MULTI-PERIOD MULTI-DIMENSIONAL KNAPSACK PROBLEM AND ITS APPLICATION TO AVAILABLE-TO-PROMISE

Hoong Chuin LAU and Min Kwang LIM
School of Computing
National University of Singapore,
3 Science Drive 2, Singapore 117543
lauhc@comp.nus.edu.sg

Abstract

This paper is motivated by a recent trend in logistics scheduling, called Available-to-Promise. We model this problem as the multi-period multi-dimensional knapsack problem. We provide some properties for a special case of a single-dimensional problem. Based on insights obtained from these properties, we propose a two-phase heuristics for solving the multi-dimensional problem. We also propose a novel time-based ant colony optimization algorithm. The quality of the solutions generated is verified through experiments, where we demonstrate that the computational time is superior compared with integer programming to achieve solutions that are within a small percentage of the upper bounds.
Keywords: Multi-dimensional knapsack problem, Logistics, Tabu search, Ant colony.

1. Introduction

The motivation of this work arises in an e-Commerce ordering system or an e-Market place where customer requests arrive dynamically (through, say the web). The key requirement of the so-called Available-to-promise capability is that suppliers can "promise" to meet batches of online customer requests with the available inventories (in the warehouse) over a certain planning horizon. Each request comprises a collection of items, the quantity of each item, and a specific period when this request needs to be fulfilled. A request may be fulfilled at some period later (but not earlier) than the prescribed period. A late penalty cost will be incurred which is period-dependent (usually, the later the fulfilment the higher the penalty). Hence the net profit, defined as the total revenue minus the total penalty cost, is also period-dependent.

For available-to-promise to be practically useful, suppliers must be able to examine the current inventory profile and make online decisions whether they can promise fulfilment. Given the NP-hardness of the underlying problem, global optimality is unlikely to be found quickly. This paper proposes and compares two heuristics that obtain near-optimal solution quickly.

We model the problem as a generalization of the multi-dimensional knapsack problem (MKP), which we call the multi-period multi-dimensional knapsack problem (MPMKP). MPMKP can be seen as a generalization of the well-studied dynamic capacitated lot-sizing problem in manufacturing, where there is a single customer and a single item to supply. This problem has been extended in different ways, including the work of Lau and Song (2002) which consider extension to multiple customers and time windows. In this paper, we consider multi-customer, multi-item composite demands with monotonically decreasing profit function.

Our objective here is to maximize the net profit by fulfilling a subset of all requests over the discrete planning horizon subject to item availability constraints. This problem can be modeled as an MPMKP, where a request can be regarded as an object to be added to an MKP of certain period while the available quantities of all items at the beginning of this period can be regarded as the capacity constraint of this knapsack. Each period's MKP is related to the one of the next period in that the remaining inventory of any item at the end of this period is carried forward to the next period given that this period is not the last one of the discrete planning horizon.

We assume that the reader is familiar with the concepts of tabu search and ant colony optimation.

This paper is organized as follows. In the next section, we present a brief literature review and notations used in the paper. We begin by studying a special case, the multi-period single-dimensional knapsack problem. Some combinatorial properties and a heuristic with performance bound for this special case are presented. Next, we propose two meta-heuristic approaches for the MPMKP: an efficient Tabu Search, and a novel Ant Colony Optimization algorithm that handles the notion of time explicitly. The quality of the solutions obtained is verified through extensive numerical experiments on extended benchmark problems.

2. Literature Review and Preliminaries

While many research works are available in the literature for MKP, to our best knowledge, there is no published research work for the MPMKP. Shih (1979) presented a branch and bound algorithm for MKP. Another branch and bound algorithm was developed by Gavish and Pirkul (1985) where various relaxations of the problem were used, their algorithm was compared with the exact algorithm of Shih (1979) and was found to be faster by at least one order of magnitude. Other previous exact algorithms, with only limited success reported, include dynamic programming. Loulou and Michaelides (1979) presented a greedy method based on Toyoda's primal heuristic (1975). Balas and Martin (1980) used linear programming by relaxing the integrality constraints and heuristically setting the fractional solution to become integral while maintaining feasibility. Pirkul (1987) presented a heuristic algorithm which makes use of surrogate duality. Freville and Plateau (1994) presented an efficient preprocessing algorithm for MKP. Freville and Plateau (1997) presented a heuristic for the special case, the bidimensional knapsack problem, their heuristic incorporated a number of components including problem reduction, a bound based surrogate relaxation and partial enumeration. A number of papers involving the use of tabu search to solve MKP have appeared such as Dammeyer and Voss (1993) and Aboudi and Jornsten (1994).

Before we present the mathematical formulation of the MPMKP, we introduce some notation:

T : number of periods in the planning horizon
N : total number of requests over the planning horizon
M : total number of items
t^j : the prescribed period where request j is due to be fulfilled
a_{ij} : order size for item i in request j
b_{it} : incoming quantity of item i at the beginning of period t
p_{jt} : net profit if request j is fulfilled in period t.

In this paper, we assume that the profit function p_{jt} of any request j is a concave function with respect to $t(1 \le t \le T)$ and partial delivery is not allowed. MPMKP becomes a linear program (which is easy to solve) if partial deliveries are allowed. The maximum profit of the request j will be attained at period t^j and the profit decreases monotonically from period t^j onwards with increasing time. Without loss of generality, we assume that for any request j, $p_{j,t^j} > p_{j,t^j+1}$ if $t^j < T$. Since we assume that the inventory of any item can be carried forward at no additional cost and there is no inventory capacity constraint at the warehouse, there is no need to consider early fulfillment, and hence we assume $p_{jt} = 0$ for all $1 \le t < t^j$ if $t^j > 1$. The MPMKP can be mathematically formulated as:

$$\max Z = \sum_{j=1}^{N} \sum_{t=1}^{T} p_{jt} x_{jt} \text{ s.t.}$$

$$\sum_{j=1}^{N} a_{ij} x_{jt} + s_{it} = b_{it} + s_{i,t-1} \quad 1 \le i \le M, 1 \le t \le T \quad (1)$$

$$\sum_{t=1}^{T} x_{jt} \le 1 \quad 1 \le j \le N \quad (2)$$

where $x_{jt} \in \{0,\ 1\}, s_{i0} = 0, s_{it} \ge 0$. If $x_{jt} = 1$, it means that request j is fulfilled at period t, otherwise, it is not fulfilled at period t. The s_{it} is a slack variable, *i.e.*, the remaining inventory level at the end of period t for item i. Without loss of generality, we also assume that the initial inventory level at the beginning of the planning horizon is zero for all M items, *i.e.*, $s_{i0} = 0$. Constraints (1) are inventory balance constraints. Constraints (2) mean that each request can be fulfilled at no more than one period.

From the above formulation, it is obvious that if $T = 1$, the MPMKP is reduced to an MKP. If $T = 1$ and $M = 1$, it is reduced to the well known 0-1 knapsack problem (KP). Since the KP is NP-hard, the MPMKP is NP-hard.

3. Multi-period single-dimensional knapsack problem

In this section, we will study a special case, the multi-period single-dimensional knapsack problem (MPKP). We first provide some basic properties of the continuous relaxation in subsection 3.1. By using these properties, we propose a heuristic with performance bound for solving MPKP in subsection 3.2.

3.1 Continuous relaxation

Consider the continuous relaxation of the MPKP (i.e. x_{jt} is a real number and $0 \le x_{jt} \le 1$), which is denoted by C-MPKP. In the following, we will characterize the structure of an optimal solution for C-MPKP.

Lemma 1 *Consider an optimal solution for C-MPKP and let* $\Theta = \{t_1, t_2, \ldots, t_k\}$ *where* $1 \le t_1 < t_2 < \ldots < t_k \le T$ *denote the set of periods with zero remaining inventory level at the end of the respective periods. Then,*

1. *Every request partially or fully satisfied at period* t, *for* $t = 1, \ldots, t_1$, *was a request which was due at period* t.
2. *For any* $i \ge 2$, *any request fully or partially satisfied at period* $t_{i-1}+1$ *was due at one of the periods* $1, \ldots, t_{i-1}+1$. *Any request satisfied at some period* $t_{i-1} + 1 < t \le t_i$ *was due at period* t.

Proof. Suppose that $X = \{x_{jt} |\ 1 \le j \le N,\ 1 \le t \le T\}$ is an optimal solution to C-MPKP, and let $t_1 < t_2 < \ldots < t_k$ be the periods, if any, such that the remaining inventory level at the end of any period t_i

is zero and the remaining inventory level at any period $t \neq t_i (1 \leq i \leq k)$ is positive.

If $k = 0$, it is trivial. Hence we assume that $k > 0$.

For t_1, suppose that there is some request j due in period t_a and satisfied at period t_b where $1 \leq t_a < t_b \leq t_1$. Since $p_{j,t_a} > p_{j,t_b}$, and the remaining inventory at the end of period t_a is positive, we could remove some part of the request j from period t_b and fulfill it at period t_a, thus contradicting the optimality of X. Therefore, the result in part (1) is correct. A similar argument proves part (2). $\diamondsuit$

By using the results in Lemma 1, we propose the following greedy algorithm for solving C-MPKP. Note that each request can be partially or completely satisfied at some period.

Greedy algorithm A: (solving C-MPKP)

while *(there are some unsatisfied requests and some inventories)* **do**

1. *Find request j^* and T_{j^*} such that $\frac{p_{j^*T_{j^*}}}{a_{1j^*}} = max\{ \frac{p_{jt}}{a_{1j}} |$ request j can be completely or partially satisfied in period t $\}$.*
2. *Satisfy request j^* at period T_{j^*} as much as possible.*
3. *Update the inventory of each item at the beginning of each period and the list of unsatisfied requests.*

The feasible solution obtained by using the above algorithm may be not an optimal one for C-MPKP. But, if the profit function is a single point function, i.e., $p_{jt} = 0$ if $t \neq t^j$, we can show in the following that the solution obtained by using *Greedy Algorithm A* is an optimal one for C-MPKP. In practical terms, this means that late fulfillment does not add any profit. Even for this special profit function, the corresponding MPKP is a generalization of the 0-1 knapsack problem. Although the single point profit function for any request is very special, any solution approach for solving it should also consider the coordination between different periods, *i.e.*, to decide whether the remaining inventory at current period should be used to fulfill some request due in current period or carried forward to fulfill some other request due in later period.

Lemma 2 *If the profit function for each request is a single point function, C-MPKP is optimally solved by Greedy Algorithm A.*

Proof. Note that the optimal solution structure described in Lemma 1 is further simplified if the profit function for each request is a single point function, *i.e.*, the satisfied requests at each period t are due in at period t. If $k = 0$, it is trivial. In the following, we assume that $k > 0$ and call periods $(1, \ldots, t_1)$ as cycle 1, ..., periods $(t_{k-1} + 1, \ldots, t_k)$ as cycle k, and periods $(t_k + 1, \ldots, T)$ as cycle $k + 1$.

We prove this lemma by induction on T, the total number of periods.

For $T = 1$, the problem becomes the continuous relaxation of the 0-1 knapsack problem. If we replace any satisfied request by any other unsatisfied request, the total profit will not be increase. Hence it is true.

Suppose that for any $T \leq n - 1 (n \geq 2)$ the lemma is true. We shall show that it is also true for $T = n$.

If we replace some request fulfilled in the first cycle by some other unfilled request or carry some inventory consumed in the first cycle to the next cycle, the objective value of the new solution will be less than or equal to the objective value of the solution obtained by using *Greedy Algorithm A*. This can be seen from the procedure of the *Greedy Algorithm A* and the fact that any fulfilled request j must be fulfilled at period t^j (its due in period). Hence, there will be some optimal solution such that the first cycle is part of this solution. The remaining problem will be reduced to a C-MPKP over a planning horizon of $n - t_1$ periods. By using the assumption for $T \leq n - 1$, we complete the proof. $\diamondsuit$

3.2 A greedy heuristic for MPKP

We can generate two related MPKPs by modifying the profit function p_{jt} of any request $j (1 \leq j \leq N)$. From our assumption, we know that p_{jt} is concave and will attain the maximum at the period t^j. Thus, if we replace p_{jt} by $p^u_{jt} = p_{j,t^j}$, the objective value Z^u of an optimal solution will be an upper bound of Z^*, the optimal objective value of the original MPKP. Similarly, if we replace p_{jt} by the single point function $p^l_{jt} = p_{j,t^j}$ if $t = t^j$ and $p^l_{jt} = 0$ otherwise, the objective value Z^l of an optimal solution will be a lower bound of Z^*.

An upper bound of Z^u can be computed in the following way.

Lemma 3 *If the profit function for any request $j (1 \leq j \leq N)$ is a constant function p^u_{jt}, we have $U - max_{\{1 \leq j \leq N\}} p^u_{jt} \leq Z^u \leq U$ where U is the optimal objective value of the continuous relaxation.*

Proof. As the profit of any request is constant with respect to period $t (1 \leq t \leq T)$, we can delay the fulfillment of all requests until the last period and carry forward all the inventories to the last period. Hence, the MPKP is equivalent to the 0-1 knapsack problem. Clearly, the deviation of the optimal value from the upper bound U obtained by continuous relaxation is at most $max_{\{1 \leq j \leq N\}} p_{j1}$, since there is at most one request that is not completely satisfied in the latter solution, and that request contributes a profit value of at most $max_{\{1 \leq j \leq N\}} p_{j1}$ to the objective. $\diamondsuit$

The *Greedy Algorithm A* can be easily modified to solve the MPKP. Henceforth, if some request cannot be completely satisfied at any period with positive net profit, we say that the best profit for this request is $-\infty$, otherwise, the best profit of any unsatisfied request j is defined as $max\{p_{jt} |$ request j can be completely satisfied at period t with positive net profit

}. The maximizer is denoted as T_j. We say that a request is non-active if the best profit of the request is $-\infty$; otherwise, this request is active.

Greedy algorithm B: (solving MPKP)

1. *Compute the best profit and T_j for every request j.*
2. *Set the requests with best profit $-\infty$ to non-active.*
3. **while** *(there are some active unsatisfied requests)* **do**
   a. *Find request j^* such that $\frac{p_{j^*T_{j^*}}}{a_{1j^*}} = max\{\frac{p_{jT_j}}{a_{1j}}|$ request j is active and unsatisfied in period t $\}$.*
   b. *Satisfy request j^* completely at period T_{j^*}.*
   c. *Update the inventory for each item at the beginning of each period, the best profit and the active status of any unsatisfied request.*

It is obvious that *Greedy Algorithm B* is the discrete version of *Greedy Algorithm A*. By using Z^l and Z^u defined above, we can obtain the performance bound of the above *Greedy Algorithm B* for the MPKP.

Theorem 1 *Let Z^* and Z^H be the objective values of the optimal solution and the feasible solution obtained by using Greedy Algorithm B, respectively. Then we have $Z^l \le Z^H \le Z^* \le Z^u$.*

Proof. It is obvious that $Z^H \le Z^* \le Z^u$ from the definition of Z^u. About $Z^l \le Z^H$, we only note that the candidates of all active requests at each iteration of *Greedy Algorithm B* if the profit function of request $j(1 \le j \le N)$ is $p^l_{jt}(1 \le t \le T)$ is a subset of the candidates of all active requests at the same iteration of *Greedy Algorithm B* if the profit function of request $j(1 \le j \le N)$ is $p_{jt}(1 \le t \le T)$. $\diamondsuit$

4. Multi-period multi-dimensional knapsack problem

In this section, we will provide a two-phase solution approach for the MPMKP by using the insights obtained in the previous section. The first phase is a greedy heuristic which is a generalization of *Greedy Algorithm B*, followed by a local improvement phase via tabu search. We then propose another heuristic based on ant colony optimization, and compare the results obtained in the subsequent section.

4.1 A greedy heuristic for MPMKP

The basic idea is first to transform an MPMKP to an MPKP, and then obtain a feasible solution of the MPMKP by adapting *Greedy algorithm B* proposed for the MPKP in section 3. This idea is not new, almost all solution approaches for solving MKPs come from this idea: first transform an MKP to a KP by proposing a multiplier for each item and computing the weighted capacity of the knapsack and the weighted weight of each object, then provide a feasible solution for MKP by using some solution approach for the KP.

In our following *Greedy Algorithm C*, the multiplier of item $m(1 \le m \le M)$ at period t is simply defined as the used inventory r_m for item m if there is at least one request satisfied at period t, otherwise, we simply define the multiplier for each item is 1. Therefore, the multiplier does not depend on the request. It is the same for all requests at period t. Suppose that $u_{mt}(1 \le m \le M,\ 1 \le t \le T)$ are the multipliers for item m in period t. The weighted weight w_{jt} of request j in period t is defined as:

$$w_{jt} = \sum_{i=1}^{M} u_{it} a_{ij}.$$

We call request j active in period t if request j can be satisfied at period t with positive net profit. Otherwise, we call the request j non-active. Similarly, if request j can be satisfied in at least one period with positive net profit, we call it active, otherwise, we call it non-active.

Greedy algorithm C: (solving MPMKP)

1. *Set the active status for all requests.*
2. *Compute the multipliers for all resources at each period.*
3. **while** *(there are some active unsatisfied requests)* **do**
   a. *Find request j^* and period t^* such that $\frac{p_{j^*t^*}}{w_{j^*t^*}} = max\{\frac{p_{jt}}{w_{jt}}|$ request j is active and unsatisfied in period t $\}$.*
   b. *Satisfy request j^* at period t^*.*
   c. *Update the inventory for each item, the multipliers for all items at period t^*, and update the active status of each unsatisfied request.*

4.2 Tabu Search Improvement for MPMKP

By using *Greedy algorithm C* described in subsection 4.1, we obtain an initial feasible solution for the MPMKP. In the following we propose a tabu search improvement phase. First, we will introduce two operators (moves), namely the relocate operator and the exchange operator.

Relocate Operator: Suppose request $i(1 \le i \le N)$ is satisfied at period $t_i(1 \le t_i \le T)$. We remove this request from period t_i and satisfy it at period $t' \ne t_i(1 \le t' \le T)$. If this can be done feasibly, we define the profit saving of this relocate operation as $(p_{it'} - p_{it_i})$, otherwise, we define the profit saving of this relocate operation as $-\infty$.

Exchange Operator: Suppose there are two requests i and j which are satisfied at periods t_i and $t_j(\ne t_i)$, respectively. We remove request i from period t_i and satisfy it at period t_j, at the same time, we also remove request j from period t_j and satisfy

it at period t_i. If this can be done feasibly, we define the profit saving of this exchange operation as $(p_{i,t_j} + p_{j,t_i}) - (p_{i,t_i} + p_{j,t_j})$, otherwise, we define the profit saving as $-\infty$.

Tabu List: It is a list of moves executed during the last L steps where L is the length of the list. The status of each move is either *tabu* or *non-tabu*. If a move is in the tabu list, the status of this move will be *tabu*, otherwise, its status is *non-tabu*.

Aspired Criteria: Each move A is associated with an aspired profit value x_A initialized to zero. When a move is being executed, there are two possibilities. If $x_A = 0$, we call move A *aspired* and update $x_A = max\{$ *current total profit + profit saving of* A, $0\}$. If $x_A > 0$, we call move A aspired if *current total profit + profit saving of* $A > x_A$ and update $x_A = max\{$ *current total profit + profit saving of* A, $x_A\}$.

4.3 Ant Colony Optimization for MPMKP

Ant Colony Optimization (ACO) (Dorigo and Di Caro (1999)) is a meta-heuristic inspired from the observation of nature, in this instance the foraging behavior of a colony of ants. Typically the pheromone trail, a key component in the ACO operation, is specified using a node-to-node matrix, which is not appropriate for scheduling problems involving time. We propose in this paper a new scheme for the pheromone trail that explicitly handle the notion of time to improve the performance of ACO for scheduling problems.

Intuitively, a natural extension of ACO to scheduling problems is what we call Scheme A, where in addition to the standard pheromone trail presenting a value from one node to another, one adds a third dimension of time to the pheromone trail structure. However, Scheme A suffers an obvious problem in that it is computationally intensive. This can be derived by considering a generic ACO implementation using x iterations, y ants, and n nodes. The performance of such algorithm is $O(xyn^2)$, since the operation incurs a n^2 pheromone trail computation. By adding the time dimension of size T, the algorithm complexity becomes $O(xyn^2T)$.

In this paper, we propose the following Scheme B for solving scheduling problems: Modify the pheromone trail from a node-to-node trail to a *node-to-period* trail. This pheromone trail will specify the attractiveness of placing a node in a time-period. The computational complexity becomes $O(xynT)$. Since the number of time-periods T is typically small compared to the number of nodes n, we achieve a computational speedup. Furthermore, this structure is logically better when dealing with time, since it directly informs the algorithm where the best period to place nodes is. This scheme makes sense for problems like MPMKP where sequencing of nodes (within a knapsack) is not a concern.

The computation of the transition probabilities is given as follows:

$$\eta_{jt} = \begin{cases} \frac{p_{jt}}{\phi_{jt}}, & \text{if } \phi_{jt} \geq 0, \\ 0, & \text{otherwise,} \end{cases}$$

$$\phi_{jt} = \sum_{i=1}^{M} \frac{a_{ij}}{c_{it}}$$

where η_{jt} represents the local heuristics for moving from node i to node t; ϕ_{jt} represents the sum tightness of all items ($1 \leq i \leq M$) on request j; and c_{it} represents the cumulative quantity of item i left at the beginning of period t, after subtracting quantities that have been allocated in previous periods.

5. Experimental Results

We now present numerical experiments to verify the algorithms presented in Secion 4.

The OR Library (http://www.ms.ic.ac.uk/jeb/pub/) contains 0-1 MKP instances generated with varying size and tightness, denoted by the parameter α. The problems are divided into 9 sets, each with a given M (the number of items) and n (the number of requests). There are 30 test cases within each set, the first 10 are generated with $\alpha = 0.25$, the next 10 with $\alpha = 0.5$, and the last 10 with $\alpha = 0.75$. The parameters used in each test case are generated as follows (U is a random number generator with two input values representing the lower and upper bounds):

$$\begin{aligned} a_{ij} &= U(0, 1000) \\ b_{it} &= \alpha \sum_{i=1}^{n} a_{ij}, \alpha \in \{0.25, 0.5, 0.75\} \\ p_{jt} &= \sum_{i=1}^{M} a_{ij}/M + 500U(0, 1) \end{aligned}$$

Since this set of benchmark is for 0-1 MKPs, they can only be used as test cases for the single-period multi-dimensional knapsack problem. We extended to 7-period MPMKPs (equivalent to 7 days for a week). For example, in the following table, we show how to generate a set of 9 test cases from the 30 test cases in the mknapcb$X$ ($4 \leq X \leq 9$) test sets. The first test case is generated using the first 7 test cases in mknapcb$X$ (abbrev. mX). The rest of the test cases are generated by choosing other 7 test cases from the mX test set as described in Table 1. As there are n requests due in each period $t(1 \leq t \leq 7)$, the profit for any of these n requests in period t is the same as the OR library. We define the profit function $p_{jt'} = 0$ if $t' < t$ and $p_{jt'} = p_{jt} - 0.1(t' - t)p_{jt}$ if $t' \geq t$.

In the interest of space, we will present experimental results for only one set of 250-request 30-item m8 test

Table 1 The nine test cases generated from mX

Instance	mX test case chosen	α
mX-1-n-M	1,2,3,4,5,6,7	0.25
mX-2-n-M	2,3,4,5,6,7,8	0.25
mX-3-n-M	3,4,5,6,7,8,9	0.25
mX-4-n-M	11,12,13,14,15,16,17	0.50
mX-5-n-M	12,13,14,15,16,17,18	0.50
mX-6-n-M	13,14,15,16,17,18,19	0.50
mX-7-n-M	21,22,23,24,25,26,27	0.75
mX-8-n-M	22,23,24,25,26,27,28	0.75
mX-9-n-M	23,24,25,26,27,28,29	0.75

cases. Suffice to say that all other test cases conducted exhibit very similiar behavior. We denote the objective value of our TS and ACO heuristics by Z_{TS} and Z_{AC} respectively (rounded to the nearest integer). To measure the quality of our solutions, we compare with upper bound values obtained by CPLEX, denoted by UB[1]. The run times are denoted by T_{TS} and T_{AC} respectively (in seconds). The result illustrates that both TS and ACO can quickly obtain near-optimal (within 6% deviation of the upper bound).

Table 2 Results of test cases generated from m8

Instance	UB	Z_{TS}	Z_{AC}	T_{TS}	T_{AC}
1-250-30	416534	399999	401184	145	18
2-250-30	416728	401153	405882	140	55
3-250-30	416489	399017	403577	165	19
4-250-30	775929	747851	752445	560	33
5-250-30	773089	750229	745073	599	33
6-250-30	770849	747512	749205	599	33
7-250-30	1075402	1039310	1031131	816	66
8-250-30	1078888	1054980	1045941	858	68
9-250-30	1078300	1052480	1039319	838	46

6. Conclusion

With SCM and JIT becoming business norms, suppliers must be able to react to customer demands in real-time. What we have presented in this paper is a step in this direction. It might be pointed out that this problem is not as interesting as its online version from the academic point of view. Nevertheless, we developed this methodology in consultation with commercial companies which perform online decision making based on aggregate customer requests arriving over time, rather than individual arrivals. Hence, we believe the methodology proposed in this paper is practical and sound.

[1]CPLEX took too long to obtain optimal solutions for many of the large-scale instances, so we terminte pre-maturely after running each instance with a maximum of 2 hours CPU time.

Acknowledgement. The authors like to thank Seet Chong Lua, Yuyue Song and Roland Yap for their contributions on an early draft of this paper.

References

[1] Aboudi, R. and K. Jornsten. (1994). Tabu search for general zero-one integer programs using pivot and complement heuristic, *ORSA Journal on Computing*, 6, 82-93.

[2] Balas, E. and C. H. Martin. (1980). Pivot and complement–A heuristic for 0-1 programming, *Management Science*, 26, 86-96.

[3] Dammeyer, F. and S. Voss. (1993). Dynamic tabu list management using reverse elimination method, *Annals of Operations Research*, 41, 31-46.

[4] Dorigo and Di Caro, (1999). Ant Colony Optimization: A New Meta-Heuristic, *Proc. 1999 Congress on Evolutionary Computation*, 1470-1477.

[5] Freville, A. and G. Plateau. (1994). An efficient preprocessing procedure for the multidimensional 0-1 knapsack problem, *Disc. Appl. Math.,* 49, 189-212.

[6] Freville, A. and G. Plateau. (1997). The 0-1 bidimensional knapsack problem: toward an efficient high-level primitive tool, *Journal of Heuristics*, 2, 147-167.

[7] Gavish, B. and H. Pirkul. (1985). Efficient algorithms for solving multiconstraint Zero-One knapsack problems to optimality, *Mathematical Programming*, 31, 78-105.

[8] Lau H. C. and Song Y. (2002). Combining Two Heuristics to Solve a Supply Chain Optimization Problem, *Proc. 15th European Conference on Artificial Intelligence (ECAI)*, Amsterdam, 581-585.

[9] Loulou, R. and E. Michaelides. (1979). New greedy-like heuristics for the multidimensional 0-1 knapsack problem, *Operations Research*, 27, 1101-1114.

[10] Pirkul, H. (1987). A heuristic solution procedure for the multiconstraint Zero-One knapsack problem, *Naval Research Logistics*, 34, 161-172.

[11] Shih, W. (1979). A branch and bound method for the multiconstraint Zero-One knapsack problem. *Journal of the Operational Research Society*, 30, 369-378.

[12] Toyoda, Y. (1975). A simplified algorithm for obtaining approximate solutions for zero-one programming problems, *Management Science*, 21, 1417-1427.