INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa • University of Ottawa

Constraint Based Master Scheduling Problem

Author: Charles Michael Potter

A THESIS

Submitted to the School of Graduate Studies and Research in partial fulfillment of the requirement for the degree of

Masters

In

Computer Science

Ottawa-Carleton Institute for Computer Science

Department of Computer Science

School of Information Technology and Engineering

University of Ottawa

OTTAWA, ONTARIO

Table of Contents

Acknowledgements

I would like to extend my sincere thanks to my supervisor, Dr. Sylvia Boyd, whose guidance, patience, and support was greatly appreciated. Her dedication was inspiring and respected. I would also like to thank Manugistics and specifically Ingrid Bongartz for the test data.

Finally, I would like to thank my wife Laura, without whom I could not have persevered. Her patience, love and support were crucial to the completion of this thesis.

Abstract

In the manufacturing industry, many problems pertain to part management, assembly and scheduling. Products today largely consist of a series of components which themselves can, in turn, consist of more components. The assembly of a product becomes an exercise in assembling all of the sub-components in a timely fashion. Sometimes these sub-components are available in stock, while other times, they must be manufactured using other sub-components. Thus, inventory management of these sub-components becomes crucial in the manufacturing of products on schedule.

This thesis looks at the problem of managing inventory so as to maximize a profit function associated with customer orders whose ordered parts are assembled on time. Because this problem is known to be NP-hard, we propose efficient heuristic methods for this problem. Two of these methods are implemented and tested against each other.

1 Introduction

1.1 Introduction and motivation

A common problem that is faced in many aspects of the manufacturing industry is the problem of inventory allocation, scheduling and resource management. Many companies who specialize in manufacturing products have the arduous task of managing an optimal amount of inventory in the form of parts and assemblies, while ensuring they can meet all customer demands.

What makes this problem complex is the number of factors that affect the balance between meeting production requirements and minimizing production cycles and excess inventory. These factors include: the complexity and relationships between the parts and assemblies in inventory, the amount of each, the time it takes to acquire them through assembly or purchase, the number and frequency of demand for these parts, and so forth.

There have been a number of simulation models developed which analyze various aspects of these production systems. The goal of these simulation models was to manage this balance between inventory and product delivery.

One company who is looking at this area is located in Ottawa. Manugistics is a provider of supply chain management software tools, and has worked on providing solutions to this problem that they refer to as the constraint based master scheduling problem henceforth, the CBMSP.

In this thesis, some preliminary work is reported by Boyd, Holte and Wang[1]. We attempt to extend this preliminary work and in particular provide a heuristic for solving the CBSMP. The heuristic we present is much more robust than the one mentioned by Boyd, Holte and Wang in that it effectively handles variations and certain situations that occur in the problem which were not handled in their version. One of the key components of the heuristic involves mapping portions of the part hierarchy to a Multiple Knapsack Problem(MKP). This Multiple Knapsack Problem is known to be NP-hard implying that it is very unlikely that an efficient solution to the problem can be found. Using this transformation, we can solve these sub-graphs using known heuristics for the MKP.

We also use the implementation of this heuristic on real manufacturing data to show how the heuristic offers better solutions than more greedy approaches.

1.2 Overview of currently known methods

Because of the production applicability of the CBMSP, most of the inventory allocation, scheduling and resource management research has been performed with an attempt to place the solutions into production. A literature search was conducted looking for inventory, inventory allocation, resource allocation, manufacturing, lead-time, lot sizing, supply chain management, and so forth. Most of the results of this search yielded papers in the areas of simulation, and process engineering that specifically spoke to dissimilar variations of the CBMSP defined in this paper. A brief summary of the papers reviewed follows.

In [1] the General Lot sizing and Scheduling Problem[2] (GLSP) is discussed, which studies, minimizing inventory hold costs and sequence-dependant setup costs subject to the lot sizing and scheduling of many products on a single capacitated machine. The lots are tied to customer orders. In addition, the problem introduces the concept macro and micro periods. Macro-periods refer to external fluctuating demands and inventory costs. Micro-periods refer to internal dynamics such as the changeover of system state or change of production capacity, etc. Each macro-period divides into a fixed number of distinct micro-periods of variable length.

Fleischmann and Meyr[2] provide ILP models for a series of lot sizing problems. All problems share the same objective functions that try to minimize the setup cost and inventory cost.

Mathur[3] proposes a design and operation of a simulation model to compare build and setup costs against inventory carrying costs with varying lot sizes. Many industries in manufacturing are striving to adopt a "Just-in-Time" (JIT) philosophy and the "Pull System" concept for reducing inventory-carrying costs. To that end, by reducing lot sizes, it is possible to minimize inventory build up. Although this research does not directly relate to the CBMSP which is the focus of this thesis, its efforts emphasize the fact that the manufacturing industry is striving for maintaining minimal inventory while meeting all production demands. This effort only proves to show the importance of the role the CBMSP plays in the manufacturing industry.

Drevna[4] developed simulation models to assist in the analysis of high-volume production systems. These models were used to provide results to balance material

handling resources system wide and to develop least-cost and highest-performance alternatives. The models were used to evaluate material delivery and distribution facilities used in the production of consumer products. Many companies have developed highly detailed simulation models to reflect the performance of their production systems. Drevna indicates that a number of companies have realized success in making key decisions using these models, and help analyze just-in-time and push/pull operations. Drevna was able to prove how these detailed models were able to capture system information for the total production operations.

Fu and Healy[5] conduct a periodic review of an inventory system reporting on deterministic "retrospective" and gradient based approaches for simulation. Their analysis looks at the inventory control of an item that is measured in continuous units (e.g. pounds). Once every period, the inventory is reviewed and orders are placed to replenish it as required. If the inventory on hand plus the amount on order falls below a level s, the new order is placed to replace the inventory to bring it back to an acceptable threshold. The idea behind the retrospective approach is to solve a deterministic optimization problem as if the outcomes are known in advance. The gradient based approach estimates the gradient of the performance measure of interest and adjusts the constraints according to the gradient during the evolution of the simulation. The results of their review indicates that the each approach excel depending on the problem.

Veeramani[6] examined resource management in large computer-controlled manufacturing systems constrained under auction-based shop-floor control. Using simulation, Veeramani shows how an auction-based control manufacturing system can

adapt to different system loads. An auction-based decision making scheme amongst autonomous entities in the system is proposed. Decisions are made between the entities based on information exchanged. Each work piece is bid upon by each machine on the shop floor. The work piece selects the machine with best bid to perform the task. Results show that this system creates a robust system, with good scalability.

1.3 Contributions

The main contributions of this thesis are the following:

1. A HEURISTIC FOR CBMSP. A heuristic is proposed which performs a level-wise analysis of the part graph, maximizing the profit function to deliver orders on time.

2. AN IMPLEMENTATION. An implementation of the heuristic is done, with tests documenting its effectiveness.

3. AN EXTENSION. The basic model is extended to support the concept of dynamic supply of inventory into the part graph. The corresponding transformation and heuristic for CBSMP is enhanced to accommodate dynamic supply.

1.4 Outline of thesis

The thesis is organized as follows:

- Chapter 2 presents a formulation of the model for the CBMSP. This model is defined and represented in graphical form. In addition, definitions are presented and an ILP model of the CBMSP is presented.

- Chapter 3 provides background for the flat CBMSP, and how it can be modeled as a Multiple Knapsack Problem(MKP). Background for the MKP is presented along with a heuristic for solving the problem.

- Chapter 4 formalizes the heuristic for solving the CBMSP which incorporates the heuristics described in Chapter 3. It proceeds to provide a example of its function, an interpretation of the results, as well as document aspects of its function.

- Chapter 5 describes the implementation of the heuristic documented in Chapter 4, and describes the testing performed on the data provided by Manugistics.

- Chapter 6 extends the CBMSP heuristic by introducing the concept of dynamic supply, in which inventory is supplied to the part hierarchy over time.

- Chapter 7 documents conclusions, and areas for future work.

2 Problem Formulation and Notation

2.1 Problem Definition

We are given a set of parts $j = 1,2,\ldots, J$. These parts are used to fulfill orders, or are used to assemble other parts. These parts form a hierarchy such that each assembled part j contains a list of child parts $i = 1,2,\ldots, I$ called *sub-assemblies* for j. If no sub-assemblies exist for a part j it is assumed to be purchased. For each sub-assembly part i of part j, we are given a quantity a_{ij} denoting the amount of part i needed to assemble one part j. In addition, each part in the hierarchy is given an initial inventory I_{i0}. We assume that parts that are assembled cannot be purchased.

Time is represented as equal discrete periods $t=1,2,\ldots,T$. For each part j there exists an integral positive lead time v_j. For assembled parts, the lead-time is used to indicate the number of periods needed to assemble part j once all its sub-assemblies are available. For purchased parts the lead-time is used to indicate the number of periods needed to purchase the parts and introduce them into the hierarchy, with the assumption that there is no limit on the amount which can be purchased.

There also exists a list of customer orders $k = 1, 2, \ldots, K$. Each order k requires a specified quantity o_k of a single part m_k. In addition, each customer order has a due date d_k which is the time period order k is due. Each order may have a weight p_k used to indicate a "profit" a particular order may have. The profit is used to weight customer

orders such that orders with a high "profit" have a higher priority for being delivered on time. By default, we assume $p_k = 1$.

Parts are supplied through:

- using available inventory;
- assembling parts from sub-assemblies; and
- purchasing in the cases where no sub-assemblies exist.

The objective for the CBMSP is to find a set of customer orders that can be delivered on time for which the total sum of the profits p_k is maximized. In the case where $p_k = 1$ for all k, the objective is to maximize the number of customer orders delivered on time.

The following table summarizes the notation used to represent the CBMSP:

J	Number of parts $j = 1, 2, \ldots, J$
K	Number of customer orders $k = 1, 2, \ldots, K$
T	Number of time-periods $t = 1, 2, \ldots, T$
a_{ij}	The amount of part i to produce one part j
v_j	The integral lead time to supply part j either through purchasing or assembly
I_{j0}	The initial inventory for part j at time-period 0.
d_k	The time-period for which customer order k is due

m_k	The specific part ordered by customer order k
o_k	The quantify of part m_k required for customer order k
p_k	The "profit" of supplying customer order k on time.

2.2 Graphical Representation

Based upon the problem given, the problem will be shown as a directed graph of nodes and edges. Future references in this paper will use the following graph notation.

2.2.1 Parts

The parts are depicted as circular nodes in the graph and fall into two distinct classes: those assembled from other parts, and those that are purchased. Purchased parts correspond to nodes that have no edges entering them in the graph. Refer to Figure 1.

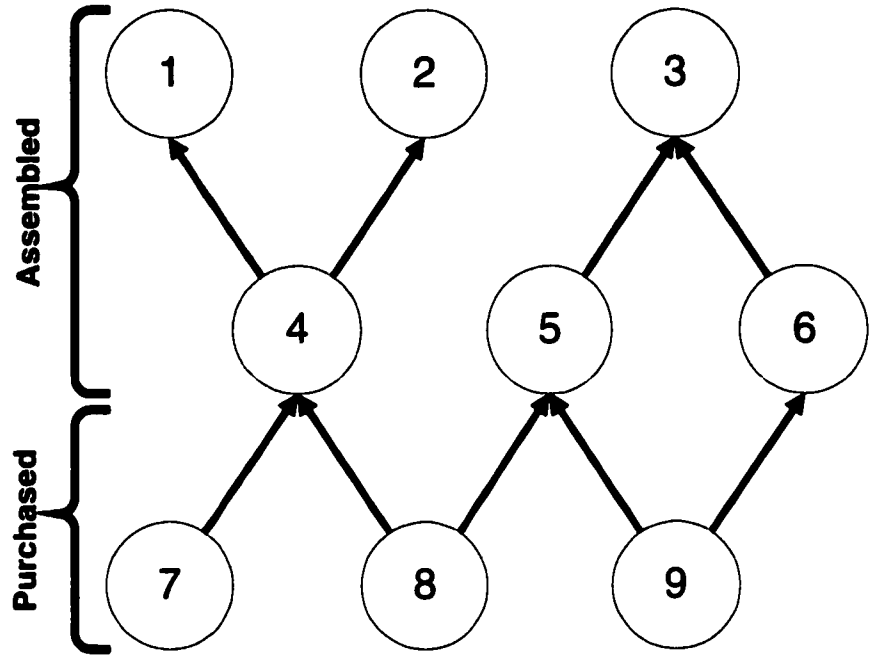


Figure 1 Part Hierarchy of Assembled and Purchased Parts

2.2.2 Orders

Orders are represented at the top of the graph as boxes. They are connected to the corresponding ordered part m_k via a dashed line. Refer to Figure 2.

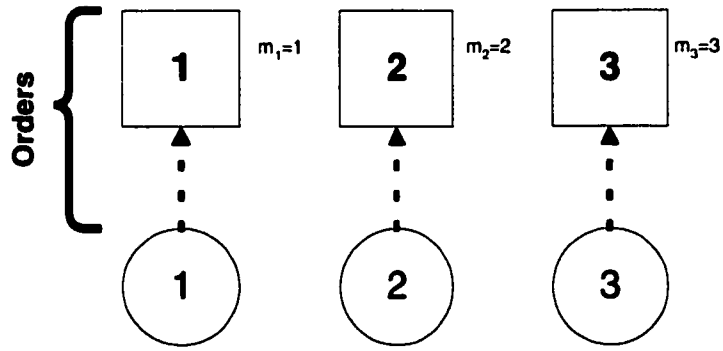


Figure 2 Orders

It is possible for multiple orders to exist for any given part.

2.2.3 Assemblies

Assemblies in the part hierarchy are shown through solid edges in the graph. The sub-assemblies for part j consist of all parts i for which there is a solid edge from i to j in the graph. The quantity a_{ij} of each part i required to assemble a part j is drawn as a weight on the edge. Refer to Figure 3.

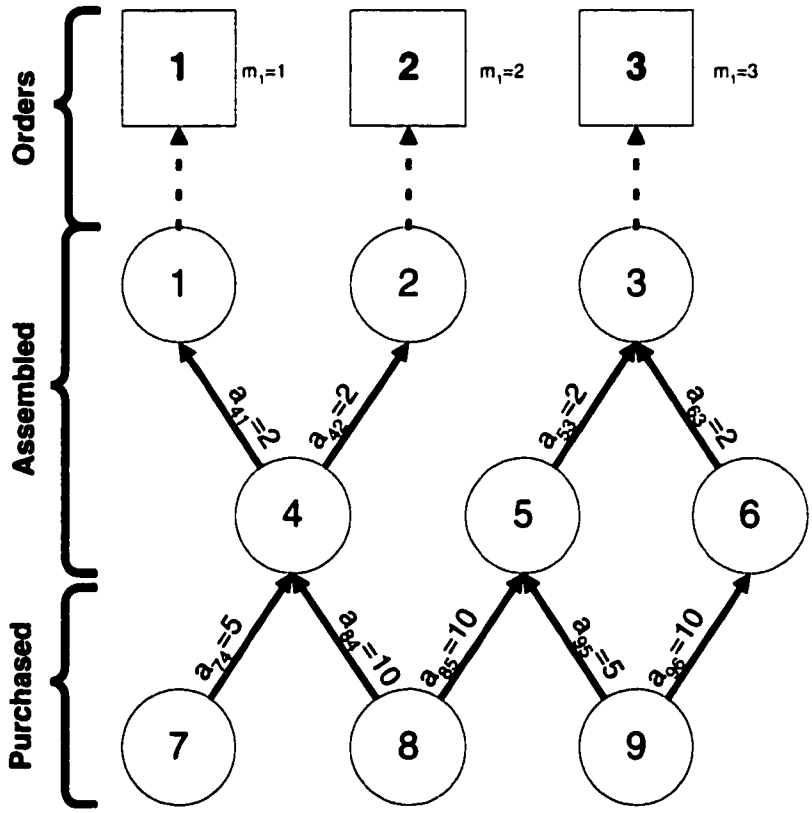


Figure 3 Assemblies

2.2.4 Inventory, Lead Time, and Demands

Each part node is decorated with the initial inventory Ij_0 and integral lead time v_j. Each customer order is decorated with the due period d_k and quantity o_k.

2.2.5 Definitions

The following definitions will be used to describe a position of a part(node) in the part hierarchy(graph).

The $\boldsymbol{depth_j}$ of a part j in the graph will be defined as the length (in edges) of the shortest path from j to any order $k \in K$.

The $\boldsymbol{distance_j}$ of a part j will be defined as the length (in edges) of the longest path from j to any order $k \in K$.

A part j resides in ***level h*** if there exists a path of length h from j to some order $k \in K$ where $depth_j \leq \boldsymbol{h} \leq distance_j$. A part can reside in more than one level.

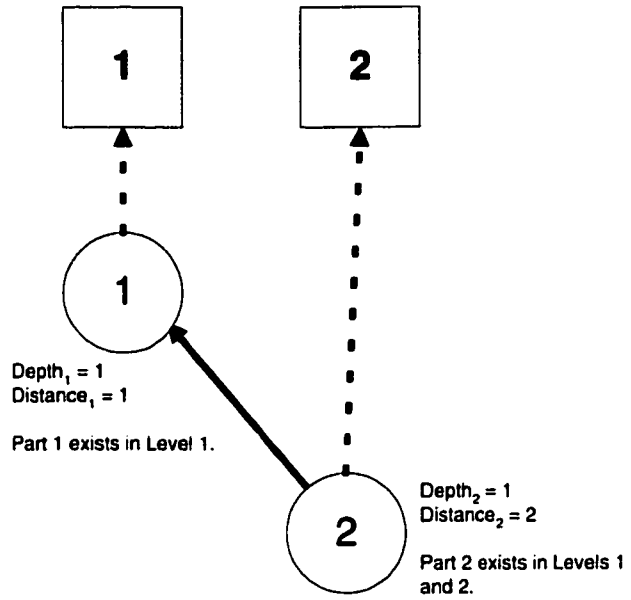


Figure 4. Example of Depth, Distance and Level

2.2.6 Example

Figure 4 depicts the part hierarchy complete with orders, parts, inventories, lead times, due periods, and assembly quantities.

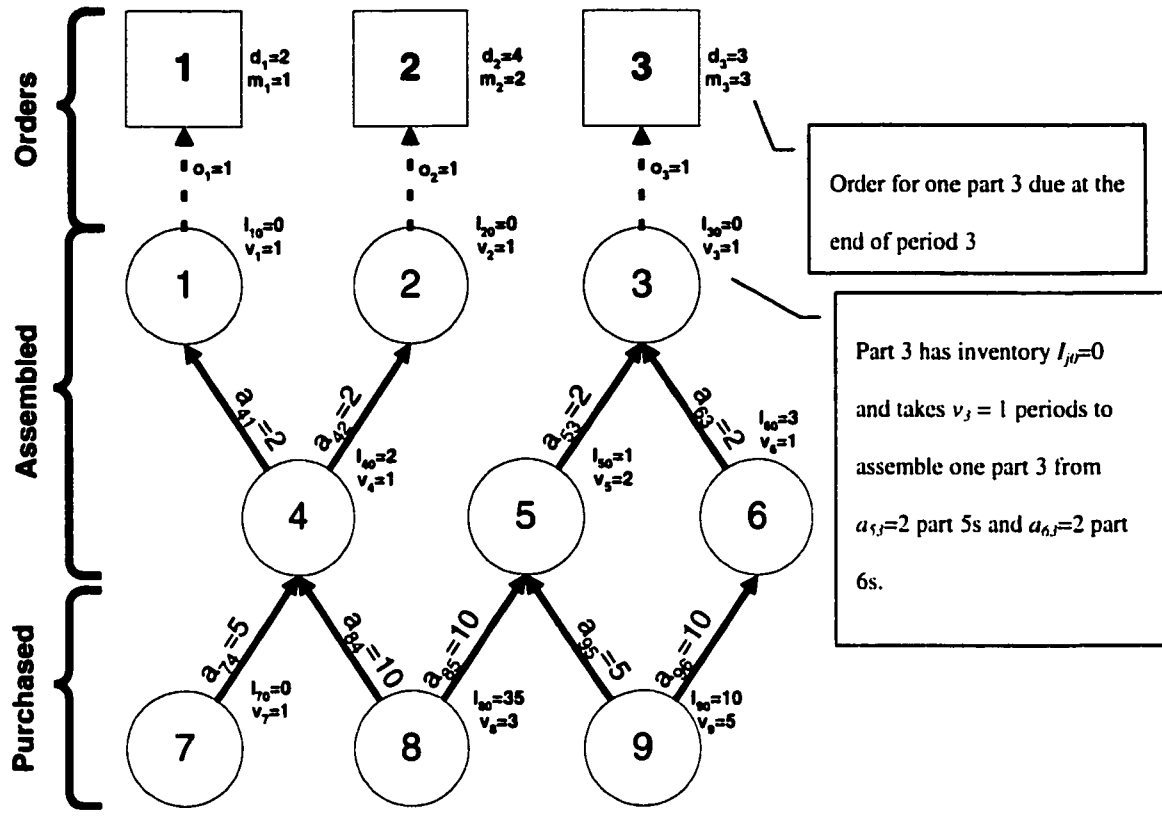


Figure 5. Example General CBMSP

2.3 ILP Model for CBMSP

Boyd, Holte and Wang[1] formulated the CBMSP as an integer linear problem. The model is formulated by taking the information for a customer order k (i.e. quantity o_k, due date d_k, and order part m_k) and using d_{jkt}. Variable d_{jkt} represents the quantity of part j required for customer order k during time-period t. From this, it is asserted that:

$$d_{jkt} = \begin{cases} o_k, \text{ if customer order k has } m_k = j, \text{ and } d_k = t; \\ 0, \text{ otherwise} \end{cases}$$

In addition, parts have two kinds of demands, external and internal. The quantity of a part directly needed for customer orders are external demands. Internal demands refer to the quantity of a part needed to assemble other parts in the hierarchy.

The complete ILP model for the CBMSP is described as follows:

J	Number of parts $j = 1, 2, ..., J$
K	Number of customer orders $k = 1, 2, ..., K$
T	Number of time-periods $t = 1, 2, ..., T$
a_{ij}	The amount of part i to produce one part j
S_j	The set of immediate descendants of part j.
v_j	The integral lead time to supply part j either through purchasing or assembly
I_{j0}	The initial inventory for part j at time-period 0.
d_{jkt}	The quantity of item j required for customer order k during time-period t
p_k	The profit of supplying customer order k on time.

Decision variables:

$$x_k = \begin{cases} 1, \text{if customer order k is delivered on time} \\ 0, \text{otherwise.} \end{cases}$$

q_{jt} = the production quantity for item j in time-period t.

I_{jt} = the inventory for item j at the end of time-period t.

The ILP Model:

$$\text{maximize} \sum_{k=1}^{K} p_k x_k$$

subject to:

$$I_{jt} = I_{j,t-1} + q_{jt} - \sum_{k=1}^{K} d_{jkt} x_k - \sum_{i \in S_j} a_{ji} q_{i,t-1+v_i}; \text{ for all } j=1,...J; t=1,...,T \quad (1)$$

$$I_{jt} \geq \sum_{i \in S_j} a_{ji} q_{i,t+v_i}; \text{ for all } j=1,...,J; t=0,...,T-1 \quad (2)$$

$$x_k \in \{0,1\}$$

$$I_{jt}, q_{jt} \geq 0, \text{ integer, for all } j=1,...,J; t=1,...,T.$$

Constraint (1) represents the inventory balances. The inventory of a part at the end of the current period is calculated as the inventory at the end of the previous period plus the amount produced in the current period, minus the amount used for satisfying external and internal demands. The terms are described as follows:

$I_{j,t-1}$	The inventory at the end of the previous period
q_{jt}	The amount produced this period
$\sum_{k=1}^{K} d_{jkt} x_k$	Represents the quantity of item j needed to satisfy its external demand in time-period t. For a particular order k, the quantity of part j needed for the end of period t is equal to the external demand (d_{jkt}). Then they multiply d_{jkt} by x_k and subsequently summed over all k = 1 to K.

$\sum_{i \in S} a_{ji} q_{i,t-l+v_i}$	Represents the quantity of item j needed to satisfy its internal demand. The amount of j directly needed for one part i (a_{ji}) is multiplied by the quantity of i produced to date ($q_{i,t-l+vi}$). Note that the lead-time for i is considered. The summation is performed over all parents of j (i.e. $i \in S_j$)

Constraints (2) ensures the inventory of part j at the end of period t is sufficient for all parents of part j (i.e. parts $i \in S_j$) in the period t+1.

When $q_{jt} = 0$ and $t > T$, there is no need to produce item j after considering T.

Due to the number of constraints, solving the ILP became computationally prohibitive as the number of orders, parts and assemblies increases.

3 The Flat CBMSP

3.1 Description Flat CBMSP

A CBMSP is said to be *flat* if the part hierarchy consists of only customer orders attached to a single set of purchased parts. In other words, in a flat CBMSP, all parts are in level 1 only.

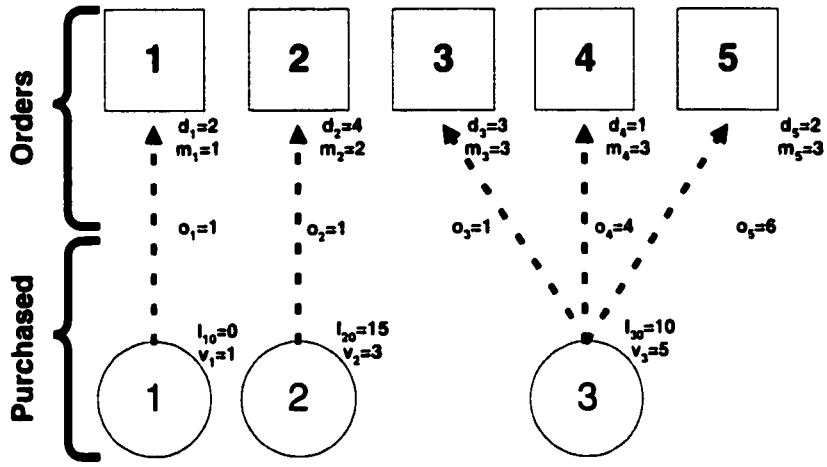


Figure 6. Flat CBMSP[1]

[1] The flat CBMSP presented in [1] upon which this model is based consists of orders for parts that are assembled from other purchased parts. In this representation of the flat CBMSP the ordered parts are purchased.

3.2 Transforming a Flat CBSMP into a Multiple Knapsack Problem

3.2.1 Description of MKP

Knapsack problems often arise as sub-problems of many combinatorial problems. These problems often model industrial situations such as cargo loading, capital budgeting, cutting stock, and project selection. Given n different objects and a knapsack of capacity b, where the object k has weight r_k and profit p_k, the goal is to find a combination of non-negative integers $x_1, x_2, \ldots x_n$ representing quantities of objects 1,2, .., n such that the total weight fits the knapsack, i.e. $r_1x_1 + r_2x_2 + \ldots + r_nx_n \leq b$, and the profit $p_1x_1 + p_2x_2 + \ldots + p_nx_n$ is maximized. In the case where $x_i \in \{0,1\}$ the problem has been referred to as the 0/1 knapsack problem[7]. The multiple knapsack variant of this problem has multiple knapsacks $i \in \{1, .., m\}$ each with a capacity of b_i, and each object has a weight r_{ik} for each $i \in \{1, \ldots, m\}$, and $k \in \{1\ldots, n\}$.

The multiple knapsack problems can be formulated as:

$$\text{maximize} \quad \sum_{k=1}^{n} p_k x_k \qquad (1)$$

$$\text{subject to} \quad \sum_{k=1}^{n} r_{ik} x_k \leq b_i, \qquad i = 1, \ldots, m \quad (2)$$

$$x_k \in \{0,1\}, \qquad k = 1, \ldots, n \quad (3)$$

where each of the m constraints (2) is referred to as the knapsack constraint. This problem has also been referred to as the m-dimensional knapsack problem,

multidimensional knapsack problem, the multi-knapsack and the multidimensional zero-one knapsack problem.

Typically, p_k reflects the profit attributed to object k which consumes r_{ik} units of a resource i, where b_i is the initial amount of that resource. The objective is to find a subset of the n objects such that the total profit is maximized and all resource (i.e. knapsack) constraints are met[8].

3.2.2 Background for MKP

The Multiple Knapsack Problem (MKP) is known to be an NP-hard problem[9] in which case it is considered highly unlikely that an efficient polynomial-time method exists for this problem. For this reason, large problems require heuristic solutions.

The majority of the studies relating to the multiple knapsack problem have dealt with a solutions for a single resource (m=1). For the most part effective algorithms have been developed for obtaining near-optimal solutions. When $m \geq 2$ effective optimal solution algorithms have only been demonstrated on problems where m is small. For cases when m and n are large, exact and heuristic methods have diminishing effectiveness.

Chu and Beasley[8] provided a literature survey of both exact and heuristic algorithms for solving the MKP. Some of their findings are summarized as follows:

3.2.2.1 Exact Algorithms

Branch and bound algorithms were developed for the MKP by Shih[10], and Gavish and Pirkul[11]. In addition, a number of dynamic programming algorithms were developed including an enumeration algorithm based on Fourier-Motzkin elimination[12] and an iterative scheme in which linear programs were solved to generate sub-problems that were subsequently solved by implicit enumeration[13].

3.2.2.2 Heuristic Algorithms

Loulou and Michaelides[13] described a greedy method based on Toyoda's primal heuristic[14]. Primal heuristics begin with a zero solution, after which variables are assigned a value of one, according to a given rule, while ensuring the solutions remains feasible. This approach resembles the greedy heuristic mentioned in Section 3.3.1.

Magazine and Oguz[15] presented a heuristic combining the ideas of Senju and Toyoda's dual heuristic[16] with Everett's generalized Lagrange multiplier approach[17]. Dual heuristics begin with the all-ones solutions, and setting variables to zero in a step-wise fashion according to the heuristic rules until a feasible solution is found.

Rinnooy and Kan *et al.*[18] proposed a class of generalized greedy algorithms in which the items were selected according to the decreasing ratios of their profit and a weighted sum of their resource coefficients.

Pirkul[19] presented a heuristic algorithm which utilizes surrogate duality.

Freville and Plateau[20] presented an efficient preprocessing algorithm for the MKP known as AGNES.

Glover and Kochenberger[21] presented a heuristic based on tabu searching.

Thiel and Voss [22] applied genetic algorithms to the MKP. Simple heuristic operators based upon local search algorithms were used along with a hybrid algorithm based on combining a GA with a tabu search.

Chu and Beasley[8] developed a heuristic operator which utilizes problem-specific knowledge and incorporated it into a standard genetic algorithm approach.

Morales, et al.[7] applied parallel algorithms to the MKP problem for the implementation on distributed systems.

3.2.3 Representing a Flat CBMSP as a MKP

Recall that a flat CBMSP consists of a set of customer orders $k \in \{1, \ldots, K\}$ and a set of parts $i \in \{1, \ldots, I\}$ required for customer orders (recall that each order k is for a single part m_k). Let $b_i = I_{i0}$ represent the initial inventory for all parts in i. For all parts i and orders k, let r_{ik} represents the amount of part i's inventory which is required to ensure order k is assembled on time. If $i \neq m_k$, then clearly $r_{ik} = 0$. If $i = m_k$ then if the lead-time v_i for i is less than or equal to the due date d_k for k, then clearly the required part i for order k can be purchased, and hence $r_{ik} = 0$, and if $v_i > d_k$, then order k requires $r_{ik} = o_k$ of part i.

In other words,

$$r_{ik} = \begin{cases} o_k & \text{if } v_i > d_k \text{ and } i = m_k, \\ 0 & \text{otherwise.} \end{cases}$$

The following table summarizes the notation:

$k \in \{1, \ldots, K\}$	The set of customer orders.
$i \in \{1, \ldots, I\}$	The set of parts required for the assembly of customer order parts
b_i	The initial inventory of part i = 1, ..., I
r_{ik}	The quantity of inventory for i = 1, .. I required to ensure that customer order k = 1, ..., K is on time.
p_k	The profit of ensuring customer order k = 1, .., K is filled on time. Note that p_k = 1 if all orders are equally profitable.

Decision variables:

$$x_k = \begin{cases} 1, \text{if customer order k is on time} \\ 0, \text{otherwise} \end{cases}$$

for all k = 1, .., K.

The MKP problem corresponding to the flat CBMSP:

$$\max \sum_{k=1}^{K} p_k x_k$$

s.t.

$$\sum_{k=1}^{K} r_{ik} x_k \le b_i, \text{ for i} = 1, .., \text{I}$$

$$x_k \in \{0,1\} \text{ for } k = 1,2,...,K$$

The constraint simply ensures that we never exceed b_i for each i when calculating the amount of inventory to ensure that customer orders are delivered on time.

3.2.3.1 Example

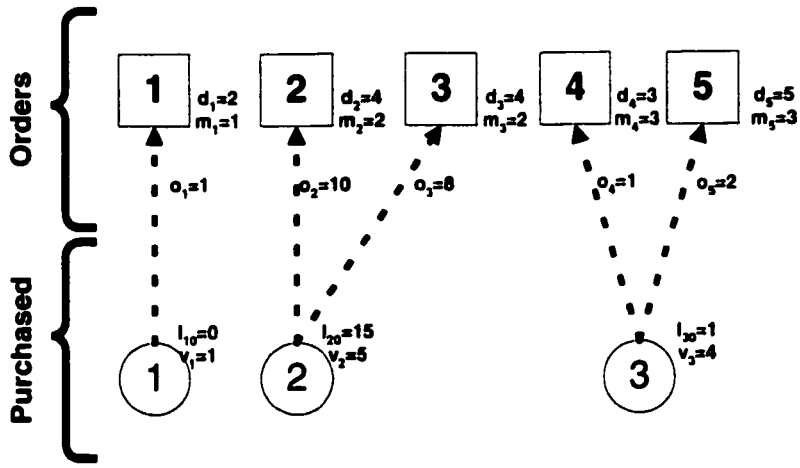


Figure 7. Flat CBMSP

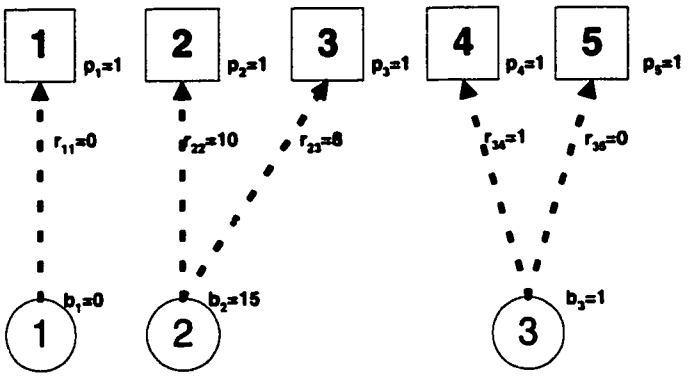


Figure 8. Flat CBMSP represented as a MKP[2]

3.3 A Detailed look at a Heuristic for the MKP

3.3.1 Greedy Hill-Climbing Method

A heuristic for the MKP proposed by Boyd, Holte, and Wang[1] describes a solution set as the set of orders that are to be satisfied. This set is grown incrementally from an empty set. With each iteration of the heuristic algorithm, orders that are not already in the solution set are evaluated. The "highest-scoring" feasible one is then added to the solution set.

The heuristic evaluates an order k by computing the ratio of the $BENEFIT_k$ of satisfying the order k against the COST of consuming its resources. The $BENEFIT_k$ is simply p_k,

---

[2] In this example, we assume $p_k = 1$ for all $k \in K$.

being the profit of satisfying order k. The COST is computed as the depletion (k,i) = r_{ik}/b_i. In addition, the following assumptions are made:

- $p_k > 0$ where p_k reflects the profit attributed to satisfying item k.
- $r_{ik} \leq b_i$ for all $i \in I = \{1, .., m\}$, and $k \in K = \{1, ..., n\}$, where order k consumes r_{ik} units of a resource i, where b_i is the initial amount of resource i. If $r_{ik} > b_i$ then the demand exceeds the availability of the resource thus making it impossible for the order to be completed. If we do have $r_{ik} > b_i$ for any order k any time in the algorithm, we simply mark order k as not satisfied and remove it from the problem.
- $b_i < \sum_{k=1}^{n} r_{ik}$ for all $i \in I = \{1, .., m\}$. If $\sum_{k=1}^{n} r_{ik} \leq b_i$, this indicates that there is enough resource i to satisfy all orders k, and we simply remove resource i.
- For every order k, we assume $\sum_{i=1}^{m} r_{ik} > 0$. If we have an order k for which $\sum_{i=1}^{m} r_{ik} = 0$, such an order requires no resources and can be marked as satisfied.

The motivation behind computing the depletion is to normalize r_{ik} with respect to the available amount of resource i. Depletion (k,i) can be thought of as the size of the "step" in the direction i if order k is added to the solution set. The depletion (k,i) for all orders k and resources i are combined by computing the length of the vector represented by the depletion (k,i) values. The length is computed as:

$$COST_k = \left(\sum_i depletion(k,i)^2 \right)^{\frac{1}{2}}$$

From the assumptions, $COST_k > 0$ and $b_i > 0$.

In the case where $p_k = BENEFIT_k = 1$ the corresponding $BENEFIT_k/COST_k$ ratio is:

$$\frac{BENEFIT_k}{COST_k} = \frac{1}{\left(\sum_i depletion(k,i)^2\right)^{\frac{1}{2}}}$$

Once the highest solution that satisfies all constraints of the MKP is computed, the corresponding order k is added to the solution set and the b_i are reduced by r_{ik} for all $i \in I$. The heuristic repeats as long as orders can be added to the solution set.

3.3.1.1 Example

Refer to the following example:

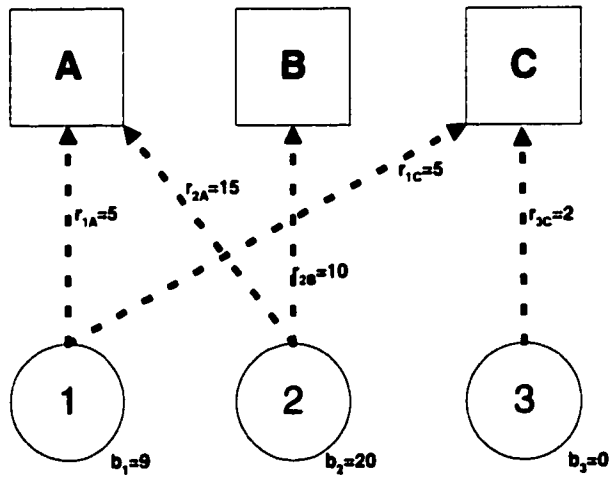


Figure 9. Multiple Knapsack Example

Figure 6 depicts a graph which represents a multiple knapsack problem with n = 3 items {A, B, C} and m = 3 resources {1, 2, 3}. In the graph, r_{ik} reflects the amount of resource i is needed by order k and b_i reflects the amount of resource i that is available. Note that $r_{ik} = 0$ for edges not shown. In this example $p_k = \text{BENEFIT}_k = 1$ for all k = {A, B, C}. Employing the greedy hill-climbing heuristic described in Section 3.3.1 we get:

Pass 1: Solution Set = { }

$$COST_A = \left(\sum_{i=1}^{3} depletion(A,i)^2 \right)^{\frac{1}{2}}$$

$$depletion(A,1) = \frac{r_{1,A}}{b_1} = \frac{5}{9}$$

$$depletion(A,2) = \frac{r_{2,A}}{b_2} = \frac{15}{20}$$

$$COST_A = \left(\left(\frac{5}{9} \right)^2 + \left(\frac{15}{20} \right)^2 \right)^{\frac{1}{2}} = 0.933$$

$$\frac{BENEFIT_A}{COST_A} = \frac{1}{.933} = 1.07$$

$$COST_B = \left(\sum_{i=1}^{3} depletion(B,i)^2 \right)^{\frac{1}{2}}$$

$$depletion(B,2) = \frac{r_{2,B}}{b_2} = \frac{10}{20}$$

$$COST(B) = \left(\left(\frac{10}{20} \right)^2 \right)^{\frac{1}{2}} = 0.5$$

$$\frac{BENEFIT_B}{COST_B} = \frac{1}{.5} = 2$$

$$COST_C = \left(\sum_{i=1}^{3} depletion(C,i)^2 \right)^{\frac{1}{2}}$$

$$depletion(C,1) = \frac{r_{1,C}}{b_1} = \frac{5}{9}$$

Since $r_{3,C} > b_3$, order 3 can never be satisfied and is removed.

Order B scored the highest feasible solution (2), therefore order B is marked satisfied and resource 2 is reduced by r_{2B} = 10 units.

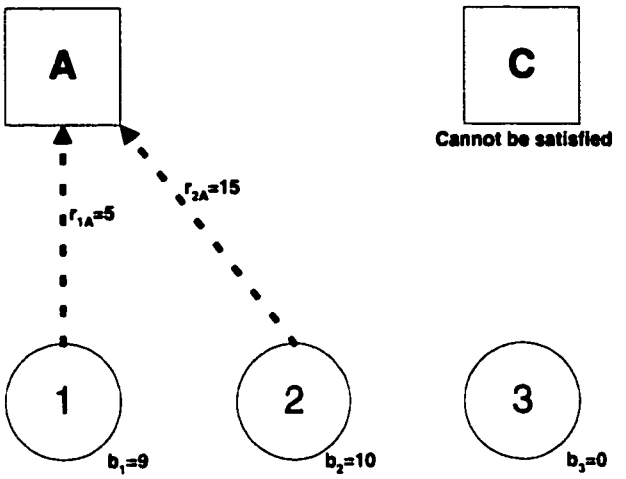


Figure 10. Example after Item B has been satisfied

Pass 2 : Solution Set = {B}

$$COST_A = \left(\sum_{i=1}^{3} depletion(A,i)^2 \right)^{\frac{1}{2}}$$

$$depletion(A,1) = \frac{r_{1,A}}{b_1} = \frac{5}{9}$$

Since $r_{2,A} > b_2$, object A cannot be satisfied and is removed.

Item A cannot be satisfied as resource 2 does not have enough inventory. Likewise, Item C cannot be satisfied as resource 3 does not have enough inventory. Both items violate the constraint that states that $r_{ik} \leq b_i$ for all $i \in I$ and $k \in K$. Since no items were added to the solution set, we stop with a solution set = {B}.

3.3.2 Complexity of Hill-Climbing Method

In the worst case one order $k \in K$ will be added to the solution set per iteration through the parts $i \in I$, so there will be O(K) iterations. If the MKP to be solved was a complete

graph (i.e. all orders k ∈ K are connected to all parts i ∈ I), then we have a complexity of O(K·I) due to the fact that we must compute the depletion for all parts i connected to and order k. Therefore, the overall complexity for the Greedy Hill-Climbing Method is $O(|K|^2|I|)$.

3.4 Transforming General CBMSP into a MKP

Consider the following graph:

1

Figure 11. General CBMSP with no internal inventories

In this example, the inventories for nodes = {1,2,3,4,5,6} are all zero. It is possible to transform this graph into a MKP.

3.4.1 General CBMSP Transformation to MKP

Given a general CBMSP with no internal inventories, it is possible to convert this to an equivalent MKP using the following transformation:

1. Let K represent the set of customer orders for the general CBMSP.

2. Let I represent the purchased items in the hierarchical structure for the general problem (i.e. the ones at the bottom of the hierarchical structure).

3. For each i ∈ I let $b_i = I_{io}$.

4. Calculate r_{ik} for all $i \in I$, $k \in K$. To do this consider the graphical representation G of the general CBMSP and do the following to calculated r_{ik}:

   a. Find each distinct path P from i to m_k in G.

   b. For each of these paths P, find the required amount A_p of i for that path by multiplying the given amounts a_{uv}, for all edges uv in P, and multiplying the result by the given customer order amount o_k.

   c. For each path P from i to m_k in G, we can also calculated the lead-time V_p for the path by summing the given lead times v_j for all nodes j in P, including i.

   d. If V_p is less than or equal to the given due time d_k for k, we do not need to use inventory for item i for that path to help make the order on time. Using the inventory will make no difference in satisfying order k, since sending the inventory towards k along path P will not satisfy the order any earlier. If $V_p - v_i$ is greater than the given due time d_k, order k cannot be satisfied. If $V_p > d_k$ and $V_p - v_i \leq d_k$, then we must use A_p units of the inventory of i for path P to satisfy order k on time. This corresponds to "pushing" A_p units of items i up along path P to order k.

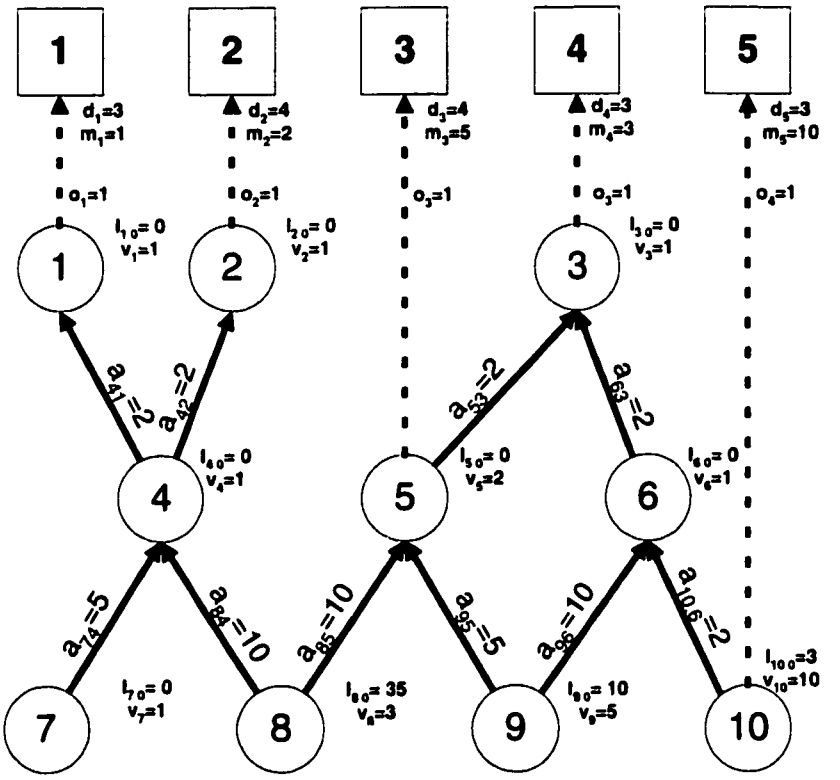
   In summary:

$$r_{ik} = \begin{cases} 0 & \text{when } V_p \leq d_k \text{ for all paths P from i to } m_k, \text{ or if there are not paths from i to } m_k, \\ \sum A_p & \text{where P is a path from i to } m_k \text{ s.t. } V_p - v_i \leq d_k \text{ and } V_p > d_k, \\ \infty & \text{otherwise.} \end{cases}$$

Note that the value $r_{ik} = \infty$ indicates that order k cannot be satisfied under any circumstances and thus can be removed from the hierarchy.

3.4.1.1 Example

Using our example:

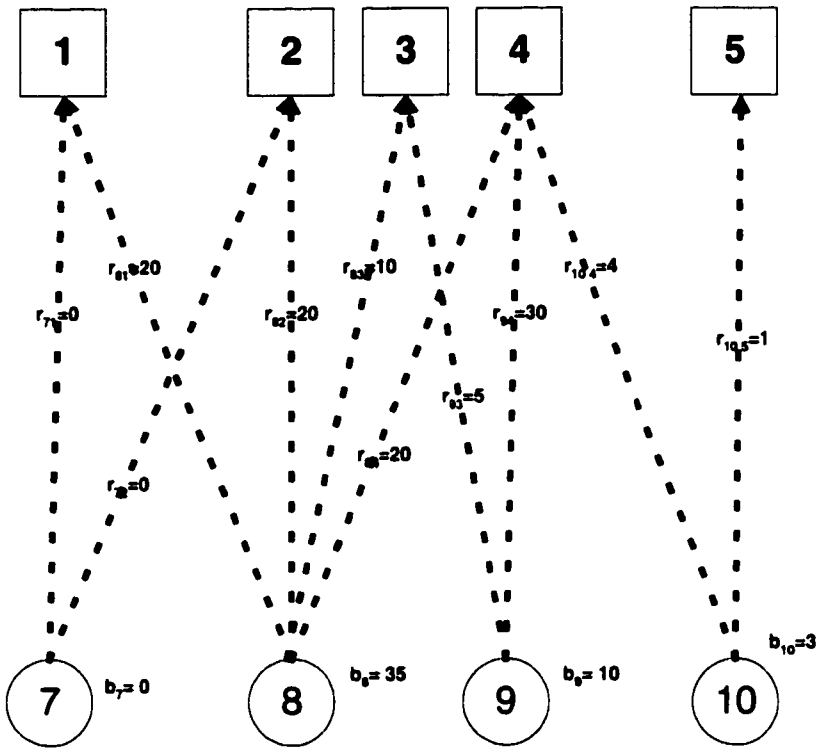


1. Let K = {1,2,3,4,5}

2. Let I = {7,8,9,10}

3. $b_i = I_{i,0}$ for all $i \in I$.

4. r_{ik}, $i \in I$, $k \in K$ are shown beside the corresponding edge, and the values b_i, $i \in I$ are shown below the corresponding nodes. (Note that edges with $r_{ik} = 0$ could be left out.)



r_{71} $= 0$ because $V_{71} = v_1 + v_4 + v_7$ $= 1+1+1$, and $V_{71} \le d_1$	r_{72} $= 0$ because $V_{72} = v_2 + v_4 + v_7$ $V_{72} = 1+1+1$, and $V_{72} \le d_2$	r_{81} $= o_1 \cdot a_{41} \cdot a_{84}$ $= 1 \cdot 2 \cdot 10$ $= 20$	r_{82} $= o_2 \cdot a_{42} \cdot a_{84}$ $= 1 \cdot 2 \cdot 10$ $= 20$
r_{83} $= o_3 \cdot a_{85}$ $= 1 \cdot 10$ $= 10$	r_{93} $= o_3 \cdot a_{95}$ $= 1 \cdot 5$ $= 5$	r_{84} $= o_4 \cdot a_{53} \cdot a_{85}$ $= 1 \cdot 2 \cdot 10$ $= 20$	r_{94} $= o_4 \cdot a_{53} \cdot a_{95}$ $+ o_4 \cdot a_{63} \cdot a_{96}$ $= 1 \cdot 2 \cdot 5 + 1 \cdot 2 \cdot 10$ $= 10 + 20$ $= 30$

r_{104} $= o_4 \cdot a_{63} \cdot a_{106}$ $= 1 \cdot 2 \cdot 2$ $= 4$	r_{105} $= o_5$ $= 1$		

3.4.1.2 Complexity of General CBMSP Transformation to MKP

The general CBMSP transformation to MKP can be accomplished through a recursive Depth-First-Search (DFS). The DFS is extended to calculate the V_p and A_p as it searches through the part hierarchy. The complexity of the DFS can be described as **O(|V| + |E|)** where |V| is the total number of nodes in the graph (including parts and orders) and |E| is the number of edges in the part graph.

3.5 Interpreting the Solution for MKP as a Solution for Flat CBMSP

If we were to solve the resulting MKP found in example 3.4.1.1, using the heuristic proposed in Section 3.3.1, we would end up satisfying orders 1, 3 and 5. Orders 2 and 4 cannot be satisfied as there is not enough inventory to satisfy the demands. In order to interpret the solution for the MKP as a solution for the general CBMSP, we use the original graph and consider the following:

1. The Satisfied Orders

This can be determined as the set of orders added to the solution set when the MKP is solved. Any order not in the solution set cannot be satisfied.

From our example, orders 1, 3 and 5 were in the solution set indicating that they were satisfied.

2. When the Order was satisfied

For orders that were included in the solution set, the order was satisfied using parts in one of two ways:

- For a part j needed to satisfy order k, there was enough inventory b_j to meet the demand $r_{jk} > 0$. The time it takes to satisfy the order using this part is $V_{j \to k} - v_j$. This is the lead-time path from part j to order k in the original graph **excluding** lead-time v_j for part j.

- The longest path $V_{j \to k}$ from part j to order k was less than or equal to the due period d_k, indicating that there was enough time to purchase and assemble the parts required satisfying the order. Recall, for this path $r_{jk} = 0$.

For an order in the solution set, all longest j to k lead-times are computed based on the above cases. The maximum of longest lead-time path over all j corresponds to the period in which the order was delivered.

For Order 1 in our example, the lead-time path from part 7 is $V_{7\,1} = 3$, the lead-time path from part 8 is $V_{8\,1} - v_8 = 5-3 = 2$. The maximum of $\{V_{7\,1}, V_{8\,1}\}$ is 3 indicating the order 1 was delivered on day 3.

For Order 3 in our example, the lead-time path from part 8 is $V_{8\,3} - v_8$=2. The lead-time path from part 9 is $V_{9\,3} - v_9$=2. The maximum of {$V_{8\,3}$, $V_{9\,3}$} is 2 indicating the order 3 was delivered on day 2.

For Order 5 in our example, the lead-time path from part 10 is $V_{10\,4} - v_{10}$=0. This is the maximum and only lead-time path. This means that order 5 was delivered on day 0.

3. The Parts Used

The parts used to satisfy an order include the connected parts in the MKP, and all higher level parts upon which the order is dependent.

The parts used to satisfy order 1 included parts 1, 4, 7, and 8. Order 3 needed part 5, and order 5 needed part 10.

4. The Quantity of Parts Needed

The quantity of each part j required to satisfy order k is computed from each of the j to k paths P in the original hierarchy. To calculate the amount of part j used, calculate the amount required A_{jk} by multiply the given amount a_{uv} for all edges uv between j and k.

From our example Order 1 required:

o_1 = 1 of part 1; $o_1 \cdot a_{41}$ = 1·2 = 2 of part 4; $o_1 \cdot a_{41} \cdot a_{74}$ = 1·2·5 = 10 of part 7; and

$o_1 \cdot a_{41} \cdot a_{84}$ = 1·2·10 = 20 of part 8.

Order 3 required:

o_3 = 1 of part 5;$o_3 \cdot a_{85}$ = 10 of part 8; $o_3 \cdot a_{95}$ = 5 of part 9.

Order 5 required:

o_{10} = 1 of part 10.

5. The Quantity of Parts Purchased

For each purchased part j in level h=H, identified in Step 4. The number of parts purchased is the difference between the quantity required and the $I_{j\,0}$.

Order 1 required 10 of part 7, 10 – $I_{7\,0}$ = 10 of which was purchased; 20 of part 8, 0 of which was purchased

Order 5 required 1 of part 10, 0 of which was purchased.

4 Revised Heuristic for CBMSP

The following Chapter describes a heuristic for the CBMSP, along with examples, and discussion regarding its function.

4.1 General Discussion of CBMSP Heuristic

In this section, we describe a heuristic for the CBSMP. This heuristic is based on using the algorithm for transforming a general CBSMP into a MKP (Section 3.4.1). The heuristic performs a level-wise "collapsing" of the part hierarchy. Each level of the general CBSMP is collapsed into a MKP. Satisfied orders are determined by solving the resulting MKP. This process repeats until all levels of the graph have been processed or all orders have been satisfied.

The heuristic presented here is a much improved and more robust version of the algorithm presented by Boyd, Holte and Wang[1]. In the original CBMSP heuristic, the level-wise collapsing occurred by modifying the original graph eliminating parts in levels $< h$. This caused lead-times and inventory amounts needed to perform the 0-h MKP transformation to be lost, resulting in incorrect transformations to MKP. Secondly, there was no provision in the original heuristic for parts to exist at multiple levels as was the case in the customer data used to test the heuristic. Finally, the heuristic provided here provided a better method for handling the inventory reallocation step presented in the original algorithm.

The idea of the CBMSP heuristic is as follows: First, we check if any orders can be satisfied on time by using the available inventory in the level 1 parts. If not, we determine if we can satisfy orders by building level 1 parts, using parts deeper in the part hierarchy. To accomplish this, we calculated the longest-lead time path between the purchased parts (i.e. the bottom nodes in the hierarchy) and the parts in level 1. The longest path to a level 1 part i represents the earliest time in which, the part i can be provided. Using this information, we can transform the orders and parts at level 1 into a MKP as described in Section 3.4.1. We solve the MKP attempting to satisfy as many orders as possible.

Next, we consider the level 2 parts. Again, we determine if there is enough inventory available in level 2 that can help satisfy the remaining orders. We do this by "collapsing" levels 0,1 and 2. This is accomplished by calculating the longest lead-time paths from level 2 parts to purchased parts at the bottom of the hierarchy. At the same time, we ignore any inventory still present in level 1 parts. We proceed to transform levels 0,1, and 2 into a MKP using the General CBMSP transformation to MKP from Section 3.4.1 and subsequently solve it satisfying as many orders as possible.

For each level h we create the appropriate MKP using the method described to collapse levels 0,1,...,h. We call this the 0-h MKP. When all levels have been considered we stop.

4.1.1 Inventory Allocation

At this point, there still may be inventory that was left over and then ignored later. Although this inventory was not used by the 0-h MKP, making the most of this inventory proved to have a significant influence on the effectiveness of the heuristic. In the original algorithm offered by Boyd, Holte and Wang[1], the inventory for level h nodes, was "given away" in a greedy fashion. When we analyzed the algorithm closely, we saw that their approach was flawed and that there was room for improvement.

We considered the following approaches for dealing with left over inventory in the part hierarchy:

4.1.1.1 Pushing Down Inventory

In this approach to dealing with the leftover inventory, we attempt to "push" the inventory in the level h nodes down to the level h+1 parts by "disassembling" it. This was accomplished by taking each level h node and augmenting the inventory of all level h+1 nodes. For each node u in *level* h+1, the new inventory pushed down from nodes j in level h was calculated using the formula:

$$I_{u_0} = I_{u_0} + \sum I_{j_0} \times a_{uj} \text{ for all nodes j and u.}$$

The ability to "push down" inventory from level h nodes to level h+1 nodes, is restricted by the fact that the act of "disassembling" the parts in level h is misleading. What is actually happening, is that the leftover inventory for part j in level h can still be used to assemble any orders that directly or indirectly require the part in order to be completed. If

the sub-parts of j are required by the same orders for completion, "pushing" the inventory down to sub-parts in level h+1, simply means deferring the allocation of part j's inventory to level h+1. However, if the parents of the sub-parts of j include other parts in addition to j, then "pushing down" is not possible. Consider the following:

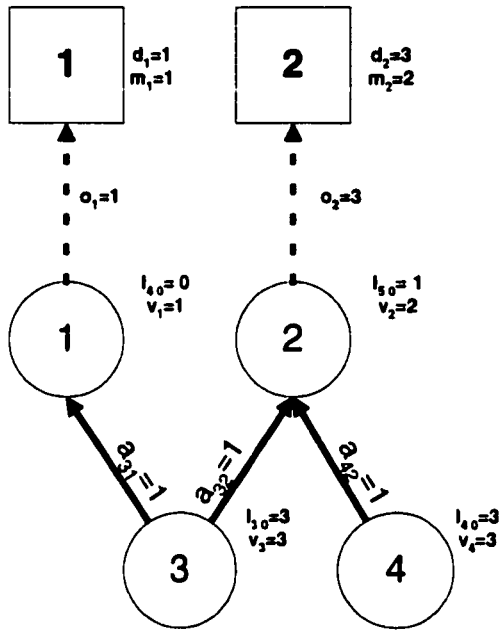


Figure 12. Inventory cannot be pushed down.

In this example, the parent of part 4 is part 2, the parents of part 3 are 1 and 2. It can be seen that from the example that we cannot push inventory from part 2 down to part 3 as it would then become available for part 1 to satisfy order 1. This inventory can only be used to assemble part 2. It is possible to "push" the inventory down to part 4 because part 2 is the only parent of part 4. The inventory pushed down to part 4 can still be used to satisfy order 2.

Consider the following example:

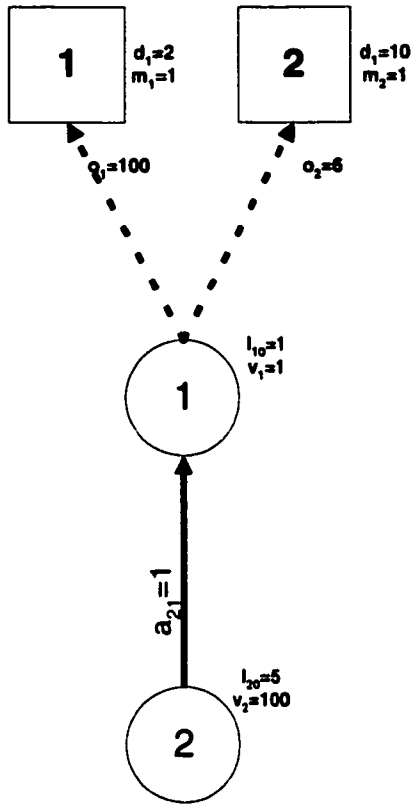


Figure 13. An example to exhibit the benefits of pushing down inventory.

From Figure 13, it can be seen that it will be impossible to satisfy Order 1, as there isn't enough inventory in the part hierarchy. In addition, the longest lead-time path from Order 1 to part 2 is 101, which is greater than the due date of 2. However, it is possible to make order 2 delivered on time as there is enough Part 2 to make 6 of Part 1 thus satisfying Order 2's demand.

By pushing Part 1's inventory to Part 2, we get:

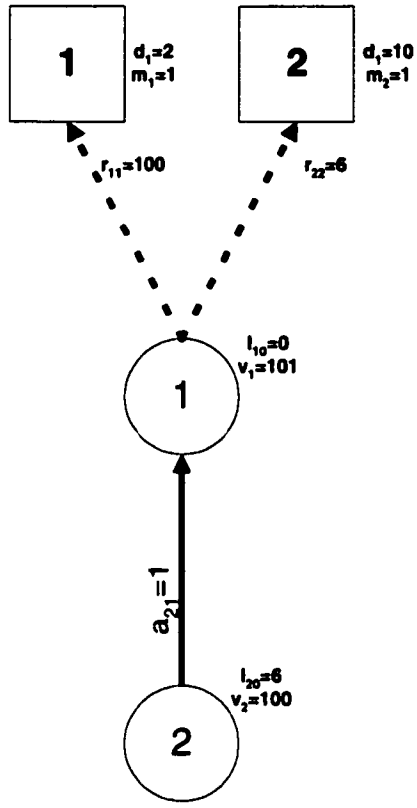


Figure 14. Part 1 inventory is pushed down to Part 2.

By doing this, we are deferring the decision to allocate leftover inventory until the next iteration. Following the algorithm to completion, Order 2 receives the inventory from part 2, making it on time, and maximizing the number of delivered orders.

Thus, we see that the push down approach can be used whenever the nodes i at level h-1 is connected to nodes j in level h and each node in j is connected to only node i in level h-1. In this case, the push down approach improves upon the original heuristic's inventory reallocation step by deferring the decision of how to best allocate the inventory until the next iteration of the heuristic. By deferring the decision, the inventory becomes available in the h+1 nodes for allocation in the next iteration of the revised heuristic. This step is

also computationally simple compared to the old approach, and no inventory is lost from the hierarchy.

4.1.1.2 Pushing Up Inventory

As the push down approach cannot always be used, we also looked at another approach to handle leftover inventory in the hierarchy. In this approach, leftover parts are used at the end of the algorithm to assemble other parts. This is accomplished by "pushing" this part's inventory "upwards" along arcs to parts that need the inventory for assembly. That is to say, we simply start making parts in the hierarchy by consuming the inventory of the connected sub-parts in the part hierarchy.

Starting with the purchased parts (bottom nodes), this inventory is pushed up along edges to parts higher in the hierarchy. If a part i receives enough inventory from all connected sub-parts j, one or more of these parts i can be assembled. The corresponding inventory of the sub-parts j is reduced accordingly. Consider the following:

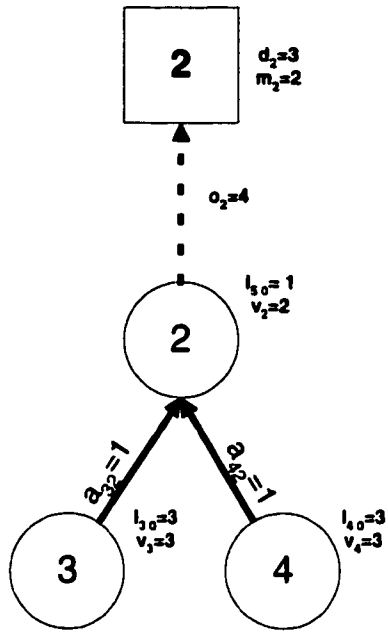


Figure 15. Example to show Pushing Inventory Up from Parts 3 and 4 to 2.

In the above example, it is possible to construct 3 part 2s using inventory "pushed up" from parts 3 and 4. The time it takes to supply the additional part 2's is lead-time $v_2 = 2$.

There are a number of considerations when "pushing up" inventory in the part hierarchy. These include:

1. The order or orders k ultimately being satisfied with the pushed up inventory;

2. Ensuring a part can be assembled because it receives the necessary inventory from all connected sub-parts;

3. Ensuring that just enough inventory is pushed to a given part;

4. The lead-time it takes to assemble the part from inventory is small enough to ensure an order can still be satisfied on time; and

5. Whether or not it is possible to purchase parts to help in the assembly.

The problem of determining the path along which a part's inventory can be pushed, so that an order can be satisfied, is similar to the problem of choosing which order to assign the inventory in the transformed 0-h MKP. Similarly, the problem of determining which part to give inventory to can also be modeled as a MKP. Finally, when considering the amount of inventory to assign to a part, the amount required is dictated by the demands on that part.

To describe the "push-up" approach developed consider this example:

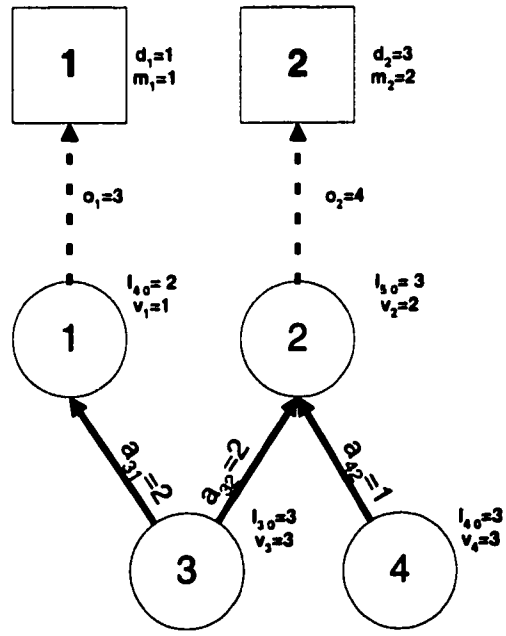


Figure 16. Pushing inventory upwards

The approach requires determining a 0-H MKP where H is the level of purchased parts. Using the 0-H MKP, we can determine which order would most likely be satisfied by taking the highest scoring order determined using the heuristic method for solving the MKP described in Section 3.3.1. The highest scoring order in the 0-H MKP would be the order, which is the "closest" to being satisfied. Using the Greedy Hill-Climbing heuristic described in Section 3.3.1 the highest-scoring order is the one with the minimum $COST_k$ > 1 (Order 2).

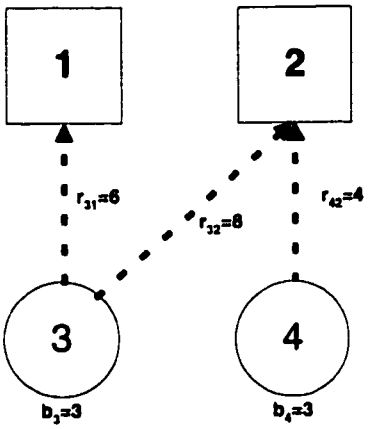


Figure 17. 0-H MKP

Using this order k, we construct a graph consisting of only the part hierarchy required to satisfy it. The hierarchy contains only the parts j for which, V_p-$v_j \leq d_k$ where V_p is the longest lead-time path from j to order k (See Section 3.4.1). This ensures that the order k part graph contains only the left over inventory that can be used to satisfy order k by the due date d_k.

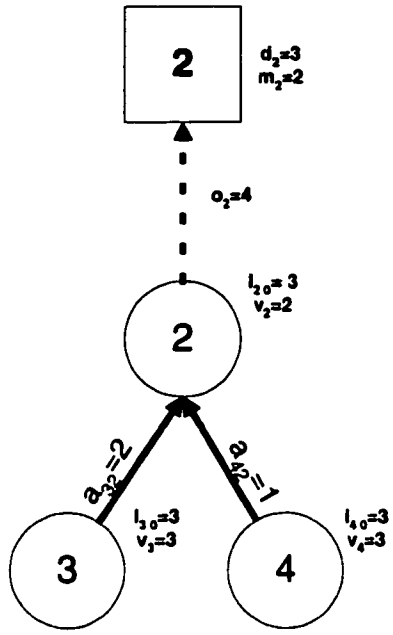


Figure 18. Order 2 Part Graph

At the same time, however, we must keep track of the fact that we are introducing new inventory for part i in level h-1 by assembling parts j in level h. This is recorded from the bottom of the part hierarchy (level h=H) up to the ordered part m_k. The first thing to consider is whether purchased parts can be purchased for the assembly of other parts. This is done by using the 0-H MKP from the original graph. For each $r_{jk} = 0$ for part j in level h=H and order k, then we can purchase parts while still satisfying the order within the due-period d_k. In this case we can set the $I_{j\,0} = \infty$ indicating that there will always be enough part j inventory to push up. If $r_{jk} > 0$ for part j in level h=H and order k, then we cannot purchase any parts. In our example, neither parts 3 or 4 can purchase additional inventory.

Starting at level h=H, we determine which level h-1 parts should have inventory pushed to them. This is accomplished by creating the 0-(h-1) MKP from the order 2 part graph (Figure 19. 0-1 MKP). The transformation to the 0-(h-1) MKP uses the longest lead-time paths computed from each bottom (purchased) nodes to node i in level h-1 in the original graph.

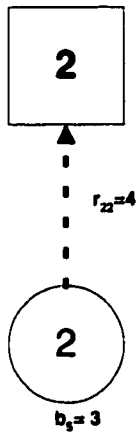


Figure 19. 0-1 MKP

First we rank all of the parts i in level h-1 in decreasing order of b_i / r_{ik} for order k. This ordered list permits us to give inventory to parts i that are ' close" to satisfying the order k demands placed on them. In this example, h=H=2 and the only part in h-1 is part 2. Therefore, this part receives any inventory that can be pushed up to it.

We consider each part i in level h-1 one at a time in the order established. Using the 0-(h-1) MKP we calculate the amount of inventory we need to push up to part i from its sub-parts j in level h. This would be $r_{ik} - b_i$. If $b_i \geq r_{ik}$, then clearly we do not need to push any inventory up to part i, and we consider the next candidate i in our ordering. Otherwise, we let w_{ji} represent the number of part i, which could potentially be built by

pushing up inventory from part j i.e $w_{ji} = \left\lfloor I_{j0} / a_{ji} \right\rfloor$. The maximum number of part i which can be built by pushing up from sub-parts will be: $build_i = \min(w_{ji}: j \text{ a sub-part of } i)$. If $build_i$ is greater than the amount of part i needed, i.e. if $build_i > r_{ik} - b_i$, then we let $build_i = r_{ik} - b_i$. The 0-1 MKP for our example is shown in Figure 18. We can see that the amount of part 2 needed through pushing up would be $r_{22} - b_2 = 1$.

We have $w_{42} = \left\lfloor I_{40} / a_{42} \right\rfloor 3/1 = 3$ and $w_{32} = \left\lfloor I_{30} / a_{32} \right\rfloor = 3/2 = 1$, and thus $build_2 = 1$.

The next step, before considering the next part i in our ordering, is to adjust the inventories of our current part i and its sub-parts j. This will reflect the fact that we have built $build_i$ units of i by pushing up inventory. We increase $b_i = I_{i0}$ by $build_i$, and for each sub-part j, the new inventory is $I_{j0} - a_{ji} \cdot build_i$. For our example this means $b_2 = I_{20} = 4$, $I_{30} = 1$ and $I_{40} = 2$.

Once we have processed all parts i in level h-1, we check if we now have $b_i \geq r_{ik}$ for all i. If so, order k is now satisfied. Otherwise, we decrement h and repeat the process until h=1. Now we are at the ordered part m_k and can determine if the order k is delivered by simply ensuring that $I_{m_k 0} \geq o_k$. If the order can be satisfied, we simply adjust the inventory in the original graph to reflect the inventory in the order k part graph, and move to the next highest-scoring order k in the 0-H MKP. In our example, $I_{m_k} = 4$ which is $\geq o_4 = 4$. Therefore, part 2 can be satisfied. The original graph updated with the inventory from the order 2 part graph is as follows:

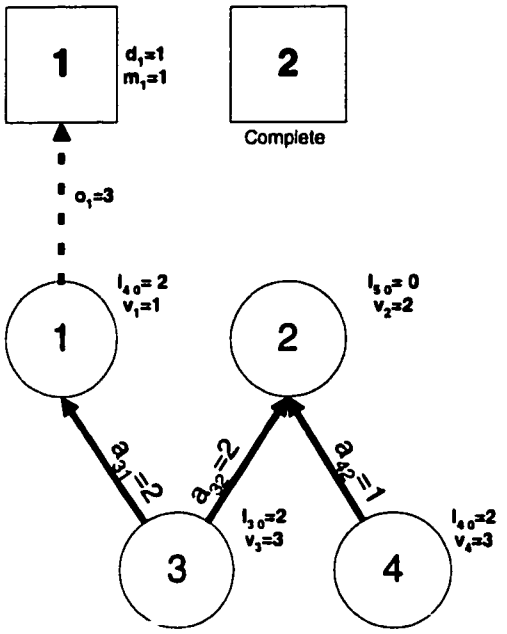


Figure 20. Original Graph updated from Order 2 Part Graph.

This process would repeat trying to push inventory up to order 1.

This approach deals, effectively, with the limitation of the pushing down inventory approach, and has been incorporated as a final step in the heuristic.

4.2 Detailed Description of CBMSP Heuristic

The following summarizes the steps of the CBMSP heuristic:

- The CBMSP has levels 0,1,..., H where the customer orders are at level 0. Recall that a part j resides in level h if there exists a path of length h from j to some order k ∈ K. In addition, recall a part can reside in more than one level.

- For each level h = 1,2,..., H we transform levels 0, 1, ..., h into an MKP which we call a *level 0-h MKP*. This assumes that all inventories for parts above level h are ignored and that all parts below level h are assembled or purchased.

- Each level 0-h MKP can be solved using an exact or heuristic method such as the Greedy Hill-Climbing Heuristic (Section 3.3.1).

- Using the solution from the 0-h MKP, we identify the on-time customer orders and the inventory they require. For level h items which are no longer needed to assemble other parts(i.e. those parts j that reside at distance$_j$ h), we sometimes are able to re-allocate leftover inventory to level h+1 parts by "disassembling" level h parts and incrementing the inventory of corresponding h+1 parts.

- Next, we increment h and repeat the process.

- Once h=H, we attempt to push any remaining inventory in the part hierarchy up to unfilled orders that were "close" to being satisfied. This is done by pushing inventory level by level from the purchased parts upward towards the ordered parts while considering the lead-time along the way. If enough inventory is accumulated and the total assembly time is less than the due date, these orders are marked as satisfied.

4.2.1 The Algorithm

For each level h = 1, 2, ... H perform the following:

1. For each node j in level h, find a longest bottom(purchased part) to j node path, where the length of the path is identified as the sum of the v_i's for all nodes i in the path, including j. Let the label $v_{bottom,j}$ for j be the length of this longest path. *This represents the earliest time that part j can be ready assuming everything below it is assembled or purchased.*

2. If h = 1, we can transform levels 0 and 1, into a level 0-1 MKP directly by using the General CBMSP transformation to MKP algorithm described in Section 3.4.1. In this transformation we consider the level 1 items as purchase parts with purchase time $v_{bottom,j}$ and ignore the other items in the hierarchy. Otherwise, if h > 1, then we will consider the hierarchy consisting of levels 0, 1, ..., h. We transform the levels 0,1, ..., h into a 0-h MKP using the General CBMSP transformation to MKP algorithm described in Section 3.4.1. In this transformation we consider the level h parts as purchased parts with purchase time $v_{bottom,j}$ and ignore the items not in levels 0,1, ..., h. We also ignore the inventories for items in levels 1,2, ..., h-1. When calculating the i to k paths, we only consider paths of exactly length h.

3 Solve the 0-h MKP from Step 2 using an exact or heuristic method such as the Greedy Hill-Climbing Algorithm described in Section 3.3.1. For each order k for which x_k = 1 (the order is filled on time) we mark the order complete in our problem and remove it from the original hierarchy. In the original hierarchy, remove any inventory in the level h nodes, which has been used to make order k on time.

4 Recall that some nodes in level h may also exist in other levels higher than h (i.e. levels h+1, h+2, ..., H). Consider the level h nodes, which do not exist in other levels

high than h. For some of these nodes we can give away left over inventory as follows:

a. Remove all nodes j at level h and their adjacent edges for which the inventory I_{j0} is enough to satisfy the demands r_{jk} of all connected orders k and which are not needed to assemble other parts.

b. For each node j in level h with leftover inventory, we determine if all connected nodes u in level h+1 have j as the only parent. If so, compute the new inventory for each node u in level h+1 by "disassembling" the parts and "pushing" the inventory down to node u using the formula:

$$I_{u_0} = I_{u_0} + I_{j_0} \times a_{ju}$$

5 If there are any level h nodes that need assembling (i.e. have children), then we increment h, and go back to step 1.

6 Using the 0-H MKP, we select the next highest scoring unfilled order k computed by Step 3. Using this order k, we construct a graph consisting of only the parts required to satisfy this order such that for each part j, V_p-$v_j \leq d_k$ where V_p is the longest lead-time path from j to order k. The order k part graph contains only the left over inventory that can be used to satisfy order k by the due date d_k. Using the order k part graph, do the following:

a. For any purchased nodes in the order k part graph, we consider the 0-H MKP based on the original graph. If $r_{jk} = 0$, then we can supply as much inventory as

needed from part j to satisfy the order k, therefore, set $I_{j0} = \infty$. Otherwise, we can only use the inventory I_{j0} currently available.

b. Otherwise, create the 0-(h-1) MKP from the order k part graph consisting of parts i which are assembled from parts j that exist only in level h. Next, rank each part i in level h-1 in decreasing order by b_i/r_{ik}. This ordered list permits us to give inventory to parts i in level h-1, which are closest to meeting the demands placed upon them. For each parts i in the ordered list we do the following: If $r_{ik} < b_i$, then go to the next part i. Otherwise, for each connected part j which exists only in level h compute:

$$w_{ji} = \left\lfloor I_{j0} / a_{ji} \right\rfloor$$

Let $build_i = \min (w_{ji} : \text{where j is a connected part in level h of part i in level h-1})$. If $build_i > r_{ik}-b_i$, then let $build_i = r_{ik} - b_i$. Next, adjust all inventories by adding $build_i$ to b_i and I_{i0}, and decreasing the inventories I_{j0} for sub-parts j by $a_{ji} \cdot build_i$. We proceed to the next part i in the ordered list until all h-1 have been considered.

c. Check to see if $b_i \geq r_{ik}$ for all nodes i in level h-1. If so, order k is now satisfied and can be marked complete. If not, we decrement h. If h = 1, proceed to Step d, otherwise go to step b.

d. If for the ordered node m_k, if $I_{m_k 0} \geq o_k$ then the order is satisfied and can be marked as complete. We update the original hierarchy with the new inventory and

repeat Step 6 until all orders have been considered at which time we stop. Any remaining orders cannot be completed on time.

4.3 An Example of CBMSP Heuristic

Using this example in which we assume $p_k = 1$:

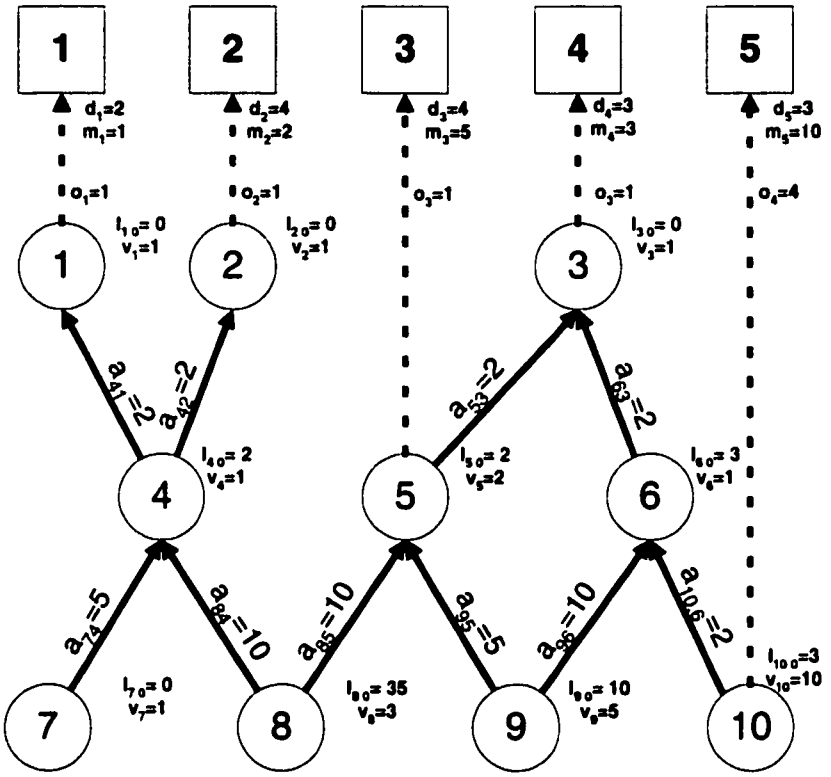


Figure 21. Original Graph with 10 Parts and 5 Orders

Step	Example
1. For each node j in level h, find a longest bottom(purchased part) to j node path, where the length of the path is identified as	h = 1 I = {1, 2, 3, 5}

the sum of the v_i's for all nodes i in the path, including j. Let the label $v_{bottom,j}$ for j be the length of this longest path. *This represents the earliest time that part j can be ready assuming everything below it is assembled or purchased.*	$V_{bottom,1} = v_1 + v_4 + v_8 = 1 + 1 + 3 = 5$ $V_{bottom,2} = v_2 + v_4 + v_8 = 1 + 1 + 3 = 5$ $V_{bottom,5} = v_5 + v_9 = 2 + 5 = 7$ $V_{bottom,3} = v_3 + v_6 + v_{10} = 1 + 1 + 10 = 12$ $V_{bottom,10} = v_{10} = 10$

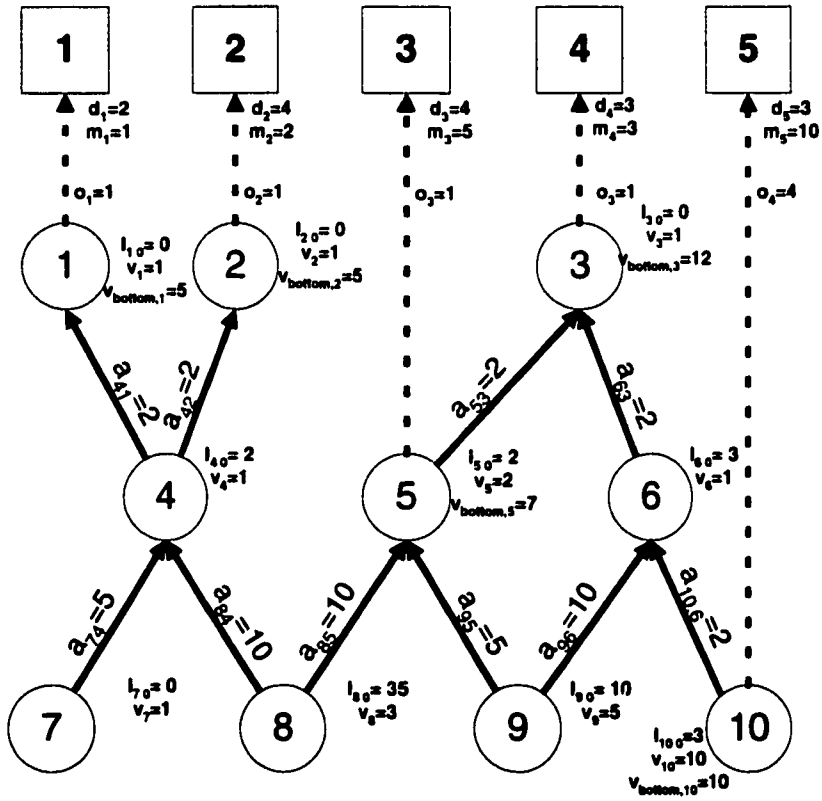


Figure 22. h = 1, Modified Graph with $v_{bottom,j}$ after Step 1

2. If h = 1, we can transform levels 0 and 1, into a level 0-1 MKP directly by using the General CBMSP transformation to MKP algorithm described in Section 3.4.1. In this transformation we consider the level 1 items as purchase parts with purchase time $v_{bottom,j}$ and ignore the other items in the hierarchy.	h = 1

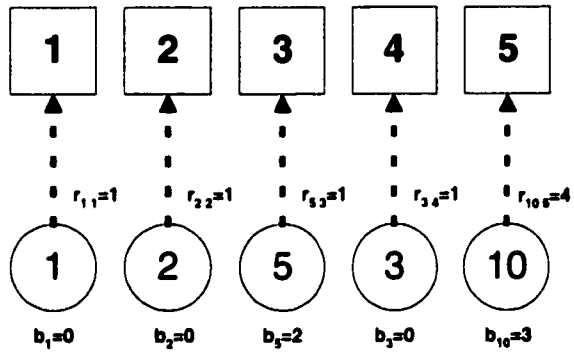


Figure 23. h =1, 0-1 MKP after Step 2

3. Solve the 0-h MKP from Step 2 using an exact or heuristic method such as the Greedy Hill-Climbing Algorithm described in Section 3.3.1. For each order k for which $x_k = 1$ (the order is filled on time) we	h = 1 Using the Greedy Hill-Climbing Algorithm. SOL = {3} ∴ Orders 3 can be delivered.

mark the order complete in our problem and remove it from the original hierarchy. In the original hierarchy, remove any inventory in the level h nodes, which has been used to make order k on time.	$I_{5\,0}$ is reduced by 1 Note: Because order 3 has been satisfied, parts 8 and 9 no longer exist at level 2.
4. Recall that some nodes in level h may also exist in other levels higher than h (i.e. levels h+1, h+2, ..., H). Consider the level h nodes, which no not exist in other levels high than h. For some of these nodes we can give away left over inventory ...	h = 1 $I_{1\,0} = I_{2\,0} = I_{3\,0} = 0$ ∴ no left over inventory in level 1. $I_{10\,0}$ is needed to assemble order 6 and also resides in level 3 ∴ this node is not considered.
5. If there are any level h nodes that need assembling (i.e. have children), then we increment h, and go back to step 1.	h = 2, goto Step 1
Step 1.	h = 2 I = {4, 5, 6} $V_{bottom,4} = v_4 + v_8 = 1 + 3 = 4$ $V_{bottom,5} = v_5 + v_9 = 2 + 5 = 7$

	$V_{bottom,6} = v_6 + v_{10} = 1 + 10 = 11$
2. Otherwise, if h > 1, then we will consider the hierarchy consisting of levels 0, 1, ..., h. We transform the levels 0,1, ..., h into a 0-h MKP using the General CBMSP transformation to MKP algorithm described in Section 3.4.1. In this transformation we consider the level h parts as purchased parts with purchase time $v_{bottom,j}$ and ignore the items not in levels 0,1, ..., h. We also ignore the inventories for items in levels 1,2, ..., h-1. When calculating the i to k paths, we only consider paths of exactly length h.	h = 2

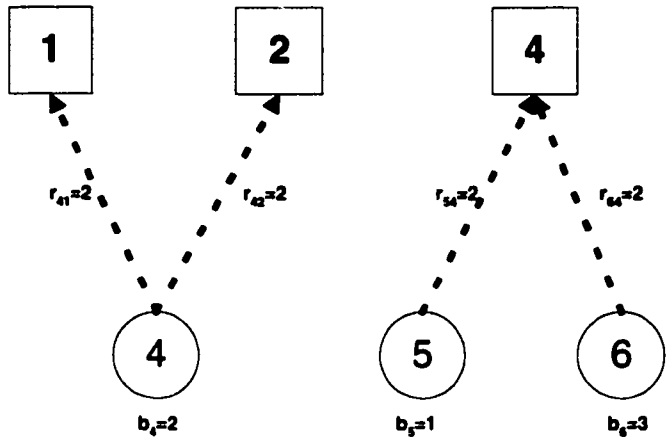


Figure 24. h = 2, 0-2 MKP, After Step 2

Step 3.	h = 2 Using the Greedy Hill-Climbing Algorithm. SOL = { 1 } ∴ Order 1 delivered. L_{10} is reduced by 2.
4. Recall that some nodes in level h may also exist in other levels higher than h (i.e. levels h+1, h+2, ..., H). Consider the level h nodes, which no not exist in other levels high than h. For some of these nodes we can give away left over inventory as	h = 2 Part 6 can be removed.

follows:	
a. Remove all nodes j at level h and their adjacent edges for which the inventory I_{j0} is enough to satisfy the demands r_{jk} of all connected orders k and which are not needed to assemble other parts.	

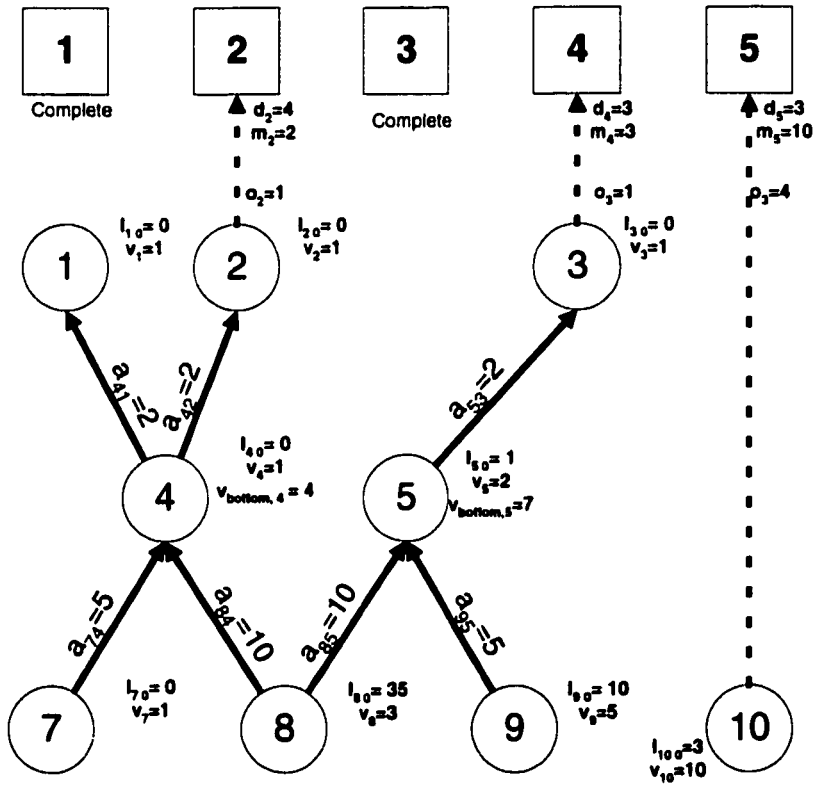


Figure 25. h = 2, After Step 4

Step 5.	h = 3, goto Step 1.

Step 1.	h = 3 I = {7, 8, 9, 10} $V_{bottom,7} = v_7 = 1$ $V_{bottom,8} = v_8 = 3$ $V_{bottom,9} = v_9 = 5$ $V_{bottom,10} = v_{10} = 10$
Step 2.	h = 3 Considering levels 0, 2, and 3.

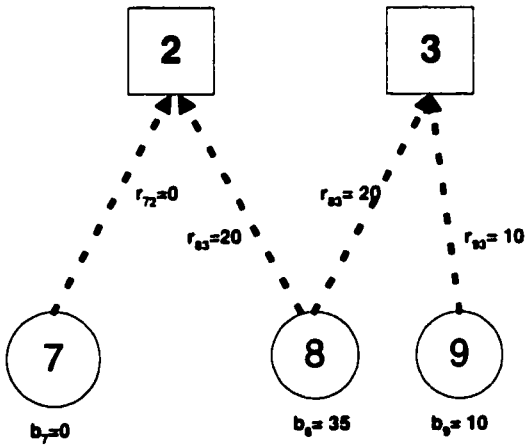


Figure 26. h=3, 0-3 MKP, After Step 2

Step 3	h = 3 Using the Greedy Hill-Climbing Algorithm. SOL = { 2 } Reduce I_{80} by 20

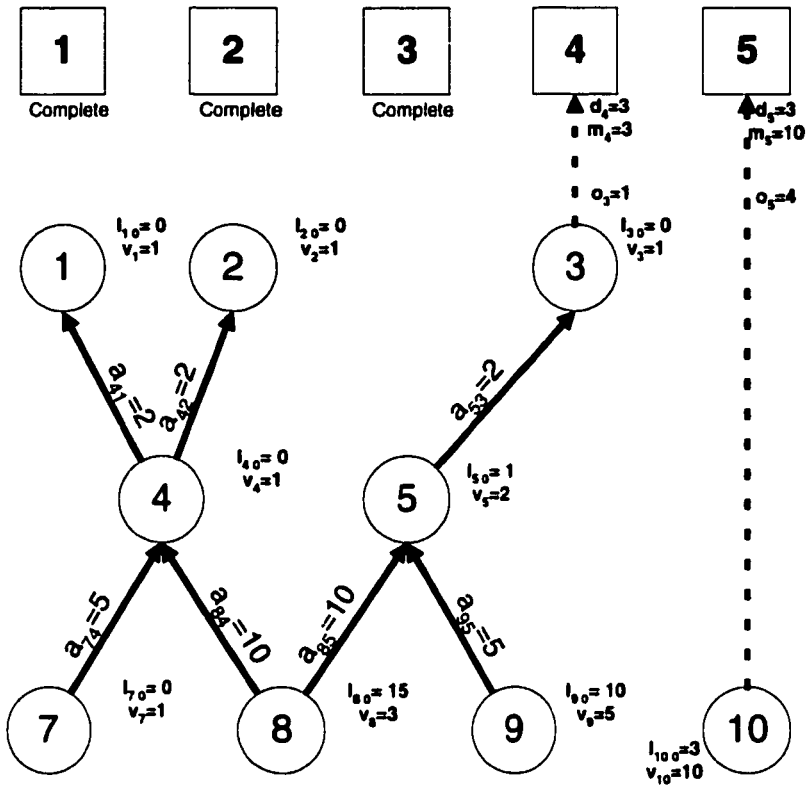


Figure 27. h=3. After step 3.

Step 4	h=3 Part 9 can be removed. {7,8,9,10} are all purchased parts, ∴ no inventory to push down to the next level.
5. If there are any level h nodes that need assembling (i.e. have children), then we increment h, and go back to step 1.	There are no more parts that require assembling with order 4 remaining.

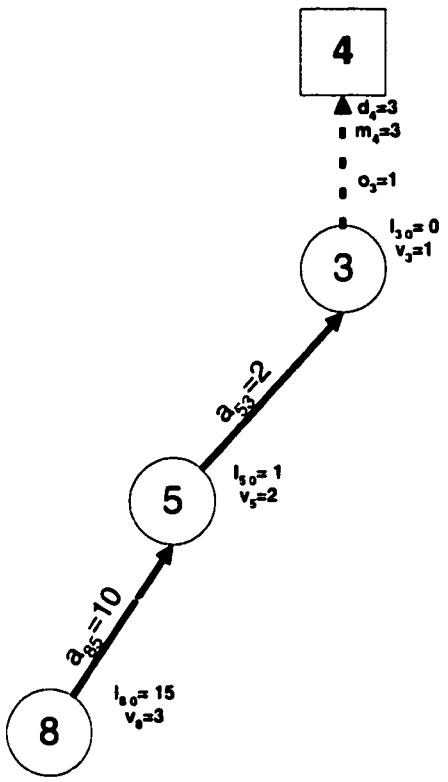


Figure 28. Order 4 part graph

6. Using the 0-H MKP, we select the next highest scoring unfilled order k computed by Step 3. Using this order k, we construct a graph consisting of only the parts required to satisfy this order such that for each part j, V_p-$v_j \leq d_k$ where V_p is the longest lead-time path from j to order k. The order k part graph contains only the left over inventory that can be used to	Order 4 remains h = H = 3 $r_{j4} > 0$ for all nodes j.

satisfy order k by the due date d_k. Using the order k part graph, do the following: a. For any purchased nodes in the order k part graph, we consider the 0-H MKP based on the original graph. If $r_{jk} = 0$, then we can supply as much inventory as needed from part j to satisfy the order k, therefore, set $I_{j0} = \infty$. Otherwise, we can only use the inventory I_{j0} currently available in level H nodes	

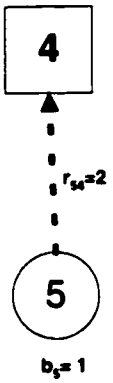


Figure 29. h=3, 0-2 MKP before Step 6b

6b. Otherwise, create the 0-(h-1) MKP from the order k part graph consisting of	h = 3

parts i which are assembled from parts j that exist only in level h. Next, rank each part i in level h-1 in decreasing order by b_i/r_{ik}. This ordered list permits us to give inventory to parts i in level h-1, which are closest to meeting the demands placed upon them. For each parts i in the ordered list we do the following: If $r_{ik} < b_i$, then go to the next part i. Otherwise, for each connected part j which exists only in level h compute:

$$w_{ji} = \left\lfloor I_{j0} / a_{ji} \right\rfloor$$

Let $build_i = \min (w_{ji} : \text{where j is a connected part in level h of part i in level h-1})$. If $build_i > r_{ik}-b_i$, then let $build_i = r_{ik} - b_i$. Next, adjust all inventories by adding $build_i$ to b_i and I_{i0}, and decreasing the inventories I_{j0} for sub-parts j by $a_{ji} \cdot build_i$. We proceed to the next part i in the ordered list until all h-1 have been considered.

$$w_{85} = \left\lfloor I_{80} / a_{85} \right\rfloor = 1$$

$build_5 = \min(w_{85}) = 1$

$I_{50} = I_{50} + build_5 = 2$

$b_5 = I_{50} = 2$

$I_{80} = I_{80} - build_5 \cdot a_{85} = 15 - 10 = 5$

6c. Check to see if $b_i \geq r_{ik}$ for all nodes i in level h-1. If so, order k is now satisfied and can be marked complete. If not, we decrement h and go to step b.	$b_5 \geq r_{54}$ (refer to step 6b) $\therefore$ Order 4 can be delivered No more orders remain so we stop.

4.4 Interpreting the Results

Using the method for interpreting the solution for the MKP as a solution for the flat CBMSP(Section 3.5), it is possible to interpret the solution for the previous example as follows:

1. The Satisfied Orders: The solution set = {1,2,3,4}

2. When the Order was Satisfied:

Order 1	Order 2
Part 4 had enough inventory to meet the demand $r_{41} \leq b_4$ (0-2 MKP - Step 3).	Part 8 had enough inventory to meet the demand $r_{82} \leq b_8$ (0-3 MKP, Step 4).
$\therefore V_{4\rightarrow 1}$-$v_4$ = 2-1 = 1	$\therefore V_{8\rightarrow 2}$-$v_8$ = 5-3 = 2
The order was delivered in period 1.	Part 7 had r_{72} = 0 (0-3 MKP, Step 3) $\therefore V_{7\rightarrow 2} = 3$ The ordered was delivered in period 3.

Order 3	Order 4
Part 5 had enough inventory to meet the demand $r_{53} \leq b_5$ (0-1 MKP - Step 3).	Part 3 had some of the inventory that had to be pushed up from parts 5, 8 and 9. (Step 6)
$\therefore V_{5\rightarrow 3}\text{-}v_5 = 3\text{-}2 = 1$	$\therefore V_{5\rightarrow 4}\text{-}v_5 = 1$
The order was delivered in period 1.	$\therefore V_{8\rightarrow 4}\text{-}v_8 = 3$
	$\therefore V_{9\rightarrow 4}\text{-}v_9 = 3$
	The order was delivered in period 3.

3. The Parts Used:

Order 1 required part 4;

Order 2 required parts 4, 7, and 8;

Order 3 required part 5;

Order 4 required parts 5, 6, 8, and 9; and

4. The Quantity of Parts Needed:

Order 1 required: $o_1 = 1$ of part 1; and $o_1 \cdot a_{41} = 1 \cdot 2 = 2$ of part 4.

Order 2 required: $o_2 = 1$ of part 2; $o_2 \cdot a_{42} = 1 \cdot 2 = 2$ of part 4; $o_2 \cdot a_{42} \cdot a_{74} = 1 \cdot 2 \cdot 5 = 10$ of part 7; and $o_2 \cdot a_{42} \cdot a_{84} = 1 \cdot 2 \cdot 10 = 20$ of part 8.

Order 3 required: $o_5 = 1$ of part 5.

Order 4 required: $o_5 = 1$ of part 3; $o_5 \cdot a_{53} - I_{5\,0} = 1$ of part 5; $o_3 \cdot a_{63} = 2$ of part 6; $(o_5 \cdot a_{53} - I_{5\,0}) \cdot a_{85} = 1 \cdot 10 = 10$ of part 8; and $(o_5 \cdot a_{53} - I_{5\,0}) \cdot a_{85} = 5$ of part 9.

5. The Quantity of Parts Purchased:

Order 2 required 10 of part 7, all of which was purchased.

4.5 Complexity Analysis for the Heuristic for CBMSP

Recall that the heuristic for CBMSP utilizes:

- The general CBMSP transformation to MKP (Section 3.4.1) which was shown to be **O(|V| + |E|)** ; and
- The Greedy Hill-Climbing heuristic (Section 3.3.1) for solving the MKP which was shown to be $\mathbf{O(|K|^2 |I|)}$.

These two components were executed as the graph was iterated through level by level by the heuristic. During each iteration the following was performed:

Step	Complexity
Determine the max lead-time paths (Step 1)	**O(|V| + |E|)** A Depth-First-Search (DFS) was performed.

A general transformation to 0-h MKP	**$O(|V| + |E|)$** This is an absolute worst case in which the entire graph must be traversed for each transformation to 0-h MKP, which would not happen.
Solving the resulting 0-h MKP	**$O(|K|^2 |I|)$ or $O(|V|^3)$** Again, this is a worst case in which the graph is fully connected including orders and parts.
Pushing down Inventory	**$O(|V|)$** This case assumes that the graph consists of one path from each order k to purchased parts.

This will repeated for each level of in the part hierarchy up to level h=H. The overall complexity for the heuristic excluding the push up step is **$O(h \cdot |V|^3)$**. In the last step of the heuristic, the following occurs:

Step	Complexity
Pushing Up Inventory	**$O(h \cdot |V|^3)$**

	This steps involves forming and solving 0-(h-1) MKP for each level h = {0,1,..., h-1} in the graph. In the worst case, all orders must be considered and the graph is fully connected.

Overall, the complexity for the CBMSP heuristic is $\mathbf{O(h \cdot |V|^3)}$. It should be noted that this complexity is pessimistic, in that it assumes that no orders were solved via steps 1 through 5 of the CBMSP heuristic, and all inventory had to be "pushed up" using step 6.

5 Application in Industry

Using the CBMSP heuristic presented in Chapter 4, the next step was to provide an implementation and test it against production data provided by a local area company. This testing would take the heuristic and pit it against a basic approach that transforms the general CBMSP into the 0-H MKP (Section 3.4.1) and solves it using the Greedy Hill-Climbing heuristic (3.3.1). In the worst case, where there was no inventory present in the part hierarchy, the CBMSP heuristic would achieve the same solution as the basic approach. The goal of these tests were to show how much better the CBMSP heuristic would do in the average case than an approach that ignored inventory internal to the part hierarchy.

5.1 The CBMSP Application

In order to test both the original version of the Flat CBMSP heuristic and the revised CBMSP heuristic, both were implemented in an application written in C++ using Microsoft Foundation Classes. This application has the following features:

- A fully functional graph editor supporting the creation, deletion, moving of nodes (parts and orders), and edges that make up the part hierarchies;
- The ability to load and save part graphs in binary format;
- The ability to import graphs from real life customer data;
- The ability to optimize the layout of these graphs;

- The ability to execute each of the heuristics and interactively review their results; and

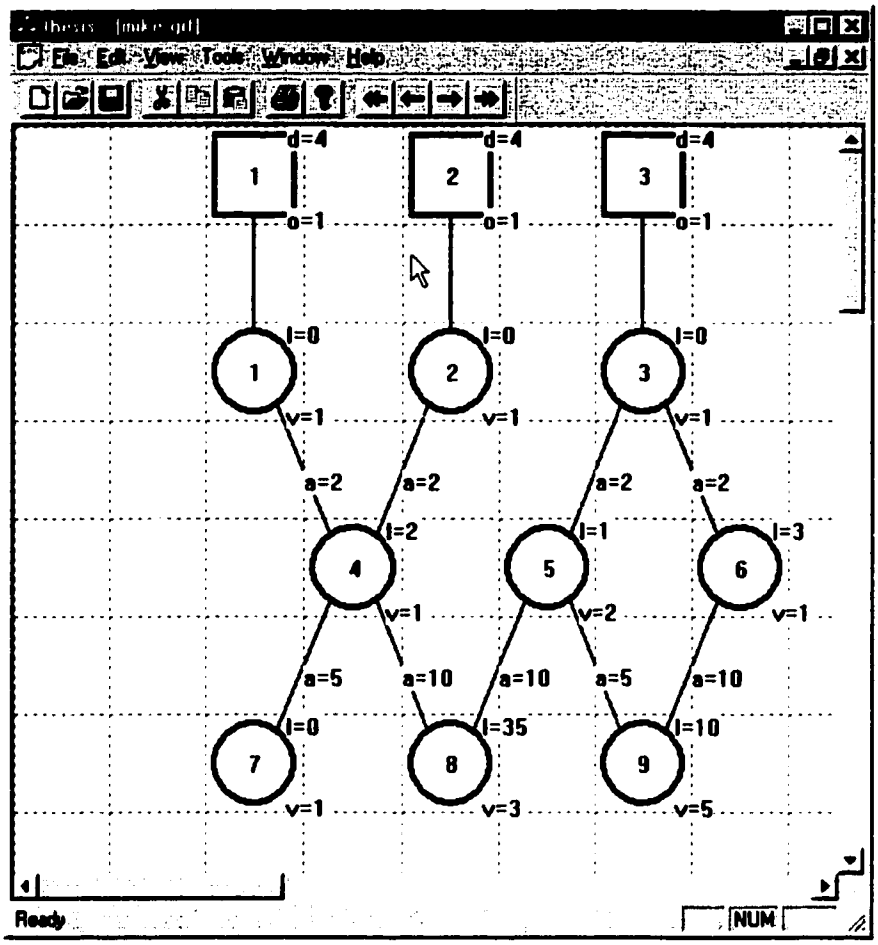


Figure 30. The CBMSP Application

5.1.1 Implementation of Part Graph

The graph used to represent the part hierarchies, and orders, was based upon the following class diagram:

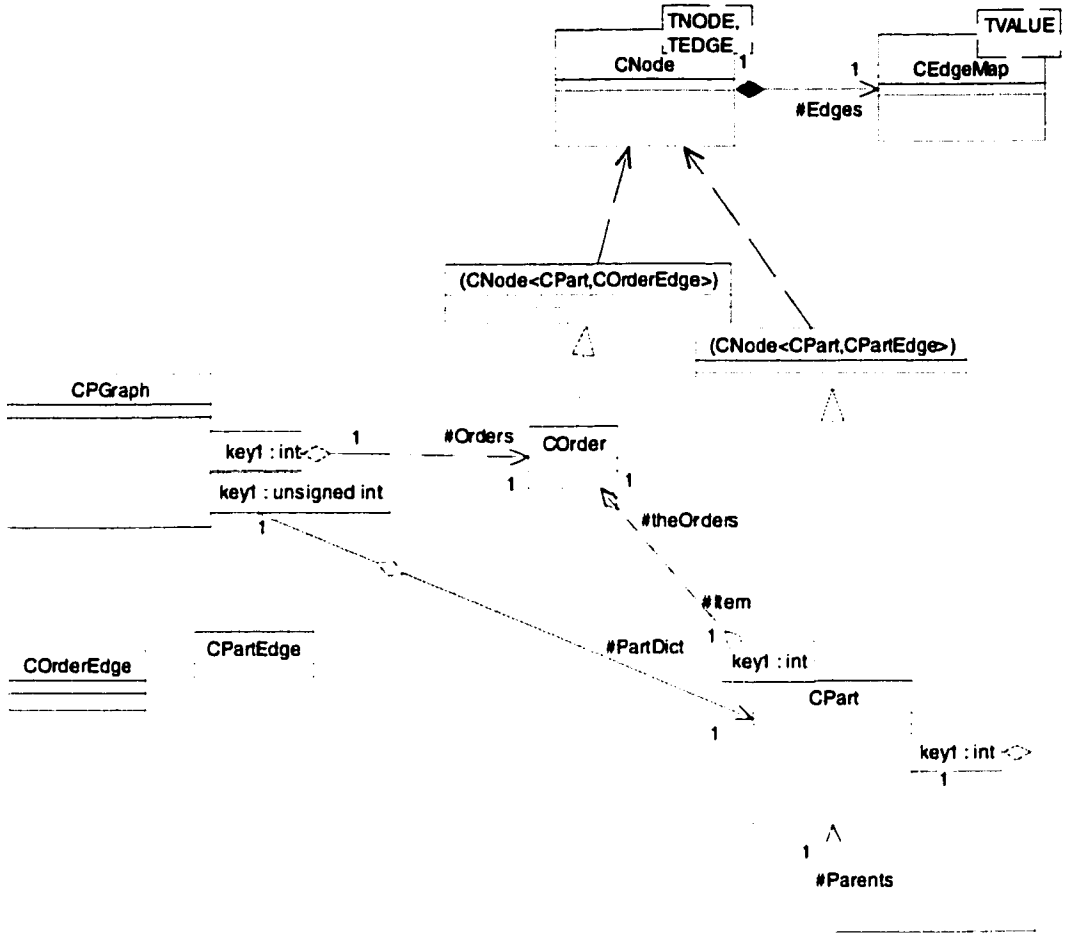


Figure 31. UML Class Diagram for Graph

Class	Purpose
CPGraph	The part graph derived from CGraph. This class handles all parts and orders that exist in the graph.
CNode	The template class from which the parts and orders are derived. This class

	implements the edge associations through CEdgeMap
COrder	The Order Class. This class derives from CNode and contains references to one or more Parts through CEdgeMap. In addition, this class contains specific information about Due Date, Qty ordered, and the ordered part.
CPart	The Part Class. This class derives from CNode and contains specific information about inventory and lead-time.
CEdgeMap	A class that manages a "keyed" index of either COrderEdges or CPartEdges to other parts in the hierarchy.
COrderEdge/CPartEdge	Edge classes, which contain specific information about required inventory, and edge lead-times.

5.1.2 Implementation of Flat and Revised CBMSP Heuristics

The implementation of both the flat and revised CBMSP heuristics use the graph implementation described in the previous section.

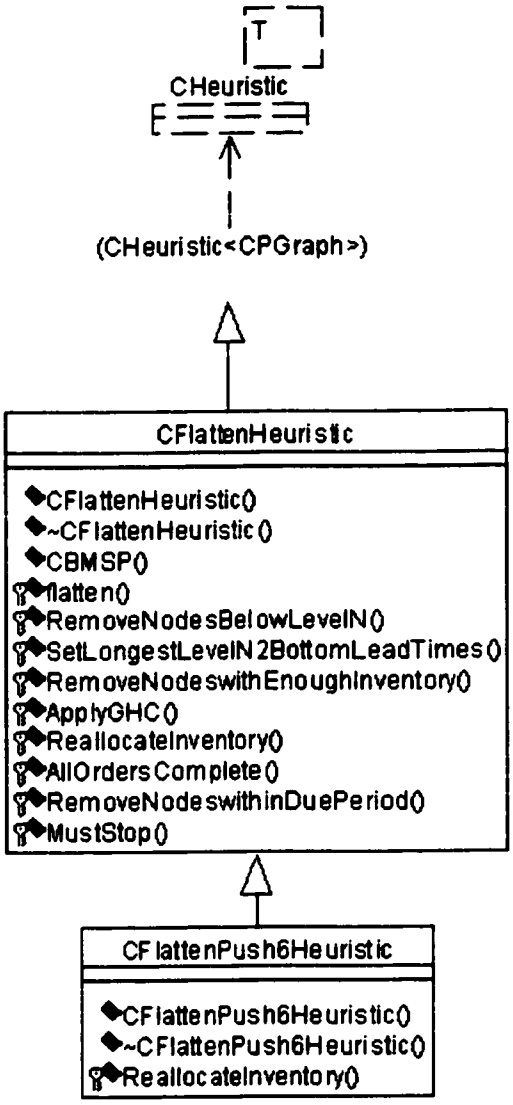


Figure 32. UML Class diagram for the Heuristics

The heuristics were implemented such that specific steps of the algorithms could be overridden through inheritance. Each class is defined as follows:

Class	Purpose

CflattenHeuristic	Implements the original CBMSP heuristic with support for Multiple *Level* part hierarchies.
CFlatten6Heuristic	Implements the revised Inventory Reallocation or "push" and "pull" enhancements
CDJHeuristic	Implements the Greedy CBMSP algorithm.

To execute each heuristic an instance of the heuristic is created given an instance of a CPGraph. The heuristic is then executed on the graph.

5.2 Manugistics

Research into this area began at the request of a company called Manugistics[23] Their solutions include support for a number of features for supply chain management including Constraint-Based master planning.

5.3 Running CBMSP on their Data

Manugistics provided us with 3 test cases to use to test the implementation of the revised CBMSP heuristic. They are described as follows:

Test Case	Number of Parts	Number of Connections	Number of Orders	Number of Levels

DB1	44909	148339	5769	9
DB2	10784	43104	6691	7-8
DB3	101	160	47	4

5.3.1 Mapping the Data to the CBMSP Model

There were a number of differences between the test data and the model defined in this paper. The most significant difference was the fact that the test data contained more information about the part hierarchy that was accounted for in the general definition of the model. In addition, the structure of the test cases differed and required some cleaning/conversion so that the CBMSP application could import the part graphs.

5.3.1.1 Test Case 1: DB1 and DB2

The test data came in the form of 6 flat ASCII files:

5.3.1.1.1 Description of the Data

5.3.1.1.1.1 Item Master Table (IM)

This table defines the set of parts in the part hierarchy.

Field names are as follows:

Field	CBMSP	Description

IM-PARTIDX	Part	Part number
IM-ONHAND	Inventory	Quantity on hand
IM-TYPE	N/A	P=purchased, A=assembled, T=transferred (treat T as assembled?)
IM-PUR-LT	Lead-time	Purchase lead time (if type=P)
IM-PROD-LT	Lead-time	Production lead time (if type=A)
IM-TCOST	N/A	Cost of this part
IM-MS-TYPE	N/A	P=pseudo, N=not a pseudo (pseudo's are "virtual" parts)

The fields IM-TCOST, and IM-MS-TYPE did not map to the CBMSP model. If the IM-TYPE was P(Purchased), then IM-PUR_LT was used as the lead time for the part. If it was A(Assembled), then IM-PROD-LT was used as the lead time. The concept of transferred parts (the case were IM-TYPE = T(Transferred)) was not supported in the CMBSP so the type was treated as an assembled part using IM-PROD-LT.

5.3.1.1.1.2 Product Structure Table (PS)

This table contains the list of edges defining the connections between the parts in the part hierarchy.

Field names are as follows:

Field	CBMSP Equivalent	Description
PS-PARIDX	Part	Parent part number
PS-COMIDX	Part	Component (child) part number
PS-QPA	a_{ij}	Quantity per assembly

5.3.1.1.1.3 Customer Orders Table (CO)

This table reflects the customer orders placed on the part hierarchy.

Field names are as follows:

Field	CBSMP Equivalent	Description
PO-PARTIDX	Part	Part number
PO-REQ-DT	Due Date	Required date
PO-QTY	Quantity	Quantity

PO, MS and EU tables are identical to the CO table.

The PO table describes a schedule for receiving purchased parts. The CBMSP model doesn't maintain this information, as these parts in themselves are orders for parts not contained within the hierarchy.

The MS table is used for order forecasting to estimate when orders will occur. The CBMSP has no provision for order forecasting.

5.3.1.1.2 Data Conversion Experiences

In order to read in the data, a parser was added to the CBMSP application which read the IM.rp, PS.rp and CO.rp respectively. The next step was to eliminate all parts from the graph that were orphans. These were parts having no parents or children. In addition, parts that were not contributing to an order (i.e. having no high level parts with a customer order associated with them) were also removed. The resulting graph was then formatted for display in the graph editor.

Once the graph was loaded, a number of observations were made. These included:

- Several thousand parts in the hierarchy had zero or production lead-times specified;
- A number of parts had negative inventory;
- The due dates for all orders were expressed as MM/DD/YYYY, and did not include a starting date. In addition the data contained several years of orders;
- Of the total number of orders in the CO.rp tables, many parts were ordered several times. In some cases, parts were ordered several times in the same day.
- The part hierarchies were too rich with respect to inventory. In many cases, the ordered part had enough inventory to satisfy all orders for it. In other cases, the ordered parts were purchased parts; and

- Parts existed at one or more levels in the part hierarchy.

Given this, an attempt was made to extract a sub-graph from DB1 and DB2 that would provide an "interesting" case for the CBMSP. This sub-graph attempted to avoid parts with negative orders and zero lead-times. A sub-set of the orders was identified by choosing a calendar date and converting all order due-dates to an integer offset from that date. For the purpose of the test, the subset of orders contained as few duplicates as possible in an attempt to satisfy the other constraints on the part hierarchy.

In short, this proved to be a challenge. For the most part the graph lacked depth. In the cases where there was a hierarchy present, it contained zero lead-time parts and had too much inventory. Due to the size of the data files for both of these examples, and the simplicity of the hierarchies, a request for another example was made to Manugistics.

5.3.1.2 Test Case 2: DB3

The test data came in the form of 9 flat ASCII files:

5.3.1.2.1 Description of the Data

5.3.1.2.1.1 Item Master

This table contains the key information about the parts in the hierarchy.

Field	CBSMP	Description
ID	Order Id	Order Identifier

Site ID	N/A	Site of the Part
Description	N/A	Description
On-Hand	Inventory	The amount of inventory on hand.
Source	Assembled or Purchased	BUY, MAKE, TRANSFER
Lead Time Type	N/A	Indicates if the lead-time is fixed.
Lead Time	Lead-time	Integer purchase, manufacture, or transfer lead-time.
Buffered Lead Time	N/A	Buffered lead-time
Variable Lead Time	N/A	Variable lead-time
Move Lead Time Setup Lead Time Queue Lead Time	N/A	Lead-times for manufactured items
Order Policy	N/A	L4L, POS (lot-for-lot, periods of supply)

Order Days	N/A	Number of days for fixed days order policy.
Base Order Quantity Min Order Quantity Max Order Quantity Mult Order Quantity	N/A	Order Policy parameter
Shrinkage	N/A	Shrinkage factor This number is a percentage that represents losses occurring during manufacturing. Any demand for an item should be inflated by this amount. e.g., If shrinkage = 5%, then NetDemandQty = OrigDemandQty * 1.05.
Yield	N/A	Yield factor This number is a percentage that represents the effective yield from a manufacturing process. It is used only if an entry in the Product Structure table, where the current item is the parent item, has a zero value.

Planning Fence	N/A	Planning fence Planning fences limit when you can start producing an item.
Safety Code	N/A	None, fixed qty, lead time, number of days
Safety Qty	N/A	qty of safety stock

Of the fields in this table, only ID, On-Hand, Source, and Lead-Time were mapped to the CBMSP model. The remaining fields contain information not supported by the CBMSP.

5.3.1.2.1.2 Product Structure

This table describes the parent-child relationships as well as the build of material relationships between parts.

Field	**CBSMP**	**Description**
Parent Item ID	Parent Part Id	reference Item Master for parent
Parent Site ID	N/A	site of parent item
Child Item ID	Sub-Part	reference Item Master for child
Child Site ID	N/A	site of child item
Start Effectively Date	N/A	The dates in which this records is in effect.

End Effectively Date		
QPA	a_{ij}	quantity per assembly
Mix	N/A	Mix factor
Shrinkage	N/A	Shrinkage factor
Yield	N/A	Yield factor
Offset	N/A	Lead-time offset

Of the fields in the Product Structure table, Parent Item Id, Child Item Id, and QPA were mapped to the CBMSP. Like the item master table, the remaining fields are not supported by the CBMSP model.

5.3.1.2.1.3 Customer Orders

This table contains the list of customer orders.

Field	**CBSMP**	**Description**
ID	Order Id	Order Identifier
Item ID	Part	Name of the ordered Part
Site ID	N/A	Site of the Part
Quantity	Quantity	Amount of part required.

Due Date	Due Date	The due date of the order.
Request Date	N/A	The requested date of the order.
Priority	Priority	The priority of the order.

In the Customer Order table, ID, Item Id, Quantity, Due Date and Priority were mapped to the CBMSP model.

5.3.1.2.1.4 Other Tables

There were several other tables included in this test case. None were mapped to the CBMSP as they either contained information that was not supported by the CBMSP model, or customer order information that was accounted for by the customer order table.

1. **Forecast**: This table contains the list of forecast orders. It is similar to the customer order table and can be treated as a set of external demands for parts.

2. **Inventory**: This table provides the current inventory of the parts in the hierarchy. It can be used instead of the on-hand field in the Item Master table.

3. **Purchase Order**: This table provides a schedule for supplies for purchased items.

4. **Purchase Requisition**: This is another schedule for supplies for purchased items. It is treated separately because requisitions have not been released to suppliers.

5. **Work Order Supply**: This is a schedule of supplies for assembled parts.

6. **Work Order Demand**: This is a demand hierarchy for each work order supply. This table is used instead of the product structure relationship because a work order might be partially "kitted" (i.e. some child items are already allocated to that work order). Another reason is that the work order may have different parent-child relationships than the product structure.

5.3.1.2.2 Data Conversion Experiences

Unlike DB1 and DB2, this test case had a string identifier for each order and part in the hierarchy. These had to be converted to a unique numerical code in order to be able to import the hierarchy into the CBMSP application. The tables of the test cases were imported into a relational database, and then SQL was constructed to convert them to the format of the customer order, item master, and product structure tables used for DB1 and DB2.

Unlike DB1 and DB2, DB3 does not have any negative inventory, nor zero lead-times. In addition, the hierarchy does not have so much inventory as to not challenge the CBMSP heuristic. Of the 47 customer orders, only 7 are for distinct parts.

Another advantage of DB3 over DB1 and 2 is that due its smaller size, it is easier to test and analyze the graph. The size of the DB1 and 2 made it time consuming to analyze the graph for any features that could not be supported by the CBMSP model and heuristic.

5.3.2 Test Results

5.3.3.1 Test Description

For the purposes of testing the CBMSP heuristic described in Section 4.1, the heuristic was implemented in the CBMSP application described in Section 5.1. It was tested against an approach that simply performed the General CBMSP Transformation to MKP (Section 3.4.1) from all orders to purchased parts at the bottom of the hierarchy. This transformation was followed by executing the Greedy Hill-Climbing Method (Section 3.3.1). The goal of the tests was to illustrate the amount of improvement the CBMSP heuristic provides through it's level-wise flattening of the part hierarchies.

The tests were performed using DB3 provided by Manugistics. This test case provided an "interesting" part hierarchy in which to execute the heuristic. As was mentioned previously this test consisted of 101 parts, and 47 orders, with the part hierarchy being 4 levels deep.

The test case was executed by the CBMSP heuristic and the base line approach described above. The solution set obtained by each approach was subsequently compared to determine the number of satisfied orders.

Observations

The results of the testing are presented in the following table:

Test	Approach	Satisfied Orders (47)
DB3	CBMSP	24
DB3	Basic Approach	9

As shown in the above table, the CBMSP heuristic performed significantly better than the Basic Approach satisfying 24 orders of 47 where the basic approach could satisfy 9 of the orders. The main reason for this result was that the level-wise processing of the part hierarchy made better use of available inventory. It should be noted that for DB3, the "push up" leftover inventory step of the algorithm (i.e. Step 6 of the CBMSP) had no affect on the final result (i.e. for this example the opportunity to "push up" leftover inventory never arose).

6 CBMSP with Dynamic Supply

6.1 The Flat CBMSP with Dynamic Supply

The model we have considered thus far for the supply of parts assumes each part $i \in I$ has an initial inventory b_i and additional supply of part i may be purchased with a lead time v_i. We can extend this model to include dynamically supplying the inventory of parts i. The dynamic supply model assumes that in addition to being able to purchase part i in time v_i there is also an amount S_{i_t} of item i supplied in time period t for t = 0,1, ..., T.

6.2 Representing a Dynamic Flat CBMSP as an MKP

It is possible to represent the dynamic flat CBMSP supply problem as a MKP by using ideas presented in [1][3]. To do this we create a copy i_t for every item $i \in I$ for every time period $t \in T$. For each part i_t b_{i_t} is defined as the accumulated total supply of i which has been supplied over time-periods 0 to t inclusive, i.e. $b_{i_t} = \sum_{j=0}^{t} s_{i_j}$ for all $i \in I$, $t \in T$. This means that once the supply i is used up at time period t, we must take into account that it is used up out of the accumulated supply for periods after t as well. Also, b_{i_0} reflects the

---

[3] The model described in [1] differs from the one described here in that it replaces the purchased parts with a supply vector over time t. In addition model for the dynamic flat described here allows the order parts m_k to be purchased.

initial inventory at time t = 0. The flat CBMSP with dynamic supply can be modeled as an MKP as follows:

$$\text{maximize} \quad \sum_{k=1}^{K} p_k x_k$$

$$\text{subject to} \quad \sum_{k=1}^{K} r_{i,k} x_k \le b_{i_t}, \text{for all } i = 1,2,...,I, t = 0,1,...,T$$

$$\text{where } x_k \in \{0,1\}, k = 1,...,K.$$

Here $r_{ik} = \begin{cases} \infty & \text{if } d_k > T, \\ o_k & \text{if } t \ge d_k \text{ and } i = m_k \text{ and } v_i > d_k, \\ 0 & \text{otherwise.} \end{cases}$

for all $i \in I, t \in T$, and $k \in K$.

Here, $r_{i,k}$ represents the amount of resource i needed from total supply of i in time period t. When it is convenient, we will use r_{ik} to denote the amount of resource i needed to satisfy order k for time periods $t \ge d_k$. Therefore, $r_{ik} = o_k$ for periods $t \ge d_k$ where there exists a path from part i to ordered part m_k such that the lead time v_i is greater than the due date d_k.

6.2.1 Example

Consider the following example:

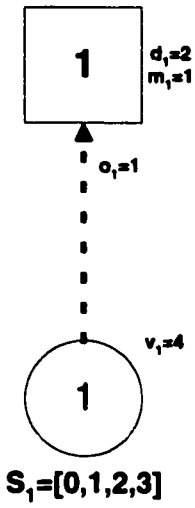


Figure 33. Inital graph.

In this example, there one customer order and one part. In this example, t=3, so S_1 reflects the inventory arriving for part 1. For example, $S_{1_2} = 2$, indicating that when t=2, 2 units of part 1 become available. The part starts out with no initial inventory. If we transform this into a MKP, the transformed graph appears as:

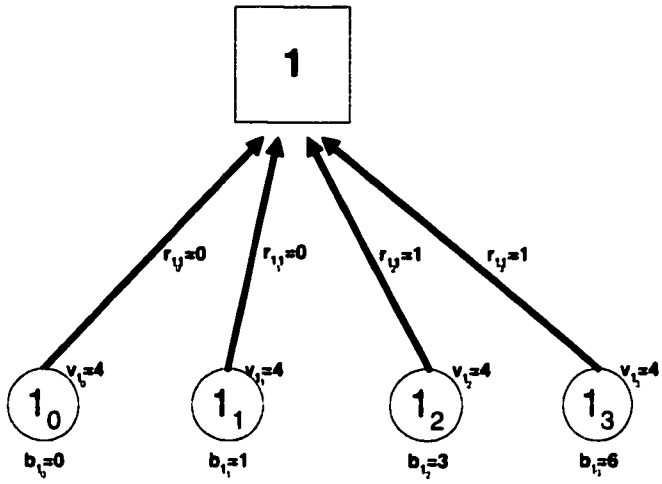


Figure 34. After transformation to a MKP

In this example, $r_{1,1} = o_1$, for $t \geq \{d_1 = 2\}$. This means that for nodes 1_0, and 1_1 $r_{1_0,1} = 0$, and $r_{1,1,1} = 0$. Notice that $b_{1_1} = 6 = b_{1_1} + b_{1_2} + b_{1_1} + b_{1_0}$ where b_{1_0} is the initial inventory in the graph. Because $r_{1,1} = o_1$, for $t \geq \{d_1 = 2\}$ it is possible to simplify the representation of the graph by introducing a "super node" for which r_{ik} apply. We can remove these edges from the graph when $r_{i,k} = 0$.

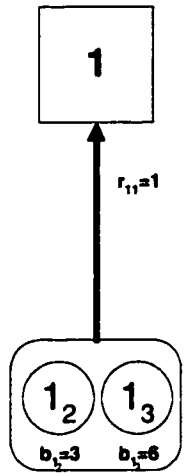


Figure 35. Simplified MKP graph with "super nodes"

Once the transformation to MKP has been completed, it can be solved using a heuristic like the one described in section 3.3.1.

6.3 Transforming General Dynamic CBMSP into a MKP

If we ignore internal inventory, and supply, it is possible to transform the Dynamic General CBMSP into a MKP similar to what was done in Section 6.2. In order to perform the transformation, a copy i_t of each purchase part $i \in I$ for $t \in T$ in the original graph is created. As mentioned in Section 6.2. Recall that b_{i_t} is equal to the accumulated inventory received up to and including time-period t. This accumulated inventory includes the initial inventory (i.e. when t = 0).

Subsequently, the transformation of general CBMSP to MKP described in 3.4.1 can be employed. The calculation of $r_{i,k}$ is determined as:

$$r_{i,t,k} = \begin{cases} \infty, & \text{if } d_k - (V_p - v_i) < 0 \\ \sum A_p, & \text{where P is a path from i to } m_k \text{ s.t. } V_p - v_i \le d_k, V_p > d_k, \text{ and } t \ge d_k - (V_p - v_i) \\ 0, & \text{when } V_p \le d_k \text{ for all paths P from i to } m_k \text{ or there is no path from i to } m_k. \end{cases}$$

for all $i \in I$, $t \in T$, and $k \in K$.

Here, the $r_{i,t,k}$ is equal to $\sum A_p$ where an i to m_k path P exists such that the longest lead-time path V_p not including v_i is less than the due date d_k. V_p itself must be greater than d_k, while t must be greater than the due date d_k minus longest lead-time path V_p not including v_i.

6.3.1 Example

Consider the following example:

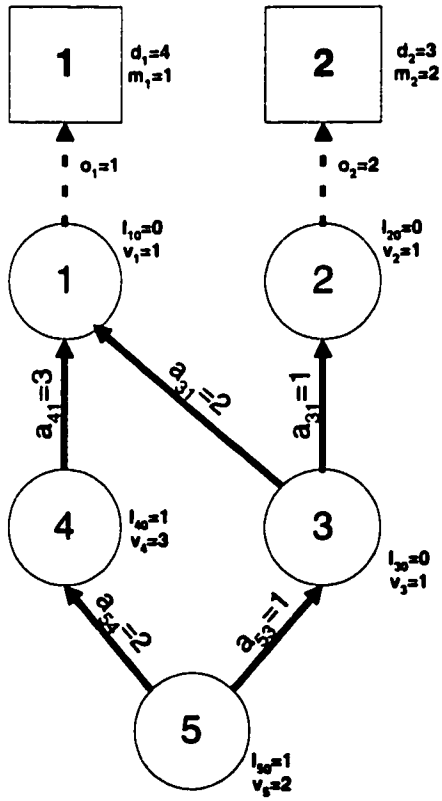


Figure 36. Initial Graph

In this example, $S_1 = [0, 0, 0, 0, 0]$, $S_2 = [0, 0, 0, 0, 0]$, $S_3 = [0, 0, 0, 0, 0]$, $S_4 = [0, 0, 0, 0, 0]$, and $S_5 = [1, 1, 1, 0, 1]$.

Applying the transformation of the general Dynamic CBMSP into a MKP we would get:

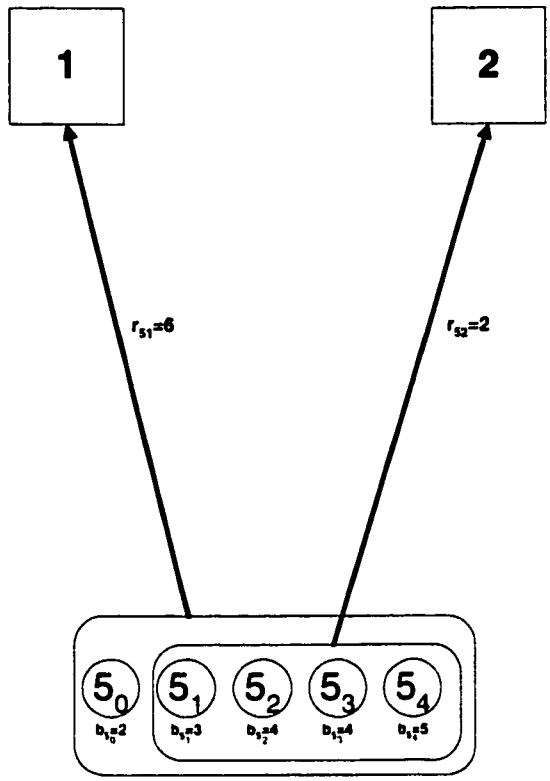


Figure 37. h=3, After Step 3.

Since $d_1 - (v_1 + v_4) = 0$, $\therefore$ $r_{51} = 6$ for $t \geq 0$. Because $d_2 - (v_2 + v_3) = 1$, $r_{52} = 2$ for $t \geq 1$.

6.4 The Dynamic CBMSP

There are a number of modifications to the CBMSP heuristic presented in Section 4.1.2 to support dynamic supply of part inventory. The CBMSP heuristic focuses on being able to satisfy customer orders with available part inventory. In the case where there is

insufficient inventory, the heuristic makes an overall worst case evaluation of the total time it takes to purchase and assemble parts in order to complete a customer order. By assessing the due date (d_k) with respect to the maximum lead-time path from customer order to purchase part, the heuristic decides whether or not it is possible to satisfy the order in the maximum allowable time.

Through the introduction of dynamic supply, comes the possibility to satisfy customer orders in less than the maximum allowable time to satisfy the customer order. As parts in the hierarchy receive new inventory, which doesn't happen in the CBMSP described in Chapter 4, it reduces the requirement to purchase and assemble parts up the length of the entire part hierarchy. This reduces the overall time to provide the parts necessary to satisfy the customer orders.

6.4.1 Changes to the Heuristic

Below is the heuristic presented in Section 4.1.2 with the changes required to support dynamic supply highlighted:

For each level h = 1, 2,..., H perform the following:

1. **Compute the accumulated inventory $AI_{i,t}$ for each part $i \in I$ and $t \in T$ as follows:**

$$AI_{i,t} = \sum_{j=0}^{T} S_{i_j}$$

where AI_{i_0} is the initial inventory, and S_i is the supply vector for part i over time t. In addition, AI and S are defined as the accumulated inventory matrix, and supply matrix respectively.

2. For each node j in level h, find a longest bottom (purchased part) to j node path, where the length of the path is identified as the sum of the v_i's for all nodes i in the path, including j. Let the label $v_{bottom,j}$ for j be the length of this longest path. *This represents the earliest time that part j can be ready assuming everything below it is assembled or purchased.*

3. If h = 1, we can transform levels 0 and 1, into a level 0-1 MKP directly by using the **Dynamic** General CBMSP transformation to MKP algorithm described in Section 6.2. In this transformation we consider the level 1 items as purchase parts with purchase time $v_{bottom,j}$ and ignore the other items in the hierarchy. Otherwise, if h > 1, then we will consider the hierarchy consisting of levels 0, 1, ..., h. We transform the levels 0,1, ..., h into a 0-h MKP using the **Dynamic** General CBMSP transformation to MKP algorithm described in Section 6.3. In this transformation we consider the level h parts as purchased parts with purchase time $v_{bottom,j}$ and ignore the items not in levels 0,1, ..., h. We also ignore the inventories for items in levels 1,2, ..., h-1. When calculating the i to k paths, we only consider paths of exactly length h.

4. Solve the MKP representing the flat level 0-h CBMSP from Step 3 using an exact or heuristic method such as the Greedy Hill-Climbing Algorithm described in Section 3.3.1. For each order k for which $x_k = 1$ (the order is filled on time) we mark the

order complete in our problem. In the original hierarchy, remove any inventory in the level h nodes, which has been used to make order k on time.

5. Recall that some nodes in level h may also exist in other levels higher than h (i.e. levels h+1, h+2, ..., H). Consider the level h nodes, which do not exist in other levels high than h. For some of these nodes we can give away left over inventory as follows:

   a. Remove all nodes j_t at level h and their adjacent edges for which the inventory b_{j_t} (from the MKP in Step 3) is enough to satisfy all demands $r_{j_t k}$ of all connected orders k and which are not needed to assemble other parts.

   b. For each node j in level h with leftover inventory, we determine if connected nodes u in level h+1 have j as the only parent. If so, **recalculate the accumulated inventory vector** for each node u in level h+1 by "disassembling" the parts and "pushing" the inventory down to node m using the formula:

$$AI_{u,t} = AI_{u,t} + AI_{j,t} \times a_{ju} \text{ , for } t \in \{1 \ldots T\}$$

6. If there are any level h nodes that need assembling (i.e. have children), then we increment h, and go back to step 1.

7. Using the 0-H MKP, we select the next highest scoring unfilled order k computed by Step 3. Using this order k, we construct a graph consisting of only the parts required to satisfy this order such that for each part j, $V_p - v_j \leq d_k$ where V_p is the longest lead-time path from j to order k. The order k part graph contains only the left over

inventory that can be used to satisfy order k by the due date d_k. **We set the inventory for each part i in the graph to be $AI_{i,d_k-(V_p-v_i)}$ which represent the latest accumulated inventory we can consider for pushing up the part hierarchy.**

Using the order k part graph, do the following:

a. For any purchased nodes in the order k part graph, we consider the 0-H MKP based on the original graph. If $r_{jk} = 0$, then we can supply as much inventory as needed from part j to satisfy the order k, therefore, set $AI_{i,d_k-(V_p-v_i)} = \infty$.

   Otherwise, we can only use the inventory $AI_{i,d_k-(V_p-v_i)}$ currently available.

b. Otherwise, create the 0-(h-1) MKP from the order k part graph consisting of parts i which are assembled from parts j that exist only in level h. Next, rank each part i in level h-1 in decreasing order by b_i/r_{ik}. This ordered list permits us to give inventory to parts i in level h-1, which are closest to meeting the demands placed upon them. For each parts i in the ordered list we do the following: If $r_{ik} < b_i$, then go to the next part i. Otherwise, for each connected part j which exists only in level h compute:

$$w_{ji} = \left\lfloor AI_{j,d_k(V_p-v_j)} \Big/ a_{ji} \right\rfloor$$

   Let $build_i = \min (w_{ji}$: where j is a connected part in level h of part i in level h-1). If $build_i > r_{ik}-b_i$, then let $build_i = r_{ik} - b_i$. Next, adjust all inventories by adding $build_i$ to b_i and $AI_{i,d_k-(V_p-v_i)}$, and decreasing the inventories $AI_{i,d_k-(V_p-v_i)}$ for sub-

parts j by $a_{ji} \cdot build_i$. We proceed to the next part i in the ordered list until all h-1 have been considered.

c. Check to see if $b_i \geq r_{ik}$ for all nodes i in level h-1. If so, order k is now satisfied and can be marked complete. If not, we decrement h. If h = 1, proceed to Step d, otherwise go to step b.

d. If for the ordered node m_{k_t}, if $AI_{m_{k_t}, d_k - (V_{ik} - v_i)} \geq o_k$, then the order is satisfied and can be marked as complete. We update the original hierarchy with the new inventory and repeat Step 6 until all orders have been considered at which time we stop. Any remaining orders cannot be completed on time.

6.4.2 Example

Consider the following example solved using the dynamic CBMSP:

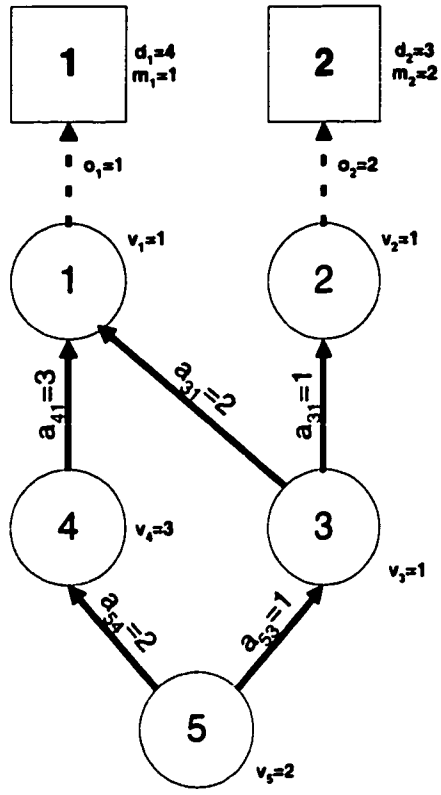


Figure 38. Initial Graph

In this example, S_1 = [0, 0, 0, 0, 0], S_2 = [0, 0, 1, 0, 0], S_3 = [0, 0, 2, 1, 2], S_4 = [1, 0, 2, 0, 0], and S_5 = [2, 1, 1, 0, 1].

3. Compute the accumulated inventory $AI_{i,t}$ for each part $i \in I$ and $t \in T$ as follows: $$AI_{i,t} = \sum_{j=0}^{T} S_{i_j}$$ **Where AI_{i_0} is the initial inventory, and S_i is the supply vector for part i over time t. In addition, AI and S are defined as the accumulated inventory matrix, and supply matrix respectively.**	S is an i×t matrix where $t \in \{0,...,4\}$ or 5 periods and $i \in I$. $$S = \begin{array}{c} \text{Part } i = \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 & 2 \\ 0 & 0 & 2 & 0 & 0 \\ 2 & 1 & 1 & 0 & 1 \end{bmatrix}$$ $$AI = \begin{array}{c} \text{Part } i = \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 3 & 5 \\ 1 & 1 & 3 & 3 & 3 \\ 2 & 3 & 4 & 4 & 5 \end{bmatrix}$$
2. For each node j in level h, find a longest bottom(purchased part) to j node path, where the length of the path is identified as the sum of the v_i's for all nodes i in the path, including j. Let the label $v_{bottom,j}$ for j be the length of this longest path. *This represents the earliest time that part j can be ready assuming everything below it is assembled or purchased.*	h = 1 $v_{bottom,1} = 6$, $v_{bottom,2} = 4$
3. If h = 1, we can transform levels 0 and 1,	h = 1

into a level 0-1 MKP directly by using the **Dynamic** General CBMSP transformation to MKP algorithm described in Section 6.2. In this transformation we consider the level 1 items as purchase parts with purchase time $v_{bottom,j}$ and ignore the other items in the hierarchy.	

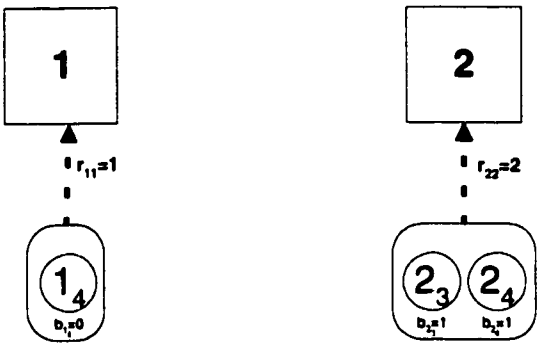


Figure 39. Step 3 (h=1) After transformation to MKP[4]

4. Solve MKP representing the flat level 0-h CBMSP from Step 3 using an exact or heuristic method such as the Greedy Hill-	Because $r_{ik} \geq b_i$ for $i \in I$ and $k \in K$ $\therefore$ Solution Set = { }.

[4] Because $d_1 = 4$ then nodes 1_t were created for $t \geq 4$. Likewise, because $d_2 = 3$ then nodes 2_t were created for $t \geq 3$.

Climbing Algorithm described in Section 3.3.1. For each order k for which $x_k = 1$ (the order is filled on time) we mark the order complete in our problem. In the original hierarchy, remove any inventory in the level h nodes, which has been used to make order k on time.	
5. Recall that some nodes in level h may also exist in other levels higher than h (i.e. levels h+1, h+2, ..., H). Consider the level h nodes, which no not exist in other levels high than h. For some of these nodes we can give away left over inventory as follows: b. For each node j in level h with leftover inventory, we determine if connected nodes u in level h+1 have j as the only parent. If so, **recalculate the accumulated inventory vector** for each node u in level h+1 by "disassembling" the parts and "pushing" the inventory down to node m using the formula: $AI_{u,t} = AI_{u,t} + AI_{j,t} \times a_{ju}$, for $t \in \{1...T\}$	No nodes can push inventory down.

6. If there are any level h nodes that need assembling (i.e. have children), then we increment h, and go back to step 2.	

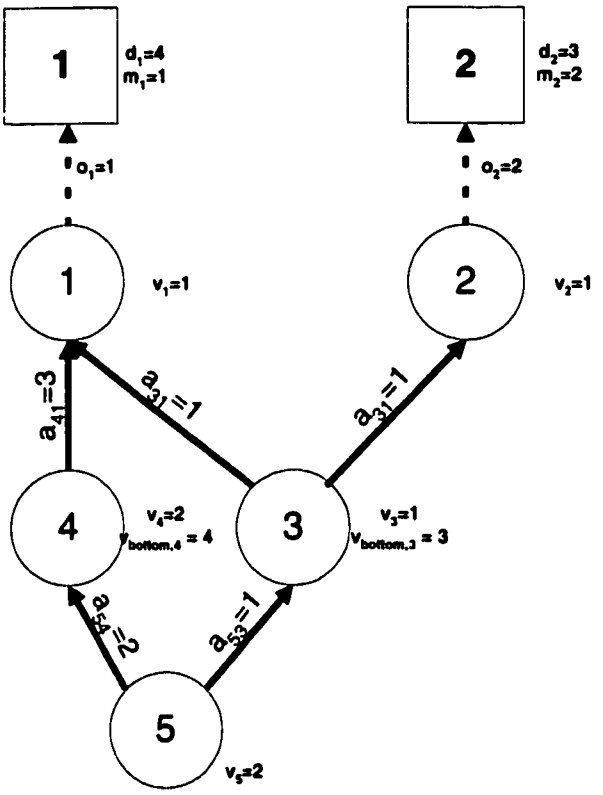


Figure 40. h = 2, After Step 2.

3. Otherwise, if h > 1, then we will consider the hierarchy consisting of levels 0, 1, …, h. We transform the levels 0,1, …, h into a 0-h MKP using the **Dynamic**	h=2 $v_{bottom,4} = 4$, $v_{bottom,3} = 3$

General CBMSP transformation to MKP algorithm described in Section 6.3. In this transformation we consider the level h parts as purchased parts with purchase time $v_{bottom,j}$ and ignore the items not in levels 0,1, ..., h. We also ignore the inventories for items in levels 1,2, ..., h-1. When calculating the i to k paths, we only consider paths of exactly length h.	

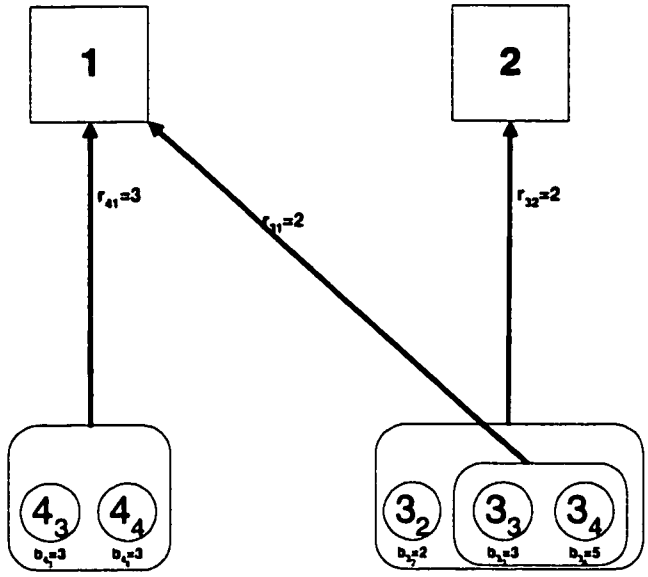


Figure 41. h=2, After Step 3[5].

[5] Since $d_1 - v_1 = 3$ then nodes, 4_t, 3_twere created for $t \geq 3$ and $d_2 - v_2 = 2$ then nodes 3_t were created for $t \geq 2$.

Step 4.	Order 1 can be satisfied. $\therefore AI_3 = [0, 0, 0, 1, 3]$ Since $r_{32} \geq b_{3_t}$ where $t = 2,3,4$ Therefore, Order 2 cannot be satisfied.
Step 5.	No Inventory can be pushed down.
Step 6.	Go to Step 2.
Step 2.	$h = 3$ $v_{5,bottom} = v_5 = 2$

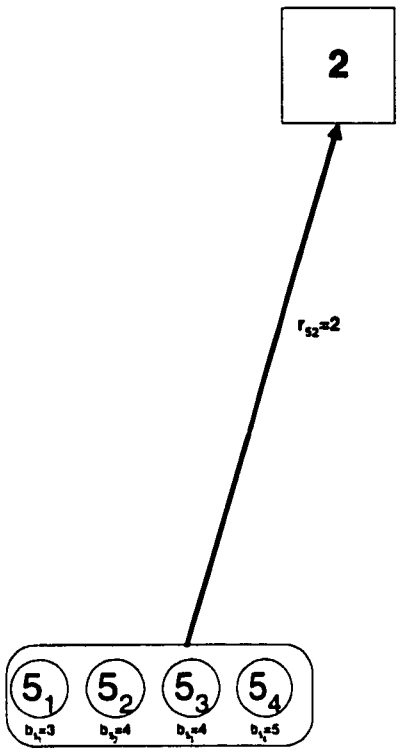


Figure 42. h=3, After Step 3[6].

Step 4.	$r_{52} \leq b_{5_t}$ where t = 1,2,3,4 $\therefore$ Order 2 can be satisfied..
Step 7	There are no remaining orders so we stop.

The heuristic finishes satisfying orders 1 and 2.

---

[6] Since $d_2 - (v_2 + v_3) = 1$, $r_{52} = 2$ for $t \geq 1$.

7 Conclusions and Future work

7.1 Conclusion

This paper has attempted to give the reader an appreciation of the constraint-based master scheduling problem. This was done through a presentation of the background of the problem, and the types of research currently being performed. We proceeded to present a model for the CBMSP and propose a heuristic based upon a series of transformations to Multiple Knapsack Problems followed by subsequent solving of these problems using a greedy heuristic method. The complexity for the heuristic was discussed and was shown to be $\mathbf{O(h \cdot |V|^3)}$ where h is the number of levels in the part hierarchy and V is the number of parts. In addition, a methodology for interpreting the results of the heuristic was proposed in order to determine the orders that were delivered, when, and the parts required.

Through implementation, an attempt was made to show how the heuristic could be used on real production data. An application implementing the heuristic along with a graphical environment for observing and testing the heuristic was developed and applied against test cases from the company Manugistics. Manugistics offers solutions relating to supply chain management in which the constraint based master scheduling problem is a facet. This testing showed that the data given reflected a model that was similar to the one described, but had minor differences relating to such things as order forecasting, and multi-purpose lead-times per part. It also supported other indicators such as shrinkage, yield, planning fence, and so forth.

A series of observations of the performance of the heuristic were discussed, revealing how the handling of leftover inventory internal to the part hierarchy impacted the overall effectiveness of the algorithm. Through testing, this assertion was reinforced by comparing an approach that ignored this internal inventory with the CBMSP heuristic. In fact, the left over internal inventory had a significant effect on the heuristics ability to find an optimal solution.

Finally, an extension to the model was discussed relating to the dynamic supply of inventory for parts within the part hierarchy. The necessary changes to the transformation to MKP and the CBMSP heuristic were provided to enhance the heuristic to support this extension.

7.2 Future Research

Other areas of future research may include:

1. The objective for CBMSP heuristic was to maximize the number of orders delivered on time. Future research could consider other objective functions such as minimizing the amount of internal inventory, or inventory ordering strategies that can satisfy cyclic ordering patterns which may exist in production situations.

2. Testing could be expanded to include comparisons with the ILP implemented through some ILP packages as well as other types of production data.

3. The CBMSP heuristic implementation and incorporate it into a production system. The goal of this would be to see how well the heuristic performs under continuous

load with production data. Second, it would useful to monitor the heuristic's behavior and effectiveness over a period of time.

4. Order Forecasting could be incorporated into the model in which a mechanism for giving priority to orders that have high demand.

5. Parts could be allowed to be both assembled or purchased.

6. Impacting the idea of parts existing at different geographical locations. This incorporates the concept of sites and transfer lead-times into the model.

7. Considering the notion that only a specific number of parts can be introduced within a lead-time period.

8. The dynamic supply extension of the CBMSP could be implemented and tested. This extension is a likely requirement in a production system. Monitoring only the static inventory state fails to consider the impact of supply that in reality occurs.

Bibliography

[1] Boyd S., Holte R.C., Wang, Y.(1997), "The Constraint Based Master Scheduling Problem", Department of Computer Science, University of Ottawa

[2] B. Fleischmann, and H. Meyr, "The general lotsizing and scheduling problem", August 1996, Dept. for Production and Logistics, University of Augsburg, Germany.

[3] M. Mathur, "Inventory Cost Model for 'Just-In-Time' Production", *Proceedings of the 1994 Winter Simulation Conference*, ed. J.D. Tew, S. Manivannan, D.A. Sadawski and A.F. Seila.

[4] M.J. Drevna, "Assessment of Alternative Inventory Methodologies for High-Volume Production Systems", *Proceedings of the 1994 Winter Simulation Conference*, ed. J.D. Tew, S. Manivannan, D.A. Sadawski and A.F. Seila.

[5] M.C. Fu, K.J Healy, "Simulation Optimization of (s,S) Inventory Systems", *Proceedings of the 1994 Winter Simulation Conference*, ed. J.D. Tew, S. Manivannan, D.A. Sadawski and A.F. Seila.

[6] D. Veeramani, "Task and Resource Allocation Via Auctioning", *Proceedings of the 1994 Winter Simulation Conference*, ed. J.D. Tew, S. Manivannan, D.A. Sadawski and A.F. Seila.

[7] Morales D., Roda J., Almeida F., Rodriguez C., Garcia F., "Integral Knapsack Problems: Parallel Algorithms and their Implementation on Distributed Systems", Dpto. Estadistica, Investigacion Operativa y Computacion, Universidad de La Laguna

[8] P.C. Chu and J.E. Beasley, "A Genetic Algorithm for the Multiconstraint Knapsack Problem", The Mangement School, Imperial College

[9] M.R. Garey, and D.S. Johnson, "Computers and intractability: a guild to the theory of NP-completeness". *Freeman*, 1979.

[10] W. Shih. "A branch and bound method for the multiconstraint zero-one knapsack problem". *Journal of the Operational Research Society*, 30:369-378, 1979.

[11] B. Gavish and H.Pirkul. Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality. *Mathematical Programming*, 31:78-105, 1985.

[12] A.V. Cabot, An enumeration algorithm for knapsack problems. *Operations Research*. 18:306-311, 1979.

[13] A.L. Soyster, B. Lev, and W. Slivka, Zero-one programming with many variables and few constraints. *European Journal of Operational Research*, 2:195-201, 1978.

[13] R. Loulou, E. Michaelides. "New greedy-like heuristics for the multidimensional 0-1 knapsack problem", *Operations Research* 27:1101-1114, 1979.

[14] Y. Toyoda. "A simplified algorithm for the obtaining approximate solutions to zero-one programming problems", *Management Science*, 21:1417-1427, 1975.

[15] M.J. Magazine, O. Oguz, "A heuristic algorithm for the multidimensional zero-one knapsack problem." *European Journal of Operational Research*, 16:319-326, 1984.

[16] S. Senju, Y. Toyoda, "An approach to linear programming with 0-1 variables", *Management Science*, 15:196-207, 1968.

[17] H. Everett, "Generalized lagrange multiplier method for solving problems of optimum allocation of resources", *Operations Research*, 11:399-417, 1963.

[18] A.H.G. Rinnooy Kan, L. Stougie and C. Vercellis, "A class of generalized greedy algorithms for the multi-knapsack problem", *Discrete Applied Mathematics*, 42:279-290, 1993.

[19] H. Pirkul, "A heuristic solution procedure for the multicontraint zero-one knapsack problem", *Naval Research Logistics*, 34:161-172, 1987.

[20] V. Gavish, H. Pirkul, "Allocation of databases and processors in a distributed computing system in J. Akoka", editor, *Management of Distributed Data Processing*, pages 215-231, North-Holland, 1982.

[21] F. Glover and G.A. Kochenberger, "Critical event tabu search for multidimensional knapsack problems", In 1.11. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory and Applications*, pages 407-427, Kluwar Academic Publishers, 1996.

[22] J. Thiel, S. Voss, "Some experiences on solving multiconstraint zero-one knapsack problems with genetic algorithms", *INFOR*, 32:226-242, 1994.

[23] http://www.manu.com/USMB/index.html