba Redüktör	0.201	0.244	0.185	0.178
Varan Tulumba	0.192	0.205	0.189	0.166
Pitsan	0.069	0.062	0.067	0.117
Süper Teknik	0.161	0.226	0.139	0.118
Gökçe Brülör	0.085	0.109	0.073	0.106
Dalgakıran	0.121	0.062	0.148	0.113

Table B.5 Customer Evaluations with AHP-Sutem Export Cutomers

	Customer Evaluation	Strategic Cooperation	Infra-structure and Management Functions of Buyer Firm	Supply Chain Management System
CUSTOMERS		0.105	0.637	0.258
Pujal	0.146	0.117	0.140	0.172
Sepiee	0.185	0.249	0.189	0.147
Sermes	0.164	0.175	0.166	0.154
MTPC	0.126	0.098	0.126	0.137
Zest	0.079	0.068	0.069	0.107
Küenle	0.155	0.179	0.152	0.150
Elsto	0.147	0.113	0.158	0.133

APPENDIX C

NUMERIC SAMPLES FOR DIFFERENT DISSAGREGATION MODELS

C.1.Calculations Related to The Weights and Demands of Customers

Here, the disaggregation was performed for the third month of the year. The information about demand of buyers and capacity considerations was provided by TEE.

Total Monthly Aggregate Capacity of SUTEM Plant : 30.000 units

Total (Export+ Local) Demand of SUTEM Plant Customers : 26.000

TABLE C.1.1 Weights of SUTEM Plant customers

Export Customers	Weight	Local Customers	Weight
Sepiee	0,1846	Toba Redüktör	0,2010
Sermes	0,1638	Varan Tulumba	0,1920
Küenle	0,1546	Standart Pompa	0,1702
Elsto	0,1469	Süper Teknik	0,1614
Pujal	0,1458	Dalgakıran	0,1213
MTPC	0,1258	Gökçe Brülör	0,0852
Zest	0,0787	Pitsan	0,0689
Total	1,00	Total	1,00

TABLE C.1.2 Demands of SUTEM Plant customers

Export Customers	Demand	Local Customers	Demand
Sepiee	7853	Toba Redüktör	2329
Sermes	2035	Varan Tulumba	1800
Küenle	3473	Standart Pompa	2753
Elsto	1632	Süper Teknik	636
Pujal	1055	Dalgakıran	382
MTPC	274	Gökçe Brülör	1271
Zest	276	Pitsan	233

In implementation of three disaggregation model, it is assumed that the weight of export and local customers are equal for SUTEM Plant. Therefore, all weights are multiplied with 0,5 in the calculations. Furthermore, a sensitivity analysis is carried out for different values for the weights of local customers and export customers as 0,25 and 0,75.

C.2. Disagregation Models with Equal Weights for Local and Export Customers

C.2.1. Weighted Allocation Model

$CAP_t = 30000$

In this model the weights of local and export customers are equal for SUTEM Plant and they are multiplied with 0.5 in the calculations.

TABLE C.2.1.1 Weights of Sutem Plant Customers

Export Customers	Weight	New Weight: 0,5*Weight	Local Customers	Weight	New Weight: 0,5*Weight
Sepiee	0.1846	0.0923	Toba Redüktör	0.201	0.1005
Sermes	0.1638	0.0819	Varan Tulumba	0.192	0.096
Küenle	0.1546	0.0773	Standart Pompa	0.1702	0.0851
Elsto	0.1469	0.07345	Süper Teknik	0.1614	0.0807
Pujal	0.1458	0.0729	Dalgakıran	0.1213	0.06065
MTPC	0.1258	0.0629	Gökçe Brülör	0.0852	0.0426
Zest	0.0787	0.03935	Pitsan	0.0689	0.03445
Total	**1**	**0.5001**	**Total**	**1**	**0.5**

TABLE C.2.1.2 Allocations for Sutem Plant Export Customers

Export Customers	Weight	Allocation	
		CAPt*Weight	Real Demands:
Sepiee	0.0923	2769	7853
Sermes	0.0819	2457	2035
Küenle	0.0773	2319	3473
Elsto	0.07345	2203.5	1632
Pujal	0.0729	2187	1055
MTPC	0.0629	1887	274
Zest	0.03935	1180.5	276
Total	**0.5001**	**15003**	**16598**

TABLE C.2.1.3 Allocations for Sutem Plant Local Customers

Local Customers	Weight	Allocation	
		CAPt*Weight	Real Demands:
Toba Redüktör	0.1005	3015	2329
Varan Tulumba	0.096	2880	1800
Standart Pompa	0.0851	2553	2753
Süper Teknik	0.0807	2421	636
Dalgakıran	0.06065	1819.5	382
Gökçe Brülör	0.0426	1278	1271
Pitsan	0.03445	1033.5	233
Total	**0.5**	**15000**	**9404**

C.2.2. Ordered Allocation Model

CAP_t = 30000

In this model, it is assumed that the weights of local and export customers are equal for SUTEM Plant and they are multiplied with 0,5 in the calculations.

CAP t for local customers=15000

CAP t for export customers=15000

TABLE C.2.2.1. Weights of Sutem plant customers

Export Customers	Weight	Order	Local Customers	Weight	New Weight:
		0,5*Weight			0,5*Weight
Sepiee	0.1846	0.0923	Toba Redüktör	0.201	0.1005
Sermes	0.1638	0.0819	Varan Tulumba	0.192	0.096
Küenle	0.1546	0.0773	Standart Pompa	0.1702	0.0851
Elsto	0.1469	0.07345	Süper Teknik	0.1614	0.0807
Pujal	0.1458	0.0729	Dalgakıran	0.1213	0.06065
MTPC	0.1258	0.0629	Gökçe Brülör	0.0852	0.0426
Zest	0.0787	0.03935	Pitsan	0.0689	0.03445
Total	**1**	**0.5001**	**Total**	**1**	**0.5**

TABLE C.2.2.2. Allocations for Sutem plant export customers

Export Customers:			
	Weight	Demands:	Allocation
Sepiee	0.1846	7853	7853
Sermes	0.1638	2035	2035
Küenle	0.1546	3473	3473
Elsto	0.1469	1632	1632
Pujal	0.1458	1055	7
MTPC	0.1258	274	0
Zest	0.0787	276	0
Total	**1.0002**	**16598**	**15000**

Remaining Capacity= 0

TABLE C.2.2.3. Allocations for Sutem plant local customers

Local Customers:			
	Weight	**Demands:**	**Allocation**
Toba Redüktör	0.201	2329	2329
Varan Tulumba	0.192	1800	1800
Standart Pompa	0.1702	2753	2753
Süper Teknik	0.1614	636	636
Dalgakıran	0.1213	382	382
Gökçe Brülör	0.0852	1271	1271
Pitsan	0.0689	233	233
Total	**1**	**9404**	**9404**

Remaining Capacity= 5596

C.2.3. Regular Knapsack Model

CAP t = 30000 Total Demad=26000

CAPt > Total Demand

In this model, it is assumed that the weights of local and export customers are equal for SUTEM Plant and they are multiplied with 0,5 in the calculations.

TABLE C.2.3.1. Weights of Sutem plant customers

Export Customers	Weight	Order	Local Customers	Weight	New Weight:
		0,5*Weight			0,5*Weight
Sepiee	0.1846	0.0923	Toba Redüktör	0.201	0.1005
Sermes	0.1638	0.0819	Varan Tulumba	0.192	0.096
Küenle	0.1546	0.0773	Standart Pompa	0.1702	0.0851
Elsto	0.1469	0.07345	Süper Teknik	0.1614	0.0807
Pujal	0.1458	0.0729	Dalgakıran	0.1213	0.06065
MTPC	0.1258	0.0629	Gökçe Brülör	0.0852	0.0426
Zest	0.0787	0.03935	Pitsan	0.0689	0.03445
Total	**1**	**0.5001**	**Total**	**1**	**0.5**

TABLE C.2.3.2. Allocations for Sutem plant export customers

Export Customers	Weight	Demands:	Allocation:	Qjt-Djt
Sepiee	0.0923	7853	7853	0
Sermes	0.0819	2035	2035	0
Küenle	0.0773	3473	3473	0
Elsto	0.07345	1632	1632	0
Pujal	0.0729	1055	1055	0
MTPC	0.0629	274	274	0
Zest	0.03935	276	276	0
Total	**0.5001**	**16598**	**16598**	**0**

TABLE C.2.3.3. Allocations for Sutem plant local customers

Local Customers	Weight	Demand	Allocation	Qjt-Djt
Toba Redüktör	0.1005	2329	2329	0
Varan Tulumba	0.096	1800	1800	0
Standart Pompa	0.0851	2753	2753	0
Süper Teknik	0.0807	636	636	0
Dalgakıran	0.06065	382	382	0
Gökçe Brülör	0.0426	1271	1271	0
Pitsan	0.03445	233	233	0
Total	**0.5**	**9404**	**9404**	**0**

C.3. Disagregation Models (Weights of Export Cust. < Local Cust.)

C.3.1. Weighted Allocation Model

CAP t = 30000

In this model, it is assumed that the local cutomers have 3 times more important than export customers in SUTEM Plant. Local customers weights are multiplied with 0,75 and export customers weights are multiplied with 0,25 in the calculations.

TABLE C.3.1.1. Weights of Sutem plant customers

Export Customers	Weight	New Weight: 0,25*Weight	Local Customers	Weight	New Weight: 0,75*Weight
Sepiee	0.1846	0.04615	Toba Redüktör	0.201	0.15075
Sermes	0.1638	0.04095	Varan Tulumba	0.192	0.144
Küenle	0.1546	0.03865	Standart Pompa	0.1702	0.12765
Elsto	0.1469	0.036725	Süper Teknik	0.1614	0.12105
Pujal	0.1458	0.03645	Dalgakıran	0.1213	0.090975
MTPC	0.1258	0.03145	Gökçe Brülör	0.0852	0.0639
Zest	0.0787	0.019675	Pitsan	0.0689	0.051675
Total	**1**	**0.25005**	**Total**	**1**	**0.75**

TABLE C.3.1.2. Allocations for Sutem plant export customers

Export Customers	Weight	Allocation CAPt*Weight	Real Demands:
Sepiee	0.04615	1384.5	7853
Sermes	0.04095	1228.5	2035
Küenle	0.03865	1159.5	3473
Elsto	0.03673	1101.75	1632
Pujal	0.03645	1093.5	1055
MTPC	0.03145	943.5	274
Zest	0.01968	590.25	276
Total	**0.2501**	**7501.5**	**16598**

TABLE C.3.1.3. Allocations for Sutem plant local customers

Local Customers	Weight	Allocation	
		CAPt*Weight	Real Demands:
Toba Redüktör	0.15075	4522.5	2329
Varan Tulumba	0.144	4320	1800
Standart Pompa	0.12765	3829.5	2753
Süper Teknik	0.12105	3631.5	636
Dalgakıran	0.09098	2729.25	382
Gökçe Brülör	0.0639	1917	1271
Pitsan	0.05168	1550.25	233
Total	**0.75**	**22500**	**9404**

C.3.2. Ordered Allocation Model

CAP t = 30000

In this model, it is assumed that the local cutomers have 3 times more important than export customers in SUTEM Plant. Local customers weights are multiplied with 0,75 and export customers weights are multiplied with 0,25 in the calculations.

CAP t for local customers = 500

CAP t for export customers = 7500

TABLE C.3.2.1. Weights of Sutem plant customers

Export Customers	Weight	Order	Local Customers	Weight	New Weight:
		0,25*Weight			0,75*Weight
Sepiee	0.1846	0.04615	Toba Redüktör	0.201	0.15075
Sermes	0.1638	0.04095	Varan Tulumba	0.192	0.144
Küenle	0.1546	0.03865	Standart Pompa	0.1702	0.12765
Elsto	0.1469	0.036725	Süper Teknik	0.1614	0.12105
Pujal	0.1458	0.03645	Dalgakıran	0.1213	0.090975
MTPC	0.1258	0.03145	Gökçe Brülör	0.0852	0.0639
Zest	0.0787	0.019675	Pitsan	0.0689	0.051675
Total	**1**	**0.25005**	**Total**	**1**	**0.75**

TABLE C.3.2.2. Allocations for Sutem plant export customers

Export Customers:			
	Weight	Demands:	Allocation
Sepiee	0.1846	7853	7500
Sermes	0.1638	2035	0
Küenle	0.1546	3473	0
Elsto	0.1469	1632	0
Pujal	0.1458	1055	0
MTPC	0.1258	274	0
Zest	0.0787	276	0
Total	**1.0002**	**16598**	**7500**

Remaining Capacity= 0

TABLE C.3.2.3. Allocations for Sutem plant local customers

Local Customers:			
	Weight	Demands:	Allocation
Toba Redüktör	0.201	2329	2329
Varan Tulumba	0.192	1800	1800
Standart Pompa	0.1702	2753	2753
Süper Teknik	0.1614	636	636
Dalgakıran	0.1213	382	382
Gökçe Brülör	0.0852	1271	1271
Pitsan	0.0689	233	233
Total	**1**	**9404**	**9404**

Remaining Capacity= 0

C.3.3 Regular Knapsack Model

CAP t = 30000 Total Demad=26000

CAPt > Total Demand

In this model, it is assumed that the local cutomers have 3 times more important than export customers in SUTEM Plant. Local customers weights are multiplied with 0,75 and export customers weights are multiplied with 0,25 in the calculations.

TABLE C.3.3.1. Weights of Sutem plant customers

Export Customers	Weight	Order	Local Customers	Weight	New Weight:
		0,25*Weight			0,75*Weight
Sepiee	0.1846	0.04615	Toba Redüktör	0.201	0.15075
Sermes	0.1638	0.04095	Varan Tulumba	0.192	0.144
Küenle	0.1546	0.03865	Standart Pompa	0.1702	0.12765
Elsto	0.1469	0.036725	Süper Teknik	0.1614	0.12105
Pujal	0.1458	0.03645	Dalgakıran	0.1213	0.090975
MTPC	0.1258	0.03145	Gökçe Brülör	0.0852	0.0639
Zest	0.0787	0.019675	Pitsan	0.0689	0.051675
Total	**1**	**0.25005**	**Total**	**1**	**0.75**

TABLE C.3.3.2. Allocations for Sutem plant export customers

Export Customers	Weight	Demands:	Allocation:	Qjt-Djt
Sepiee	0.04615	7853	7853	0
Sermes	0.04095	2035	2035	0
Küenle	0.03865	3473	3473	0
Elsto	0.03673	1632	1632	0
Pujal	0.03645	1055	1055	0
MTPC	0.03145	274	274	0
Zest	0.01968	276	276	0
Total	**0.2501**	**16598**	**16598**	**0**

TABLE C.3.3.3. Allocations for Sutem plant local customers

Local Customers	Weight	Demand	Allocation	Qjt-Djt
Toba Redüktör	0.15075	2329	2329	0
Varan Tulumba	0.144	1800	1800	0
Standart Pompa	0.12765	2753	2753	0
Süper Teknik	0.12105	636	636	0
Dalgakıran	0.09098	382	382	0
Gökçe Brülör	0.0639	1271	1271	0
Pitsan	0.05168	233	233	0
Total	**0.75**	**9404**	**9404**	**0**

C.4 Disagregation Models (Weights of Export Cust. > Local Cust.)

C.4.1. Weighted Allocation Model

CAP t = 30000

In this model, it is assumed that the local cutomers have 3 times less important than export customers in SUTEM Plant. Local customers weights are multiplied with 0,25 and export customers weights are multiplied with 0,75 in the calculations.

TABLE C.4.1.1. Weights of Sutem plant customers

Export Customers	Weight	New Weight: 0,75*Weight	Local Customers	Weight	New Weight: 0,25*Weight
Sepiee	0.1846	0.13845	Toba Redüktör	0.201	0.05025
Sermes	0.1638	0.12285	Varan Tulumba	0.192	0.048
Küenle	0.1546	0.11595	Standart Pompa	0.1702	0.04255
Elsto	0.1469	0.110175	Süper Teknik	0.1614	0.04035
Pujal	0.1458	0.10935	Dalgakıran	0.1213	0.030325
MTPC	0.1258	0.09435	Gökçe Brülör	0.0852	0.0213
Zest	0.0787	0.059025	Pitsan	0.0689	0.017225
Total	**1**	**0.75015**	**Total**	**1**	**0.25**

TABLE C.4.1.2. Allocations for Sutem plant export customers

Export Customers	Weight	Allocation CAPt*Weight	Real Demands:
Sepiee	0.13845	4153.5	7853
Sermes	0.12285	3685.5	2035
Küenle	0.11595	3478.5	3473
Elsto	0.11018	3305.25	1632
Pujal	0.10935	3280.5	1055
MTPC	0.09435	2830.5	274
Zest	0.05903	1770.75	276
Total	**0.7502**	**22504.5**	**16598**

TABLE C.4.1.3. Allocations for Sutem plant local customers

Local Customers	Weight	Allocation	
		CAPt*Weight	Real Demands:
Toba Redüktör	0.05025	1507.5	2329
Varan Tulumba	0.048	1440	1800
Standart Pompa	0.04255	1276.5	2753
Süper Teknik	0.04035	1210.5	636
Dalgakıran	0.03033	909.75	382
Gökçe Brülör	0.0213	639	1271
Pitsan	0.01723	516.75	233
Total	**0.25**	**7500**	**9404**

C.4.2. Ordered Allocation Model

CAP t = 30000

In this model, it is assumed that the local cutomers have 3 times less important than export customers in SUTEM Plant. Local customers weights are multiplied with 0,25and export customers weights are multiplied with 0,75 in the calculations.

CAP t for local customers = 7500

CAP t for export customers = 22500

TABLE C.4.2.1. Weights of Sutem plant customers

Export Customers	Weight	Order	Local Customers	Weight	New Weight:
		0,75*Weight			0,25*Weight
Sepiee	0.1846	0.13845	Toba Redüktör	0.201	0.05025
Sermes	0.1638	0.12285	Varan Tulumba	0.192	0.048
Küenle	0.1546	0.11595	Standart Pompa	0.1702	0.04255
Elsto	0.1469	0.110175	Süper Teknik	0.1614	0.04035
Pujal	0.1458	0.10935	Dalgakıran	0.1213	0.030325
MTPC	0.1258	0.09435	Gökçe Brülör	0.0852	0.0213
Zest	0.0787	0.059025	Pitsan	0.0689	0.017225
Total	**1**	**0.75015**	**Total**	**1**	**0.25**

TABLE C.4.2.2. Allocations for Sutem plant export customers

Export Customers:			
	Weight	**Demands:**	**Allocation**
Sepiee	0.1846	7853	7853
Sermes	0.1638	2035	2035
Küenle	0.1546	3473	3473
Elsto	0.1469	1632	1632
Pujal	0.1458	1055	1055
MTPC	0.1258	274	274
Zest	0.0787	276	276
Total	**1.0002**	**16598**	**16598**

Remaining Capacity=5902

TABLE C.4.2.3. Allocations for Sutem plant local customers

Local Customers:			
	Weight	**Demands:**	**Allocation**
Toba Redüktör	0.201	2329	2329
Varan Tulumba	0.192	1800	1800
Standart Pompa	0.1702	2753	2753
Süper Teknik	0.1614	636	618
Dalgakıran	0.1213	382	0
Gökçe Brülör	0.0852	1271	0
Pitsan	0.0689	233	0
Total	1	**9404**	**7500**

Remaining Capacity=0

C.4.3. Regular Knapsack Model

CAP t = 30000 Total Demad=26000

CAPt > Total Demand

In this model, it is assumed that the local cutomers have 3 times less important than export customers in SUTEM Plant. Local customers weights are multiplied with 0,25 and export customers weights are multiplied with 0,75 in the calculations.

TABLE C.4.3.1. Weights of Sutem plant customers

Export Customers	Weight	Order	Local Customers	Weight	New Weight:
		0,75*Weight			0,25*Weight
Sepiee	0.1846	0.13845	Toba Redüktör	0.201	0.05025
Sermes	0.1638	0.12285	Varan Tulumba	0.192	0.048
Küenle	0.1546	0.11595	Standart Pompa	0.1702	0.04255
Elsto	0.1469	0.110175	Süper Teknik	0.1614	0.04035
Pujal	0.1458	0.10935	Dalgakıran	0.1213	0.030325
MTPC	0.1258	0.09435	Gökçe Brülör	0.0852	0.0213
Zest	0.0787	0.059025	Pitsan	0.0689	0.017225
Total	**1**	**0.75015**	**Total**	**1**	**0.25**

TABLE C.4.3.2. Allocations for Sutem plant export customers

Export Customers	Weight	Demands:	Allocation:	Qjt-Djt
Sepiee	0.13845	7853	7853	0
Sermes	0.12285	2035	2035	0
Küenle	0.11595	3473	3473	0
Elsto	0.11018	1632	1632	0
Pujal	0.10935	1055	1055	0
MTPC	0.09435	274	274	0
Zest	0.05903	276	276	0
Total	**0.7502**	**16598**	**16598**	**0**

TABLE C.4.3.3. Allocations for Sutem plant local customers

Local Customers	Weight	Demand	Allocation	Qjt-Djt
Toba Redüktör	0.05025	2329	2329	0
Varan Tulumba	0.048	1800	1800	0
Standart Pompa	0.04255	2753	2753	0
Süper Teknik	0.04035	636	636	0
Dalgakıran	0.03033	382	382	0
Gökçe Brülör	0.0213	1271	1271	0
Pitsan	0.01723	233	233	0
Total	**0.25**	**9404**	**9404**	**0**

C.5

Table C.5.1Comparison between disaggregation models

	Local Customers	Export Customers	Local Customers	Export Customers	Local Customers	Export Customers
Weight:	0.25	0.75	0.5	0.5	0.75	0.25
Total Allocation in Weighed Allocation Model	7500	22500	15000	15000	22500	7500
Real Demand – Allocation in Total	1904	-5902	-5596	1598	-13096	9098
Unsatisfied Customers	4	1	1	2	0	4
Total Allocation in Ordered Allocation Model	7500	22500	15000	15000	22500	7500
Real Demand – Allocation in Total	1904	-5902	-5596	1598	-13096	9098
Unsatisfied Customers	4	0	0	3	0	6
Total Allocation in Regular Knapsack Model	9404	16598	9404	16598	9404	16598
Real Demand - Allocation in Total	0	0	0	0	0	0
Unsatisfied Customers	0	0	0	0	0	0

Al three methods are inspected, Regular Knapsack Model is the only one satisfying both export and local customers. The other two methods differ in the number of customers satisfied. The difference between the real demand and the allocation in total gives an idea about the excess items in the allocations. If this value is positive, this means that there is a deficit amount in the allocation and some customers are not satisfied.

APPENDIX D

SUPRIME PSEUDO-CODES

```
for N4=0 to N4num;
        {
        for N3=0 to N3num;
                {
                N2 ← 0;
                while (N2 < N2num)
                        {
                        call generate_neighbour();
                        while (new neighbor is TABU and N1 < N1num)
                                {
                                call generate_neighbour();
                                N1 ← N1 + 1;
                                }
                        N1 ← 0;
                        if TABU then
                                {
                                move to the oldest tabu neighbour;
                                N2 ← N2sum + 1;
                                }
                        else
                                if Δ> 0 then
                                        {
                                        make the move;
                                        N2 ← N2sum + 1;
                                        }
                                else
                                        {
                                        N2 ← N2 + 1;
                                        move back to the incumbent solution
                                        }
                                endif;
                        endif;
        }
        if no move yet then  calculate PA
                {
                if the best nonimproving solution accepted then make the move;
                else
                        move back to the incumbent solution
                endif;
        endif;
        N2←0;
        }
                next;
                new price;
                }
        next;
```

Figure D.1 Pseudo-code for **SUPRIME**

if DECREASE **then**

select j* and t* randomly;

$\Delta \leftarrow$ some random quantity;

$DS_{j^*t^*} \leftarrow DS_{j^*t^*} - \Delta$;

select i* randomly;

if $I^-_{i^*j^*t^*} > 0$ **then** $X_{i^*j^*t^*} \leftarrow min\{X_{i^*j^*t^*} + I^-_{i^*j^*t^*}, (IS_{j^*t^*-1} + DS_{j^*t^*})/a_{i^*j^*}\}$;

endif;

if $I^+_{i^*j^*t^*} > 0$ **then**

if $h_{i^*j^*t^*} < hm_{j^*t^*}$ **then** $X_{i^*j^*t^*} \leftarrow (IS_{j^*t^*-1} + DS_{j^*t^*})/a_{i^*j^*}$;

else $X_{i^*j^*t^*} \leftarrow max\{0, X_{i^*j^*t^*} - I^+_{i^*j^*t^*}\}$;

endif;

else

if $h_{i^*j^*t^*} < hm_{j^*t^*}$ **then** $IS_{j^*t^*} \leftarrow IS_{j^*t^*} - \Delta$;

else $X_{i^*j^*t^*} \leftarrow X_{i^*j^*t^*} - \Delta/a_{i^*j^*}$;

endif;

endif;

endif;

if INCREASE **then**

select j* and t* randomly;

$\Delta \leftarrow$ some random quantity

$DS_{j^*t^*} \leftarrow DS_{j^*t^*} + \Delta$; without violating (14) and (16):

select i* randomly;

if $\pi_{i^*j^*t^*} + hm_{j^*t^*} - h_{i^*j^*t^*} - c_{i^*j^*t^*} > 0$ **then** $X_{i^*j^*t^*} \leftarrow (IS_{j^*t^*} + DS_{j^*t^*})/a_{i^*j^*}$

else

if $I^-_{i^*j^*t^*} > 0$ **then** $X_{i^*j^*t^*} \leftarrow min\{X_{i^*j^*t^*} + I^-_{i^*j^*t^*}, (IS_{j^*t^*-} + DS_{j^*t^*})/a_{i^*j^*}\}$

else

if $c_{i^*j^*t^*} + h_{i^*j^*t^*} > hm_{j^*t^*}$ **then** $IS_{j^*t^*} \leftarrow IS_{j^*t^*} + \Delta$;

else $X_{i^*j^*t^*} \leftarrow X_{i^*j^*t^*} + \Delta/a_{i^*j^*}$;

endif;

endif;

endif;

endif;

Figure D.2 Pseudo-code for procedure **(neighbour)**

APPENDIX E

SUPRIME CODES

```
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <stdio.h>
#include <values.h>

// #include <iomanip.h>
long cumQ;
int numbernegIN;
float alpha=0.9;
float T;
enum Boolean { False, True };
struct JT {int jj; int tt;};
int si, sj, st, satir, sutun;
//
int tabu_list_size;
float Rdbest1;
float p_lowerv = 2;
float p_upperv = 8;
int num_of_consecutives = 10;
int count_i = 0;
int count_d = 0;
long **Q;          //*Quotas of buyer j in period t
long ***d;         //*Demand of product i in period t of buyer j
long **DS;         //-Total amount purchased by buyer j in t
long **dummyDS; //-dummy DS buyer j in t
long ***X;         //-Manufactured amount of product i in period t of
buyer j
long **IS;         //*Inventory of the supply item in period t of buyer j
long ***IP; //*Inventory of product i in period t of buyer j
long ***IN; //*Shortage of product i in period t of buyer j
long **a;          //*Amount of the supply item       in 1 unit of item i
float *z;          //-Discounted cost of buyer j
float *TC;         //*Total cost of buyer j
float TP;          //*Total profit
float Alpha;       //*improvement of cost
float ***Pie;      //*Shortage cost of product i in period t of buyer j
long ***Y;         //-Binary variable for setup cost
float ***s;        //*fixed cost of producing product i of buyer j in
period t
float ***c;        //*unit variable cost of product i of buyer j in period
t
float ***h;        //*unit holding cost of product i of buyer j in period
t
float pu;          //*Undiscounted purchased price
float discount;//*Discounted price ratio
float **hm;        //*Unit holding cost of supply item in period t
//float *US;             //Undiscounted purchase amount of the supply item
in period t
float p;                //*given price
float m;       //*given profit percentage
float Rd=-MAXFLOAT;           //-Objective function value
////////////////////////////////////////////////////////////////
long **DSbest;          //-Total amount purchased by buyer j in t
long ***Xbest;     //-Manufactured amount of product i in period t of
buyer j
long ***IPbest;
long ***INbest;
```

```
float *zbest;                    //-Discounted cost of buyer j
float Rdbest= -MAXFLOAT;  //-Objective function value
float **ISbest;
////////////////////////////////////////////////////////////////
long **DSgecerli;
long ***Xgecerli;
long **ISgecerli;
float *zgecerli;
float Rdgecerli=-MAXFLOAT;          //-Objective function value
////////////////////////////////////////////////////////////////
long **DSrotten;
long ***Xrotten;
float protten;
float *zrotten;
float Rdrotten=-MAXFLOAT;           //-Objective function value
////////////////////////////////////////////////////
long **DSglob;
long ***Xglob;
long ***IPglob;
long ***INglob;
float *zglob;
float Rdglob = -MAXFLOAT;           //-Objective function value
float pglob;
long **ISglob;
//////////////////////

float **tabulist;      ////
////////////////////////////////////////////////////////////////
float *compare_gecici;///??????????/
///////////////  integer gets up 32000 something
long max_iter;                  //-maximum number of iteration before the
program exists.
long iterationcount=0;  // iteration counter variable
long feasiblecount=0;   // feasible sol'n after iteration counter
variable.
int *feas;
int *neg;
int ***negi;
//int *negdelta_costs;

float *delta_costs;
float sum3=0;
int ilkneg=0;
time_t ttt;
int tempcount=0;
int check_tabu();
int accept_or_not();
float sum1, sum2, dum;
long stepnumber=0;
int numberofnegIN=0;
long **case2DS;   //xx
long ***case2IN;     //xx
long ***case2IP;     //xx
long **case2IS;    //xx
long ***case2X; //xx
float *case2z; //xx
long ***case2Y;     //xx
float *case2TC;       //xx
float case2Rd;
```

```
float case2TP;
float case2sum1;
float case2sum2;
float case2sum3, case2dum;
long N1=0, N2=0, N3=0, N4=0;

struct IJT {int s_i, s_j, s_t;};
float pgecerli;
clock_t start, end;

//////////////
float beta=0.5;
int cons_imp_num=0;
int cons_disimp_num=0;
//////////////
int previousfeasible=0;
int runpart=0;


//FILE *outfile;

// Function declarations
void Create_All_Arrays();
float*** Create_Array();
float** Create_Array(int tk, int tl);
float* Create_Array(int tk);

long*** Create_Array_long();
long** Create_Array_long(int tk, int tl);
long* Create_Array_long(int tk);

int flag000=0;

void calcumQ();

int*** Create_Array_int();
int* Create_Array_int(int);
void calculate_new_DS(int,int);
void Initialize();
void Check_Feasibility();
void Compute_IS();
int mini(float ***p, int *index, int tj, int tt);
void Compute_I();
void Compute_Y();
void Compute_Rd();
void Tinitial();
void MyError (char str[80]);
//void print(float ***p);
//void print(float **p, int ti, int tj);
//void print(float *p, int ti);
Boolean Feasible1();
Boolean Feasible2();
void Get(char str[13], long ***p, int j);
void Get(char str[13], float ***p, int j);
void Get(char str[13], long **p, int tk, int tl);
void Get(char str[13], float **p, int tk, int tl);
void Destroy_Array(float*** p);
void step1();
void Generate_New_Solution();
void Increase_DS();
```

```
void Decrease_DS();
JT Find_Some_DS();
void Random_Decrease_DS();
void Increase_DS();
int Rand_Inc(int);
JT Any_DS();
JT case1submain();
JT case3submain();
void Decrease_Found(JT);
JT Get_Random_DS();
void Decrease_Found(JT);
void Store_best();
void Modify_random();
void Compute_IS_update();
void nice_DSc(int, int);
void function1(int, int);
//void function2(int, int);
void function3(int, int);
void nice_DSc(int, int);
//void run_n_IS_VAR(int);
void run_n_IS_YOK(int);

////////////////////////////////
//ofstream  fout1("outff.dat");
//ofstream  fout2("outff2.dat");
////////////////////////////////
//ofstream  fout1;
ofstream  fout2;
////////////////////////////////


void artirf(int,int);
void onestep();
void generate_neighbor();
void move_new_tabu(int);
void decrease_index();
void keep_the_best_rotten_apple();
int outgoing();
void move_gecerli_to_gecici();
void update_procedure();
void assign_best_rotten_apple_to_new_solution();
void artir2(int, int, int);
void artir3(int, int, int);
int select_neg_diff();
void move_gecici_to_gecerli();
void neg_mat_sifirla();
int neg_IN_mat_form();
int main_case2();
void move_current_matrices_to_case2_matrices();
IJT get_i_j_t_case2_sub1();
int max_pie(int,int);
void update_case2();
Boolean C2Feas1();
Boolean C2Feas2();
void move_case2_matrices_to_current_matrices();
void C2Compute_IS();
void C2Check_Feasibility();
void C2Compute_I();
void C2Compute_Y();
void C2Compute_Rd();
```

```
int randomn(int);
void calculate_new_DSc3(int,int);
void functionlc3(int,int);
void  Rand_Init();
void Rand_Init2();
void make_a_move();
void generate_p();
void Print_Best_Soln();
void Accept_the_oldest_tabu();
float randomfloat(float);
void initializefornewp();
void printbestofall();
//void refresh_beta_for_imp();
//void refresh_beta_for_disimp();
void main1();
void main2();
void moveglobtogecerliandgecici();
long gettoplam(int);


main()
{
//T = cumQ * (p_upperv + p_lowerv)/2; //////?????????
//T = (p_lowerv+p_upperv)/2;
flag000=4;
alpha = 0.85;
//p//fout1.open("part1.dat");
fout2.open("part1sum.dat");
runpart=1;
main1();
//p//fout1.close();
fout2.close();
//p//fout1.open("part2.dat");
fout2.open("part2sum.dat");
flag000=0;
alpha = 0.95;
runpart=2;
//////p//fout1<<endl<<endl<<"PART2"<<endl<<endl;
//T = (p_lowerv+p_upperv)/2;
T = Rdglob*0.10;
Rdbest1=Rdglob;
main2();
return 0;
}

void MyError (char str[80])
{
      // cout << "Error : " << str;
   //p//fout1 << "Error : " << str;
   fout2 << "Error : " << str;
/*    if (runpart == 1)
            {
            ////p//fout1.close();
            fout2.close();
            ////p//fout1.open("part2.dat");
            fout2.open("part2sum.dat");
            flag000=0;
            alpha = 0.95;
            runpart=2;
            //p//fout1<<endl<<endl<<"PART2"<<endl<<endl;
```

```
            T = Rdglob*0.10;
            Rdbest1=Rdglob;
            main2();
            }
   else
            {
      end=clock();
            printbestofall();
            //p//fout1.close();
            fout2.close();
      }*/
      exit(1);
}


void main1()
{
   start=clock();
      Rand_Init();  //gerekli matrixleri vs. olusturur. initial sol'n
bulur.
   calcumQ();
   Tinitial();
   for (N4=0; N4<50; N4++)
      {
for (N3=0; N3<1600; N3++)    //1000 yapcam.
            {
N2=0;
            make_a_move();
         }
      initializefornewp();
      /* Yukardaki fonksiyonda DSbest, zbest, Rdbest, pbest glob
      degiskenlere tasinacak ve Rdbest neg  max degeri alacak.*/
      //generate_p();
      }
      printbestofall();
      fout2.close();
}


void main2()
{
   moveglobtogecerliandgecici();
   float yp_upperv = p_upperv;
   float yp_lowerv = p_lowerv;
      float deltap = p_upperv - p_lowerv;
   deltap=deltap/8;

      p_lowerv = pglob - deltap;
      p_upperv = pglob + deltap;
   if (yp_upperv<p_upperv) p_upperv = yp_upperv;
   if (yp_lowerv>p_lowerv) p_lowerv = yp_lowerv;


//    Rand_Init();  //gerekli matrixleri vs. olusturur. initial sol'n
bulur.
   for (N4=0; N4<40; N4++)
      {
for (N3=0; N3<1000; N3++)    //1000 yapcam.
            {
N2=0;
```

```
            make_a_move();
          }
      initializefornewp();
//generate_p();
      }
      end=clock();
      printbestofall();
      //p//fout1.close();
      fout2.close();
}


void initializefornewp()
{
int i, t, j;
///besti best of all a ata;/////
if (Rdbest>Rdglob)
      {
      Rdglob=Rdbest;
   pglob=p;
      for(j=0;j<sj;j++)
      {
            zglob[j]=zbest[j];
            for(t=0;t<st+1;t++)
         {
            DSglob[j][t]=DSbest[j][t];
         ISglob[j][t]=ISbest[j][t];
                  for(i=0;i<si;i++)
            {
               Xglob[i][j][t]=Xbest[i][j][t];
            INglob[i][j][t+1]= INbest[i][j][t+1];
                        IPglob[i][j][t+1]= IPbest[i][j][t+1];
            }

         }
            }
      }

Rdbest = -MAXFLOAT;
Rdgecerli = -MAXFLOAT;

for(j=0;j<sj;j++)
      {
      zbest[j]=0;
   zgecerli[j]=0;
      for(t=0;t<st+1;t++)
      {
         DSbest[j][t]=0;
      DSgecerli[j][t]=0;
            for(i=0;i<si;i++)
            {
                  Xbest[i][j][t]=0;
         Xgecerli[i][j][t]=0;
         }
            }
      }
```

```
Tinitial();       ////////////XXXXXXXXXXXXXXXXx

Rand_Init2();

}

void Tinitial()
{
/*if (runpart==1)  T = cumQ *10000* (p_upperv + p_lowerv)/2;     ////1000
kalkacak.
else T = Rdbest1*0.1; */

if (runpart==1)
      {
      T = 0;
      for (int jjj=0; jjj<3; jjj++) T += TC[jjj]*1000;
   T += TP*1000;
   T = T/20;
      }

      else T = Rdbest1*0.1;
}



void make_a_move()
{
////p//fout1 <<"make_a_move()"<<endl;
int g;
int accept;

while (N2<10)
      {
//    //p//fout1 <<"\nN2N2N2N2N2N2N2=="<<N2<<endl;
      generate_neighbor();
   g=check_tabu();
//
   while ((g==0) && (N1<10))
      {
//      //p//fout1 <<"\nTABU!!   N1="<<N1<<endl;
      generate_neighbor();
      N1++;
         g=check_tabu();
      }
   N1=0;
   if (g==0)
      {
      Accept_the_oldest_tabu();//assign old tabu values to the actual
 variables
      update_procedure(); //make the move
            N2=100000;
      }
   else  //not a tabu
      {
      if (Rd>Rdgecerli)          //(Rd>Rdbest) idi originali..
            {
         update_procedure(); //make_the_move();
         N2=100000;
//       refresh_beta_for_imp();
         }
```

```
      else
         {
         N2++;
//         //p//fout1 <<"   keep the best nonimproving "<<N2<<endl;
         keep_the_best_rotten_apple();
         move_gecerli_to_gecici(); //back to origin
//         //p//fout1 <<"   go back to the previous solution "<<N2<<endl;
         }
      }
      }

if (N2!=100000)  //hareket yapilamadi
      {
   accept=accept_or_not(); // nonimoroving;i kabul edeyim mi?
      if (accept==1)          //kabul ettim
         {
//       //p//fout1<<"#DISIMPROVEMENT ACCEPTED"<<endl;
         assign_best_rotten_apple_to_new_solution();
            update_procedure();
      Compute_Rd();

            } //accept_the_non_improving();
      else                     //non improving rejected
      {
      move_gecerli_to_gecici();

      /* *******Burada ileride problem cikabilir N3 degerini kac oldugu
onemli**** */
      N3=32000;      //  donguden kurataracak bir rakam girilir..../
      //      donguden kurataracak bir rakam girilir..../
      }
   }

Rdrotten=-MAXFLOAT;
N2=0;
}

void Rand_Init()
{
      srand((unsigned) time(&ttt));

   int MM1=4;
   int MM2=5;
   int stepnumber=0;
   int stepnumber2=0;
   step1();   // ilk matrisler matrisler olusturulur...
   while (stepnumber<MM1)
      {
            generate_p();
      stepnumber++;
      while (stepnumber2<MM2)
            {
                  onestep();
                  stepnumber2++;
         }
      stepnumber2=0;
         }
      move_gecerli_to_gecici();
      Print_Best_Soln();
//    //p//fout1.close();
```

```
}

void Rand_Init2()
{
      int MM1=4;
   int MM2=5;
   int stepnumber=0;
   int stepnumber2=0;
//   step1(); buna gerek olmayabilir..
   while (stepnumber<MM1)
      {
            generate_p();
      stepnumber++;
      while (stepnumber2<MM2)
            {
                  onestep();
                  stepnumber2++;
         }
      stepnumber2=0;
         }
      move_gecerli_to_gecici();
      Print_Best_Soln();
//    //p//fout1.close();
//return 0;
}


void generate_p()
{
   float p_upper=p_upperv;
      float p_lower=p_lowerv;
      int rast=randomn(10000);
      p=p_lower+(p_upper-p_lower)*rast/10000;
//   //p//fout1<<"####price =\t"<<p<<endl;
}

void onestep()    /// hayir bu main degil!!! Bu sadece 1 adim..
{
      Initialize();
      Check_Feasibility();
      Compute_I();
      Compute_Y();
      Compute_Rd();
//    //p//fout1 <<iterationcount<< "  Rd= " << Rd <<" Sum3= " << sum3 <<
endl;
                        if (Rd > Rdgecerli)   //improvement var.
                        {
//            //p//fout1<<"rand. init. #(Rd > Rdgecerli)#"<<endl;
            move_gecici_to_gecerli();  /// dikkat gecici gecerliye
atanacak..!! //tabu guncellenecek
                        }
      Store_best();
}

/*
void Hile1()
{
      for(int j=0;j<sj;j++)
      for(int t=0;t<st;t++)
      if  (DS[j][t]>=10) DS[j][t] -= 10;
```

```
}

*/

void step1()
{
      int i, j;        // define i and j as integer variables
      ifstream fin1("para.dat");  // look for parameters file.

      if(!fin1) MyError("Could not find file");
      fin1 >> si;           // read number of items
      // cout << si << endl;
      fin1 >> sj;           // read number of suppliers
      fin1 >> st;           // read time horizon
//
      fin1>> tabu_list_size;                //get the size of tabu list.
//

//
      satir=tabu_list_size;
      sutun=( si*sj*st + sj*st + sj + 2);
   // X[i][j][t] DS[j][t] z[j] ve 1 tabu icin, 1 price
//
      Create_All_Arrays(); //call function Create_All_Arrays() to create
arrays
      fin1 >> pu;
      fin1 >> p;
      fin1 >> m;
      fin1 >> Alpha;
      fin1 >> TP;

      for (j=0;j<sj;j++) fin1 >> TC[j];
      fin1 >> discount;
      fin1 >> max_iter;
      for(i=0;i<si;i++)
            for(j=0;j<sj;j++) fin1 >> IP[i][j][0];
      for(i=0;i<si;i++)
            for(j=0;j<sj;j++) fin1 >> IN[i][j][0];
      for(j=0;j<sj;j++) fin1 >> IS[j][0];

      fin1.close();

      Get("d1.dat", d, 0);
      Get("d2.dat", d, 1);
      Get("d3.dat", d, 2);
      Get("d4.dat", d, 3);
      Get("d5.dat", d, 4);
      Get("d6.dat", d, 5);
      Get("d7.dat", d, 6);
      Get("d8.dat", d, 7);
      Get("d9.dat", d, 8);
      Get("d10.dat", d, 9);

      Get("c1.dat", c, 0);
      Get("c2.dat", c, 1);
      Get("c3.dat", c, 2);
      Get("c4.dat", c, 3);
      Get("c5.dat", c, 4);
      Get("c6.dat", c, 5);
```

```
   Get("c7.dat", c, 6);
   Get("c8.dat", c, 7);
   Get("c9.dat", c, 8);
   Get("c10.dat", c, 9);

   Get("h1.dat", h, 0);
   Get("h2.dat", h, 1);
   Get("h3.dat", h, 2);
   Get("h4.dat", h, 3);
   Get("h5.dat", h, 4);
   Get("h6.dat", h, 5);
   Get("h7.dat", h, 6);
   Get("h8.dat", h, 7);
   Get("h9.dat", h, 8);
   Get("h10.dat", h, 9);

   Get("s1.dat", s, 0);
   Get("s2.dat", s, 1);
   Get("s3.dat", s, 2);
   Get("s4.dat", s, 3);
Get("s5.dat", s, 4);
   Get("s6.dat", s, 5);
   Get("s7.dat", s, 6);
   Get("s8.dat", s, 7);
Get("s9.dat", s, 8);
   Get("s10.dat", s, 9);

   Get("pi1.dat", Pie, 0);
   Get("pi2.dat", Pie, 1);
   Get("pi3.dat", Pie, 2);
   Get("pi4.dat", Pie, 3);
   Get("pi5.dat", Pie, 4);
   Get("pi6.dat", Pie, 5);
   Get("pi7.dat", Pie, 6);
   Get("pi8.dat", Pie, 7);
   Get("pi9.dat", Pie, 8);
   Get("pi10.dat", Pie, 9);

   Get("q.dat", Q, sj, st);
   Get("a.dat", a, si, sj);
   Get("hm_jt.dat", hm, sj, st);

   Initialize();

//   Hile1();  // kaldirilacak.....
   Check_Feasibility();

   //    // cout << "Girdim4......." << endl;
   Compute_I();

   //    // cout << "Girdim5......." << endl;

   Compute_Y();

   //    // cout << "Girdim6......." << endl;

   Compute_Rd();
   // cout << "Rd= " << Rd;
move_gecici_to_gecerli();//void
Store_best();
```

```
} //step1--- ex main


void Create_All_Arrays()
{
      Q = Create_Array_long(sj, st);/*call function appropiate function
for creating
      arrays move the add to the pointer Q  */
      d=Create_Array_long();
      X=Create_Array_long();
      IP=Create_Array_long();
      IN=Create_Array_long();
      Pie=Create_Array();
      Y=Create_Array_long();
      s=Create_Array();
      c=Create_Array();
      h=Create_Array();
      DS=Create_Array_long(sj, st);
      dummyDS=Create_Array_long(sj, st);
      IS=Create_Array_long(sj, st);
      a=Create_Array_long(si, sj);
      hm=Create_Array(sj, st);
      z=Create_Array(sj);
      TC=Create_Array(sj);

   ISbest=Create_Array(sj,st);
      DSbest=Create_Array_long(sj, st);
      Xbest=Create_Array_long();
   INbest=Create_Array_long();
   IPbest=Create_Array_long();
      zbest=Create_Array(sj);

   DSrotten=Create_Array_long(sj, st);
   Xrotten=Create_Array_long();
   zrotten=Create_Array(sj);

      feas=Create_Array_int(sj);
      negl=Create_Array_int();
   neg=Create_Array_int(sj);
//    negdelta_costs=Create_Array_int(sj);

      delta_costs=Create_Array(sj);
///////
      DSgecerli=Create_Array_long(sj, st);
      Xgecerli=Create_Array_long();
   ISgecerli=Create_Array_long(sj,st);
      zgecerli=Create_Array(sj);

////////
      DSglob=Create_Array_long(sj, st);
      ISglob=Create_Array_long(sj, st);
      Xglob=Create_Array_long();
      zglob=Create_Array(sj);
   INglob=Create_Array_long();
   IPglob=Create_Array_long();
////////
      tabulist=Create_Array(satir, sutun);
      compare_gecici=Create_Array(sutun);
```

```
////////
   case2X=Create_Array_long();
      case2IP=Create_Array_long();
      case2IN=Create_Array_long();
      case2Y=Create_Array_long();
      case2DS=Create_Array_long(sj, st);
      case2IS=Create_Array_long(sj, st);
      case2z=Create_Array(sj);
      case2TC=Create_Array(sj);


} //Create_All_Arrays()

float*** Create_Array()
{
      int i, j, t;
      float ***p;  //create a pointer to pointer which points to floats
      p = new float** [si];  // create an array of pointers
      if (!p) MyError("Alloc Err");
      for(i=0;i<si;i++)
      {
            p[i] = new float* [sj];
            if (!p[i]) MyError("Alloc Err");
            }

            for(i=0;i<si;i++)
                  for(j=0;j<sj;j++)
         {
                  p[i][j] = new float[st+1];
                  if (!p[i][j]) MyError("Alloc Err");
                  }

      for(i=0;i<si;i++)
      for(j=0;j<sj;j++)
      for(t=0;t<st+1;t++)
            p[i][j][t]=0;

      return p;
}

long*** Create_Array_long()
{
      int i, j, t;
      long ***p;  //create a pointer to pointer which points to floats
      p = new long** [si];  // create an array of pointers
      if (!p) MyError("Alloc Err");
      for(i=0;i<si;i++)
      {
            p[i] = new long* [sj];
            if (!p[i]) MyError("Alloc Err");
            }

            for(i=0;i<si;i++)
                  for(j=0;j<sj;j++)
         {
                  p[i][j] = new long[st+1];
                  if (!p[i][j]) MyError("Alloc Err");
                  }
```

```
	for(i=0;i<si;i++)
	for(j=0;j<sj;j++)
	for(t=0;t<st+1;t++)
		p[i][j][t]=0;

	return p;
}


int*** Create_Array_int()
{
	int i, j, t;
	int ***p;  //create a pointer to pointer which points to int
	p = new int** [si];  // create an array of pointers
	if (!p) MyError("Alloc Err");
	for(i=0;i<si;i++) {
		p[i] = new int* [sj];
		if (!p[i]) MyError("Alloc Err");
	}

	for(i=0;i<si;i++)
	for(j=0;j<sj;j++) {
		p[i][j] = new int[st+1];
		if (!p[i][j]) MyError("Alloc Err");
	}

	for(i=0;i<si;i++)
	for(j=0;j<sj;j++)
	for(t=0;t<st+1;t++)
		p[i][j][t]=0;

	return p;
}


float** Create_Array(int tk, int tl)
{
	int k, l;
	float **p;
	p = new float* [tk];
	for(k=0;k<tk;k++)
   {
		p[k] = new float [tl];
		if (!p[k]) MyError("Alloc Err");
	}

	for(k=0;k<tk;k++)
	for(l=0;l<tl;l++)
		p[k][l]=0;

	return p;
}

float* Create_Array(int tk)
{
	float *p;

	p = new float [tk];
	if (!p) MyError("Alloc Err");
```

```
      for(int k=0;k<tk;k++) p[k]=0;

      return p;
}

///////
long** Create_Array_long(int tk, int tl)
{
      int k, l;
      long **p;
      p = new long* [tk];
      for(k=0;k<tk;k++)
   {
            p[k] = new long [tl];
            if (!p[k]) MyError("Alloc Err");
      }

      for(k=0;k<tk;k++)
      for(l=0;l<tl;l++)
            p[k][l]=0;

      return p;
}

long* Create_Array_long(int tk)
{
      long *p;

      p = new long [tk];
      if (!p) MyError("Alloc Err");

      for(int k=0;k<tk;k++) p[k]=0;

      return p;
}

///////

int* Create_Array_int(int tk)
{
      int *p;
      p = new int [tk];
      if (!p) MyError("Alloc Err");
      for(int k=0;k<tk;k++) p[k]=0;
      return p;
}


/*void Initialize()
{
   //p//fout1<<  "Initilize()//DS[][] & X [][][]"<<endl ;
      int i, j, t;
      for(j=0;j<sj;j++)
            for(t=0;t<st;t++)
      {
      DS[j][t]=Q[j][t];
      //p//fout1<<  " DS["<<j<<"]["<<t<<"] = Q["<<j<<"]["<<t<<"] =
"<<"\t"<<DS[j][t]<<endl;
      }
```

```
      for(i=0;i<si;i++)
            for(j=0;j<sj;j++)
                  for(t=0;t<st;t++)
         {
         X[i][j][t]=d[i][j][t];
            //p//fout1<<  " X["<<i<<"]["<< j <<"]["<< t <<"] = d["<< i
<<"]["<< j <<"]["<<t<<"]="<<"\t"<<d[i][j][t]<<endl ;
         }
      Compute_IS();
}
*/


float randomfloat(float f)
{
      int j;
      j = ((rand() % 10000)+1);  // just a random number between 0 and
kk-1
   float ex;
   ex=f*j/10000;
   ex=floor(ex);
      return ex;
}


void Initialize()    //DS random; X <- d
{
//   //p//fout1<<  "Initilize()//DS[][] & X [][][]"<<endl ;
      int i, j, t;
      for(j=0;j<sj;j++)
            for(t=0;t<st;t++)
      {
      DS[j][t]=randomfloat(Q[j][t]);
//    //p//fout1<<  " DS["<<j<<"]["<<t<<"] = Q["<<j<<"]["<<t<<"] =
"<<"\t"<<DS[j][t]<<endl;
      }
      for(i=0;i<si;i++)
            for(j=0;j<sj;j++)
                  for(t=0;t<st;t++)
         {
          X[i][j][t] = d[i][j][t];
//          //p//fout1<<  " X["<<i<<"]["<< j <<"]["<< t <<"] = d["<< i
<<"]["<< j <<"]["<<t<<"]="<<"\t"<<d[i][j][t]<<endl ;
         }
      Compute_IS();
}

void Get(char str[13], long ***p, int j)
{
      ifstream fin1(str);
      if(!fin1) MyError("Could not find file1");
      for(int i=0;i<si;i++)
            for(int t=0;t<st;t++)
                  fin1 >> p[i][j][t];
      fin1.close();
}
```

```
void Get(char str[13], float ***p, int j)
{
      ifstream fin1(str);
      if(!fin1) MyError("Could not find file1");
      for(int i=0;i<si;i++)
            for(int t=0;t<st;t++)
                  fin1 >> p[i][j][t];
      fin1.close();
}

void Get(char str[13], long **p, int tk, int tl)
{
      ifstream fin1(str);
      if(!fin1) MyError("Could not find file2");
      for(int k=0;k<tk;k++)
            for(int l=0;l<tl;l++)
                  fin1 >> p[k][l];
      fin1.close();
}


void Get(char str[13], float **p, int tk, int tl)
{
      ifstream fin1(str);
      if(!fin1) MyError("Could not find file2");
      for(int k=0;k<tk;k++)
            for(int l=0;l<tl;l++)
                  fin1 >> p[k][l];
      fin1.close();
}

 /*********************************************************/
void Check_Feasibility()  ////varsa neg. IS'leri 0 veya poz. yapar...
{
////p//fout1<<  " Check feasibility "<<endl ;
   int salak_t;
      int i, j, t;
      int ind;
      int *index;
      index = new int[si];
      for(i=0;i<si;i++) index[i]=0;
      for(j=0;j<sj;j++)
   {
            for(t=0;t<st;t++)
      {
        salak_t = t;
                  while (IS[j][salak_t+1]<0)
        {
//    //p//fout1<<  "dikkat IS["<< j<< "][" << (salak_t+1) << "] = " <<
"\t"<<IS[j][salak_t+1]<<endl ;
                        // kac tane i kullandigina gore durdurabiliriz.
                        ind=mini(Pie, index, j, salak_t);
//    //p//fout1<< "min pi index i = " << ind <<endl ;

            DS[j][salak_t] = min (Q[j][salak_t], X[ind][j][salak_t]-
IS[j][salak_t]);
            Compute_IS();
            if (IS[j][salak_t+1]<0) //hala negatif ise...

               {
```

```
                X[ind][j][salak_t] = IS[j][t] + DS[j][t]; //X = ISjt-1 +
DSjt
                        index[ind]=0;
                }
                        Compute_IS();
//////////////////////////////
                    };
                    for(i=0;i<si;i++) index[i]=0;
            }
      }

} //Check_Feasibility

void Compute_IS()
{
//    //p//fout1<<  "Compute_IS()//"<<endl ;
      int i, j, t;
      float sum;
      for(j=0;j<sj;j++)
            for(t=0;t<st;t++)
      {
                  sum=0;
                  for (i=0;i<si;i++) sum +=a[i][j] * X[i][j][t];
                  IS[j][t+1] = IS[j][t] + DS[j][t] - sum;
      //          //p//fout1<<  " IS["<<j<<"]["<<(t+1)<<"] =
"<<"\t"<<IS[j][t+1]<<endl ;

            }

//    //p//fout1<<  " ===e=n=d==c=o=m=p=u=t=e=IS====    "<<endl ;

} // Compute_IS
/*
void Compute_IS_update()
{
   //p//fout1<<  "Compute_IS()//"<<endl ;
      int i, j, t;
      float sum;
      for(j=0;j<sj;j++)
            for(t=0;t<st;t++)
      {
                  sum=0;
                  for(i=0;i<si;i++) sum +=a[i][j] * X[i][j][t];
                  IS[j][t+1] = IS[j][t] + DS[j][t] - sum;
                  //p//fout1<<  " IS["<<j<<"]["<<(t+1)<<"] =
"<<"\t"<<IS[j][t+1]<<endl ;
            }

      //p//fout1<<  " ===e=n=d==c=o=m=p=u=t=e=IS====    "<<endl ;

} // Compute_IS_update()
*/

int mini(float ***p, int *index, int tj, int tt)
{
      float val= MAXFLOAT;
      int storei;

      for(int i=0; i<si; i++)
            if ((p[i][tj][tt]<val)&&(index[i]==0))
```

```
        {
                        val=p[i][tj][tt];
                        storei=i;
                }
        index[storei]=1;
        return storei;
}


void Compute_I()
{
        int i, j, t;
        float sum;
        for(i=0;i<si;i++) {
                for(j=0;j<sj;j++) {
                        for(t=0;t<st;t++) {
                                sum = IP[i][j][t] - IN[i][j][t] + X[i][j][t] -
d[i][j][t];
                                if (sum>0) {
                                        IP[i][j][t+1] = sum;
                                        IN[i][j][t+1] = 0;
                                }
                                else {
                                        IN[i][j][t+1] = -sum;
                                        IP[i][j][t+1] = 0;
                                }
                        }
                }
        }
} //Compute_I

void Compute_Y()
{
        int i, j, t;

        for(i=0;i<si;i++) {
                for(j=0;j<sj;j++) {
                        for(t=0;t<st;t++) {
                                if (X[i][j][t]>0) Y[i][j][t]=1;
                                else Y[i][j][t]=0;
                        }
                }
        }
} // Compute_Y


void Compute_Rd()
{
   float M1, M2;
   M1=1000; M2=1000;
        int i, j, t;
//      float sum1, sum2, sum3=0;
//      float sum1, sum2, sum3, dum;
        sum3=0;
        for(j=0;j<sj;j++)
                {
                sum1=0; sum2=0;
                for(t=0;t<st;t++)
                        {
```

```
                        for(i=0;i<si;i++)
                        {
                        sum1 +=  s[i][j][t] * Y[i][j][t]
                                 + c[i][j][t] * X[i][j][t]
                                 + h[i][j][t] * IP[i][j][t+1]
                                 + Pie[i][j][t] * IN[i][j][t+1];
                        }
                    sum2 += p * DS[j][t] + hm[j][t] * IS[j][t+1];

                    sum3 += p * DS[j][t];


                    }
              z[j] = sum1 + sum2;
              }
        Rd = sum3;
     dum=sum3;     // *m silindi !!!!!!
        if (TP>dum) Rd -= M1*(TP-dum);       //
        for (j=0;j<sj;j++)
        if (z[j]>(Alpha*TC[j]))
        Rd -= M2 * (z[j]-Alpha*TC[j]);
}//void Compute_Rd()



Boolean Feasible1()
{
      float sum=0;
      Boolean retval=False;
      for(int t=0;t<st;t++)
            for(int j=0;j<sj;j++) sum += (p * DS[j][t]); //m
silindi.......
                  if (sum>TP) retval=True;

      return retval;
}

Boolean Feasible2()
{
      Boolean retval=True;
      for (int j=0; j<sj; j++)
      {
            feas[j]=1;
            delta_costs[j] = z[j] - Alpha * TC[j];
            if (delta_costs[j] > 0)
                  {
                  feas[j]=0;  //check for which buyers the second
constraint is infeasible
                  retval=False;
                  }
      }

return retval;
}


int Max_Delta_Cost()
{
      int rj;
```

```
      float dummy = - 3.4e+35;
      for (int j=0; j<sj; j++)
      {
      if (delta_costs[j]> dummy)
            {
            dummy = delta_costs[j];
            rj = j;
            }
      }
if (delta_costs[rj]<0) MyError("Yanlis j");
return rj;
}

int Max_Delta_2_Cost()   //
{
      int rj;
      float dummy = - 3.4e+35;
      for (int j=0; j<sj; j++)
      {
      if (ilkneg==1) if (neg[j]==0) continue;
      if (-delta_costs[j]> dummy)
            {
            dummy = -delta_costs[j];
            rj = j;
            }
      }
return rj;
}


Boolean Dies()
{
      Boolean inc = True;
      int i;
//    time_t ttt;
//    srand((unsigned) time(&ttt));
      i = (rand() % 100);     // random integer between 0-99
      if (i < 50);            // first 50 (0 to 49) increase
      else inc = False;       // second 50 (50 to 99) decrease
      return inc;
}

float nice_DS(int ji, int ti)
{
      float xport;
      if (ti == -1)
      {
      xport = 0;
      }
            else
            {
            float add1;
            add1 = -TP;
            for (int j=0; j<sj; j++)
                  for (int t=0; t<st; t++)
                        add1 += (m*p*DS[j][t]);
            add1 -= (m*p*DS[ji][ti]);
            if (add1<0) add1=0;
//       long dmmy = floor(add1/(m*p));
            xport = min ((add1/(m*p)), (float)IS[ji][ti+1]);
```

```
            }
      return xport;

}

float nice_DSa(int ji, int ti)
{
      float Xort=0;
      float xport;
      if (ti == -1)
      {
      xport = 0;
      }
            else
            {
            float add1;
            add1 = -TP;
            for (int j=0; j<sj; j++)
                  for (int t=0; t<st; t++)
                        add1 += (m*p*DS[j][t]);
            add1 -= (m*p*DS[ji][ti]);
            if (add1<0) add1=0;

      for (int i=0; i<si; i++)
   Xort=X[i][ji][ti];
   Xort=Xort/si;
   int f=(randomn(1000)+1);
   xport = min ((add1/(m*p)), Xort);
//
   if (flag000==4) xport = Xort;
// yeni03 is different by the above statement...

//   flag000=0;

   xport = xport *f/1000; /////random
            //xport = min ((add1/(m*p)), IS[ji][ti+1]);
//    xport = (add1/(m*p));
            }
      return xport;

}

float nice_DSac3(int ji, int ti)
{
      float Xort=0;
      float xport;
      if (ti == -1)
      {
      xport = 0;
      }
            else
            {
            float add1;
            add1 = -TP;
            for (int j=0; j<sj; j++)
                  for (int t=0; t<st; t++)
                        add1 += (m*p*DS[j][t]);
            add1 -= (m*p*DS[ji][ti]);
            if (add1<0) add1=0;
```

```
      for (int i=0; i<si; i++)
   Xort=X[i][ji][ti];
   Xort=Xort/si;
//   xport = min ((add1/(m*p)), Xort);
   int f=(randomn(1000)+1);
   xport = Xort*f/1000;   /// random
            //xport = min ((add1/(m*p)), IS[ji][ti+1]);
//    xport = (add1/(m*p));
            }
      return xport;

}

int randomn(int kk)
{
      int j;
      j = (rand() % kk);    // just a random number between 0 and kk-1
      return j;
} // just returns the random number

void Generate_New_Solution()
{
      int durum=0;
      int j;
      int t;
      iterationcount++;
      Boolean res1=Feasible1();
   Boolean res2=Feasible2();
//    Boolean increase;

      if ((res1==True)&&(res2==False))  durum = 1;
      if ((res1==False)&&(res2==True))  durum = 2;
      if ((res1==False)&&(res2==False)) durum = 3;
      if ((res1==True)&&(res2==True))
      {
      int y=randomn(2);
            //p//fout1 << "** C A S E 4 --> Rd = " << Rd << "   "
<<iterationcount<<endl;
            if (y==1)
               {
            if (runpart ==1) flag000 = 4; else flag000 = 0;
                  durum=1;
            previousfeasible=1;
                        //p//fout1 << "\t\t\t TREATED AS CASE 1 "
<<iterationcount<<+endl;
            }
            else
            {
            if (runpart ==1) flag000 = 4; else flag000 = 0;
               durum=2;
                        //p//fout1 << "\t\t\t TREATED AS CASE 2 "
<<iterationcount<<endl;
               }
      }

      switch (durum)
      {
       case 1 :

                        //    if((res1==True)&&(res2==False)) //case 1
```

```
            {
      JT DUET;
            //p//fout1 << "** C A S E 1 --> Rd = " << Rd << "  "
<<iterationcount<<endl;
            Store_best();
      DUET = caselsubmain();
            calculate_new_DS(DUET.jj,DUET.tt);
            } //case1
            break;
///********CASE 2
            case 2:
      {
      neg_mat_sifirla();
      numbernegIN = neg_IN_mat_form();  //hem IN leri 1/0 matrx yapiyor,
kac eleman oldugunu veriyor
      int r=main_case2();
      while (r==0) r=main_case2();
      }
      break;

//********CASE 3
      case 3:
      {
                  //p//fout1 << "** C A S E 3 --> Rd = " << Rd << "  "
<<iterationcount<<endl;
            Store_best();
      ///                                    sub case random
      JT DUET;
      DUET = case3submain();
      j=DUET.jj;
      t=DUET.tt;
            calculate_new_DSc3(j,t);

      }
      break;
/*          case 4:
            {
            //p//fout1 << "** C A S E 4 --> Rd = " << Rd << "  "
<<iterationcount<<endl;

            Store_best();
            Modify_random();
            }
            break;
*/
            default: printf(" B U G !!! \n ");
      }
}    //Generate_New_Solution()

JT case3submain()
{
      int j,t;
      JT DUET;
            j=select_neg_diff();                  // choose random j with
positive difference
            //sub case max. difference
            //j = Max_Delta_Cost();
            t = randomn(st);
      DUET.jj=j;
```

```
      DUET.tt=t;
      return DUET;
}

JT caselsubmain()
{
      int j,t;
      /*Dikkat: Eger bir onceki cozum feasible ise j secerken infeasible
olan
      bir tane bulamaz onun icin onceki cozumun feasible oldugu
anlasilirsa
      rassal bir j'ye gonderebiliriz */
            JT DUET;
      if (previousfeasible==1)
            {
//          previousfeasible=0;
         j=randomn(sj);
//         //p//fout1<<"random j because the previous ";
//         //p//fout1<<"solution was feasible j = "<<j<<endl;
         }
      else
            {
      /*  DIKKAT j olarak ya pozitif fark olanlardan rassal bir tane
sececek
veya max. fark olanini sececek Asagida gerekli satiri aktive edecegiz,
digerini
iptal edecegiz..*/
      ///                                    sub case random
            j=select_neg_diff();                  // choose random j with
positive difference
            //sub case max. difference
            //j = Max_Delta_Cost();
            }
            t = randomn(st);

      DUET.jj=j;
      DUET.tt=t;
//      if (previousfeasible==1) previousfeasible=0;
      return DUET;
}


void run_n_IS_YOK(int j)
{
int t=st-1;
float remain = Q[j][t] - DS[j][t];
while (remain <= 0)
      {
      t--;
      if (t ==-1)
            {
            MyError("Bulamadi...");
            }
      remain = Q[j][t] - DS[j][t];
      }

float add1=TP;
      for (int j1=0; j1<sj; j1++)
            for (int t1=0; t1<st; t1++)
                  add1 -= p*DS[j1][t1];
```

```
//    float xport = min ((add1/p), IS[ji][ti+1]);
  //  return xport;

      DS[j][t]+= min(remain, (add1/p));

//    //p//fout1<< "nyok*x* DS["<<j<<"]["<<t<<"] = "<<DS[j][t]<<endl ;

}

/*void run_n_IS_VAR(int i, int j, int t) .
{
            int count = 0;
            t = randomn(st);
            while (IS[j][t+1]>=0)
            {
            count++;
            if (count > (10000*st)) MyError("Check if error in case 2");
            t=randomn(st);
            }
            function2(j,t);
}
*/

int negatif_is_var_mi(int j)
{
      int l=0;
      for (int t=0; st; t++)
            {
            if (IS[j][t+1]<0)
                  {
                  l=1;
                  break;
                  }
            }
return l;
}


void calculate_new_DS(int j, int t)
{
JT DUET;
int yt=randomn(2);
if (count_d >= num_of_consecutives) yt = 0;   ///10 defa ustuste decrease
geldi ise..
if (count_i >= num_of_consecutives) yt = 1;  ///10 defa ustuste increase
geldi ise..

if (yt==1)
      {
   count_d++;
   count_i=0;
   if (previousfeasible==1) previousfeasible=0;
    function1(j,t);
//   //p//fout1<<"RANDOM DECISION : DECREASE---"<<endl;
      }
      else
      {
   int dongu=0;
   while (DS[j][t]==Q[j][t])
```

```
      {
      dongu++;
      if (dongu>120)
            {
//       //p//fout1<< "all DS = Q"<<endl<<"mandotary decrease
option"<<endl;
            count_d++;
               count_i=0;
               if (previousfeasible==1) previousfeasible=0;
                  function1(j,t);   /*
************************************ */
//             //p//fout1<<"RANDOM DECISION : DECREASE---"<<endl;
            }
      if (dongu>120) break;
      DUET = caselsubmain();
      j=DUET.jj;
      t=DUET.tt;
      }
   if (dongu <= 120)
      {
         count_i++;
         count_d=0;
         if (previousfeasible==1) previousfeasible=0;
            function3(j,t);
//       //p//fout1<<"RANDOM DECISION : INCREASE+++"<<endl;
      }
      }
}


void calculate_new_DSc3(int j, int t)
{
//int flag=0;
//int X=0;
JT DUET;
int yt=randomn(2);
if (count_d >= num_of_consecutives) yt = 0;
if (count_i >= num_of_consecutives) yt = 1;

if (yt==1)
      {
   count_d++;
   count_i=0;
      function1c3(j,t);
//   //p//fout1<<"RANDOM DECISION : DECREASE---"<<endl;
      } .
      else
      {
//   count_i++;
//   count_d=0;
   int dongu=0;
   while (DS[j][t]==Q[j][t])
      {
      dongu++;
      if (dongu>40000)
            {
         count_d++;
               count_i=0;
               if (previousfeasible==1) previousfeasible=0;
```

```
                    function1c3(j,t);    /*
************************************ */

         }
      if (dongu>40000) break;
      DUET = case3submain();
      j=DUET.jj;
      t=DUET.tt;
      }
   if (dongu <= 40000)
      {
         count_i++;
         count_d=0;
            function3(j,t);
//       //p//fout1<<"RANDOM DECISION : INCREASE+++"<<endl;
      }
      }
}

//case 1a   if IS(j,t)>0.

void function1(int j, int t)
{

int i=0;

double toplam = gettoplam(j);

      int k=t;
      while (DS[j][k]<= 0)
                  {
                  --k;
                  if (k==-1)
                        {
//k =-1 e kadar duserse aradigimiz yok demektir. Degisiklik yapma
//nice_DS'de...
////// D I K K A T ********** no positive IS exists.
//MyError("Bulmadik abi....");
            ++k;
                        break;
                        }
                  }

//    float dec = floor(nice_DSa(j, k));   //bir kontrol et! k'daki
eklenmeyecek!!!
//    if (dec >= DS[j][k]) dec = DS[j][k];
//    if (dec!=0) dec = randomn((int)dec)+1;

   long dec = min (toplam, DS[j][k]);

      DS[j][k] -= dec;

if (IN[i][j][k]>0) X[i][j][k] = ((X[i][j][k] + IN[i][j][k+1]), (IS[j][k]
+ DS[j][k]));
if (IP[i][j][k]>0)
      {
      if (h[i][j][k]> hm[j][t])
      X[i][j][k] = max ((long)0, (X[i][j][k] - IP[i][j][k+1]));
      else
            X[i][j][k] = IS[j][k] + DS[j][k];
```

```
      }
if ( (IP[i][j][k]==IN[i][j][k]) && (IP[i][j][k]==0) )
      {
   if (h[i][j][k] >= hm[j][t]) X[i][j][k] = max((long)0, (X[i][j][k]-
dec));
   if (h[i][j][k] < hm[j][t]) {}
   }
}


void function1c3(int j, int t)
{

int i = 0;

double toplam= gettoplam(j);



      int k=t;
      while (DS[j][k]<= 0)
                  {
                  --k;
                  if (k==-1)
                        {
                        //k =-1 e kadar duserse aradigimiz yok demektir.
Degisiklik yapma
                        //nice_DS'de...
                        ////// D I K K A T ********** no positive IS
exists.
                        //MyError("Bulmadik abi....");
            ++k;
                        break;
                        }
                  }

   long dec = min (toplam, DS[j][k]);

      DS[j][k] -= dec;

if (IN[i][j][k]>0) X[i][j][k] = min ((X[i][j][k] + IN[i][j][k+1]),
(IS[j][k] + DS[j][k]));
if (IP[i][j][k]>0)
      {
      if (h[i][j][k]> hm[j][t])
      X[i][j][k] = max ((long)0, (X[i][j][k] - IP[i][j][k+1]));
      else
            X[i][j][k] = IS[j][k] + DS[j][k];
      }
if ( (IP[i][j][k]==IN[i][j][k]) && (IP[i][j][k]==0) )
      {
   if (h[i][j][k] >= hm[j][t]) X[i][j][k] = max((long)0, (X[i][j][k]-
dec));
   if (h[i][j][k] < hm[j][t]) {}
   }
}
```

```
/*
///////// bunun duzeltmeye ihtiyaci olabilir.!!!!!!!!!!!
void function2(int i, int j,int t)
{
//MyError("error in IS? Detection point function2");

   delta2=compute_Delta2();
      float remain = Q[j][t] - DS[j][t];
      float dec =  min ((double)remain, delta2);
      DS[j][t] += dec;
      //    X[ei][j][t] += a;
      //p//fout1<< "f2*x* DS["<<j<<"]["<<t<<"] = "<<DS[j][t]<<endl ;

      //    return dec;
}
  */
void function3(int j, int t) //IS[j][t+1]=0 durumunda yapilacaklar..
{
int i;
int elde_t;
int ei;
float elde_pi=-1;
//int X= 10;
int INflag=0;
//float a;

for (i=0; i<si; i++)
{
if ( (IN[i][j][t+1]>0) && (Pie[i][j][t]>elde_pi) )
      {
   INflag=1;
   elde_t=t;
   ei=i;
   elde_pi=Pie[i][j][t];
   }
}

if (INflag==1) artir2(ei, j, elde_t);
      else                              //positive IN bulamadi
                                                                  //son
atanmis random t ile yapacak bunu..
      {
   elde_pi=-1;
      for (i=0; i<si; i++)
      {
      if (Pie[i][j][t]>elde_pi)
            {
            ei=i;
            elde_pi=Pie[i][j][t];
         }
      }
   artir3(ei,j,t);
      }//else

}//function3(int j, int t)

long gettoplam(int j)
{
      double toplam=0;
      for (int ttt= 0; ttt<st; ttt++)
```

```
            {
            int per = randomn(2);
            if (per == 1) toplam += d[0][j][ttt];
            }
            return toplam;
}

void artir3(int ei, int j, int t)
{

double toplam = gettoplam(j);

long aa = min(toplam, (Q[j][t] - DS[j][t]));

   DS[j][t] += aa;
   if (Pie[ei][j][t]+hm[j][t]-h[ei][j][t]-c[ei][j][t] > 0)  X[ei][j][t] =
IS[j][t]+DS[j][t];
   else
   {
//   if (IN[ei][j][t+1]>0) X[ei][j][t] = min((X[ei][j][t]+
IN[ei][j][t+1]), (IS[j][t]+DS[j][t]));
   if (IN[ei][j][t+1]>0) X[ei][j][t] = (IS[j][t]+DS[j][t]);
   else
      {
      if (c[ei][j][t]+h[ei][j][t]> hm[j][t]) IS[j][t+1]=IS[j][t+1]+aa;
      else X[ei][j][t]=X[ei][j][t]+aa;
      }
   }

////p//fout1 << "C2 sub2.artir3"<<endl;
////p//fout1 << "DS and X["<<ei<<"]["<<j<<"]["<<t<<"] are increased by
"<< aa<<endl;
////p//fout1<<"DS["<<j<<"]["<<t<<"] = "<<
DS[j][t]<<"\tX["<<ei<<"]["<<j<<"]["<<t<<"] = "<< X[ei][j][t]<<endl;
}

void artir2(int i, int j, int t)
{
//int ii;
long artis;
long remainingQ;
long aaaa;
//float ort=0;
//int per;
long toplam = gettoplam(j);

aaaa = IN[i][j][t] + d[i][j][t]; /// IN[i][j][t-1] indis farkindan dolayi
IN[i][j][t] yazildi.
////// yukarida degisiklik yapildi..
remainingQ = Q[j][t] - DS[j][t];
artis=min((long)remainingQ, toplam);

DS[j][t]+=artis;

X[i][j][t]=min((long)(DS[j][t]+IS[j][t]), (long)aaaa); //IS[j][t-1]

////p//fout1 << "C2 sub1 (func.artir2)"<<endl;
////p//fout1 << "DS and X["<<i<<"]["<<j<<"]["<<t<<"] are increased by "<<
artis<<endl;
```

```
////p//fout1 << "DS["<<j<<"]["<<t<<"] = "<<
DS[j][t]<<"\tX["<<i<<"]["<<j<<"]["<<t<<"] = "<< X[i][j][t]<<endl;
}


/*
{
int klm = randomn (2);
if (klm == 0)
      {
      int xt=t;
      while (DS[j][xt] <= 0)
                  {
                  --xt;
                  if (xt==-1)
                        {
                   // -1 cikarsa yani bisey bulamaz ise hic bir
degisiklik yapilmayacak
                   ////// D I K K A T ********** no positive IS
exists.
                   //MyError("Bulmadik abi.2...");
                        break;
                        }
                  }
      nice_DSc(j, xt);
      }
      else
      artirf(j,t);
}
*/

void artirf(int j,int t)
{
      float dec;
      float remain = Q[j][t] - DS[j][t];
      if (remain <= 0) dec = 0;
            else
            {
             dec = ( (rand() % (int)remain) + 1 );
        //   dec= dummy1*remain/1000
            }
      DS[j][t] += dec;
//    //p//fout1<< "artirf *x* DS["<<j<<"]["<<t<<"] = "<<DS[j][t]<<endl ;

      //    return dec;
}

void nice_DSc(int ji, int ti)
{

      float xport;
      if (ti == -1)
      {
      xport=0;
      }
      else
      {
      float add1;
      add1=-TP;
```

```
      for (int j=0; j<sj; j++)
            for (int t=0; t<st; t++)
                  add1 += m*p*DS[j][t];

      add1 -= m*p*DS[ji][ti];

      xport = min ((add1/(p*m)), (float)DS[ji][ti]);
      }
      DS[ji][ti] -= xport;

      //    return xport;
}


void Modify_random()
{
      int delta;
      JT duejt;
      duejt=Any_DS();
      Boolean inc_dec=Dies();
      if (inc_dec == True)
   {
   int ex=Q[duejt.jj][duejt.tt]-DS[duejt.jj][duejt.tt];
   if (ex<0) MyError("Increase_DS()error");
   delta = Rand_Inc(ex);
      DS[duejt.jj][duejt.tt]+=delta;
   }//DS[duejt.jj][duejt.tt]+= delta;

/*          else if (DS[duejt.jj][duejt.tt]>delta)
DS[duejt.jj][duejt.tt]-=delta;
                              else DS[duejt.jj][duejt.tt]=0;
*/
      else
            {
               int ex=DS[duejt.jj][duejt.tt];
            if (ex<0) MyError("Increase_DS()error");
               delta = Rand_Inc(ex);
                  DS[duejt.jj][duejt.tt]-=delta;
            }

////p//fout1 << "** CASE4-> Rd= " << Rd << " iter # = "
<<iterationcount<<" delta="<<delta <<endl;
////p//fout1<< "*xx* DS["<<duejt.jj<<"]["<<duejt.tt<<"] = "<<
DS[duejt.jj][duejt.tt]<<endl;
}

void Increase_DS()
{
      JT duejt;
      duejt=Any_DS();
   int ex=Q[duejt.jj][duejt.tt]-DS[duejt.jj][duejt.tt];
   if (ex<0) MyError("Increase_DS()error");
   int imp = Rand_Inc(ex);
      DS[duejt.jj][duejt.tt]+=imp;

}  //void increaseDS()

int Rand_Inc(int max_inc)
{
//
```

```
   int retv;
      if (max_inc == 0)
      retv = 0;
   else
            retv=((rand() %max_inc)+1);
      return retv;
}   //Rand_Inc()

void Decrease_DS()
{
      JT DUETJT;
      DUETJT = Find_Some_DS();
      if ((DUETJT.jj == -1)&& (DUETJT.tt == -1)) Random_Decrease_DS();
/* Buraya cok dikkat edilmesi lazim hata olmasin*/
      else Decrease_Found(DUETJT);

}//void DecreaseDS()

JT Any_DS()
{
      JT duetjjtt;
      int rj = (rand() %sj);
      int rt = (rand() %st);
      duetjjtt.jj=rj;
      duetjjtt.tt=rt;
      return duetjjtt;  //JT Any_DS()
}


void Random_Decrease_DS()
{
      JT JJTT;
      JJTT=Get_Random_DS();
      int decre = (rand() %250); //random number between 0-249
      DS[JJTT.jj][JJTT.tt]-=decre;
}



JT Get_Random_DS()
{
//    time_t ttt;
//    srand((unsigned) time(&ttt));

      JT duetjjtt;
      int rj, rt;
      int count=0;
      rj = (rand() %sj);
      rt = (rand() %st);
      while (DS[rj][rt] == 0)
            {
            rj = (rand() %(sj));
            rt = (rand() %(st));
            count++;
            if (count > 10000) MyError("loop:Get_Random_DS");
            }
      duetjjtt.jj=rj;
      duetjjtt.tt=rt;
      return duetjjtt;
}
```

```
void Decrease_Found(JT DU)
{
      int j, t;
      j=DU.jj; t=DU.tt;
      if (DS[j][t] > IS[j][t+1]) DS[j][t] -= IS[j][t+1];
            else DS [j][t] = 0;
}



JT Find_Some_DS()
{
      JT duet;
      duet.jj=0; duet.tt=0;
//    float **dummypointer;
      int rdom;
      int dummy_c=0;
      int tot_of_ones=0;
      int j,t;
      for (j=0; j<sj; j++)
            for (t=0; t<st; t++)
            if ((DS[j][t]!=0)&&(IS[j][t+1]!=0))
                  {
                  dummyDS[j][t]=1;
                  tot_of_ones++;
                  }
            if (tot_of_ones == 0)
                  {
                        duet.jj = -1;
                        duet.tt = -1;
                  }           // if there exists no
            else                          //  appropriate DS,IS pair
                  {                         //
//                time_t ttt;
//                srand((unsigned) time(&ttt));
                  rdom = (rand() % (tot_of_ones+1));    // random element
of the dummyDS

                  //while (dummy_c < rdom)

                  for (j=0; j<sj; j++)
                  {
                        for (t=0; t<st; t++)
                        {
                              if (dummyDS[j][t]==1) dummy_c +=1;
                              if (dummy_c == rdom) break;
                        }
                  if (dummy_c == rdom) break;
                  }

                        duet.jj=j;
                        duet.tt=t;
                        }
       return duet;

} // Find_Some_DS()

//*****STORE BEST

void Store_best()
```

```
{
static float feasiblecount=0;
static float improvenumber=0;
if (Rd>0) feasiblecount++;
int i,j,t;

if (Rdbest < Rd)

      {
      improvenumber++;
      Rdbest = Rd;
      //p//fout1<< "*************************" << endl ;
      //p//fout1 << "Iteration Number      = " << iterationcount <<endl;
      //p//fout1 << "Feasible Sol'n #      = " << feasiblecount <<endl;
      //p//fout1 << "Improved Iteration # = " << improvenumber <<endl;
      //p//fout1 << "Feasible Iteration # = " << feasiblecount <<endl;
      //p//fout1 << "Rd =" << Rd<<"\tp= "<<p<<endl;
      // cout << "Rd =" << Rd <<"\tp= "<<p<< endl;
   //p//fout1 << "TP =" << TP << endl;
   //p//fout1 << "sum(j,t)P[j][t]DS[j][t] =" << sum3 << endl;
   //p//fout1 << "TP-sum(j,t)P[j][t]DS[j][t] =" << (TP-sum3) << endl;
   for (j=0; j<sj; j++)
      //p//fout1<< "TC[" <<j<< "]=" << TC[j] << "\t";
   //p//fout1<< "\n";
   for (j=0; j<sj; j++)
      //p//fout1<< "aTC[" <<j<< "]=" << TC[j]*Alpha << "\t";
   //p//fout1<< "\n";
   for (j=0; j<sj; j++)
      //p//fout1<< "z-aTC[" <<j<< "]=" << (z[j]-TC[j]*Alpha) << "\t";
   //p//fout1<< "\n";

   for (j=0; j<sj; j++)
      //p//fout1<< "z[" <<j<< "]=" << z[j] << "\t";
   //p//fout1<< "\n";

      for (i=0; i<si; i++)
            for (j=0; j<sj; j++)
                  for (t=0; t<st; t++)
                  {
                  Xbest[i][j][t] = X[i][j][t];
                  //p//fout1<< "X[" << i<< "][" << j << "][" << t <<"] =
"<< X[i][j][t]<< endl;
                  IPbest[i][j][t+1] = IP[i][j][t+1];
                  INbest[i][j][t+1] = IN[i][j][t+1];
                  }
      for (j=0; j<sj; j++)
            for (t=0; t<st; t++)
            {
            DSbest[j][t] = DS[j][t];
  //        //p//fout1 << "DS[" << j << "][" << t <<"] = " << DS[j][t]
<< endl;
            }

      for (j=0; j<sj; j++)
            for (t=0; t<st; t++)
            {
            ISbest[j][t+1] = IS[j][t+1];
  //        //p//fout1 << "IS[" << j << "][" << (t+1) <<"] = " <<
IS[j][t+1]   << endl;
            }
```

```
//    for(j=0;j<sj;j++)
//          for(t=0;t<st;t++)
            //    //p//fout1<<  " IS["<<j<<"]["<<(t+1)<<"] =
"<<"\t"<<IS[j][t+1]<<endl ;



      for (j=0; j<sj; j++)
            {
            zbest[j]= z[j];
            //p//fout1 << "Z[" << j << "] " << z[j] << endl;
            }
      }
else
      {
      //p//fout1<< "*worse then the best*" << endl ;
      //p//fout1 << "Iteration Number     = " << iterationcount <<endl;
      //p//fout1 << "Feasible Sol'n #     = " << feasiblecount <<endl;
      //p//fout1 << "Improved Iteration # = " << improvenumber <<endl;
//    //p//fout1 << "Rd =" << Rd<<endl;
   //p//fout1 << "Rd =" << Rd<<"\tp= "<<p<<endl;
      // cout << "Rd =" << Rd <<"\tp= "<<p<< endl;
      //p//fout1 << "TP =" << TP << endl;
   //p//fout1 << "sum(j,t)P[j][t]DS[j][t] =" << sum3 << endl;
   //p//fout1 << "TP-sum(j,t)P[j][t]DS[j][t] =" << (TP-sum3) << endl;
/*   for (j=0; j<sj; j++)
      //p//fout1<< "TC[" <<j<< "]=" << TC[j] << "\t";
   //p//fout1<< "\n";
      for (j=0; j<sj; j++)
      //p//fout1<< "z-aTC[" <<j<< "]=" << (z[j]-TC[j]*Alpha) << "\t";
   //p//fout1<< "\n";

   for (j=0; j<sj; j++)
      //p//fout1<< "z[" <<j<< "]=" << z[j] << "\t";
   //p//fout1<< "\n";

      for (i=0; i<si; i++)
            for (j=0; j<sj; j++)
                  for (t=0; t<st; t++)
                  {
//                Xbest[i][j][t] = X[i][j][t];
                  //p//fout1<< "X[" << i<< "][" << j << "][" << t <<"] =
"<< X[i][j][t]<< endl;
                  }
      for (j=0; j<sj; j++)
            for (t=0; t<st; t++)
            {
//          DSbest[j][t] = DS[j][t];
            //p//fout1 << "DS[" << j << "][" << t <<"] = " << DS[j][t]
<< endl;
            }
      for(j=0;j<sj;j++)
            for(t=0;t<st;t++)
                  //p//fout1<<  " IS["<<j<<"]["<<(t+1)<<"] =
"<<"\t"<<IS[j][t+1]<<endl ;

      for (j=0; j<sj; j++)
            {
```

```
//          zbest[j]= z[j];
            //p//fout1 << "Z[" << j << "] " << z[j] << endl;
            }

*/
   }

}


void calcumQ()
{
      cumQ = 0;
   int j, t;
      for(j=0;j<sj;j++) for(t=0;t<st;t++) cumQ = cumQ + Q[j][t];
}


void Destroy_Array(float*** p)
{
      int i, j;
      for(i=0;i<si;i++)
      for(j=0;j<sj;j++) delete [] p[i][j];
      for(i=0;i<si;i++) delete [] p[i];
      delete [] p;
}

int check_tabu()
{
      int ex = 0;
      int ayni;
      int ccount = 0;
      for (int sat=0; sat<satir; sat++)
      {
            ayni=1;
            for (int sut=1; sut < sutun; sut++) //??????
            {
                  if  (fabs(compare_gecici[sut] - tabulist[sat][sut]) >
0.000001)
         {
                        ayni = 0;
                        break;
                  }
            }
            ccount+= ayni;
      }
      if (ccount < 0.0001)
            ex = 1;

   //p//fout1<< "check TABU z[j]'s "<<endl;
   for (int sat=0; sat<satir; sat++)
   {
   int kkk = (si*sj*st + sj*st+ 1);
   int kkl = (si*sj*st + sj*st + sj +1);
   for (int sut=kkk; sut<kkl; sut++)
      {
      //p//fout1 <<" tabu["<<sat<<"]["<<sut<<"] "<<
tabulist[sat][sut]<<endl;
      }
```

```
   }

      return ex;
}

void move_gecerli_to_gecici()
{
      for(int j=0;j<sj;j++)
      {
            z[j]=zgecerli[j];
            for(int t=0;t<st+1;t++)
            {
         IS[j][t]= ISgecerli[j][t];
                  DS[j][t]= DSgecerli[j][t];
                  for(int i=0;i<si;i++)
                        X[i][j][t]=Xgecerli[i][j][t];
            }
      }
Rd=Rdgecerli;
p=pgecerli;
}

void move_gecici_to_gecerli()
{
      for(int j=0;j<sj;j++)
      {
            zgecerli[j]=z[j];
            for(int t=0;t<st+1;t++)
            {
         ISgecerli[j][t]= IS[j][t];
                  DSgecerli[j][t]= DS[j][t];
                  for(int i=0;i<si;i++)
                        Xgecerli[i][j][t]=X[i][j][t];
            }
      }
Rdgecerli=Rd;
pgecerli=p;
}

void update_procedure()
{
      move_gecici_to_gecerli();
      decrease_index();
      int sat=outgoing();          // neydi bunlar????
      move_new_tabu(sat);   //
   Compute_IS();
   Check_Feasibility();
   Compute_I();
   Compute_Y();

}

void move_new_tabu(int sat)
{
for (int sut=0; sut<sutun; sut++)
      tabulist[sat][sut] = compare_gecici[sut];
tabulist[sat][0]=satir;
}

void Accept_the_oldest_tabu()
```

```
{
int oldest=outgoing();
//   //p//fout1 <<" Accept the oldest tabu.. oldest="<<oldest<<endl;

      int k=0, i, j, t;
      for(i=0;i<si;i++)
            for(j=0; j<sj; j++)
                  for(t=0;t<st;t++)
                     X[i][j][t]=tabulist[oldest][++k];
      for(j=0;j<sj;j++)
            for(t=0;t<st;t++)
                  DS[j][t]=tabulist[oldest][++k];
      for(j=0;j<sj;j++)
            z[j]=tabulist[oldest][++k];
   p=tabulist[oldest][++k];

}

int outgoing()
{
      int ex;
      int maxi=(satir+1);
      for (int sat=0; sat<satir; sat++)
            if (tabulist[sat][0] < maxi)
                  {
                  maxi = tabulist[sat][0];
                  ex=sat;
                  }

      return ex;
}

int select_neg_diff()
{
int j=randomn(sj);
while (delta_costs[j] <=0)
      j=randomn(sj);

return j;
}

void decrease_index()
{
      for (int sat=0; sat<satir; sat++)
            {
            if (tabulist[sat][0] < 0.1) continue;
            if (tabulist[sat][0] < 1.1)
                  for (int sut=0; sut<sutun; sut++)
                        tabulist[sat][sut] = 0;
            else tabulist[sat][0] = (tabulist[sat][0]-1);
            }
}

void keep_the_best_rotten_apple()
{
      if (Rd>Rdrotten)
      {
      for(int j=0; j<sj; j++)
      {
            zrotten[j]=z[j];
```

```
            for(int t=0;t<st+1; t++)
            {
                  DSrotten[j][t]= DS[j][t];
                  for(int i=0;i<si;i++)
                        Xrotten[i][j][t]=X[i][j][t];
            }
      }
Rdrotten=Rd;
protten=p;
      }

}


void assign_best_rotten_apple_to_new_solution()
{
      for(int j=0; j<sj; j++)
      {
            z[j]=zrotten[j];
            for(int t=0;t<st+1; t++)
            {
                  DS[j][t]= DSrotten[j][t];
                  for(int i=0;i<si;i++)
                        X[i][j][t] = Xrotten[i][j][t];
            }
      }
Rd = Rdrotten;
p=protten;
}



float cal_temp()
{
      static float temperature=1;
   if (tempcount ==9)
   {
   tempcount =0;
   temperature=temperature*0.9;
   }
   return temperature;
}

int accept_or_not()
{
//    static float T = 1; bunu da global yaptim cunku fiyat degistiginde
initial valuelara donecek..
   if (T <= 0.001) Tinitial(); ///// T cok kuculurse yeniden
yukseltiyoruz....

      int acc=0;
 // burada exponansiyel fonksiyonu kullanarak kabul veya redde karar
verecegiz.
//    float temp;
      double ras;
//    temp=cal_temp();
   int xyz=randomn(1000);
      ras=(double)xyz/1000;
      double PA;
//   float gec=fabs(Rdgecerli);
```

```
   float del=Rdrotten-Rdgecerli;
//    float payda;
      PA=exp((double)(del/T));     // (Rdrotten < Rdglob) durumu
   if (ras <= PA)
      {
         acc =1;
//    refresh_beta_for_disimp();
//      if (beta>=1) beta = 0.5;
//      //p//fout1 << "beta = "<< beta<< endl;
      T=T*alpha;
//    T= T /(1+beta*T);
      }
//p//fout1<< "T =\t"<< T << "\tR=\t" << ras << "\t  PA = " << PA <<
"\tRdgecerli\t" << Rdgecerli << "\tRdrotten\t"<< Rdrotten <<"\tdelta\t"<<
del<<endl;
/////////fout2<< "T =\t"<< T << "\tR=\t" << ras << "\t  PA = " << PA <<
"\tRdgecerli\t" << Rdgecerli << "\tRdrotten\t"<< Rdrotten <<"\tdelta\t"<<
del<<endl;
      return acc;
}

/*void refresh_beta_for_imp()
{
float alpha = 0.001;
cons_imp_num++;
cons_disimp_num=0;
if (cons_imp_num==10)    // burasi da soft kontrol olmali degisken ismi
kullanilmali
      {
   cons_imp_num=0;
   beta=beta*(1-alpha);
   //p//fout1<<"alpha = "<<alpha<<endl;
   //p//fout1<<"beta = "<<beta<<endl;
   }
}
*/


/*
void refresh_beta_for_disimp()
{
float alpha = 0.001;
cons_disimp_num++;
cons_imp_num=0;
if (cons_disimp_num==10)  //burasi soft kontrol olmali
      {
   cons_disimp_num=0;
   beta=beta*(1+alpha);
   //p//fout1<<"beta = "<<beta<<endl;
   }
}
*/

void generate_neighbor()
{
      int i, j,t;
      Generate_New_Solution();
   Compute_IS();
      Check_Feasibility();
      Compute_I();
```

```
      Compute_Y();
      Compute_Rd();
      //p//fout1 <<iterationcount<< "  Rd= " << Rd <<" Sum3= " << sum3 <<
endl;
      int k=0;
      for(i=0;i<si;i++)
            for(j=0;j<sj;j++)
                  for(t=0;t<st;t++)
                        compare_gecici[++k] = X[i][j][t];
      for(j=0;j<sj;j++)
            for(t=0;t<st;t++)
                  compare_gecici[++k] = DS[j][t];
      for(j=0;j<sj;j++)
            compare_gecici[++k] = z[j];
   compare_gecici[++k] = p;

// Ilk field bos digerleri dolu olacak sanirim.
// DIKKAT DIKKAT
// BURADA BIR PROBLEM CIKABILIR>>>>>>>>>>>>>

// return 0;
}

void neg_mat_sifirla()
{
      for (int i=0; i<si; i++)
            for (int j=0; j<sj; j++)
                  for (int t=0; t<st; t++)
                  neg1[i][j][t]=0;
ilkneg=0;
}

int neg_IN_mat_form()
{
numberofnegIN=0;
for (int i=0; i<si; i++)
   for (int j=0; j<sj; j++)
            for (int t=0; t<st; t++)
            if (IN[i][j][t+1]<0)
            {
            neg1[i][j][t]=1;
            ilkneg=1;
       ++numberofnegIN;
            }
            else neg1[i][j][t]=0;
      return numberofnegIN;
}

int main_case2()
{
      float increment;
   int ret;
      //p//fout1 << "** C A S E 2 --> Rd = " << Rd << "   "
<<iterationcount<<endl;
      Store_best();
   move_current_matrices_to_case2_matrices();
//    int j,t;
   int ii;
   int jj;
   int tt;
```

```
      //          int jjj=randomn(sj);
//    neg_mat_sifirla();
//    int numberofnegIN = neg_IN_mat_form();  //hem IN leri 1/0 matrx
yapiyor, kac eleman oldugunu veriyor
      if (numbernegIN >0)     //bir tane negatif buldu en azindan..
      {
   IJT abc = get_i_j_t_case2_sub1();// rassal i,j,t abc.i, j, t olarak
alindi //alinan ise matrix'ten silindi.
      ii=abc.s_i;
   jj=abc.s_j;
   tt=abc.s_t;
//    //p//fout1 <<"negIN var"<<endl;
//    //p//fout1<<ii<<"\t"<<jj<<"\t"<<tt<<endl;

   }
   else
      {
      jj=randomn(sj);
      tt=randomn(st);
      //ii=max_pie(jj,tt);  // bu max pie'degil max h olacak. bu hali
yanlis..
      ii=randomn(si);       //ii rassal secildi..
//      //p//fout1 <<"negIN yok"<<endl;
//      //p//fout1<<ii<<"\t"<<jj<<"\t"<<tt<<endl;

      }
   float RemQ = Q[jj][tt] - DS[jj][tt];

int don = 0;
   while (RemQ == 0)
///
      {
   don++;
   if (don>120)
//    {
//    MyError("Infinite loop: line 2327");             ///
//    }
         break;
      if (numbernegIN >0)     //bir tane negatif buldu en azindan..
///
      {
///
   IJT abc=get_i_j_t_case2_sub1();// rassal i,j,t abc.i, j, t olarak
alindi  ///
      ii=abc.s_i;
///
   jj=abc.s_j;
///
   tt=abc.s_t;
///
//    //p//fout1 <<"negIN var"<<endl;
///
//    //p//fout1<<ii<<"\t"<<jj<<"\t"<<tt<<endl;
///
                                                                  ///
   }
///
   else
///
```

```
      {
///
      jj=randomn(sj);
///
      tt=randomn(st);
///
      //ii=max_pie(jj,tt);  // bu max pie'degil max h olacak. bu hali
yanlis..
      ii=randomn(si);       //ii rassal secildi..
///
//      //p//fout1 <<"negIN yok"<<endl;
///
//      //p//fout1<<ii<<"\t"<<jj<<"\t"<<tt<<endl;
///

///
      }
///
   RemQ = Q[jj][tt] - DS[jj][tt];
///

///
   }
///
////////////////////////////////////////////////////////////////////////////
///////

   float delta2 = (Alpha*TC[jj]-z[jj])/p;

   increment = min (RemQ, delta2);

//bu program yukardaki satir ile yeni02.den farkli...
   if (flag000==4) increment=RemQ;
   int k = randomn(1000);
   increment = ceil(k*increment/1000);
   case2DS[jj][tt] += increment;
   case2X[ii][jj][tt] += increment;
//   //p//fout1<<"wants to increment by " << increment<<endl ;
   update_case2();
   Boolean res2=C2Feas2();
   if (res2 ==True)
      {
//      //p//fout1<<" const2 is feasible. iteration done. "<<endl;
      move_case2_matrices_to_current_matrices();
         ret=1;
      }
   else
      {
      if (flag000==4)
            {
//          //p//fout1<<" const2 is not feasible. STILL iteration done.
"<<endl;
            move_case2_matrices_to_current_matrices();
                  ret=1;
         }
         else
            {
//                //p//fout1<<" const2 is infeasible. aborted "<<endl;
                     move_current_matrices_to_case2_matrices();
                ret=0;
```

```
            }
      }
//flag000 = 0;
return ret;

}

void move_current_matrices_to_case2_matrices()
{
   case2TP=TP;
      case2Rd=Rd;
      for (int j=0; j<sj; j++)
      {
            case2z[j]=z[j];
//      case2TC[j]=TC[j];
            for (int t=0; t<=st; t++)
         {

         case2IS[j][t]=IS[j][t];
         case2DS[j][t]=DS[j][t];
               for (int i=0; i<si; i++)
            {
         case2X[i][j][t]=X[i][j][t];
            case2Y[i][j][t]=Y[i][j][t];
         IP[i][j][t]=case2IP[i][j][t];
         IN[i][j][t]=case2IN[i][j][t];
            }
            }
      }
}


void move_case2_matrices_to_current_matrices()
{
   TP=case2TP;
      Rd=case2Rd;
      for (int j=0; j<sj; j++)
      {
            z[j]=case2z[j];
//      TC[j]=case2TC[j];
            for (int t=0; t<=st; t++)
         {

         IS[j][t]=case2IS[j][t];
         DS[j][t]=case2DS[j][t];
               for (int i=0; i<si; i++)
            {
         X[i][j][t]=case2X[i][j][t];
            Y[i][j][t]=case2Y[i][j][t];
         IP[i][j][t]=case2IP[i][j][t];
         IN[i][j][t]=case2IN[i][j][t];
            }
            }
      }
}

void update_case2()
{
//    move_gecici_to_gecerli();
//    decrease_index();
```

```
//    int sat=outgoing();          // neydi bunlar????
//    move_new_tabu(sat);   //
   C2Compute_IS();
   C2Check_Feasibility();
   C2Compute_I();
   C2Compute_Y();
   C2Compute_Rd();

}

void C2Compute_Y()
{
       int i, j, t;

       for(i=0;i<si;i++) {
            for(j=0;j<sj;j++) {
                 for(t=0;t<st;t++) {
                       if (case2X[i][j][t]>0) case2Y[i][j][t]=1;
                       else case2Y[i][j][t]=0;
                 }
            }
      }
} // Compute_Y

Boolean C2Feas1()
{
      float sum=0;
      Boolean retval=False;
      for(int t=0;t<st;t++)
            for(int j=0;j<sj;j++) sum += (m * p * case2DS[j][t]);
                  if (sum>TP) retval=True;

      return retval;
}

Boolean C2Feas2()
{
      Boolean retval=True;
      for (int j=0; j<sj; j++)
      {
            feas[j]=1;
            delta_costs[j] = case2z[j] - Alpha * TC[j];
            if (delta_costs[j] > 0)
                  {
                  feas[j]=0;  //check for which buyers the second
constraint is infeasible
                  retval=False;
                  }
      }

return retval;
}



void C2Compute_Rd()
{
   float M1, M2;
   M1=1000; M2=1000;
```

```
      int i, j, t;
//    float sum1, sum2, sum3=0;
//    float sum1, sum2, sum3, dum;
      case2sum3=0;
      for(j=0;j<sj;j++)
            {
            case2sum1=0; case2sum2=0;
            for(t=0;t<st;t++)
                  {
                        for(i=0;i<si;i++)
                        {
                        case2sum1 +=  s[i][j][t] * case2Y[i][j][t]
                                    + c[i][j][t] * case2X[i][j][t]
                                    + h[i][j][t] * case2IP[i][j][t+1]
                                    + Pie[i][j][t] * case2IN[i][j][t+1];
 //                       // cout << "case2Sum1= " << case2sum1 << endl;
                        }
                  sum2 += p * DS[j][t] + hm[j][t] * case2IS[j][t+1];

                  sum3 += p * case2DS[j][t];
//                sum3 += p * 0.2 * DS[j][t];

//                // cout << "Sum2= " << sum2 << endl;
//                // cout << "Sum3= " << sum3 << endl;

                  }
            case2z[j] = case2sum1 + case2sum2;
            }
      case2Rd = sum3;
   case2dum=sum3;    // *m silindi !!!!!!
      if (TP>case2dum) case2Rd -= M1*(TP-case2dum);      //
      for (j=0;j<sj;j++)
      if (case2z[j]>(Alpha*TC[j]))
      case2Rd -= M2 * (case2z[j]-Alpha*TC[j]);
}//void Compute_Rd()


void C2Compute_I()
{
      int i, j, t;
      float case2sum;
      for(i=0;i<si;i++) {
            for(j=0;j<sj;j++) {
                  for(t=0;t<st;t++) {
                        case2sum = case2IP[i][j][t] - case2IN[i][j][t] +
case2X[i][j][t] - d[i][j][t];
                        if (case2sum>0) {
                              case2IP[i][j][t+1] = case2sum;
                              case2IN[i][j][t+1] = 0;
                        }
                        else {
                              case2IN[i][j][t+1] = -case2sum;
                              case2IP[i][j][t+1] = 0;
                        }
                  }
            }
      }
} //Compute_I

void C2Check_Feasibility()
```

```
{
////p//fout1<<  " case2Check feasibility "<<endl ;

      int i, j, t;
      int ind;
      int *index;
      index = new int[si];
      for(i=0;i<si;i++) index[i]=0;

      for(j=0;j<sj;j++) {
            for(t=0;t<st;t++) {
                  while (IS[j][t+1]<0) {
//    //p//fout1<<  "IS["<< j<< "][" << (t+1) << "] = " <<
"\t"<<case2IS[j][t+1]<<endl ;

                        // kac tane i kullandigina gore durdurabiliriz.
                        ind=mini(Pie, index, j, t);
//    //p//fout1<< "min pi index i = " << ind <<endl ;
                        if (case2X[ind][j][t] > (fabs(case2IS[j][t+1]) /
a[ind][j]))
                              {
//                //p//fout1<<"case2X[ind][j][t] > (fabs(IS[j][t+1]) /
a[ind][j])" <<endl ;
               case2X[ind][j][t] -= (fabs(case2IS[j][t+1]) / a[ind][j]);
//                //p//fout1 << "decrease by "
<<"\t"<<(fabs(case2IS[j][t+1]) / a[ind][j])<< endl;
//                //p//fout1<<  "case2X["<< ind << "][" << j << "][" << t
<< "] = " << "\t"<<case2X[ind][j][t] <<endl ;
                        index[ind]=0;
                              }
                        else
            {
//            //p//fout1<<"case2X[ind][j][t] < (fabs(case2IS[j][t+1]) /
a[ind][j])" <<endl ;
            case2X[ind][j][t]=0;
//            //p//fout1<<  "case2X["<< ind << "][" << j << "][" <<  t <<
"] = " <<"\t"<<case2X[ind][j][t] <<endl ;
            }

                        C2Compute_IS();

                  };
                  for(i=0;i<si;i++) index[i]=0;
            }
      }

} //Check_Feasibility


void C2Compute_IS()
{
//   //p//fout1<<"Compute_IS()//"<<endl ;
      int i, j, t;
      float sum;
      for(j=0;j<sj;j++)
            for(t=0;t<st;t++)
      {
                  sum=0;
                  for(i=0;i<si;i++) sum +=a[i][j] * case2X[i][j][t];
                  case2IS[j][t+1] = case2IS[j][t] + case2DS[j][t] - sum;
```

```
       //          //p//fout1<<   " case2 IS["<<j<<"]["<<(t+1)<<"] =
"<<"\t"<<case2IS[j][t+1]<<endl ;
            }

//    //p//fout1<<   " ===e=n=d==c=o=m=p=u=t=e=IS====     "<<endl ;

} // Compute_IS

int max_pie(int j,int t)
{
int i;
int ei;
float elde_pi=-1;

for (i=0; i<si; i++)
if ((IN[i][j][t+1]>0)&&(Pie[i][j][t]>elde_pi))
      {
   ei=i;
   elde_pi=Pie[i][j][t];
   }
return ei;
}


IJT get_i_j_t_case2_sub1()
{
      IJT ex;
   int k;
      if (numbernegIN!=0)   k = randomn(numbernegIN);
   else k=0;
      int l=0;
      int ik=0;
      int jk=0;
      int tk=0;

      while (ik<sj)
      {
         while (jk<sj)
            {
            while (tk<st)
            {
               if (negl[ik][jk][tk]==1) ++l;
               if (l==k) break;
            negl[ik][jk][tk]=0;
               ++tk;
         }
         if (l==k) break;
         tk=0;
         ++jk;
         }
            if (l==k) break;
         jk=0;
         ++ik;
      }

   ex.s_i=ik;
      ex.s_j=jk;
      ex.s_t=tk;

--numbernegIN;
```

```
return ex;

}

void Print_Best_Soln()
{
   //p//fout1<<"\n\n\t*\t THE BEST INITIAL VALUES \t*"<<endl;
      //p//fout1<< "*sol'n with best obj. function*" << endl ;
      //p//fout1 << "Iteration Number      = " << iterationcount <<endl;
      //p//fout1 << "Feasible Sol'n #      = " << feasiblecount <<endl;
      //p//fout1 << "Rd =" << Rdgecerli<<endl<<"\tp ="<<pgecerli<< endl;
      // cout << "Rd =" << Rdgecerli <<"\tp ="<<pgecerli<< endl;
      //p//fout1 << "TP =" << TP << endl;
/*    for (int i=0; i<si; i++)
            for (int j=0; j<sj; j++)
                  for (int t=0; t<st; t++)
                  {
                  //p//fout1<< "X[" << i<< "][" << j << "][" << t <<"] =
"<< Xgecerli[i][j][t]<< endl;
                  }
      for (int j=0; j<sj; j++)
            for (int t=0; t<st; t++)
            {
            //p//fout1 << "DS[" << j << "][" << t <<"] = " <<
DSgecerli[j][t]   << endl;
            }
      for(int j=0;j<sj;j++)
            for(int t=0;t<st;t++)
                  //p//fout1<<  " IS["<<j<<"]["<<(t+1)<<"] =
"<<"\t"<<ISgecerli[j][t+1]<<endl ;

      for (int j=0; j<sj; j++)
            {
            //p//fout1 << "Z[" << j << "] " << z[j] << endl;
            }
*/
}

void printbestofall()
{
//   fout2<<"\n\n\t*\t THE BEST OF OVER ALL \t*"<<endl;
//    fout2<< "*sol'n with best obj. function over all prices*" << endl ;
      fout2 << "Iteration Number     = \t" << iterationcount <<endl;
      fout2 << "Feasible Sol'n #     = \t" << feasiblecount <<endl;
      fout2 << "Rd =\t" << Rdglob<<endl;
//    cout << "Rd =\t" << Rdglob <<"\tp =\t"<<pglob<< endl;
      fout2 << "TP =\t" << TP << endl;
   fout2 << "p =\t" << pglob<< endl;

      for (int i=0; i<si; i++)
            for (int j=0; j<sj; j++)
                  for (int t=0; t<st; t++)
                  {
                  fout2<< "X[" << i<< "][" << j << "][" << t <<"] =\t"<<
Xglob[i][j][t]<< endl;
                  }
      for (int j=0; j<sj; j++)
            for (int t=0; t<st; t++)
            {
```

```
            fout2 << "DS[" << j << "][" << t <<"] =\t" << DSglob[j][t]
<< endl;
            }
      for(int j=0;j<sj;j++)
            for(int t=0;t<st;t++)
                  fout2<<  "IS["<<j<<"]["<<(t+1)<<"] =
"<<"\t"<<ISglob[j][t+1]<<endl ;

      for (int i=0; i<si; i++)
            for (int j=0; j<sj; j++)
                  for (int t=0; t<st; t++)
                  {
                  fout2<< "IN[" << i<< "][" << j << "][" << (t+1) <<"]
=\t"<< INglob[i][j][t+1]<< endl;
                  }
      for (int i=0; i<si; i++)
            for (int j=0; j<sj; j++)
                  for (int t=0; t<st; t++)
                  {
                  fout2<< "IP[" << i<< "][" << j << "][" << (t+1) <<"]
=\t"<< IPglob[i][j][t+1]<< endl;
                  }


      for (int j=0; j<sj; j++)
            {
            fout2 << "Z[" << j << "] =\t" << zglob[j] << endl;
            }
if (runpart==2) fout2<< "\n\n CPU TIME = \t"<< (((end-
start)/CLK_TCK))<<endl;
}

void moveglobtogecerliandgecici()
{
      int j,t,i;
      Rdbest=Rdglob;
   Rd=Rdglob;
   Rdgecerli=Rdglob;

   p=pglob;
   pgecerli=pglob;

      for(j=0;j<sj;j++)
      {
            zbest[j]=zglob[j];
            for(t=0;t<st+1;t++)
         {
            DSbest[j][t]=DSglob[j][t];
         DS[j][t]=DSglob[j][t];
         DSgecerli[j][t]=DSglob[j][t];

         ISbest[j][t]=ISglob[j][t];
         IS[j][t]=ISglob[j][t];
         ISgecerli[j][t]=ISglob[j][t];

                  for(i=0;i<si;i++)
            {
               Xbest[i][j][t]=Xglob[i][j][t];
            X[i][j][t]=Xglob[i][j][t];
               Xgecerli[i][j][t]=Xglob[i][j][t];
```

```
                INNbest[i][j][t+1] = INNglob[i][j][t+1];
                INN[i][j][t+1] = INNglob[i][j][t+1];
//                INNgecerli[i][j][t+1] = INNglob[i][j][t+1];

                                IPbest[i][j][t+1] = IPglob[i][j][t+1];
                IP[i][j][t+1] = IPglob[i][j][t+1];
//                IPgecerli[i][j][t+1] = IPglob[i][j][t+1];
                }
            }
                }
}
```

APPENDIX F

SUPRIME RESULTS

TABLE F.1 Suprime for 10 buyers without setup

					SUPRIME					GAMS Solutions		
					Solution Improving		Intensification			Intensification		
					OF	price	OF	price	CPU	OF	price	Δ %
0	101	11	1	n	44512.000	8.000	44512.000	8.000	46.822	44512.000	8.000	0.000
0	101	11	1	s	41952.000	8.000	41952.000	8.000	39.953	41952.000	8.000	0.000
0	101	11	2	n	44512.000	8.000	44512.000	8.000	40.853	44512.000	8.000	0.000
0	101	11	2	s	41952.000	8.000	41952.000	8.000	38.801	41952.000	8.000	0.000
0	101	12	1	n	44512.000	8.000	44512.000	8.000	52.201	44512.000	8.000	0.000
0	101	12	1	s	41952.000	8.000	41952.000	8.000	54.576	41952.000	8.000	0.000
0	101	12	2	n	44512.000	8.000	44512.000	8.000	47.269	44512.000	8.000	0.000
0	101	12	2	s	41952.000	8.000	41952.000	8.000	46.736	41952.000	8.000	0.000
0	101	21	1	n	16305.100	3.101	16305.100	3.101	120.154	16341.039	3.108	0.002
0	101	21	1	s	32787.400	6.544	33628.800	6.704	259.387	40220.129	8.000	0.164
0	101	21	2	n	16106.300	3.063	16106.300	3.063	10.441	16341.039	3.108	0.014
0	101	21	2	s	34459.400	6.858	36719.900	7.341	248.837	40351.689	8.000	0.090
0	101	22	1	n	20646.900	3.941	21097.400	4.042	15.160	21885.661	4.062	0.036
0	101	22	1	s	39848.300	7.906	39872.000	7.906	19.089	41487.270	8.000	0.039
0	101	22	2	n	21024.500	4.058	21121.400	4.043	14.760	21885.661	4.062	0.035
0	101	22	2	s	39269.000	7.951	39622.900	7.974	18.938	41848.761	8.000	0.053
0	102	11	1	n	91568.000	8.000	91568.000	8.000	78.674	91776.000	8.000	0.002
0	102	11	1	s	94691.500	7.987	94699.500	7.987	87.751	95267.670	8.000	0.006
0	102	11	2	n	91776.000	8.000	91776.000	8.000	95.275	91776.000	8.000	0.000
0	102	11	2	s	95112.000	8.000	95112.000	8.000	78.480	95424.000	8.000	0.003
0	102	12	1	n	91472.000	8.000	91472.000	8.000	69.375	91776.000	8.000	0.003
0	102	12	1	s	94736.000	8.000	94736.000	8.000	61.513	95267.670	8.000	0.006
0	102	12	2	n	91472.000	8.000	91472.000	8.000	72.755	91776.000	8.000	0.003
0	102	12	2	s	94768.000	8.000	94776.000	8.000	60.769	95424.000	8.000	0.007
0	102	21	1	n	51344.800	5.821	51344.800	5.821	3422.120	71755.378	7.257	0.284
0	102	21	1	s	43580.100	4.888	43770.700	4.888	2969.000	56860.200	5.349	0.230
0	102	21	2	n	50875.300	5.665	50943.300	5.665	475.265	72960.461	7.215	0.302
0	102	21	2	s	42866.300	4.517	42866.300	4.517	582.485	57422.364	5.194	0.253
0	102	22	1	n	-992568.000	2.674	23060.200	2.546	4638.400	30183.102	2.822	0.236
0	102	22	1	s	25454.600	2.841	25454.600	2.841	740.731	39539.093	3.623	0.356
0	102	22	2	n	-1334400.000	2.419	22052.600	2.458	4659.530	30507.952	2.790	0.277
0	102	22	2	s	-127902.000	2.098	22724.600	2.493	4724.500	39995.663	3.574	0.432
0	103	11	1	n	104418.000	6.783	109670.000	7.227	96.366	117540.000	7.950	0.067
0	103	11	1	s	116788.000	7.802	119001.000	7.987	104.253	114040.000	8.000	0.044
0	103	11	2	n	106103.000	6.762	109009.000	6.953	1170.160	119200.000	8.000	0.085
0	103	11	2	s	120696.000	8.000	120907.000	7.979	104.218	114050.000	8.000	0.060
0	103	12	1	n	90248.900	5.646	90248.900	5.646	95.742	92266.567	6.109	0.022
0	103	12	1	s	119525.000	7.942	119882.000	7.957	95.402	113890.000	8.000	0.053
0	103	12	2	n	90347.800	5.691	90950.500	5.710	93.212	93879.097	6.216	0.031
0	103	12	2	s	119311.000	7.975	120201.000	7.992	95.979	113970.000		0.055
0	103	21	1	n	52077.300	4.201	52077.300	4.201	3807.660	72757.037	5.137	0.284
0	103	21	1	s	94904.200	7.639	94904.200	7.639	1122.170	112200.000	8.000	0.154
0	103	21	2	n	53959.900	4.237	53959.900	4.237	4089.780	77093.821	5.316	0.300
0	103	21	2	s	88126.800	6.964	94129.400	7.465	1133.780	114150.000	8.000	0.175
0	103	22	1	n	68664.500	5.399	68772.500	5.399	900.622	83427.139	5.912	0.176
0	103	22	1	s	92235.300	7.417	98109.700	7.986	1237.340	112010.000	8.000	0.124
0	103	22	2	n	70281.200	5.489	70819.100	5.489	1111.250	88541.349	6.191	0.200
0	103	22	2	s	93341.400	7.298	98333.600	7.797	1202.860	113440.000	8.000	0.133

TABLE F.2 Suprime for 3 buyers without setup

					SUPRIME					GAMS Solutions		
					Solution Improving		Intensification			Intensification		
					OF	price	OF	price	CPU	OF	price	Δ %
0	131	11	1	n	13184.000	8.000	13184.000	8.000	18.531	13184.000	8.000	0.000
0	131	11	1	s	11036.700	7.998	11036.700	7.998	20.885	11040.000	8.000	0.000
0	131	11	2	n	13184.000	8.000	13184.000	8.000	18.879	13184.000	8.000	0.000
0	131	11	2	s	11040.000	8.000	11040.000	8.000	26.095	11040.000	8.000	0.000
0	131	12	1	n	13184.000	8.000	13184.000	8.000	20.926	13184.000	8.000	0.000
0	131	12	1	s	11040.000	8.000	11040.000	8.000	23.275	11040.000	8.000	0.000
0	131	12	2	n	13184.000	8.000	13184.000	8.000	22.138	13184.000	8.000	0.000
0	131	12	2	s	11026.800	7.990	11037.200	7.998	23.547	11040.000	8.000	0.000
0	131	21	1	n	5145.460	3.723	5145.460	3.723	5.396	5147.245	3.724	0.000
0	131	21	1	s	8487.550	7.230	8612.060	7.280	23.541	9471.333	8.000	0.091
0	131	21	2	n	5138.000	3.718	5138.000	3.718	11.862	5147.245	3.724	0.002
0	131	21	2	s	8788.770	7.480	8870.070	7.555	33.849	9497.360	8.000	0.066
0	131	22	1	n	12425.600	7.986	12768.500	7.926	3.611	13078.184	8.000	0.024
0	131	22	1	s	9635.510	7.101	9674.190	7.517	33.675	10909.096	8.000	0.113
0	131	22	2	n	12402.100	7.849	12494.500	7.903	3.470	13184.000	8.000	0.052
0	131	22	2	s	9761.570	7.074	10215.300	7.402	38.457	11040.000	8.000	0.075
0	132	11	1	n	26176.000	8.000	26176.000	8.000	20.738	26176.000	8.000	0.000
0	132	11	1	s	27840.000	8.000	27840.000	8.000	14.721	27840.000	8.000	0.000
0	132	11	2	n	26176.000	8.000	26176.000	8.000	22.413	26176.000	8.000	0.000
0	132	11	2	s	27840.000	8.000	27840.000	8.000	23.549	27840.000	8.000	0.000
0	132	12	1	n	25880.000	8.000	26029.600	7.999	18.836	26176.000	8.000	0.006
0	132	12	1	s	27272.000	8.000	27304.000	8.000	14.906	27840.000	8.000	0.019
0	132	12	2	n	25880.000	8.000	25958.000	7.970	20.656	26176.000	8.000	0.008
0	132	12	2	s	27453.100	7.974	27453.100	7.974	14.533	27840.000	8.000	0.014
0	132	21	1	n	17994.800	7.029	18533.100	6.965	1325.210	24360.133	8.000	0.239
0	132	21	1	s	-352107.000	2.466	6209.760	2.107	1427.830	8461.557	2.842	0.266
0	132	21	2	n	19223.400	6.965	19223.400	6.965	1171.750	24689.550	8.000	0.221
0	132	21	2	s	6408.020	2.200	6408.020	2.200	297.947	10157.378	3.382	0.369
0	132	22	1	n	20958.400	7.999	20990.400	7.999	742.701	24855.843	8.000	0.156
0	132	22	1	s	9060.810	3.408	9081.250	3.408	227.510	12972.122	4.162	0.300
0	132	22	2	n	21021.100	7.993	21089.600	7.860	782.268	25697.964	8.000	0.179
0	132	22	2	s	10044.300	3.826	10193.500	3.826	469.214	14798.002	4.732	0.311
0	133	11	1	n	33314.300	7.315	33497.300	7.416	16.201	36790.261	8.000	0.090
0	133	11	1	s	33048.000	8.000	33056.000	8.000	17.312	33611.839	8.000	0.017
0	133	11	2	n	34007.900	7.527	34156.400	7.552	15.600	37427.519	8.000	0.087
0	133	11	2	s	33264.000	8.000	33264.000	8.000	16.651	34230.639	8.000	0.028
0	133	12	1	n	33399.400	7.504	33613.900	7.508	11.565	36790.261	8.000	0.086
0	133	12	1	s	32664.000	8.000	32732.300	7.976	14.657	33611.839	8.000	0.026
0	133	12	2	n	33340.300	7.435	33579.200	7.457	13.150	37126.348	8.000	0.096
0	133	12	2	s	33000.000	8.000	33016.000	8.000	15.731	34161.875	8.000	0.034
0	133	21	1	n	22368.100	5.852	22368.100	5.852	1536.250	30121.200	7.080	0.257
0	133	21	1	s	31070.000	7.989	31094.000	7.989	179.569	34313.722	8.000	0.094
0	133	21	2	n	23533.900	5.726	23533.900	5.726	1582.720	32481.634	7.443	0.275
0	133	21	2	s	33363.300	7.955	33363.300	7.955	13.149	34926.377	8.000	0.045
0	133	22	1	n	10650.400	2.683	10650.400	2.683	349.399	18144.255	4.040	0.413
0	133	22	1	s	28652.300	7.986	28871.700	7.945	188.301	34172.363	8.000	0.155
0	133	22	2	n	10925.700	2.813	10925.700	2.813	379.131	20054.635	4.358	0.455
0	133	22	2	s	29172.800	7.949	29180.800	7.949	183.426	34712.558	8.000	0.159

TABLE F.3 Suprime for 10 buyers with setup

					SUPRIME				
					Solution Improving		Intensification		
					OF	price	OF	price	CPU
1	101	11	1	n	44512.000	8.000	44512.000	8.000	47.620
1	101	11	1	s	41952.000	8.000	41952.000	8.000	41.397
1	101	11	2	n	44512.000	8.000	44512.000	8.000	40.392
1	101	11	2	s	41952.000	8.000	41952.000	8.000	38.644
1	101	12	1	n	44512.000	8.000	44512.000	8.000	49.617
1	101	12	1	s	41637.400	7.940	41922.700	7.994	53.021
1	101	12	2	n	44512.000	8.000	44512.000	8.000	46.119
1	101	12	2	s	41952.000	8.000	41952.000	8.000	45.632
1	101	21	1	n	25153.000	4.944	26748.500	5.409	73.505
1	101	21	1	s	31312.900	6.396	32392.800	6.700	25.228
1	101	21	2	n	29529.800	5.681	30320.300	5.977	28.397
1	101	21	2	s	30398.600	6.260	32631.400	6.749	47.144
1	101	22	1	n	-3373710.000	2.158	10807.200	2.079	115.588
1	101	22	1	s	-441717.000	2.291	14905.100	3.004	125.416
1	101	22	2	n	-3778910.000	2.607	13379.700	2.607	86.746
1	101	22	2	s	-85513.400	2.035	13521.600	2.775	98.379
1	102	11	1	n	91288.000	8.000	91456.000	8.000	57.876
1	102	11	1	s	94288.000	8.000	94288.000	8.000	54.553
1	102	11	2	n	91392.000	8.000	91392.000	8.000	64.657
1	102	11	2	s	94368.000	8.000	94368.000	8.000	59.231
1	102	12	1	n	91000.000	8.000	91000.000	8.000	60.224
1	102	12	1	s	94152.000	8.000	94152.000	8.000	55.414
1	102	12	2	n	91079.000	7.996	91079.000	7.996	59.137
1	102	12	2	s	94248.000	8.000	94280.000	8.000	56.454
1	102	21	1	n	39828.900	4.443	40330.900	4.443	241.859
1	102	21	1	s	45673.400	4.881	46112.700	4.881	248.906
1	102	21	2	n	48795.000	5.295	49160.400	5.295	250.862
1	102	21	2	s	51431.800	5.589	51431.800	5.589	291.862
1	102	22	1	n	88511.000	7.990	88798.900	7.993	248.747
1	102	22	1	s	91712.000	8.000	91728.000	8.000	256.841
1	102	22	2	n	88931.900	7.991	88947.800	7.991	256.952
1	102	22	2	s	91512.000	8.000	91533.900	7.989	254.440
1	103	11	1	n	92056.500	5.843	92085.700	5.843	104.833
1	103	11	1	s	117023.000	7.860	117046.000	7.860	105.502
1	103	11	2	n	84534.300	5.291	86690.200	5.408	106.769
1	103	11	2	s	118373.000	7.947	119561.000	7.995	105.256
1	103	12	1	n	83072.100	5.235	83357.400	5.285	89.712
1	103	12	1	s	116914.000	7.912	118208.000	7.996	96.971
1	103	12	2	n	82212.500	5.168	83521.900	5.276	95.698
1	103	12	2	s	117976.000	7.983	118274.000	7.950	94.450
1	103	21	1	n	36289.500	3.035	36289.500	3.035	35.851
1	103	21	1	s	70291.800	5.953	70541.800	5.953	51.753
1	103	21	2	n	40687.500	3.488	41127.000	3.488	36.877
1	103	21	2	s	64255.400	5.311	64494.400	5.311	57.969
1	103	22	1	n	68748.800	5.882	68748.800	5.882	46.987
1	103	22	1	s	76436.900	6.347	76563.900	6.347	58.010
1	103	22	2	n	61009.000	5.155	61194.600	5.155	42.552
1	103	22	2	s	68502.200	5.649	68733.800	5.649	59.416

TABLE F.4 Suprime for 3 buyers with setup

					SUPRIME				
					Solution Improving		Intensification		
					OF	price	OF	price	CPU
1	131	11	1	n	13182.000	7.999	13182.000	7.999	19.214
1	131	11	1	s	11040.000	8.000	11040.000	8.000	21.053
1	131	11	2	n	13155.300	7.983	13177.000	7.996	18.671
1	131	11	2	s	11016.800	7.983	11037.400	7.998	19.360
1	131	12	1	n	13184.000	8.000	13184.000	8.000	20.535
1	131	12	1	s	10993.600	7.966	11015.800	7.982	22.263
1	131	12	2	n	13184.000	8.000	13184.000	8.000	21.128
1	131	12	2	s	11040.000	8.000	11040.000	8.000	21.535
1	131	21	1	n	7981.720	6.489	8978.480	6.725	3.212
1	131	21	1	s	8245.850	7.060	9025.270	7.490	6.310
1	131	21	2	n	8255.060	6.034	8309.850	6.074	3.399
1	131	21	2	s	8438.420	6.115	8438.420	6.115	5.759
1	131	22	1	n	10552.300	7.092	10553.000	7.092	2.575
1	131	22	1	s	5588.600	4.435	5976.620	4.743	4.421
1	131	22	2	n	10205.000	6.858	10440.500	7.016	2.626
1	131	22	2	s	7018.370	5.628	7018.370	5.628	2.124
1	132	11	1	n	26176.000	8.000	26176.000	8.000	19.439
1	132	11	1	s	27840.000	8.000	27840.000	8.000	14.420
1	132	11	2	n	26176.000	8.000	26176.000	8.000	22.813
1	132	11	2	s	27840.000	8.000	27840.000	8.000	29.596
1	132	12	1	n	25962.200	7.998	25977.300	7.986	23.344
1	132	12	1	s	27248.000	8.000	27248.000	8.000	17.024
1	132	12	2	n	26032.000	8.000	26032.000	8.000	25.732
1	132	12	2	s	27312.000	8.000	27312.000	8.000	18.678
1	132	21	1	n	21887.600	7.927	21954.300	7.983	9.844
1	132	21	1	s	15185.500	5.560	16285.900	6.104	9.627
1	132	21	2	n	23454.500	7.900	23478.200	7.900	8.723
1	132	21	2	s	15591.600	5.178	15627.800	5.178	8.593
1	132	22	1	n	20852.100	7.983	20993.800	7.958	6.538
1	132	22	1	s	20616.000	8.000	20622.700	7.941	5.212
1	132	22	2	n	20635.400	7.998	21132.100	7.947	6.099
1	132	22	2	s	20533.800	7.922	20804.300	7.968	5.254
1	133	11	1	n	29612.700	6.459	29777.600	6.641	18.669
1	133	11	1	s	32777.000	7.973	32777.000	7.973	21.037
1	133	11	2	n	30678.100	6.607	30678.100	6.607	20.820
1	133	11	2	s	33365.400	7.933	33365.400	7.933	19.385
1	133	12	1	n	29605.700	6.647	30231.400	6.792	16.580
1	133	12	1	s	32584.000	8.000	32584.000	8.000	16.667
1	133	12	2	n	29944.100	6.657	30038.700	6.767	13.877
1	133	12	2	s	32640.000	8.000	32728.000	8.000	16.302
1	133	21	1	n	16076.700	4.237	16076.700	4.237	4.950
1	133	21	1	s	21698.400	5.838	21803.400	5.838	13.633
1	133	21	2	n	12748.400	3.442	12748.400	3.442	4.448
1	133	21	2	s	22392.300	5.545	22392.300	5.545	13.318
1	133	22	1	n	12849.300	3.289	12849.300	3.289	3.816
1	133	22	1	s	27879.100	7.920	28745.300	7.976	9.762
1	133	22	2	n	10232.400	2.621	10232.400	2.621	3.730
1	133	22	2	s	27829.700	7.848	28543.900	7.995	10.642

REFERENCES

1. Hax, A.C., and, D. Candea, "Hierarchical Production Planning Systems," *Production and Inventory Management*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984.

2. Schneeweiss, C., "Hierarchical Structures in Organizations: A Conceptual Framework," *European Journal of Operational Research*, Vol. 86, pp. 4-31, 1995.

3. Dempster, M. A. H., M. L. Fisher, L.Jansen, B.J. Lageweg, J.K. Lenstra, and A.H. G. Rinnoy Kan, "Analytical Evaluation of Hierarchical Planning Systems," *Operations Research,* Vol. 29, pp. 707-716, 1981.

4. Simchi-Levi, D., "Hierarchical Planning for Probabilistic Distribution Systems in Euclidean Spaces," *Management Science*, Vol. 38, No.2, pp. 198-211, Feb. 1992.

5. Günther, H. O., "The Design of an Hierarchical Model for Production Planning and Scheduling," Research Paper, Universität Mannheim, pp. 227-260, 1992.

6. Saad G. H., "Hierarchical Production Planning Systems: Extensions and Modification," *Journal of the Operational Research Society*, Vol. 41, No.7, pp. 609-624, 1990.

7. Kistner, K.P., and M. Steven, "Applications of Operations Research in Hierarchical Production Planning" in S. Axsäter, C. Schneeweiss and E. A. Silver (Eds.), *Multi-Stage Production Planning and Inventory Control*, pp. 97-113, Springer-Verlag, Berlin, 1986.

8. Erschler, J., G. Fontan, and C. Merce, "Consistency of the Disaggregation Process in Hierarchical Planning", *Operations Research*, Vol. 34, pp. 464-469, 1986.

9. Tsubone, H, H. Matsuura and Tsutsu, "Hierarchical Production Planning System for a Two-Stage Process" *International Journal of Production Research*, Vol. 29, pp. 769-785, 1991.

10. Glover, G., G. Jones, D. Karney, D. Klingman, and J. Mote, "An Integrated Production, Distribution and Inventory Planning System," *Interfaces*, Vol. 9, No. 5 pp. 21-35, 1979.

11. Bloemhof-Ruward, J. M., M. Salomon, and L. N. Van Wassenhove, "On the Coordination of Product and By-Product Flows in Two-Level Distribution Networks: Model Formulations and Solution Procedures," *European Journal of Operational Research,* Vol. 79, pp. 325-339, 1994.

12. Iyogun, P., "Lot-sizing Algorithm for a Coordinated Multi-Item, Multi-Source Distribution Problem," *European Journal of Operational Research*, Vol. 59, No. 3, pp.503-517, 1992.

13. Slats, P.A., B. Bhola, J.J.M. Evers, and G. Dijkhuizen, "Logistic Chain Modelling," *European Journal of Operational Research*, Vol. 87, pp. 1-20, 1995.

14. Nam, K., A. Chaudhury and H.R. Rao, "A Mixed Integer Model of Bidding Strategies for Outsourcing," *European Journal of Operational Research*, Vol. 87, pp. 257-273, 1995.

15. Williams, B. R., "Advanced Supplier Partnership Practices: A case Study," *APICS*, F-9, pp. 277-279, 1998.

16. Vidal, C.J. and M. Goetschalckx, "Strategic Production-Distribution Models: A Critical Review with Emphasis on Global Supply Chain Models," *European Journal of Operational Research*, Vol. 98, pp. 1-18, 1997.

17. Thomas, D.J. and P. M. Griffin, "Coordinated Supply Chain Management," *European Journal of Operational Research*, Vol. 94, pp. 1-15, 1996.

18. Maloni, M.J., and W.C. Benton, "Supply Chain Partnership: Opportunities for Operations Research," *European Journal of Operational Research*, Vol. 101, pp. 419 - 429, 1997.

19. Petrovic, D., R. Roy, and R. Petrovic, "Modeling and Simulation of Supply Chain in an Uncertain Environment," *European Journal of Operational Research*, Vol. 109, pp. 299 -3099, 1998.

20. Desirey, S. T. and S. Özatalay, "Rough-cut Value Management: Maximizing Business Value with Supply Capabilities," *APICS Proceedings'98*, Nashville, pp. 359-362, 1998.

21. Bassok,Y., A. Bixby, R. Srınıvasan, and H.Z. Wiesel, "Design of Component-Supply Contract with Commitment Revision Flexibility," *International Business Machines Journals*, Vol. 41, No. 6, pp. 693-703, 1997

22. Ng, J.K.C, W.H. Ip and T.C. Lee, "The Development of an Enterprise Resources Planning System Using a Hierarchical Design Pyramid," *Journal of Intelligent Manufacturing*, 9:5, pp. 385-399, Oct. 1998.

23. Maione G. and G. Piscitelli, "Object-oriented Design of the Control Software for a Flexible Manufacturing System" *International Journal of Computer Integrated Manufacturing*, Vol. 12, pp. 1-14, Jan.-Feb., 1999.

24. Saaty, T. L. and L. G. Vargas, *The Logic of Priorities,* Kluwer-Nijhoff Publishing, 1982.

25. Weber, C. A., Current, J. R., and W.C. Benton, "Vendor Selection Criteria and Methods", European Journal of Operational Research, Vol. 50, No.1, pp. 2-18, 1991.

26. Barbarosoğlu, G., and T. Yazgaç, "An Application of the Analytic Hierarchy Process to the Supplier Selection Problem," *Production and Inventory Management Journal*, Vol. 35, No. 2, pp.14-21, First Quarter, 1997.

27. Diakoukaki, D., G. Mavrotas, and L. Papayannakis, "Determining Objective Weights in Multiple Criteria Problems: The Critic Method," *Computers and Operations Research*, Vol. 22, No. 7, pp. 763-770, 1995.

28. Glover, F., "Future Paths for Integer Programming and Links to Artificial Intelligence," *Computers and Operations Research*, 1986.

29. Hansen, P., "The Steepest Ascent Mildest Descent Heuristic For Combinatorial Programming," *Congress on Material Methods in Combinatorial Optimization*, Capri, Italy, 1986.

30. Osman, I. H., "Heuristics for Combinatorial Optimization Problems: Developments and New Direction," *Proceedings of the Second Conference on Information Technology and Its Applications, ITA '92*, pp. 1-25, Swansea, U.K., 1993.

31. Glover, F., G.A. Kochenberger and B. Alidaee, "Adaptive Memory Tabu Search for Binary Quadratic Programs," *Management Science*, 44:3, pp. 336-345, Mar. 1998.

32. Chiang, W., J. Fitzsimmons, Z. Huang, S. X. Li, "A Game-Theoretic Approach to Quantity Discount Problems," *Decision Sciences*, Vol. 25, No. 1, pp. 153-169, Nov. 1993.

33. Leavy, B., "Two Strategic Perspectives on the Buyer-Supplier Relationship," *Production and Inventory Management Journal*, Vol. 35, No. 2, pp.47-51, Second Quarter, 1994.

34. Abad, P.L., "Supplier Pricing When The Buyer's Annual Requirements Are Fixed," *Computers and Operations Research*, Vol. 21, No. 2, pp. 155-167, 1994.

35. Li, S.X. and Z. Huang, "Managing Buyer-Seller System Cooperation with Quantity Discount Considerations," *Computers Operational Research*, Vol. 22, No. 9, pp. 947-958, 1995.

36. Shinn, W.S., H. Hwang, and S.S. Park, "Joint Price and Lot Size Determination Under Conditions of Permissible Delay in Payments and Quantity Discount for Freight Cost," *European Journal of Operational Research*, Vol. 91, pp.528-542, 1996.

37. Huang, Z., S. Li and A. Ashley "Seller-Buyer System Co-operation in a Monopolistic Market," *Journal of the Operational Research Society,* Vol. 46, pp. 1456-1470, 1995.

38. Weng, Z. K, "Modeling Quantity Discounts under General Price-Sensitive Demand Functions: Optimal Policies and Relationships," *European Journal of Operational Research,* 86, pp. 300-314, 1995.

39. Weng, Z.K., and R. T. Wong, "General Models for The Supplier's All-Unit Quantity Discount policy," *Naval Research Logistics*, Vol. 40, pp. 971-991, 1993.

40. Weng, Z. K, "Pricing and Ordering Strategies in Manufacturing and Distribution Alliances," *IIE Transactions,* 29, pp. 681-692, 1997.

41. Sueyoshi, T., "DEA Pricing System," *Journal of the Operations Research Society of Japan,* Vol. 40, No. 2, pp. 220-235, 1997.

42. Bitran, G. R. and A. C. Hax, "Disaggregation and Resource Allocation Using Convex Knapsack Problems with Bounded Variables," *Management Science,* 27, pp. 431-441, 1981.

43. Bitran, G. R. and A. C. Hax, "On the Design of Hierarchical Production Planning Systems," *Decision Sciences,* 8, pp. 28-54, 1977.

44. Van der Heijden, M.C., "Supply Rationing in Multi-Echelon Divergent Systems," *European Journal of Operational Research*, 101, pp. 532-549, 1997.

45. Schorr J.E., *Purchasing in the 21st Century,* John Wiley & Sons, Inc., NY, 1992

46. Lawler, E.L., *Combinatorial Optimization Networks and Matroids*, Holt, Rinehart and Winston, NY, USA, 1976.

47. Nicholson, T., *Optimization in Industry*, Vol.1, Longman Press, London, 1971.

48. Metropolis, W., A. Roenbluth, M. Rosenbluth, A. Teller, E. Teller, "Equations of the State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, 21, pp. 1087-1092, 1953.

49. Holland, J.H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, US, 1975.

50. Schneeweiss, C., *Hierarchies in Distributed Decision Making*, Springer, Germany, 1999.